Exponential Lower Bounds for DPLL Algorithms on Satisfiable Random 3-CNF Formulas

Dimitris Achlioptas[1,2,3] and Ricardo Menchaca-Mendez[3]

[1] University of Athens, Greece
[2] CTI, Greece
[3] University of California, Santa Cruz, USA

Abstract. We consider the performance of a number of DPLL algorithms on random 3-CNF formulas with n variables and $m = rn$ clauses. A long series of papers analyzing so-called "myopic" DPLL algorithms has provided a sequence of lower bounds for their satisfiability threshold. Indeed, for each myopic algorithm $\mathcal{A}$ it is known that there exists an algorithm-specific clause-density, $r_{\mathcal{A}}$, such that if $r < r_{\mathcal{A}}$, the algorithm finds a satisfying assignment in linear time. For example, $r_{\mathcal{A}}$ equals $8/3 = 2.66..$ for ORDERRED-DLL and 3.003... for GENERALIZED UNIT CLAUSE. We prove that for densities well within the provably satisfiable regime, every backtracking extension of either of these algorithms takes *exponential* time. Specifically, all extensions of ORDERRED-DLL take exponential time for $r > 2.78$ and the same is true for GENERALIZED UNIT CLAUSE for all $r > 3.1$. Our results imply exponential lower bounds for many other myopic algorithms for densities similarly close to the corresponding $r_{\mathcal{A}}$.

1 Introduction

The problem of determining the satisfiability of Boolean formulas is central to computational complexity. Moreover, it is of tremendous practical interest as it arises naturally in numerous settings. Random CNF formulas have emerged as a mathematically tractable vehicle for studying the performance of satisfiability algorithms and proof systems. For a given set of n Boolean variables, let B_k denote the set of all possible disjunctions of k non-complementary literals on the variables (k-clauses). A random k-SAT formula $F_k(n, m)$ is formed by selecting uniformly and independently m clauses from B_k and taking their conjunction.

We will be interested in random formulas from an asymptotic point of view, i.e., as the number of variables grows. In particular, we will say that a sequence of random events $\mathcal{E}_n$ occurs *with high probability (w.h.p.)* if $\lim_{n\to\infty} \Pr[\mathcal{E}_n] = 1$. In this context, the ratio of constraints-to-variables, $r = m/n$, known as density, plays a fundamental role as most interesting monotone properties are believed to exhibit 0-1 laws with respect to density. Perhaps the best known example is the satisfiability property.

Conjecture 1. For each $k \geq 3$, there exists a constant r_k such that for any $\epsilon > 0$,

$$\lim_{n\to\infty} \Pr[F_k(n, (r_k - \epsilon)n)] = 1, \quad \text{and} \quad \lim_{n\to\infty} \Pr[F_k(n, (r_k + \epsilon)n)] = 0 \ .$$

The satisfiability threshold conjecture above has attracted a lot of attention in computer science, mathematics and statistical physics. At this point, neither the value, nor even the existence of r_k has been established. In a breakthrough result, Friedgut [**?**] gave a very general condition for a monotone property to have a *non-uniform* sharp threshold. In particular, his result yields the statement of the conjecture if one replaces r_k with a function $r_k(n)$. For $k = 3$, the best known bounds are $3.52 < r_3 < 4.49$, due to results in [**?**] and [**?**], respectively.

A key feature of random k-CNF formulas is that their underlying hypergraph is locally tree-like for every finite density, i.e., for both satisfiable and unsatisfiable formulas. One implication of this fact is that the formula induced by any finite-depth neighborhood of any variable is highly under-constrained. As a result, unsatisfiability comes about due to long-range interactions between variables something that appears hard to capture by efficient algorithms. In particular, random formulas have been shown to be hard both for proof systems, e.g., in the seminal work of Chvátal and Szemerédi on resolution [**?**], and, more recently, for some of the most sophisticated satisfiability algorithms known [**?**]. More generally, for the connections of random formulas to proof-complexity and computational-hardness see the surveys by Beame and Pitassi [**?**] and Cook and Mitchell [**?**], respectively.

The last decade has seen a great deal of rigorous results on random CNF formulas, including a proliferation of upper and lower bounds for the satisfiability threshold. Equally importantly, random CNF formulas have been the domain of an extensive exchange of ideas between computer science and statistical physics, including the discovery of the clustering phenomenon [**?**,**?**], establishing it rigorously [**?**], and relating it to algorithmic performance [**?**]. In this work we take another step in this direction by taking a technique from mathematical physics, the *interpolation method* [**?**,**?**,**?**,**?**], and using it to derive rigorous upper bounds for the satisfiability threshold of random CNF formulas that are mixtures or 2- and 3-clauses. As we discuss below, such formulas arise naturally as residual formulas in the analysis of satisfiability algorithms and their unsatisfiability implies exponential lower bounds for the running time of a large class of algorithms. Our main result is the following.

Theorem 1. *Let F be a random CNF formula on n variables with $(1-\epsilon)n$ random 2-clauses, and $(1+\epsilon)n$ random 3-clauses. W.h.p. F is unsatisfiable for $\epsilon = 10^{-4}$.*

Our method for proving Theorem 1 involves estimating an infinite sum with no close form, any truncation of which yields a rigorous bound. The choice of 10^{-4} is rather arbitrary as our methods can deliver arbitrarily small $\epsilon > 0$, given enough computational resources. We have chosen $\epsilon = 10^{-4}$ as it can be checked readily with very modest computation.

