The Complexity of Unique k-SAT: An Isolation Lemma for k-CNFs

Chris Calabro
Russell Impagliazzo [*]
Valentine Kabanets [†]
Ramamohan Paturi[‡]
University of California, San Diego

May 19, 2003

Abstract

We provide some evidence that Unique k-SAT is as hard to solve as general k-SAT, where k-SAT denotes the satisfiability problem for k-CNFs and Unique k-SAT is the promise version where the given formula has 0 or 1 solutions. Namely, defining for each $k \geqslant 1$, $s_k = \inf\{\delta \geqslant 0 \mid \exists$ a $O(2^{\delta n})$-time randomized algorithm for k-SAT$\}$ and, similarly, $\sigma_k = \inf\{\delta \geqslant 0 \mid \exists$ a $O(2^{\delta n})$-time randomized algorithm for Unique k-SAT$\}$, we show that $\lim_{k\to\infty} s_k = \lim_{k\to\infty} \sigma_k$. As a corollary, we prove that, if Unique 3-SAT can be solved in time $2^{\epsilon n}$ for every $\epsilon > 0$, then so can k-SAT for all $k \geqslant 3$.

Our main technical result is an isolation lemma for k-CNFs, which shows that a given satisfiable k-CNF can be efficiently probabilistically reduced to a uniquely satisfiable k-CNF, with non-trivial, albeit *exponentially small*, success probability.

1 Introduction

While NP-complete search problems are presumably intractable to solve, this does not mean that all instances of these problems are difficult. Many heuristics for NP-complete problems, e.g., using back-tracking or local search techniques, take advantage of some structure in the instance. This leads to the general question: what makes an instance of a search problem easier than the worst-case? Conversely, we can ask, what types of structure cannot be exploited by heuristics? Here, we can consider both the structure of the instance (e.g. density of constraints to variables, or expansion of a formula when viewed as a hypergraph) and the structure of the solution space (e.g. how correlated are different solutions?).

One particularly natural measure of the solution space is the number of solutions. One can have different intuitions concerning how the number of solutions should affect the difficulty of the instance. On the one hand, one might suspect that the hardest instances would be those with few solutions, since they become “needles in a large haystack”. On the other hand, algorithms that use local search or narrow in on the solution through deductions might do worse when there are many unrelated solutions. These algorithms might get caught between two equally attractive solutions, like Buridan’s ass (who starves to death, unable to decide between two sources of food).

Valiant and Vazirani [VV86] show that in some formal sense, uniqueness of the solution cannot be used to solve search problems quickly. They give a randomized polynomial-time reduction from Formula SAT to instances of Formula SAT with unique solutions. This shows that a probabilistic polynomial-time algorithm for Unique Formula SAT would also give a similar algorithm for general SAT. The same follows for most other NP-complete problems, such as k-SAT, by combining this reduction with a reduction from Unique SAT to the problem (which usually preserves uniqueness).

However, there is a large gap between not being solvable in polynomial-time and being the “hardest case” of a problem. For example, it is consistent with the Valiant-Vazirani result that Unique k-SAT be solvable in $O(2^{\sqrt{n}})$ time, but general k-SAT requires $2^{\Omega(n)}$ time. This is because, when combined with the reduction from Formula SAT to k-SAT, the Valiant-Vazirani reduction squares the number of variables in the formula.

In this paper, we narrow this gap considerably. We give a randomized reduction from general k-SAT to Unique k-SAT that takes expected time $O(2^{\epsilon_k n})$ where $\epsilon_k \to 0$ as

[*] Research supported by NSF Award CCR-0098197 and USA-Israel BSF Grant 97-00188

[†] Research supported by a Postdoctoral Fellowship from the Natural Sciences and Engineering Research Council of Canada

[‡] research supported by NSF Award CCR-0098197

$k \to \infty$. This implies that if general k-SAT requires time $2^{\delta n}$ for some k and some $\delta > 0$, then, for any $\delta' < \delta$, Unique k'-SAT requires time $2^{\delta' n}$ for sufficiently large k'. Combining with results from [IPZ98], we show that if Unique 3-SAT does not require exponential time, then no search problem in SNP requires exponential time.

It should be mentioned that we do not consider here the problem of *deciding* whether a k-CNF formula has a unique solution, as in [GB97], where it is shown that the decision version of k-SAT is solvable in time $O(2^{\delta n})$ iff the decision version of Unique k-SAT is solvable in time $O(2^{\delta n})$.

