

Easily refutable subformulas of large random 3CNF formulas

Uriel Feige and Eran Ofek

Weizmann Institute of Science,
Department of Computer Science and Applied Mathematics,
Rehovot 76100, Israel,
{uriel.feige,eran.ofek}@weizmann.ac.il

Abstract. A simple nonconstructive argument shows that most 3CNF formulas with cn clauses (where c is a large enough 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 large enough constant) finds a subformula with $O(c^2n)$ clauses and then proves 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 $\text{poly}(\log(n)) \cdot n^{3/2}$ clauses. Our algorithm is simple enough to run in practice. We present some preliminary experimental results.

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 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 [6] it is shown that there exists c_n such that

for $c < c_n(1 - \epsilon)$ almost surely ϕ is satisfiable, and for $c > c_n(1 + \epsilon)$, ϕ is almost surely unsatisfiable (for some ϵ which tends to zero as n increases). It is also known that $3.52 < c_n < 4.596$ [12],[10],[11]. We will use random formulas with cn clauses (for $c > c_n(1 + \epsilon)$) to measure the performance of a refutation heuristic. Specifically, the refutation heuristic is considered good if for some $c > (1 + \epsilon)c_n$ it almost surely proves that a random formula with cn clauses is unsatisfiable.

Notice that for any n , as c is increased (for $c > c_n(1 + \epsilon)$), 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 an efficient 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 . However, Chvatal and Szemerédi [2] 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 [1] who showed that a random formula with $n^{3/2-\epsilon}$ clauses almost surely requires a resolution proof of size $2^{\Omega(n^{\epsilon/(1-\epsilon)})}$. 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 instances with $\Omega(n^2)$ clauses. This is done by fixing a variable x , and taking all the clauses which contain it. Fixing x to be true leaves about half of the selected clauses as a random 2CNF formula with $\Omega(n)$ clauses, which can be proved to be unsatisfied by a polynomial running time algorithm. The same can be done when fixing x to be false.

A new approach introduced by Goerdt and Krivelevich in [8], gave a significant improvement and reduced the bound into $(\log n)^7 \cdot n^k$ clauses for efficient refutation of $2k$ CNF formulas. This approach was later extended in [7],[9] to handle also random 3CNF formulas with $n^{3/2+\epsilon}$, $\text{poly}(\log n) \cdot n^{3/2}$ clauses respectively. In [3], [5] it is shown how to efficiently refute a random $2k$ CNF instances with at least cn^k clauses.

Further motivation for studying efficient refutability of random 3CNF formulas is given in [4]. There it is shown that if there is no polynomial time refutation heuristic that works for most 3CNF formulas with cn clauses (where c is an arbitrarily large constant) then 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.

Our refutation algorithm is based on similar techniques to those that appear in [7], [3], [4], but it has some advantages. Both the algorithms in [7], [9] 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 [7],[9]. As a result of this simplicity, we can show that our algorithm refutes formulas with $cn^{3/2}$ clauses, whereas the algorithms given in [7],[9] are claimed only to refute formulas with $\Omega(n^{3/2+\epsilon})$, $\Omega(\text{poly}(\log n) \cdot n^{3/2})$ clauses re-

spectively. An implementation of our algorithm refuted a random formula with 50000 variables and 27335932 clauses (see details in section 4).

In some other respects, our algorithm is more limited than the algorithms in [7],[9]. 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. It is plausible that the algorithms of [7],[9] (or a variant of them) does provide a strong refutation, though this issue is not explicitly addressed in [7],[9]. The ability to perform strong refutation is an important issue, and its relation to approximability is discussed in [4].

2 Preliminaries

2.1 The random model

Definition 1. In the $F_{n,p}$ model a random 3CNF formula is generated in the following way: each clause out of the $2^3 \binom{n}{3}$ possible clauses is chosen independently with probability of p .

Although we concentrate on a specific random model for generating random formulas, our algorithm succeeds also on other related random models. For example, it is not hard to see that our algorithm works also if we generate a random formula by picking exactly $cn^{3/2}$ clauses independently at random, with (or without) replacement. Details are omitted.