2 Background and Motivation

Many algorithms for finding satisfying assignments for CNF formulas operate by building a partial assignment step by step. These algorithms commit to the assignments made at each step and operate on a *residual formula*, in which clauses already satisfied have been removed, while the remaining clauses have been shortened by the removal of their

falsified literals. We call such algorithms *forward search* algorithms. During the execution of any such algorithm a partial assignment may produce clauses of size 1 (unit clauses) in the residual formula which in turn create additional *forced* choices in the partial assignment, since the variables appearing in unit clauses have only one possible assignment if the formula is to be satisfied. The choices made by a forward search algorithm when no unit clauses are present are called *free*.

A large class of natural DPLL algorithms are "myopic" in that their free-step choices are based on local considerations in terms of the underlying hypergraph. Perhaps the simplest such algorithm is ORDERRED-DLL which performs unit-clause propagation but, otherwise, sets variables in some a priori fixed random order/sign. Another example is GENERALIZED UNIT CLAUSE (GUC) [**?**,**?**], where in each step a random literal in a random shortest clause is assigned true. The key property of myopic algorithms that makes their analysis mathematically tractable is the following (indeed, this can be seen as a definition of myopic algorithms): as long as the algorithm has never backtracked, the residual formula is uniformly random conditional on its number of 2- and 3-clauses (unit-clauses are satisfied as soon as they occur).

To analyze the performance of myopic algorithms on random formulas one employs the standard technique of approximating the mean path of Markov chains by differential equations in order to keep track of the 2- and 3-clause density of the residual formula. As is well understood, in the large n limit, both of these densities behave as deterministic functions, for every myopic algorithm. In the absence of backtracking, i.e., if the algorithm continues blithely on after a 0-clause is generated, this means that for any given initial 3-clause density r, we can model the algorithm's behavior as a continuous 2-dimensional curve $(d_2^r(x), d_3^r(x))$ of the 2- and 3-clause density, where $x \in [0,1]$ denotes the fraction of assigned variables. Since the 2-SAT satisfiability threshold [**?**,**?**] is $r_2 = 1$, it follows that for any initial 3-clause density $r > 0$ and every $\gamma > 0$ such that $d_2^r(x) < 1$ for all $x \in [0, \gamma)$, the probability that no 0-clause is ever generated is bounded away from 0. Indeed, to determine the threshold $r_{\mathcal{A}}$ for each myopic algorithm it suffices to determine the largest r such that $d_2^r(x) < 1$ for all $x \in [0,1)$. This is because as long as $d_2^r(x) < 1$, w.h.p. 0-clauses are generated for trivial local reasons. In particular, as was shown by Frieze and Suen [**?**], there exists a very simple form of backtracking which never flips the value of any variable more than once, such that endowing any myopic algorithm with this backtracking boosts its probability of finding a satisfying assignment to $1 - o(1)$ for all $r < r_{\mathcal{A}}$.

To understand what happens for $r > r_{\mathcal{A}}$, let us consider what happens if one gives as input to a myopic algorithm $\mathcal{A}$ a random 3-CNF formula of density $r > r_{\mathcal{A}}$, but only runs the algorithm for $x_0 \cdot n$ steps where x_0 is such that $d_2^r(x) < 1$ for all $x \in [0, x_0)$. Up to that point, the algorithm will have either not backtracked at all, or backtracked for trivial local reasons, so that the residual formula will be a mixture of random 2- and 3-clauses in which the 2-clauses alone are satisfiable. Naturally, if the residual formula is satisfiable the algorithm still has a chance of finding a satisfying assignment in polynomial time. But what happens if this mixture, as a whole, is unsatisfiable? How fast will it discover this and backtrack? In [**?**] it was shown that the resolution complexity of unsatisfiable random mixtures of 2- and 3-clauses in which the 2-clause are satisfiable is exponential. Since every DPLL algorithm produces a resolution proof

of unsatisfiability, it follows that if the residual mixture is unsatisfiable, the algorithm will take exponential time to establish its unsatisfiability.

To delineate satisfiable from unsatisfiable mixtures, define Δ_c to be the largest Δ such that for every $\epsilon > 0$, a mixture of $(1-\epsilon)n$ 2-clauses and Δn 3-clauses is w.h.p. satisfiable. In [**?**] it was proven that $2/3 \leq \Delta_c < 2.28...$ The upper bound, combined with the differential equations analysis mentioned above was used in [**?**] to prove that if ORDERED-DLL is started with $3.81n$ random 3-clauses it will reach a stage where the residual formula has exponential resolution complexity (and, therefore, take exponential time on such formulas). Similarly, for GUC started with $3.98n$ random 3-clauses.

By establishing $\Delta_c < 1.001$, the exact same analysis as in [**?**] allows us to prove that each of these algorithms fails for much lower densities, well within the proven satisfiable regime. Specifically, while ORDERED-DLL succeeds in finding a satisfying assignment in linear time up to $8/3 = 2.66...$ we prove that it already requires exponential time at $r > 2.71$. Similarly, while GUC succeeds in linear time for $r < 3.003$, we prove that it requires exponential time at $r > 3.1$. We state both of this results more precisely in the next section, after discussing the different types of backtracking that one can consider.

We note that these two explicit results for ORDERED-DLL and GUC are simply indicative and Theorem 1 can be applied to prove similar bounds for all myopic algorithms. This includes all algorithms in [**?**] and many others. In fact, our Theorem 1 can be generalized to random mixtures of 2- and 3-clauses with a given degree sequence, thus also covering algorithms such as the one in [**?**].