One algorithm that seems to take advantage of uniqueness is the Unique k-SAT algorithm of [PPZ99]. They give an $O(2^{(1-1/k)n})$ time algorithm for Unique k-SAT, which they later extend to an algorithm for general k-SAT; the latter algorithm is (by their analysis) slightly less efficient. (Later, [PPSZ98] improved both the Unique and general algorithms.) We show that the same algorithm solves an instance of k-SAT with s solutions in expected time $\mathrm{poly}(n)(2^n/s)^{1-1/k}$, counter to the original intuition that the algorithm takes advantage of uniqueness.

1.1 Known Algorithms for k-SAT

The CNF satisfiability problem was one of the first problems proved NP-complete [Coo71]. This problem is widely believed to require a deterministic algorithm of superpolynomial, even exponential, time complexity. This belief is supported by the fact that there is no known randomized algorithm producing a certificate for the satisfiability of n-variable CNFs with probability $\geqslant \frac{1}{2}$ and that runs in significantly less time than 2^n. The best known algorithm for general CNF satisfiability runs in time $O(2^{n(1-\frac{1}{\log m})})$, where m is the number of clauses [Schuler03].

The situation is slightly better for k-SAT, where each clause of a given CNF contains at most k literals, for some constant $k \geqslant 3$. This syntactic restriction can be exploited to obtain a $O(2^{\epsilon_k n})$-time algorithm for k-SAT, for some constant $0 < \epsilon_k < 1$ dependent on k. The first such algorithms were obtained in the 1980's [Dan81, MS85]. Recently, a number of faster algorithms have been proposed [PPZ99, PPSZ98, Sch02, HSSW02, DGH$^+$]. The best known randomized algorithm for 3-SAT is given in [HSSW02]. For $k \geqslant 4$, the best known randomized algorithm is due to [PPSZ98]; its time complexity on k-CNFs with n variables is $O(2^{(1-\mu_k/(k-1))n})$, where $\mu_k > 1$ is an increasing function of k that approaches $\pi^2/6 \approx 1.644$.

The techniques of [PPSZ98] and its predecessor [PPZ99] crucially rely upon the notion of *isolation* of a satisfying assignment. The degree of isolation of a satisfying assignment $x \in \{0,1\}^n$ is the number of coordinates $1 \leqslant i \leqslant n$ such that flipping the value of x_i turns x into a falsifying assignment. Thus, a unique satisfying assignment has degree n of isolation, whereas any satisfying assignment of a tautology has degree 0 of isolation.

The argument in [PPZ99] proceeds as follows. If a given k-CNF has "many" satisfying assignments, then we are likely to find one by random sampling. On the other hand, if there are "few" satisfying assignments, then there must be a satisfying assignment of "high" degree of isolation. However, sufficiently isolated satisfying assignments have "short" descriptions (via the Satisfiability Coding Lemma), and so these assignments can be found by enumerating all possible "short" descriptions, and checking if any such description corresponds to a satisfying assignment. Combining the two cases yields a randomized algorithm of running time $\mathrm{poly}(n)2^{(1-1/k)n}$.

1.2 Unique k-SAT is a Hard Case of k-SAT

A closer look at the analysis in [PPZ99] reveals that the more satisfying assignments a given k-CNF has, the faster we can find one of them. More precisely, as we prove later in the paper (see Section 4), if a k-CNF on n variables has s satisfying assignments, then one of them can be found with probability $\geqslant \frac{1}{2}$ in time $\mathrm{poly}(n)(2^n/s)^{1-1/k}$. This result suggests that the hardest case of k-SAT may be *Unique k-SAT*, the set of all k-CNFs with exactly one satisfying assignment. In the rest of our paper, we provide further evidence towards the truth of this hypothesis.

Apart from the natural motivation to pinpoint the worst case for k-SAT, we are guided by the desire to narrow down any further research of the algorithmic complexity of k-SAT. We would like to argue that the true challenge in designing an improved k-SAT algorithm is to solve *Unique* k-SAT faster than the corresponding algorithm of [PPSZ98].

1.3 Why the Valiant-Vazirani Reduction is Insufficient

Before describing our results, we would like to discuss how the result of Valiant and Vazirani [VV86] on Unique Formula SAT is related to our problem of Unique k-SAT; here, Unique Formula SAT is the set of all Boolean formulas (not necessarily in conjunctive normal form) with exactly one satisfying assignment. [VV86] describes a polynomial-time randomized reduction that takes a Boolean formula ϕ and outputs a new Boolean formula ψ such that with probability at least an inverse polynomial, if ϕ is satisfiable, then ψ has a unique satisfying assignment, and otherwise, ψ has no satisfying assignment. This implies that Formula SAT randomly reduces to Unique

Formula SAT.