2.2 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 ‘accept’ or ‘don’t know’.
2. *Soundness*: The algorithm never outputs ‘accept’ on an instance ϕ which does not have the property P . The algorithm may output ‘don’t know’ on an instance x 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 C . We shall consider properties that are almost surely true for the random object taken from C . A certification algorithm is *complete* with respect to the probability space C if it almost surely outputs ‘accept’ on an input taken from 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 [13]. This algorithm is based on semi-definite programming; if the maximum cut in the input graph is of size at most $m(1/2 + \epsilon)$, then the algorithm outputs a certificate that the maximum cut in G is bounded by $m(1/2 + \delta(\epsilon))$, where $\delta(\epsilon) \rightarrow 0$ as $\epsilon \rightarrow 0$. A computationally simpler algorithm can be applied if the graph is random. In [5] 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 $dn(1/4 + \frac{\theta(1)}{\sqrt{d}})$, thus bounding the maximum cut by $m(1/2 + \epsilon)$ 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.

2.3 An overview of our refutation algorithm

Our algorithm builds on ideas taken from earlier work ([8], [7], [4], [5], [3], [9]). 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 to the algorithm is a random 3CNF formula ϕ with n variables and $m = cn^{3/2}$ clauses, where c is a large enough constant. The algorithm first greedily extracts from ϕ a subformula ϕ' . This is done as follows. We say that two clauses are *matched* 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 $(x_4 \vee \bar{x}_2 \vee x_3)$ are matched. ϕ' is constructed by greedily putting into ϕ' pairs of clauses that form a match, until no further matches are found in ϕ . Let m' be the number of clauses in ϕ' . A simple probabilistic argument shows that we can expect $m' = \Theta(m^2/n^2) = \Theta(c^2n)$. 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 [4], 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 [4]. 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 XOR 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 $3/4 + \epsilon$ 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} too 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 $1/2 + \epsilon$ satisfiable, and how to certify that a random 3NAE formula is not $3/4 + \epsilon$ 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 2.2. (The principle of refuting random formulas by reduction to random graph problems was introduced in [8].)

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 2.2, 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 will 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 NAE 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 $3/4 + \epsilon$ fraction of the 3NAE clauses are satisfied as 3NAE, than a $\frac{2}{3}(3/4 + \epsilon) = 1/2 + 2\epsilon/3$ 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 [13] on the whole graph, or by using the algorithm of [5] 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 $F_{n,p}$ where $p = \frac{c}{n^{3/2}}$. For convenience we will assume that the clauses of ϕ appear in a random order and that the order of literals inside each clause is random (this assumption is reasonable because we can permute the clauses and the literals inside each clause before handling ϕ). We will use $(?, w, l)$ to denote a clause in which the second and the third literals are w, l respectively and the first literal can be any literal. Let ϕ' be a subformula of ϕ constructed in the following way: for every pair of literals (w, l) we count the number of clauses in ϕ of the form $(?, w, l)$. If this number is two or more, then we take into ϕ' the first two appearances of these clauses (preserving the order of appearance). If the number of such clauses is one or less, we don't add anything to ϕ' . From each pair of clauses $(x, w, l), (y, w, l)$ in ϕ' we take the first one to the set ϕ_1 and the second to the set ϕ_2 . Each of ϕ_1, ϕ_2 is a random formula, though clauses in ϕ_i 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 forget ϕ and concentrate in refuting ϕ' . Before specifying the algorithm, we introduce additional notation and definitions which will ease the description of the algorithm.

Definition 2. Let ϕ' be a 3CNF formula with m pairs of matched clauses. Then the following graphs $G_1^1, G_2^1, G_3^1, G_1^2, G_2^2, G_3^2$ and G_{2eq} are defined as follows.

G_{2eq} : A graph whose vertices are the literals and whose edges are the following: each matched pair from ϕ_1, ϕ_2 , say $(x, w, l), (y, w, l)$ respectively, induces exactly one edge (\bar{x}, y) . Notice that we negate the literal which corresponds to the clause of ϕ_1 .

G_i^1 : A graph whose vertices are the literals and whose edges are the following: for each clause in ϕ_1 we omit the i -th coordinate and get a set of two literals which induces an edge of G_i .