2.1 Backtracking

When a path in the search tree leads to a contradiction, the algorithm must begin backtracking by undoing all the (forced) choices up to the last free choice and flipping the assignment to that variable. From there, perhaps the simplest option would be for the algorithm to act as if it had reached this point without backtracking and apply the original heuristic to decide which variable(s) to set next.

As long as the 2-clause density stays below 1 it is not hard to show that any such backtracking w.h.p. is due to trivial "local" reasons and can be fixed by changing the value of $O(\log n)$ variables (typically $O(1)$ variables suffice). From a technical point of view, though, such backtracking (minimally) disturbs the uniform randomness property of the residual formula, enough to make the statement of crisp mathematical statements cumbersome.

An alternative heuristic, due to Frieze and Suen [**?**], which we call FS-backtracking is the following: when a contradiction is reached, record the portion of the assignment between the last free choice and the contradiction; these literals become *hot*. After flipping the value of the last free choice, instead of making the choice that the original heuristic would suggest, give priority to the complements of the hot literals in the order that they appeared; once the hot literals are exhausted continue as with the original heuristic. FS-backtracking is quite natural in that this last part of the partial assignment got us into trouble in the first place.

A key property of FS-backtracking that is useful in analysis is that as long as the value of each variable in a partial assignment has been flipped at most once, the resid-

ual formula is perfectly uniformly random conditional on the number of clauses of each size. We emphasize that while the original motivation for introducing FS-backtracking is technical, such backtracking is, in fact, a genuinely good algorithmic idea. Specifically, on random 3-CNF formulas with densities between 3.8 and 4.0, large experiments show that the histogram of run-times of FS-backtracking is significantly better than simple backtracking. We will denote a forward search algorithm $\mathcal{A}$ extended with FS-backtracking by $\mathcal{A}$-FS.

Let us say that a DPLL algorithm is at a *t-stage* if precisely t variables have been set.

Definition 1. *Let $\epsilon = 10^{-4}$. A t-stage of a DPLL algorithm is* bad *if the residual formula at that stage is the union of a random 2-CNF formula with $(1-\epsilon)(n-t)$ 2-clauses and a random 3-CNF formula with $(1+\epsilon)(n-t)$ clauses, where $t \leq n/2$.*

Definition 1 is identical to that of bad stages in [**?**], except that they have $2.28 + \epsilon$ instead of our $1 + \epsilon$, since $\Delta_c \leq 2.28$ was the best known bound prior to our work. Proceeding exactly as in [**?**], i.e., by using the differential equations method to determine the smallest initial 3-clause density such that the algorithm eventually reaches a bad stage, we get the following.

Lemma 1. *Let $\Delta_{\text{ORDERED-DLL}} = 2.71$ and let $\Delta_{\text{GUC}} = 3.1$.*

1. *For each $\mathcal{A} \in \{$ORDERED-DLL,GUC$\}$, an execution of* any *backtracking extension of $\mathcal{A}$ on a random 3-CNF formula with $\Delta_{\mathcal{A}} \cdot n$ clauses reaches a* bad *t-stage with constant probability.*
2. *For each $\mathcal{A} \in \{$ORDERED-DLL,GUC$\}$, an execution of algorithm $\mathcal{A}$-FS on a random 3-CNF formula with $\Delta_{\mathcal{A}} \cdot n$ clauses reaches a* bad *t-stage w.h.p.*

Theorem 2. *Let $\Delta_{\text{UC}} = \Delta_{\text{ORDERED-DLL}} = 2.71$ and let $\Delta_{\text{GUC}} = 3.1$.*

1. *For each $A \in \{$ORDERED-DLL,GUC$\}$, an execution of* any *backtracking extension of A on a random 3-CNF formula with $\Delta_A n$ clauses takes time $2^{\Omega(n)}$ with constant probability.*
2. *For each $A \in \{$ORDERED-DLL,GUC$\}$, an execution of algorithm A-FS on a random 3-CNF formula with $\Delta_A n$ clauses takes time $2^{\Omega(n)}$ w.h.p.*

2.2 Proving Upper Bounds for Satisfiability Thresholds

The simplest upper bound on the satisfiability threshold of random k-CNF formulas comes from taking the union bound over all assignments $\sigma \in \{0,1\}^n$ of the probability that each one is satisfying. That is,

$$\Pr[F_k(n, rn) \text{ is satisfiable}] \leq \sum_{\sigma} \Pr[\sigma \text{ satisfies } F_k(n, rn)] = \left[2(1-2^{-k})^r\right]^n \to 0 \ ,$$

for all $r > r_k^*$, where $2(1-2^{-k})^{r_k^*} = 1$. It is easy to see that $r_k^*/(2^k \ln 2) \to 1$.

Note that the above argument holds even for $k = n$, in which case satisfiability reduces to the coupon collector's problem over $\{0,1\}^n$. By standard results, in this

case the number of clauses to cover the cube is very close to $2^n \ln(2^n) = n2^n \ln 2$. Aggregating these assignments, for the sake of comparison, into groups of size 2^{n-k} so that they are comparable to k-clauses, recovers the union bound above. In other words, the simplicity (and the weakness) of the union bound is that it treats the 2^{n-k} *distinct* assignments forbidden by each k-clause as a random multiset of the same size. As one can imagine, this phenomenon is stronger as the cubes are larger, i.e., for smaller values of k. For example, $r_3^* = 5.19..$, but a long series of increasingly sophisticated results has culminated with the bound $r_3 < 4.49$ by Díaz et al. [**?**]. In the extreme case $k = 1$, the birthday paradox readily implies that a collection of $\Theta(n^{1/2})$ random 1-clauses is *w.h.p.* unsatisfiable, yet the union bound only gives $r_1 \leq 1$, i.e., requires $\Omega(n)$ 1-clauses.