Unfortunately, the proof in [VV86] does *not* imply that Unique k-SAT is the worst case of k-SAT, or even that Unique CNF-SAT is the worst case of CNF-SAT. The random hashing technique of [VV86] consists of intersecting the set of satisfying assignments of a given n-variable formula ϕ by random hyperplanes over the space $\mathbb{F}_2^n$, where $\mathbb{F}_2$ is the field of two elements. On average, such a hyperplane depends on $n/2 \gg k$ of the variables. Thus, almost certainly, the resulting formula ψ will not be a k-CNF, even if the original formula ϕ is. Applying the standard algorithm for converting an arbitrary Boolean formula into a k-CNF will likely yield a k-CNF with $\Omega(n^2)$ new variables. Thus, we shall convert a satisfiable k-CNF on n variables into a uniquely satisfiable k-CNF on $\Omega(n^2)$ variables.

If there were an algorithm for Unique k-SAT on n variables with running time $O(2^{n^{o(1)}})$, then the Valiant-Vazirani reduction could be used to create an algorithm for k-SAT with running time $O(2^{n^{o(1)}})$ as well. This reduction is useless, however, if the best algorithm for Unique k-SAT on n variables runs in time $\Omega(2^{\epsilon_k n})$, for some $0 < \epsilon_k < 1$. Since the common working hypothesis is that Unique k-SAT requires time $2^{\Omega(n)}$, we need to use a different argument to show that k-SAT is no harder than Unique k-SAT.

1.4 Our Results

For each $k \geqslant 1$, let $s_k = \inf\{\delta \geqslant 0 \mid \exists$ a $O(2^{\delta n})$-time randomized algorithm for k-SAT$\}$ and, similarly, let $\sigma_k = \inf\{\delta \geqslant 0 \mid \exists$ a $O(2^{\delta n})$-time randomized algorithm for Unique k-SAT$\}$. Denoting $s_\infty = \lim_{k\to\infty} s_k$ and $\sigma_\infty = \lim_{k\to\infty} \sigma_k$, we can state our main result as follows.

Theorem 1. $s_\infty = \sigma_\infty$.

This gives strong evidence that Unique k-SAT is as hard to solve as general k-SAT. Combining Theorem 1 with the results in [IPZ98] yields the following.

Corollary 2. *If Unique* 3*-SAT*$\in \cap_{\epsilon>0}\mathrm{RTIME}(2^{\epsilon n})$*, then for every $k \geqslant 3$, k-SAT*$\in \cap_{\epsilon>0}\mathrm{RTIME}(2^{\epsilon n})$.

The main technical ingredient in the proof of Theorem 1 is a more refined analysis of the random hashing technique of [VV86]. We modify the construction of [VV86] so that it uses *sparse* hyperplanes that depend on at most k' variables, where k' depends only on k and an approximation parameter ϵ. We then show that, given a satisfiable k-CNF ϕ, the modified construction is likely to produce a new k'-CNF ψ such that all satisfying assignments of ψ are clustered in a single Hamming ball of "small" radius. Finally, we argue that a random assignment of "few" randomly chosen variables of ψ will result in a uniquely satisfiable k'-CNF ψ'. This result is formally stated as Lemma 3 (Isolation Lemma) below.

It is worth pointing out that, unlike the reduction of [VV86], our randomized reduction has only *exponentially small* success probability. However, since we are interested in $O(2^{\epsilon n})$-time algorithms, such small success probabilities suffice for our purposes.

Remainder of this paper In Section 2, we state and prove our main technical result, an Isolation Lemma for k-CNFs. Section 3 contains the proofs of Theorem 1 and Corollary 2 stated in the Introduction. In Section 4, we show that Unique k-SAT is the worst case for the k-SAT algorithm of [PPZ99].

2 Isolation Lemma for k-CNFs

Recall that $h(\epsilon) = -\epsilon\log\epsilon - (1-\epsilon)\log(1-\epsilon)$ is the binary entropy function.