G_i^2 : This graph is similar to G_i^1 but its edges are induced by ϕ_2 .

Definition 3. 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 normalized imbalance of ϕ is $\frac{\sum_{i=1}^n Im_i}{3m}$.

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

Definition 4. A 3CNF formula ϕ has the $(1 - \epsilon)$ 3XOR property if for every assignment A satisfying ϕ as 3CNF, at least $1 - \epsilon$ fraction of the clauses are satisfied as 3XOR.

Definition 5. 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 δ -fraction of the edges cross this partition.

The number of matched pairs is denoted by m (in section 2.3 we used m to denote the number of clauses in ϕ . From here on, m will denote the number of matched pairs in ϕ'). In lemma 2 it is shown that almost surely the number of matched pairs $m = 8c^2 \pm 10c\sqrt{n}$. The refutation algorithm does the following steps ($\epsilon < 1/22$ is some positive fixed constant):

1. Certify that ϕ' has the $(1 - 5\epsilon)$ 3XOR property. This is done by certifying the $(1 - 5\epsilon)$ 3XOR property for ϕ_1, ϕ_2 separately. If both ϕ_1, ϕ_2 have this property then also ϕ' has this property. Specifically, verify the following (for $i = 1, 2$):
 - (a) ϕ_i is ϵ -balanced.
 - (b) Each of the graphs G_1^i, G_2^i, G_3^i has a maximum cut bounded by $m(1/2 + \epsilon)$.
2. Certify that G_{2eq} has a maximum cut with at most $(\frac{1}{2} + \epsilon)m$ edges.

The algorithm rejects if one of the above steps fails. We will show the following facts: (1) a 3CNF formula ϕ_i with m clauses which satisfies conditions (a),(b) from step (1) of the refutation algorithm has the $(1 - 5\epsilon)$ 3XOR property; this follows from Lemma 1. (2) The random graphs $G_1^1, G_2^1, G_3^1, G_1^2, G_2^2, G_3^2, G_{2eq}$ each has a maximum cut (almost surely) bounded by $m(1/2 + \epsilon_1)$ for small $\epsilon_1 > 0$. This is done at Theorem 2. (3) There is an efficient algorithm which certifies that each of $G_1^1, G_2^1, G_3^1, G_1^2, G_2^2, G_3^2, G_{2eq}$ has a maximum cut bounded by $m(1/2 + \epsilon)$; this is shown in Theorem 3. (4) The above refutation algorithm almost surely accepts a random formula ϕ (Corollary 2). (5) A satisfiable formula will be rejected by the algorithm (Corollary 1).

Theorem 1. Let ϕ' be a 3CNF formula composed of pairs of matched clauses. Denote by G_{2eq} be the graph induced by ϕ' as described by the refutation algorithm. ϕ' is not satisfiable if all the following conditions hold:

1. ϕ' has the $(1 - \gamma)$ 3XOR property.
2. G_{2eq} has no $(1/2 + \epsilon)$ -cut.
3. $\epsilon + 2\gamma < 1/2$.

Proof. We shall show that if ϕ' is satisfiable and has the $(1 - \gamma)$ 3XOR property, then G_{2eq} has a $(1 - 2\gamma)$ -cut. Combined with the fact that G_{2eq} has no $1/2 + \epsilon$ cut we derive a contradiction (since $\epsilon + 2\gamma < 1/2$). Consider the cut induced on the vertices of G_{2eq} 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. Almost every pair of matched clauses induces an edge which crosses the cut: Let $(x, w, l), (y, w, l)$ be a pair such that both (x, w, l) and (y, w, l) are satisfied as 3XOR. It holds that: $A(x) + A(y) + 2(A(w) + A(l)) = 0 \pmod{2}$. 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} are cut edges. \square

Corollary 1 (Soundness). *Let ϕ be a 3CNF formula. If the refutation algorithm accepts ϕ , then ϕ is not satisfiable.*

Proof. Let ϕ be a formula accepted by the refutation algorithm. The algorithm verified that the extracted subformula ϕ' has the $(1 - 5\epsilon)$ XOR property and that G_{2eq} has no $(1/2 + \epsilon)$ -cut. Since $\epsilon < 1/22$ it follows that $\epsilon + 2 \cdot 5\epsilon < 1/2$, thus by theorem 1 ϕ' is not satisfiable. \square

Lemma 1 (The 3XOR lemma). *Let ϕ be a formula with m clauses and n variables. Let G_1, G_2, G_3 be the following projection graphs: G_i has $2n$ vertices associated with the literals of ϕ . The edges of G_i are derived from ϕ by removing the i -th coordinate from each clause of ϕ . If the following hold:*

1. ϕ is δ -balanced.
2. The maximum cut in each G_i is bounded by $m(1/2 + \epsilon)$.