For the special case $k = 2$, it has long been shown, independently, by Chvátal and Reed [**?**] and Goerdt [**?**] that $r_2 = 1$. In all these proofs, the fact $r_2 \leq 1$ is established by exploiting the existence of succinct certificates of unsatisfiability for 2-SAT enabling proofs that proceed by identifying "most likely" unsatisfiable subformulas in the evolution of $F_2(n, rn)$. Intriguingly, early non-rigorous arguments of statistical physics [**?**] recover the fact $r_2 \leq 1$ *without* relying on the fact that 2-SAT is in P. It is precisely this feature that we exploit in the present work.

2.3 Applying the Interpolation Method to Random CNFs formulas

To prove Theorem 1 we abandon standard combinatorial proofs of unsatisfiability and turn to a remarkable tool developed by Francesco Guerra [**?**], called the *interpolation method*, to deal with the Sherrignton Kirkpatrick model (SK) of statistical physics. Following Guerra's breakthrough, Franz and Leone [**?**], in a very important paper, applied the interpolation method to random k-SAT and random XOR-SAT to prove that certain expressions derived via the non-rigorous replica method of statistical physics for these problems, correspond to rigorous lower bounds for the free energy of each problem. As such, these expressions can, in principle, be used to derive upper bounds for the satisfiability threshold of each problem, yet this involves the solution of certain functional equations that appear beyond analytical penetration. In [**?**], Panchenko and Talagrand showed that the results of [**?**] can be derived in a simpler and uniform way, unifying the treatment of different levels of Parisi's Replica Symmetry Breaking.

In a recent paper [**?**], Bayati, Gamarnik and Tetali, showed that a combinatorial analogue of the interpolation method can be used to elegantly derive an approximate subadditivity property for a number of CSPs on Erdős-Renyi and regular random graphs. This allowed them to prove the existence of a number of limits in these problems, including the existence of a limit for the size of the largest independent set in a random regular graph. At the same time, the simplicity of their combinatorial approach comes at the cost of losing the capacity to yield quantitative bounds for the associated limiting quantities.

To overcome these problems we will apply a recently developed [**?**] *energetic* interpolation method to random mixtures of 2- and 3-clauses. This, implicitly, exploits the second order nature of random 2-SAT phase transitions to gain in computational tractability.

3 The Energetic Interpolation Method

In the more standard models of random k-SAT, the number of clauses m is fixed (not a random variable). The interpolation method requires that m is a *Poisson random variable* with mean $\mathbb{E}[m] = rn$. Since the standard deviation of the Poisson distribution is the square root of its mean we have $m = (1 + o(1))rn$ *w.h.p.*, thus not affecting any asymptotic results regarding densities.

We shall work with the random variable $H_{n,r}(\sigma)$, know as the Hamiltonian, counting the number of unsatisfied clauses in the instance for each $\sigma \in \{0,1\}^n$. We will sometimes refer to $H_{n,r}(\sigma)$ as the energy function. The goal of the method is to compute lower bounds on the following quantity

$$\xi_r = n^{-1}\mathbb{E}\left[\min_{\sigma\in\{0,1\}^n} H_{n,r}(\sigma)\right] \ . \tag{1}$$

Note that proving $\liminf_{n\to\infty} \xi_r > 0$ implies that the satisfiability threshold is upper bounded by r, since the random variable $n^{-1}\min_\sigma H_{n,r}(\sigma)$ is known to be concentrated in a $o(n)$ window [**?**].

Given $\sigma = (x_1, x_2, \ldots, x_n)$ we will write $H_{n,r}(\sigma)$ as the sum of m functions $E_a(x_{a_1}, ..., x_{a_k})$, one for each clause a. That is, $E_a(x_{a_1}, ..., x_{a_k}) = 1$ if the associated clause is not satisfied and 0 otherwise. The basic object of the energetic interpolation method is a modified energy function that interpolates between $H_{n,r}(\sigma)$ and the energy function of a dramatically simpler (and fully tractable) model. Specifically, for $t \in [0, 1]$, let

$$H_{n,r,t}(x_1, \ldots, x_n) = \sum_{m=1}^{m_t} E_{a_m}(x_{a_{m,1}}, ..., x_{a_{m,k}}) + \sum_{i=1}^{n}\sum_{j=1}^{k_{i,t}} \hat{h}_{i,j}(x_i) \ , \tag{2}$$

where m_t is a Poisson random variable with mean $\mathbb{E}[m_t] = trn$, the $k_{i,t}$'s are i.i.d. Poisson random variables with mean $\mathbb{E}[k_{i,t}] = (1-t)kr$, and the functions $\hat{h}_{i,j}(\cdot)$ are i.i.d. random functions distributed as the function of (4) below. Before delving into the meaning of the random functions $\hat{h}_{i,j}(\cdot)$, which are the heart of the method, let us first make a few observations about (2). To begin with, note that for $t = 1$, equation (2) is simply the energy function of the original model, i.e., a sum of m functions counting whether each clause has been violated or not. On the other hand, for $t < 1$, we see that, in expectation, $(1-t)m$ of these clause-functions have been replaced by k times as many $\hat{h}$-functions, each of which takes as input the value of a single variable in the assignment. A good way to think about this replacement is as a decombinatorialization of the energy function wherein (combinatorial) k-ary functions are replaced by univariate functions. As one can imagine, for $t = 0$ the model is fully tractable. In particular, if

$$\xi_r(t) = \frac{1}{n}\mathbb{E}\left[\min_{\sigma\in\{0,1\}^n} H_{n,r,t}(\sigma))\right] \ , \tag{3}$$

one can readily compute $\xi_r(0)$.