Lemma 3 (Isolation Lemma). *For every $k \in \mathbb{N}, \epsilon \in (0, \frac{1}{4})$, there is a randomized polynomial-time algorithm $I_{k,\epsilon}$ that, given an n-variable k-CNF ϕ, outputs an n-variable k'-CNF ϕ', for $k' = \max\{\lceil \frac{1}{\epsilon}\ln\frac{2}{\epsilon}\rceil, k\}$, such that*

1. *if ϕ is unsatisfiable, then so is ϕ', and*
2. *if ϕ is satisfiable, then, with probability at least $(32n2^{3h(2\epsilon)n})^{-1}$, ϕ' has a unique satisfying assignment.*

Proof. Our proof will consist of two steps. First, we intersect the set S of satisfying assignments of ϕ with randomly chosen hyperplanes on k' variables, and show that, with reasonable probability, all the surviving satisfying assignments of ϕ are contained (concentrated) in a Hamming ball of "small" radius. Then we isolate a single satisfying assignment in this Hamming ball by randomly assigning a small number of coordinates.

STEP I: CONCENTRATION. With probability at least $1/n$, we can guess an integer $s \in [\log|S|, \log|S|+1]$. For the remainder of our argument, let us assume that our guess is correct. For m to be determined later, we intersect the solution space S of ϕ with m sparse hyperplanes of the Boolean cube $\{0,1\}^n$, where each hyperplane h_i is chosen independently by the following randomized algorithm: Pick a subset $R_i \subseteq \{1,\ldots,n\}$ of size k' uniformly at random. For each $j \in R_i$, pick $c_{i,j} \in \{0,1\}$ uniformly at random. Pick $b_i \in \{0,1\}$ uniformly at random. Output the hyperplane $h_i \overset{\mathrm{def}}{=} \sum_{j\in R_i} c_{i,j}x_j = b_i$, where the summation is over the field $\mathbb{F}_2$.

Let us write the system of linear equations $h_1, \dots, h_m$ in the matrix form $Ax = b$, where A is an $m \times n$ 0-1 matrix, $x = (x_1, \dots, x_n)^t$, and $b = (b_1, \dots, b_m)^t$.

While a random sparse hyperplane is unlikely to separate two vectors $x, y \in \{0,1\}^n$ that are close to each other, it has a reasonable chance of separating x and y that are Hamming distance $\mathrm{dist}(x,y) \geqslant r \overset{\mathrm{def}}{=} \lceil \epsilon n \rceil$ apart.

Claim 4. *For any $x, y \in \{0,1\}^n$ with $\mathrm{dist}(x,y) \geqslant r$, we have*

$$\mathbf{Pr}_A[Ax = Ay] \leqslant \frac{1}{2^m} e^{\epsilon m}$$

Proof of Claim 4. For any $x, y \in \{0,1\}^n$ with $\mathrm{dist}(x,y) \geqslant r$ and any $i \in \{1, \dots, m\}$, we have

$$\begin{aligned}\mathbf{Pr}[(Ax)_i \neq (Ay)_i] &= \mathbf{Pr}[(A(x-y))_i \neq 0] \\ &= \mathbf{Pr}[(A(x-y))_i \neq 0 \mid (x-y)_{R_i} \neq 0] \\ &\quad \cdot \mathbf{Pr}[(x-y)_{R_i} \neq 0]\end{aligned}$$

where $(x-y)_{R_i}$ is the vector restricted to the coordinates from R_i. If $(x-y)_{R_i} \neq 0$, then $\mathbf{Pr}[(A(x-y))_i \neq 0] = \frac{1}{2}$. We get

$$\begin{aligned}\mathbf{Pr}[(Ax)_i \neq (Ay)_i] &= \frac{1}{2}\mathbf{Pr}[(x-y)_{R_i} \neq 0] \\ &= \frac{1}{2}\mathbf{Pr}[\exists j \in R_i \; (x-y)_j \neq 0] \\ &\geqslant \frac{1}{2}\Big(1 - \frac{\binom{n-r}{k'}}{\binom{n}{k'}}\Big) \\ &\geqslant \frac{1-(1-\epsilon)^{k'}}{2} \\ &\geqslant \frac{1-(1-\epsilon)^{(1/\epsilon)\ln(2/\epsilon)}}{2} \\ &\geqslant \frac{1-e^{-\epsilon(1/\epsilon)\ln(2/\epsilon)}}{2} \\ &\geqslant \frac{1}{2} - \frac{\epsilon}{2}.\end{aligned}$$

Thus, $\mathbf{Pr}_A[Ax = Ay] \leqslant (\frac{1}{2} + \frac{\epsilon}{2})^m \leqslant \frac{1}{2^m}e^{\epsilon m}$, as required. □

Now, let us define the set of *semi-isolated* solutions as

$$si = \{x \in S \mid \forall y \in S \; \big(\mathrm{dist}(x,y) \geqslant r \to Ax \neq Ay\big)\}.$$