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

Proof. The proof of this lemma appears at [4]; also a similar version of this lemma (for denser random $2k$ CNF formulas) appears at [3]. We repeat the proof for the sake of self-containment.

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 $\frac{3m + \sum_{i=1}^n Im_i}{2}$ (where Im_i denotes the imbalance of variable i). It follows that on average each clause is satisfied at most $3/2 + \frac{\sum_{i=1}^n Im_i}{2m} = \frac{3}{2}(1 + \delta)$.

We next show that the fraction of clauses satisfied as 3AND is at least $\frac{1}{4} - \frac{3}{2}\epsilon$. Equivalently it is enough to show that the fraction of clauses satisfied as 3NAE denoted by β is bounded by $\frac{3}{4} + \frac{3}{2}\epsilon$ (since A is a satisfying assignment). Consider the graph $G_{1+2+3} = G_1 + G_2 + G_3$ induced by taking the union of the edges of G_1, G_2, G_3 . We remind the reader that each clause of ϕ contributes a "triangle" of 3 edges to G_{1+2+3} (e.g. the clause $(x \vee \bar{y} \vee z)$ 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 the edges of G_{1+2+3} are exactly the union of the edges of G_1, G_2, G_3 (each of them has m edges), and each

G_i has no $(1/2 + \varepsilon)$ -cut. Hence also G_{1+2+3} has no $(1/2 + \varepsilon)$ -cut (at least $1/3$ of the edges of the maximum cut in G_{1+2+3} belong to G_i for some $i \in \{1, 2, 3\}$). It follows that $2\beta m \leq (1/2 + \varepsilon)3m$, giving $\beta \leq \frac{3}{4} + \frac{3}{2}\varepsilon$ 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 $\alpha_3 \geq \frac{1}{4} - \frac{3}{2}\varepsilon$ and that each clause is satisfied at most $\frac{3}{2} + \delta$ times on average, thus: $\frac{3}{2}(1 + \delta) \geq 3 \cdot \alpha_3 + 2 \cdot \alpha_2 + 1 \cdot \alpha_1$. Substituting $\alpha_1 = (1 - \alpha_3 - \alpha_2)$ and α_3 with $\frac{1}{4} - \frac{3}{2}\varepsilon$ yields that $\alpha_2 \leq \frac{3}{2}(\delta + 2\varepsilon)$. \square

In the following theorems $0 < \varepsilon_1 < \varepsilon_2 < 1/22$. ε_1 can be made arbitrarily small by increasing the density parameter c .

Theorem 2. *Let ϕ be a random CNF formula chosen from $F_{n,c/n^{3/2}}$ (with large enough c). Let ϕ', ϕ_1, ϕ_2 be subformulas derived from ϕ as described by the refutation algorithm. Then with high probability over the choice of ϕ , the subformula ϕ_i is ε_1 -balanced, and none of the respective graphs G_1^i, G_2^i, G_3^i and G_{2eq} has a $(1/2 + \varepsilon_1)$ -cut.*

It is well known that a random graph has no $(1/2 + \varepsilon_1)$ -cut, and that a random formula is ε_1 -balanced. The distributions of G_j^i, ϕ_i are "close enough" to the standard models of random graphs/formulas respectively, so that the proof techniques used for the random cases can be applied also in our case. Proof details are omitted due to lack of space.