The main idea of the interpolation method is to select the univariate functions $\hat{h}_i(\cdot)$ independently, from a probability distribution that reflects aspects of the geometry of the underlying solution space. A particularly appealing aspect of the energetic interpolation method is that it projects all information about the geometry of the solution space into a single probability p, which can be interpreted as the probability that a variable picked at random will be frozen, i.e., have the same value in all optimal assignments. The method then delivers a valid bound for *any* choice of $p \in [0,1]$ and the bound is then optimized by choosing the best value of p, i.e., performing a single-parameter search.

Let "1", "0", and " $*$ " denote the binary functions $\{h(0)=1, h(1)=0\}$, $\{h(0)=0, h(1)=1\}$, and $\{h(0)=0, h(1)=0\}$ respectively. One can think of function 1 as being 1 (unhappy) when the input is not 1, of function 0 as being 1 when the input is not 0, and of function $*$ as never being 1. Let $\mathtt{h}(x)$ be a random function in {"0", "1", " $*$ "} with $\Pr(\mathtt{h}(\cdot) = \text{"1"}) = \Pr(\mathtt{h}(\cdot) = \text{"0"}) = p/2$ and let the random function $\hat{h}(x)$ be defined as follows

$$\hat{h}(x) = \min_{y_1,\ldots,y_{k-1}} \{E(y_1,..,y_{k-1},x) + \sum_{i=1}^{k-1} \mathtt{h}_i(y_i)\} \ , \tag{4}$$

where $E(\cdot)$ is a random clause-function and the functions $\mathtt{h}_i(\cdot)$ are i.i.d. random functions distributed as $\mathtt{h}(x)$ i.e. $\hat{h}(\cdot) = \text{"1"}$ with probability $2^{-k}p^{k-1}$.

The main point of the interpolation method is that as t goes from 1 to 0, we can control in the change of $\xi_r(t)$, hence the name. Specifically, one has the following.

Theorem 3 ([?]).

$$\xi_r \geq \xi_r(0) - r(k-1)2^{-k}p^k \ . \tag{5}$$

Determining $\xi_r(0)$ is a tractable task and establishing $\xi_r > 0$ implies that $F_k(n, rn)$ is w.h.p. unsatisfiable.

For completeness, we present the proof of Theorem 3 in Appendix A.

4 Energy Density Bounds for $(2+p)$-SAT

Let $F_{2,3}(n, \epsilon, \Delta)$ denote a random CNF formula over n variables consisting of m_2 random 2-CNF clauses, where m_2 is a Poisson random variable with mean $\mathbb{E}[m_2] = (1-\epsilon)n$ and m_3 random 3-CNF clauses, where m_3 is a Poisson random variable with mean $\mathbb{E}[m_3] = \Delta n$. Thus, the energy function is now

$$H_{n,\epsilon,\Delta}(\sigma) = H^{(2)}_{n,1-\epsilon}(\sigma) + H^{(3)}_{n,\Delta}(\sigma) \ ,$$

where $H^{(2)}_{n,1-\epsilon}(\sigma)$ is the k-SAT energy function for $k=2$ and $r = 1-\epsilon$, while $H^{(3)}_{n,\Delta}(\sigma)$ is the energy function for $k=3$ and $r=\Delta$. Similarly the interpolation function is the sum of the two independent interpolation functions corresponding to $k=2$ and $k=3$, i.e.,

$$H_{n,\epsilon,\Delta,t}(x_1,\ldots,x_n) = H^{(2)}_{n,1-\epsilon,t}(x_1,\ldots,x_n) + H^{(3)}_{n,\Delta,t}(x_1,\ldots,x_n) \ . \tag{6}$$

Letting

$$\xi_{\epsilon,\Delta}(t) = n^{-1}\mathbb{E}\left[\min_{\sigma\in\{0,1\}^n} H_{n,\epsilon,\Delta,t}(\sigma)\right] \ , \tag{7}$$

the analogue of Theorem 3 for random mixtures of 2- and 3-clauses is the following.

Theorem 4. *For every value of* $p \in [0,1]$*,*

$$\xi_{\epsilon,\Delta} \geq \xi_{\epsilon,\Delta}(0) - \frac{1}{4}(1-\epsilon)p^2 - \frac{1}{4}\Delta p^3 \ . \tag{8}$$

Proof. As with Theorem 3, the probability joint distribution implicit in the expectation of $\xi_{\epsilon,\Delta}(t)$ can be written as the product of Poisson random functions, due to the independence among the random variables appearing in $H_{n,\epsilon,\Delta,t}(x_1,\ldots,x_n)$. Now, the derivative with respect to t gives rise to two independent set of equations similar to the ones in (14) and (15) for $k=2$ and $k=3$, where the base energy function is $H_{n,\epsilon,\Delta}(\sigma)$. Since all the relevant properties of the mixture are captured by its set of frozen variables, the theorem follows simply by applying the proof of (12) in Theorem 3 twice.