For each $x \in S$, we have

$$\begin{aligned}\mathbf{Pr}_A[x \in si] = \; & 1 - \mathbf{Pr}_A[\exists y \in S \\ & \qquad \big(\mathrm{dist}(x,y) \geqslant r \wedge Ax = Ay\big)] \\ \geqslant \; & 1 - \sum_{y \in S, \mathrm{dist}(x,y) \geqslant r} \mathbf{Pr}_A[Ax = Ay] \\ \geqslant \; & 1 - |S|e^{\epsilon m}/2^m\end{aligned}$$

where the last inequality is by Claim 4. By setting $m = s + \lceil 4\epsilon s \rceil + 2$, we obtain

$$\mathbf{Pr}_A[x \in si] \geqslant 1/2. \qquad (1)$$

It follows that

$$\begin{aligned}&\sum_{x \in S} \mathbf{Pr}_{A,b}[x \in si \wedge Ax = b] \\ &= \sum_{x \in S} \mathbf{Pr}_A[x \in si]\mathbf{Pr}_b[Ax = b \mid x \in si] \qquad (2) \\ &\geqslant \frac{1}{2}|S|2^{-m},\end{aligned}$$

where the last inequality is by (1) and the fact that the vector b is chosen independently of the matrix A.

On the other hand, for any given A and b, the number of semi-isolated solutions $x \in S$ such that $Ax = b$ is at most $2^{h(\epsilon)n}$, since every pair of such solutions must be Hamming distance less than r apart. This implies

$$\begin{aligned}&\sum_{x \in S} \mathbf{Pr}_{A,b}[x \in si \wedge Ax = b] \\ &\qquad \leqslant 2^{h(\epsilon)n}\mathbf{Pr}_{A,b}[\exists x \in S \; (x \in si \wedge Ax = b)]. \quad (3)\end{aligned}$$

Combining inequalities (2) and (3) yields

$$\begin{aligned}\mathbf{Pr}_{A,b}[\exists x \in si \; Ax = b] &\geqslant 2^{s-m-h(\epsilon)n-2} \\ &\geqslant 2^{-(4\epsilon+h(\epsilon))n-5},\end{aligned} \qquad (4)$$

concluding Step I of the lemma.

STEP II: ISOLATION. Now suppose that $\phi \wedge Ax = b$ is satisfiable and its solutions are in a Hamming ball of radius less than r, and hence, the diameter $d \leqslant 2\lfloor \epsilon n \rfloor$.

For assignments u and v to disjoint sets of variables, let uv denote the union of these assignments. Let uv and $u\bar{v}$ be two distinct solutions to $\phi \wedge Ax = b$ that are furthest apart. Let C be the set of variables on which uv and $u\bar{v}$ differ. Note that $|C| \leqslant d$. Fix $D \supseteq C$ such that $|D| = d$. Consider the assignment β_D to the variables in D consistent with the assignment uv: $x_i = v_i$ for $i \in C$ and $x_i = u_i$ for $i \in D - C$. The reduction chooses a random set D' of d variables and a random assignment $\alpha_{D'}$ to these variables. Define ϕ' to be the formula $\phi \wedge Ax = b \wedge x_{D'} = \alpha_{D'}$. If $D' = D$ and $\alpha_{D'} = \beta_D$, then uv is the *unique* solution to ϕ', since any other solution wv to ϕ' would also be a solution to $\phi \wedge Ax = b$, but $\mathrm{dist}(wv, u\bar{v}) > \mathrm{dist}(uv, u\bar{v})$.

The probability of correctly guessing a set of variables D containing C and correctly assigning them is at least

$$\frac{2^{-d}}{\binom{n}{d}} \geqslant 2^{-(2\epsilon + h(2\epsilon))n}. \qquad (5)$$

Combining equations (4) and (5), we conclude that the probability of correctly guessing s, A, b, D', and $\alpha_{D'}$ so that the resulting formula $\phi' \overset{def}{=} \phi \wedge Ax = b \wedge x_{D'} = \alpha_{D'}$ is uniquely satisfiable is at least $\frac{1}{32n} 2^{-3h(2\epsilon)n}$, as claimed. □

3 Applications of the Isolation Lemma

3.1 Proof of Theorem 1

Lemma 5. $s_k \leqslant \sigma_k + O(\frac{\ln^2 k}{k})$.