Theorem 3. *There is a polynomial time algorithm that checks whether a 3CNF formula is ε -balanced. There is a polynomial time algorithm that for every graph that does not have a $(1/2 + \varepsilon_1)$ -cut certifies that the size of the maximum cut is at most $(1/2 + \varepsilon_2)$.*

Proof. Checking that a formula ϕ with m clauses and n variables is ε -balanced 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 [13]. Given a graph with m edges whose maximum cut is bounded by $m(1/2 + \varepsilon_1)$ this algorithm produces a proof that the input graph has no cut of cardinality $m(1/2 + \varepsilon_2)$. This algorithm has the property that $\varepsilon_2 \rightarrow 0$ as $\varepsilon_1 \rightarrow 0$. \square

Corollary 2 (Completeness). *The refutation algorithm almost surely accepts a random formula ϕ taken from $F_{n,c/n^{3/2}}$ for big enough c (c is big enough so that $\varepsilon_2 < \frac{1}{22}$).*

Proof. We set the parameter ε from the refutation algorithm to be equal to ε_2 (by taking the constant c large enough we can make $\varepsilon_1, \varepsilon_2$ arbitrarily small). Let ϕ', ϕ_1, ϕ_2 be the subformulas derived from ϕ as described by the refutation algorithm. By theorem 2 almost surely ϕ_i is ε_1 -balanced, and none of the graphs $G_1^i, G_2^i, G_3^i, G_{2eq}$ induced by ϕ has a $(1/2 + \varepsilon_1)$ -cut. The refutation algorithm which uses the algorithms from theorem 3 will succeed in verifying that each of the graphs $G_1^i, G_2^i, G_3^i, G_{2eq}$ has no $(1/2 + \varepsilon)$ -cut and that ϕ_i is ε -balanced, since $\varepsilon_1 < \varepsilon_2 = \varepsilon$. \square

Lemma 2. *Let ϕ' be the formula derived from ϕ by the refutation algorithm (where ϕ is taken from $F_{n,\frac{c}{n^{3/2}}}$). Almost surely the number of matched pairs in ϕ' is $8c^2n \pm 10\sqrt{c^2n}$.*

Proof (sketch). The probability that $(?, w, l)$ appears at least twice in ϕ is $\frac{2c^2}{n}(1 - O(\frac{c}{\sqrt{n}}))$, thus the expected number of matched clauses is roughly $8c^2n$. Using large deviation laws (e.g. the Chernoff's bound) we derive that the number of matched pairs is concentrated around its expectation. \square

4 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}, l, 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 $A(x) + A(w) + A(l) + A(y) + A(\bar{w}) + A(l) = 0 \pmod 2$, thus $A(x) + A(y) = 1 \pmod 2$. 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} 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 4 to refute ϕ' . Before stating Theorem 4 we need the following definition.

Definition 6. Let ϕ be a 2XOR formula with m clauses and n variables. A_ϕ is the following symmetric matrix associated with ϕ . Initially A_ϕ is the zero matrix. For each clause of the forms (x, \bar{y}) or (\bar{y}, x) we add $+1$ to positions $A(x, y), A(y, x)$. For each clause of the forms (x, y) or (\bar{x}, \bar{y}) we add -1 to positions $A(x, y), A(y, x)$.

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

Theorem 4. Let ϕ' be a 3CNF formula with m pairs of matched clauses and n variables. Let ϕ_{2xor} be the 2XOR formula induced by replacing each 3CNF clause of ϕ' by 3 2XOR clauses (one for every two literals). Let ϕ_{2eq} be the 2EQ formula induced by adding pairs of matched clauses mod 2. If the following hold then ϕ' is not satisfiable:

- (1) ϕ' is δ -balanced.
- (2) The largest eigenvalues of matrices $A_{\phi_{3xor}}, A_{\phi_{2eq}}$ are bounded by $\lambda_{3xor}, \lambda_{2eq}$.
- (3) $3\delta + \frac{n}{4m}(\lambda_{3xor} + \lambda_{2eq}) < 1/2$.

Lemma 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 $(1/2 + \epsilon)$ satisfiable, for $\epsilon = n\lambda/4m$.*

Proof (sketch). Take the most satisfying assignment T and use the Rayleigh quotient of its corresponding $\{\pm 1\}$ vector (in index i we put 1 iff $T(i) = true$) to lower bound λ . \square

Lemma 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 $(1/2 + \gamma)$ satisfiable. Then ϕ has the $(1 - 3/2(\delta + 2\gamma))$ 3XOR property.*

Proof. The proof is very similar to the proof of Lemma 1, details omitted.