5 Application to $(2+p)$-SAT

In this section we give first an analytical expression for $\xi_{\epsilon,\Delta}(0)$ and then compute a lower bound for it. We have

$$\begin{aligned}
\xi_{\epsilon,\Delta}(0) &= n^{-1}\mathbb{E}\left[\min_{\sigma\in\{0,1\}^n} H_{n,\epsilon,\Delta,0}(\sigma)\right] \\
&= n^{-1}\mathbb{E}\left[\min_{\sigma\in\{0,1\}^n}\left(\sum_{i=1}^{n}\left(\sum_{j=1}^{k_{2,i}}\hat{h}_{2,i,j}(x_i)+\sum_{j=1}^{k_{3,i}}\hat{h}_{3,i,j}(x_i)\right)\right)\right] \\
&= n^{-1}\mathbb{E}\left[\sum_{i=1}^{n}\min_{x_i\in\{0,1\}}\left(\sum_{j=1}^{k_{2,i}}\hat{h}_{2,i,j}(x_i)+\sum_{j=1}^{k_{3,i}}\hat{h}_{3,i,j}(x_i)\right)\right] \\
&= n^{-1}\sum_{i=1}^{n}\mathbb{E}\left[\min_{x_i\in\{0,1\}}\left(\sum_{j=1}^{k_{2,i}}\hat{h}_{2,i,j}(x_i)+\sum_{j=1}^{k_{3,i}}\hat{h}_{3,i,j}(x_i)\right)\right] \ ,
\end{aligned}$$

where the $k_{2,i}$'s and the $k_{3,i}$'s are Poisson random variables with means $2(1-\epsilon)$ and 3Δ respectively, as defined in (6). Note now that the n expectations in the above summation are identical, thus

$$\xi_{\epsilon,\Delta}(0) = \mathbb{E}\left[\min_{x\in\{0,1\}}\left(\sum_{j=1}^{s_2}\hat{h}_{2,j}(x)+\sum_{j=1}^{s_3}\hat{h}_{3,j}(x)\right)\right] \ , \tag{9}$$

where s_2 and s_3 are Poisson random variables with means $2(1-\epsilon)$ and 3Δ respectively, and the functions $\hat{h}_{2,j}(\cdot)$ and $\hat{h}_{3,j}(\cdot)$ are i.i.d. copies of the function $\hat{h}(\cdot)$ in (4) for $k=2$

and $k = 3$, respectively, i.e., random functions in {"0", "1", " $*$ "} with $\Pr(\hat{h}_{k,j}(\cdot) =$ "1") $= \Pr(\hat{h}_{k,j}(\cdot) =$ "0") $= 2^{-k}p^{k-1}$.

Let $l_{k,0}$, $l_{k,1}$, and $l_{k,*}$ denote the number "0", "1", and "*" functions, respectively among the $\hat{h}_{k,j}(\cdot)$ functions inside the summation in (9). Conditional on the value of s_k, the random vector $(l_{k,0}, l_{k,1}, l_{k,*})$ is distributed as a multinomial random vector with s_k trials and probability vector $(2^{-k}p^{k-1}, 2^{-k}p^{k-1}, 1 - 2^{-k+1}p^{k-1})$, therefore,

$$\begin{aligned}\xi_{\epsilon,\Delta}(0) = \sum_{x=0}^{\infty}\sum_{y=0}^{\infty}\sum_{l_{2,0}=0}^{x}\sum_{l_{2,1}=0}^{x-l_{2,0}}\sum_{l_{3,0}=0}^{y}\sum_{l_{3,1}=0}^{y-l_{2,0}} &\min\{l_{2,0}+l_{3,0}, l_{2,1}+l_{3,1}\} \times \\ &\text{Poi}(2(1-\epsilon), x)\text{Multi}(l_{2,0}, l_{2,1}, x-l_{2,0}-l_{2,1}) \times \\ &\text{Poi}(3\Delta, y)\text{Multi}(l_{3,0}, l_{3,1}, y-l_{3,0}-l_{3,1}) \ ,\end{aligned}$$

where Multi$(\cdot,\cdot,\cdot)$ denotes the multinomial density function.

Changing the limits of all summations to infinity, does not change the value of $\xi_{\epsilon,\Delta}(0)$, since Multi$(\cdot,\cdot,\cdot)$ evaluates to zero for negative numbers, hence, we can interchange the order of the summations to get

$$\begin{aligned}\xi_{\epsilon,\Delta}(0) = \sum_{l_{2,0}=0}^{\infty}\sum_{l_{2,1}=0}^{\infty}\sum_{l_{3,0}=0}^{\infty}\sum_{l_{3,1}=0}^{\infty} &\min\{l_{2,0}+l_{3,0}, l_{2,1}+l_{3,1}\} \times \\ \sum_{x=0}^{\infty} &\text{Poi}(2(1-\epsilon), x)\text{Multi}(l_{2,0}, l_{2,1}, x-l_{2,0}-l_{2,1}) \times \\ \sum_{y=0}^{\infty} &\text{Poi}(3\Delta, y)\text{Multi}(l_{3,0}, l_{3,1}, y-l_{3,0}-l_{3,1}) \ .\end{aligned}$$

The last equation can be simplified by summing out the randomness in the Poisson random variables. The result is that $l_{2,0}$ and $l_{2,1}$ become two independent Poisson random variables with mean $\frac{1}{2}(1-\epsilon)p$, that is,

$$\begin{aligned}\sum_{x=0}^{\infty}\text{Poi}(2(1-\epsilon), x)&\text{Multi}(l_{2,0}, l_{2,1}, x-l_{2,0}-l_{2,1}) = \\ &\text{Poi}((1-\epsilon)p/2, l_{2,0}) \times \text{Poi}((1-\epsilon)p/2, l_{2,1}) \ .\end{aligned}$$