Proof. Let $\epsilon = \frac{2 \ln k}{k}$. Then $k \geqslant \frac{1}{\epsilon} \ln \frac{2}{\epsilon}$. For every $\gamma > 0$ and every k, we have a randomized algorithm $U_{k,\gamma}$ for Unique k-SAT that runs in time $O(2^{(\sigma_k + \gamma)n})$. Applying the Isolation Lemma 3, there is a polynomial time algorithm $I_{k,\epsilon}$ for reducing a given satisfiable k-CNF ϕ to a uniquely satisfiable k-CNF ϕ' with success probability $p \geqslant 2^{-O(h(\epsilon)n)} \geqslant 2^{-O(\frac{\ln^2 k}{k})n}$. By running the algorithm $I_{k,\epsilon}$ $O(p^{-1})$ times and applying algorithm $U_{k,\gamma}$ to every output k-CNF ϕ', we obtain a randomized algorithm for k-SAT that has running time $O(2^{(\sigma_k + \gamma + O(\frac{\ln^2 k}{k}))n})$. As $\gamma \to 0$, we have the lemma. □

Theorem 1 then follows by taking the limit of the inequality in the previous lemma as k tends to infinity.

3.2 Proof of Corollary 2

We shall need the following result.

Lemma 6 (Sparsification Lemma [IPZ98]). *For all k and $\epsilon > 0$, there is a $\mathrm{poly}(n)2^{\epsilon n}$-time algorithm $S_{k,\epsilon}$ that, given an n-variable k-CNF ϕ, returns an n-variable k-CNF ψ that has at most $c(k,\epsilon)n$ clauses, for some function c of k, ϵ, and such that*

1. *every solution to ψ is also a solution to ϕ, and*
2. *if ϕ is satisfiable, then so is ψ, with probability at least $\geqslant 2^{-\epsilon n}$.*

We also need a simple clause-width-reducing algorithm R_3 which takes as input a k-CNF ϕ in n variables and m clauses and returns a 3-CNF ψ in $n + (k-3)m$ variables whose solution set has a trivial bijective correspondence to that of ϕ. We describe R_3. For each clause $l_1 \vee \cdots \vee l_k$, where the l_i are literals, introduce $k - 3$ new variables $y_3, \ldots, y_{k-1}$. The idea is to express $l_1 \vee \cdots \vee l_k$ as

$$(l_1 \vee l_2 \vee y_3) \wedge (y_3 \vee \bar{l}_3) \wedge \cdots \wedge (y_3 \vee \bar{l}_k) \\ \wedge (\bar{y}_3 \vee l_3 \vee \cdots \vee l_k),$$

and then recurse on the last clause. The above is equivalent to

$$(l_1 \vee l_2 \vee y_3) \wedge (y_3 \leftrightarrow (l_3 \vee \cdots \vee l_k)),$$

and so we can see that for each solution x of ϕ, there is exactly one solution y to ψ and that we can compute x from y simply by throwing away the new variables.

Proof of Corollary 2. We combine sparsification, isolation, and clause-width reduction. Suppose that for each $\delta > 0$, there is a randomized algorithm $U_{3,\delta}$ that solves Unique 3-SAT in time $O(2^{\delta n})$.

For any $\epsilon \in (0, \frac{1}{4})$, the algorithm $T \overset{\mathrm{def}}{=} R_3 \circ I_{k,\epsilon} \circ S_{k,\epsilon}$, composed of the three algorithms R_3, $I_{k,\epsilon}$, and $S_{k,\epsilon}$, returns a 3-CNF with at most $c(k,\epsilon)n$ variables for some function c of k, ϵ since the construction in the Isolation Lemma adds at most $2^{k+1}n = O(n)$ clauses to the sparsified formula. Composing $U_{3,\delta}$ with T, we obtain a $O(2^{(\epsilon+\delta)n})$-time algorithm for k-SAT that has success probability $p \geqslant 2^{-O(h(\epsilon)n)}$. Running this algorithm $O(p^{-1})$ times independently, we get a randomized algorithm for k-SAT of running time $O(2^{\delta' n})$, for arbitrarily small $\delta' > 0$. □

4 The More Satisfying Assignments, the Better

Here we present a randomized algorithm for k-SAT whose expected running time on an n-variable $\mathrm{poly}(n)$-size k-CNF with s satisfying assignments is $\mathrm{poly}(n)(2^n/s)^{1-1/k}$. The algorithm is the same as that in [PPZ99]; our analysis of this algorithm is somewhat more refined.

Below, we say that a variable x is *forced* in a CNF ϕ if ϕ contains a unit clause which consists of x or $\bar{x}$ only. Clearly, the forced variable will have the same value in every satisfying assignment to ϕ.