Proof (of Theorem 4). Assume that ϕ' is satisfiable as 3CNF and show that properties (1),(2) contradict property (3). By Lemma 3 ϕ_{3xor} and ϕ_{2eq} are at most $(1/2 + e_{3xor})$, $(1/2 + \epsilon_{2eq})$ satisfied respectively ($e_{3xor} = n\lambda_{3xor}/(4 \cdot 6m)$, $e_{2eq} = n\lambda_{2eq}/4m$). Using Lemma 4 we conclude that ϕ' is $(1 - 3/2(\delta + 2e_{3xor}))$ 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 $(1/2 + \epsilon_{2eq})$ satisfied, we conclude that $1 - 3(\delta + 2e_{3xor}) \leq 1/2 + e_{2eq}$ implying $3(\delta + 2e_{3xor}) + e_{2eq} \geq 1/2$. Substituting e_{3xor} , e_{2eq} with the bounds derived from Lemma 3 we conclude that $3\delta + \frac{n}{4m}(\lambda_{3xor} + \lambda_{2eq}) \geq 1/2$ which contradicts property (3). \square

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. 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 G_{3xor} . 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 $n\sqrt{6m/n}/6m = \sqrt{\frac{n}{6m}}$.

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

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

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 is computing the largest eigenvalues of the two matrices A_{3xor}, A_{2eq} . This part took 63 minutes on an Intel Xeon CPU 1700MHz with 256K cache and 2Gbyte memory (Linux).

It may be interesting to see if other refutation algorithms (such as the ones based on resolution or on OBDDs) 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. E. Ben-Sasson and A. Wigderson. Short proofs are narrow-resolution made simple. *Journal of the ACM (JACM)*, 48(2):149–169, 2001.
2. V. Chvatal and E. Szemerédi. Many hard examples for resolution. *Journal of the ACM*, 35(4):759–768, Oct 1988.
3. A. Coja-Oghlan, A. Goerdt, A. Lanka, and F. Schadlich. Certifying unsatisfiability of random 2k-sat formulas using approximation techniques. In *Proc. of the 14th International Symposium on Fundamentals of Computation Theory*, 2003.
4. U. Feige. Relations between average case complexity and approximation complexity. In *Proc. of the 34th Annual ACM Symposium on Theory of Computing*, pages 534–543, 2002.
5. U. Feige and E. Ofek. Spectral techniques applied to sparse random graphs. Technical report, Weizmann Institute of Science, 2003.
6. E. Friedgut and J. Bourgain. Sharp thresholds of graph properties, and the k-sat problem. *JAMS: Journal of the American Mathematical Society*, 12(4):1017–1054, 1999.
7. J. Friedman, A. Goerdt, and M. Krivelevich. Recognizing more unsatisfiable random 3-sat instances efficiently. Technical report, 2003.
8. A. Goerdt and M. Krivelevich. Efficient recognition of random unsatisfiable k-SAT instances by spectral methods. In *STACS: Annual Symposium on Theoretical Aspects of Computer Science*, pages 294–304, 2001.
9. A. Goerdt and A. Lanka. Recognizing more random 3-sat instances efficiently. Manuscript, 2003.
10. M. Hajiaghayi and G.B. Sorkin. The satisfiability threshold for random 3-SAT is at least 3.52. <http://arxiv.org/abs/math.CO/0310193>, 2003.
11. S. Janson, Y. C. Stamatiou, and M. Vamvakari. Bounding the unsatisfiability threshold of random 3-sat. *Random Structures and Algorithms*, 17(2):103–116, 2000.
12. A.C. Kaporis, L.M. Kirousis, and E.G. Lalas. Selecting complementary pairs of literals. In *Proc. LICS'03 Workshop on Typical Case Complexity and Phase Transitions*, June 2003.
13. U. Zwick. Outward rotations: a tool for rounding solutions of semidefinite programming relaxations, with applications to MAX CUT and other problems. In *Proc. of the 31st Annual ACM Symposium on Theory of Computing*, pages 679–687, 1999.