Similarly, $l_{3,0}$ and $l_{3,1}$become two independent Poisson random variables with mean $\frac{3}{8}\Delta p^2$. Moreover, letting

$$\lambda = \frac{1}{2}(1-\epsilon)p + \frac{3}{8}\Delta p^2 \ ,$$

we see that $l_0 = l_{2,0} + l_{3,0}$ is itself a Poisson random variable with mean λ, since the sum of two independent Poisson random variables with means λ_1 and λ_2 is a Poisson random variable with mean $\lambda = \lambda_1 + \lambda_2$. Thus,

$$\xi_{\epsilon,\Delta}(0) = \sum_{l_0=0}^{\infty}\sum_{l_1=0}^{\infty}\min\{l_0, l_1\} \times \text{Poi}\,(\lambda, l_0) \times \text{Poi}\,(\lambda, l_1) \ ,$$

i.e., $\xi_{\epsilon,\Delta}(0)$ is the expected value of the minimum of two independent Poisson random variables l_0, l_1 with mean λ. Consequently, the bound of Theorem 4 becomes

$$\xi_{\epsilon,\Delta} \geq \mathbb{E}\left[\min\{l_0, l_1\}\right] - \frac{1}{4}(1-\epsilon)p^2 - \frac{1}{4}\Delta p^3 \quad . \tag{10}$$

Finally, we note that

$$\mathbb{E}\left[\min\{l_0, l_1\}\right] = \sum_{i=0}^{\infty} i \left(2\text{Poi}(\lambda, i) \left(1 - \sum_{j=0}^{i-1} \text{Poi}(\lambda, j) \right) - \left(\text{Poi}(\lambda, i)\right)^2 \right) \quad . \tag{11}$$

Thus, to compute lower bounds for (10) is enough to truncate (11) at any value of i. In particular, by letting $\epsilon = 0.0001$, $\Delta = 1.0001$ and $i = 50$, we get that for $p = 1.2 \cdot 10^{-3}$ the truncated version of (10) is greater than 0, implying that a random CNF formula with $0.9999n$ 2-clauses and $1.0001n$ 3-clauses is w.h.p. unsatisfiable.

A Proof of Theorem 3

Proof. Since $\xi_r(1) = \xi_r(0) + \int_0^1 \xi_r'(t)dt$ and since $r(k-1)2^{-k}p^k$ does not depend on t, it suffices to show that $-r(k-1)2^{-k}p^k$ is an lower bound for $\xi_r'(t)$, i.e., we have to show that for all $t \in [0,1]$,

$$\xi_r'(t) \geq -r(k-1)2^{-k}p^k \quad . \tag{12}$$

We begin by computing $\xi_r'(t)$. Let $\min_\sigma H_m(\sigma)$ and $\min_\sigma H_{k_i}(\sigma)$ denote the random variable $\min_\sigma H_{n,t}(\sigma)$ conditioned on the values of the random variables m_t and $k_{i,t}$ respectively, that is

$$\min_\sigma H_m(\sigma) = \min_\sigma H_{n,r,t}(\sigma)\Big|_{m_t=m} \quad \text{and} \quad \min_\sigma H_{k_i}(\sigma) = \min_\sigma H_{n,r,t}(\sigma)\Big|_{k_{i,t}=k_i}$$

and more generally

$$\min_\sigma H_{m,k_1,\ldots,k_n}(\sigma) = \min_\sigma H_{n,r,t}(\sigma)\Big|_{m_t=m,k_{1,t}=k_1,\ldots,k_{n,t}=k_n} \quad .$$

Denote the Poisson density function with mean μ as $\text{Poi}(\mu, z) = e^{-\mu}(\mu^z/z!)$. Since the random variable m_t and the random variables $k_{i,t}$ are independent, we can write the expectation in (3) as

$$\xi_r(t) = \sum_{m,k_1,\ldots,k_n} \text{Poi}(trn, m) \prod_{i=1}^{n} \text{Poi}((1-t)rk, k_i) \frac{1}{n} \mathbb{E}[\min_\sigma H_{m,k_1,\ldots,k_n}(\sigma)] \quad .$$

By differentiating $\xi_r(t)$ with respect to t we get

$$\begin{aligned} \xi_r'(t) = &\sum_{m=0}^{\infty} \frac{\partial}{\partial t}\text{Poi}(trn, m) \frac{1}{n} \mathbb{E}[\min H_m(\sigma)] + \\ &\sum_{i=1}^{n} \sum_{k_i=0}^{\infty} \frac{\partial}{\partial t}\text{Poi}((1-t)rk, k_i) \frac{1}{n} \mathbb{E}[\min H_{k_i}(\sigma)] \quad . \end{aligned} \tag{13}$$

Recall now that $(\partial/\partial t)\text{Poi}(trn, m) = -rn\text{Poi}(trn, m) + rn\text{Poi}(trn, m-1)$. Thus, the derivative with respect to t in the first summation in (13) can be written as

$$-r\sum_{m=0}^{\infty}\text{Poi}(trn,m)\mathbb{E}[\min H_m(\sigma)] + r\sum_{m=1}^{\infty}\text{Poi}(trn,m-1)\mathbb{E}[\min H_m(\sigma)] = r\sum_{m=0}^{\infty}\text{Poi}(trn,m)\left[\mathbb{E}[\min H_{m+1}] - \mathbb{E}[\min H_m]\right] \quad . \tag{14}$$