Consider the algorithm **A** given in Figure 1. Clearly, algorithm **A** is polynomial-time. We shall lower-bound the probability that this algorithm succeeds at finding a satisfying assignment.

```
INPUT: $k$-CNF $\phi$ on variables $x_1, \ldots, x_n$

  1. Pick a permutation $\sigma$ of the set $\{1, \ldots, n\}$ uniformly at random.

  2. For $i = 1$ to $n$, if the variable $x_{\sigma(i)}$ is forced, then set $x_{\sigma(i)}$ so that
     the corresponding unit clause is satisfied; otherwise, set $x_{\sigma(i)}$ to 1 (TRUE)
     or 0 (FALSE) uniformly at random.

  3. If the constructed assignment $a$ satisfies $\phi$, then output $a$.
```

Algorithm 1: ALGORITHM **A**

Lemma 7. *Let ϕ be a k-CNF on n variables that has s satisfying assignments. Then* $\mathbf{Pr}[\mathbf{A}(\phi)$ *outputs some satisfying assignment*$] \geqslant (s2^{-n})^{1-1/k}$.

Proof. Let S be the set of satisfying assignments of ϕ, $a \in S$, and let σ be any permutation. For this fixed σ, consider the run of $\mathbf{A}$ that results in outputting the assignment a. We denote by $d_\sigma(a)$ the number of variables that were *not* forced during this run of $\mathbf{A}$. Since each non-forced variable is chosen uniformly at random, we have $\mathbf{Pr}[\mathbf{A}(\phi) = a \mid \sigma] = 2^{-d_\sigma(a)}$, where the probability is over the random choices of the algorithm $\mathbf{A}$ for non-forced variables, given the fixed permutation σ.

It is easy to see that

$$\begin{aligned}&\mathbf{Pr}[\mathbf{A}(\phi) \text{ outputs some satisfying assignment}]\\ &= \sum_{a\in S} \mathbf{Exp}_\sigma[2^{-d_\sigma(a)}] \geqslant \sum_{a\in S} 2^{-\mathbf{Exp}_\sigma[d_\sigma(a)]},\end{aligned}$$

where the last step uses Jensen's inequality.

We can bound $\mathbf{Exp}_\sigma[d_\sigma(a)]$ using the argument from [PPZ99]. Let $l(a)$ denote the number of satisfying assignments a' of ϕ such that the Hamming distance $\mathrm{dist}(a, a') = 1$. Then there are $n - l(a)$ variables such that each of them occurs as a unique true literal in some clause; such a clause is called *critical* in [PPZ99]. It follows that each such variable $x_{\sigma(i)}$ will be forced for a random permutation σ if $x_{\sigma(i)}$ occurs last in the corresponding critical clause. The latter happens with probability at least $1/k$ since each clause of ϕ is of size at most k. By the linearity of expectation, we obtain that the expected number of *forced* variables is at least $(n - l(a))/k$. Hence, $\mathbf{Exp}_\sigma[d_\sigma(a)] \leqslant n - (n - l(a))/k$.

Thus,

$$\begin{aligned}&\mathbf{Pr}[\mathbf{A}(\phi) \text{ outputs some satisfying assignment}]\\ &\geqslant \sum_{a\in S} 2^{-n(1-1/k)} 2^{-l(a)/k}.\end{aligned}$$

After factoring out $2^{-n(1-1/k)}$, we need to lower-bound the sum $\sum_{a\in S} 2^{-l(a)/k}$. Let $L = \mathbf{Exp}_{a\in S}[l(a)]$. By Jensen's inequality, we obtain

$$\sum_{a\in S} 2^{-l(a)/k} \geqslant s2^{-L/k}, \tag{6}$$

where $s = |S|$.

Finally, to bound L, we use the well-known edge isoperimetric inequality due to [Har67]. This inequality implies that, for any subset S' of the n-dimensional Boolean cube $\{0,1\}^n$,

$$|\{(a, a') \mid a, a' \in S' \text{ and } \mathrm{dist}(a, a') = 1\}| \leqslant |S'| \log |S'|.$$

Hence, for $S' = S$, we obtain $\sum_{a\in S} l(a) \leqslant s \log s$, and so, $L = \mathbf{Exp}_{a\in S}[l(a)] \leqslant \log s$.

Continuing inequality (6), we get

$$\sum_{a\in S} 2^{-l(a)/k} \geqslant s2^{-(\log s)/k} = s^{1-1/k}.$$

So, $\mathbf{Pr}[\mathbf{A}(\phi)$ outputs some satisfying assignment$] \geqslant (s2^{-n})^{1-1/k}$, as required. □

Lemma 7 immediately implies the following.

Theorem 8. *There is a randomized algorithm for k-SAT that, given a satisfiable k-CNF ϕ on n variables with s satisfying assignments, will output one of the satisfying assignments of ϕ in expected time* $\mathrm{poly}(n)(2^n/s)^{1-1/k}$.

5 Open Problems

While we are able to show that $s_\infty = \sigma_\infty$, we conjecture that $\forall k, s_k = \sigma_k$. However, our techniques do not lend themselves to proving this conjecture. To prove the conjecture, it would be sufficient to prove an Isolation Lemma for k-CNF with an inverse polynomial probability of success. While it takes a family of size $Omega(2^s)$ to isolate an element of an arbitrary set of size 2^s by intersecting with a randomly chosen set from the family, it would be interesting if one can construct polynomial size families to isolate elements from sets of satisfying solutions of k-CNF.

In fact, we could hope to prove an even stronger version of the conjecture to the effect that satisfiability for formulas with many satisfying assignments is strictly easier than for the formulas with fewer solutions. To prove such a result, one might show that adding additional random constraints to a formula with many satisfying assignments preserves satisfiability while implicitly reducing the number of variables. For example, in our proof of Isolation Lemma, we added a number of random sparse linear equations, which implicitly define some variables in terms of the others. Can we show that these constraints can be used to simplify search? A less ambitious formulation would just prove the above for Davis-Putnam algorithms and their variants.

References

[Coo71] S.A. Cook. The complexity of theorem-proving procedures. In *Proceedings of the Third Annual ACM Symposium on Theory of Computing*, pages 151–158, 1971.

[Dan81] E. Dantsin. Two propositional proof systems based on the splitting method. *Zapiski Nauchnykh Seminarov LOMI*, 105:24–44, 1981. (in Russian). English translation in *Journal of Soviet Mathematics*, 22(3):1293–1305, 1983.

[DGH$^+$] E. Dantsin, A. Goerdt, E.A. Hirsch, R. Kannan, J. Kleinberg, C. Papadimitriou, P. Raghavan, and U. Schöning. A deterministic $(2 - 2/(k+1))^n$ algorithm for k-SAT based on local search. *Theoretical Computer Science*. to appear.

[GB97] E. Grandjean, H. Buning. SAT-problems and reductions with respect to the number of variables. *Journal of Logic and Computation*, 7(4):457–471, 1997.

[Har67] L.H. Harper. A necessary condition on minimal cube numberings. *Journal of Applied Probability*, 4:397–401, 1967.

[HSSW02] T. Hofmeister, U. Schöning, R. Schuler, and O. Watanabe. A probabilistic 3-SAT algorithm further improved. In *Proceedings of the Nineteenth Annual Symposium on Theoretical Aspects of Computer Science*, pages 192–202, 2002.

[IP01] R. Impagliazzo, R. Paturi. On the complexity of k-SAT. *Journal of Computer and Systems Sciences*, 62(2):367–375, 2001.

[IPZ98] R. Impagliazzo, R. Paturi, and F. Zane. Which problems have strongly exponential complexity? In *Proceedings of the Thirty-Ninth Annual IEEE Symposium on Foundations of Computer Science*, pages 653–662, 1998.

[MS85] B. Monien and E. Speckenmeyer. Solving satisfiability in less than 2^n steps. *Discrete Applied Mathematics*, 10:287–295, 1985.

[PPSZ98] R. Paturi, P. Pudlák, M.E. Saks, and F. Zane. An improved exponential-time algorithm for k-SAT. In *Proceedings of the Thirty-Ninth Annual IEEE Symposium on Foundations of Computer Science*, pages 628–637, 1998.

[PPZ99] R. Paturi, P. Pudlák, and F. Zane. Satisfiability coding lemma. *Chicago Journal of Theoretical Computer Science*, 1999. (preliminary version in FOCS'97).

[Sch02] U. Schöning. A probabilistic algorithm for k-SAT based on limited local search and restart. *Algorithmica*, 32:615–623, 2002.

[Schuler03] R. Schuler. An algorithm for the satisfiability problem of formulas in conjunctive normal form. Technical Report, Universit at Ulm, 2003.

[VV86] L. Valiant and V. Vazirani. NP is as easy as detecting unique solutions. *Theoretical Computer Science*, 47:85–93, 1986.