Similarly, the derivatives in the double sum in (13) with respect to t can be written as

$$-rk\frac{1}{n}\sum_{i=1}^{n}\sum_{k_i=0}^{\infty}\text{Poi}((1-t)rk,k_i)\left[\mathbb{E}[\min H_{k_i+1}] - \mathbb{E}[\min H_{k_i}]\right] \quad . \tag{15}$$

Now, a crucial observation is that (14) is r times the expected value of the change in $\min H$ after adding a random clause, while (15) is $-rk$ times the expected value of the change in $\min H$ after adding a single $\hat{h}$ function whose argument is a variable selected uniformly at random. Thus, to establish (12) we need to show that the expected change in $\min H$ caused by adding a random clause minus k times the expected change caused by adding a random function $\hat{h}$ is at most $-r(k-1)2^{-k}p^k$. Equivalently, we need to:

1. Consider the experiment:
   - Select: (i) a random formula H from the distribution $H_{n,r,t}$, (ii) a random clause c, (iii) a random variable $x \in \{x_1, \ldots, x_n\}$, and (iv) a random $\hat{h}$-function.
   - Let $H' = H(\sigma) + E_c$, $H'' = H(\sigma) + \hat{h}(x)$.
   - Let $Y = (\min H' - \min H) - k(\min H'' - \min H)$.
2. Prove that $\mathbb{E}Y$, over the choice of $H, c, \hat{h}$, is at most $-r(k-1)2^{-k}p^k$.

The averaging task in Step 2 above appears quite daunting, as we need to average over H. The reason we can establish the desired conclusion is that, in fact, something far stronger holds. Namely, we will prove that for *every* realization of H, the conditional expectation of Y, i.e., the expectation over only c, h and $\hat{h}$, satisfies the desired inequality.

Specifically, Let $H_0(\cdot)$ denote any realization of $H_{n,r,t}(\cdot)$. Let $C^* \subseteq \{0,1\}^n$ be the set of optimal assignments in H_0. A variable x_i is frozen if its value is the same in all optimal assignments. Let O^* be the set of frozen variables corresponding to H_0. We are going to compute the expected value in the change of $\min_\sigma\{H_0(\sigma)\}$ after adding a new factor node $E_{a_{\text{new}}}(x_{a_{\text{new},1}}, \ldots, x_{a_{\text{new},k}})$ and after adding an individual factor $\hat{h}_{\text{new}}(\cdot)$ to a variable selected u.a.r.

Adding a new factor $E_{\text{new}}(x_{1,\text{new}}, \ldots, x_{k,\text{new}})$ will change the minimum value by 1 iff all the variables appearing in E_{new} are frozen, i.e., $\{x_{a_{\text{new},1}}, \ldots, x_{a_{\text{new},k}}\} \subseteq O^*$, and the sign of all the frozen variables in the clause associated with E_{new} is not equal to its frozen value. For, otherwise, any non frozen variable could be adjusted to make the new factor zero. The probability that $\{x_{a_{\text{new},1}}, \ldots, x_{a_{\text{new},k}}\} \subseteq O^*$ is $(|O^*|/n)^k$, since each of the variables in a random clause are selected uniformly at random with replacement. Thus,

$$\mathbb{E}\left[\min_{\sigma}\{H_0(\sigma) + E(x_{a_{\text{new},1}}, \ldots, x_{a_{\text{new},k}})\}\right] - \min_{\sigma}\{H_0(\sigma)\} = 2^{-k}\left(\frac{|O^*|}{n}\right)^k \quad .$$

The change after adding a new individual factor $\hat{h}_{\text{new}}(\cdot)$ to variable selected uniformly at random can be computed in a similar way. In this case the minimum value will change by 1 only if the selected variable x is frozen and if the new factor forces the variable x to take its non-frozen value. This requires that both of the following occur:

- The variable x is frozen to the opposite sign from the one it has in the random clause $E(y_1, ..., y_{k-1}, x)$ in the added factor. This event occurs with probability $|O^*|/(2n)$.
- The $k-1$ random functions, distributed as $\mathtt{h}(\cdot)$, are all "0" or "1" functions and the $(k-1)$-tuple $(y_1^*, ..., y_{k-1}^*)$ that minimizes $\sum \mathtt{h}_i(y_i)$ does not satisfy the random clause in the factor. This event occurs with probability $(p/2)^{k-1}$.

Therefore,

$$\mathbb{E}\left[\min_{\sigma}\{H_0(\sigma) + \hat{h}_{\text{new}}(x)\}\right] - \min_{\sigma}\{H_0(\sigma)\} = 2^{-k}p^{k-1}\frac{|O^*|}{n} \quad .$$

Thus the value $\xi'(t)$ conditional on H_0 is

$$\xi'(t)|H_0 = r2^{-k}\left(\frac{|O^*|}{n}\right)^k - rk2^{-k}p^{k-1}\frac{|O^*|}{n} \quad .$$

We finish the proof by noting that

$$-\xi'(t)|H_0 - r(k-1)2^{-k}p^k = 2^{-k}\left(-r\left(\frac{|O^*|}{n}\right)^k + rkp^{k-1}\frac{|O^*|}{n} - r(k-1)p^k\right)$$

is always non-positive since the polynomial $F(x,p) = x^k - kp^{k-1}x + (k-1)p^k \geq 0$ for all $0 \leq x, p \leq 1$. To see this last statement note that:

- $F(0,p), F(1,p), F(x,0), F(x,1) \geq 0$.
- The derivative of F with respect to p is 0 only when $p = x$, wherein $F(x,x) = 0$.