Solving Random Satisfiable 3CNF Formulas in Expected Polynomial Time *†

Michael Krivelevich ‡ Dan Vilenchik §

Abstract

We present an algorithm for solving 3SAT instances. Several algorithms have been proved to work **whp** (with high probability) for various SAT distributions. However, an algorithm that works **whp** has a drawback. Indeed for typical instances it works well, however for some rare inputs it does not provide a solution at all. Alternatively, one could require that the algorithm always produce a correct answer but perform well on average. *Expected polynomial time* formalizes this notion. We prove that for some natural distribution on 3CNF formulas, called planted 3SAT, our algorithm has *expected polynomial* (in fact, almost linear) running time. The planted 3SAT distribution is the set of satisfiable 3CNF formulas generated in the following manner. First, a truth assignment is picked uniformly at random. Then, each clause satisfied by it is included in the formula with probability p. Extending previous work for the planted 3SAT distribution, we present, for the first time for a satisfiable SAT distribution, an *expected polynomial time* algorithm. Namely, it solves all 3SAT instances, and over the planted distribution (with $p = d/n^2$, $d > 0$ a sufficiently large constant) it runs in expected polynomial time. Our results extend to k-SAT for any constant k.

1 Introduction and Results

1.1 The Planted 3SAT Distribution A 3CNF formula over the variables $x_1, x_2, ..., x_n$ is a conjunction of clauses $C_1, C_2, ..., C_m$ where each clause is a disjunction of three literals. Each literal is either a variable or its negation. A 3CNF is satisfiable if there is a boolean assignment to the variables s.t. every clause contains at least one literal which evaluates to true. 3SAT is the language of all satisfiable 3CNF formulas.

Although 2SAT is known to be in P, 3SAT is one of the most famous NP-complete problems [14]. Understanding the complexity of the satisfiability problem on various natural instances is a fundamental problem in computer science. Thus, while the problem is believed unsolvable in polynomial time in the worst case, many heuristics have been proposed and some of them seem to perform quite well "on average".

Algorithmic theory of random structures has been the focus of extensive research in recent years (see [21] for a comprehensive survey). As part of this trend, uniformly random 3CNFs (generated by selecting at random $m = m(n)$ clauses over the variables $\{x_1, ..., x_n\}$) have also caught the eyes of many researchers. Random 3SAT is known to have a sharp satisfiability threshold in the clause-to-variable ratio [20]. Namely, a random 3CNF with clause-to-variable ratio below the threshold is satisfiable **whp** (with high probability) and one with ratio above the threshold is unsatisfiable **whp**. This threshold is not known exactly (and not even known to be independent of n). The threshold is known to be at least 3.42 [24] and at most 4.5 [25]. Experimental results predict the higher end of the interval [15]. In this work we concentrate on formulas with constant (above the threshold) clause-to-variables ratio. At such ratios, most formulas are unsatisfiable, and analyzing the distribution of random satisfiable instances seems hard. Thus, we focus on the *planted* 3SAT distribution, denoted throughout by $\mathbb{P}_{p,n}$.

A planted instance in $\mathbb{P}_{p,n}$ is generated by first picking at random a truth assignment φ to the n variables. Next, each clause satisfied by φ is included with probability p. In this work we consider $p \geq d/n^2$, where d is a sufficiently large constant. The distribution is highly interesting in its own right. It has been studied empirically in [9], and is the analog of the planted clique, planted bisection, and planted coloring distributions studied e.g. in [1], [2], [7], [17]. The planted 3SAT distribution is studied in [26], where an algorithm based on greedy variable assignment rule is proved to work **whp** for dense instances ($p = \Theta(1/n)$). The authors conjecture that some modification of their algorithm will work for sparser instances. [19], drawing on techniques from [1] (spectral and combinatorial techniques), proves this conjecture.

1.2 Expected Polynomial Time An algorithm that works **whp** has, nevertheless, a drawback. Indeed for typical instances it works well, however for some

*Supported in part by USA-Israel BSF Grant 2002-133, and by Grant 64/01 from the Israel Science Foundation

†Part of the second author's PhD thesis prepared at Tel Aviv University under the supervision of Prof. Michael Krivelevich

‡School Of Mathematical Sciences, Sackler Faculty of Exact Sciences, Tel-Aviv University, Tel-Aviv 69978, Israel.

§School of Computer Science, Sackler Faculty of Exact Sciences, Tel-Aviv University, Tel-Aviv 69978, Israel

rare inputs it does not provide a solution at all. Alternatively, one could require that the algorithm always produce a correct answer but perform well on average. Formally, an algorithm $\mathcal{A}$ with running time $t_{\mathcal{A}}(I)$ on an input instance I, has *expected polynomial running time* over a distribution $\mathcal{D}$ on the inputs, if $\sum_I t_{\mathcal{A}}(I) \cdot Pr_{\mathcal{D}}[I]$ is polynomial. Note that an expected polynomial time algorithm also works **whp**, the converse however isn't necessarily true. In many cases, in order for an algorithm to run in expected polynomial time, a distinction between typical instances (which are easy, and require polynomial time) and atypical ones (which are rare, but may require superpolynomial time) is due. Moreover, loosely speaking, the amount of work put in should be inversely proportional to how atypical the instance is. To perform this dichotomy, a deeper understanding of the underlying probability space is required, as well as a more careful analysis of properties of the random instance. Therefore, by designing algorithms that work in expected polynomial time, one might expect to obtain more robust and efficient algorithms. Several papers discussed algorithms with expected polynomial time, we selectively mention some: [4], [6], [8], [10], [11], [12], [13], [16], [22], [27], [28], [29], [31].

1.3 Related Work The coloring algorithm in [1] works **whp** for sparse planted k-colorable graphs, however it is not an expected polynomial time algorithm. An expected polynomial time algorithm for coloring planted k-colorable graphs is given in [10] for relatively dense graphs (average degree $np = d \cdot k \cdot \log n$). [8] improves upon this result, giving an expected polynomial time algorithm for sparse k-colorable graphs (average degree $np = d \cdot k^2$, which is constant when k is). Analogously for the planted 3SAT setting, the algorithm presented in [19] is not an expected polynomial time algorithm. Although expected polynomial time is not discussed in [26], slightly changing the algorithm makes it such. However, [26] works for dense formulas, namely when $p = d/n$.

1.4 Our Results Combining ideas from [8] and [19], we devise an algorithm for solving (i.e. finding a satisfying assignment) satisfiable 3CNF instances. We prove that our algorithm has expected polynomial running time over $\mathbb{P}_{d/n^2,n}$. As far as the authors know, this is the first work to address the issue of expected polynomial time algorithms for satisfiable SAT distributions. Formally we prove,

Theorem 1.1. *There exists an algorithm SAT that solves 3SAT, and runs in expected polynomial time over the planted 3SAT distribution with $p \geq d/n^2$, d a sufficiently large constant.*

The theorem extends to any constant k, while in this case d depends on k.

The remainder of the paper is structured as follows. In §2, the motivation behind the algorithm, and the algorithm itself are given. In §3, a few properties of $\mathbb{P}_{d/n^2,n}$ are analyzed. These properties come useful when estimating the expected running time of the algorithm in §4. Finally, in §5, we discuss some possibly interesting issues for further research.

2 The Algorithm

Before presenting the algorithm, we offer some intuition. The following discussion assumes the input is sampled according to $\mathbb{P}_{d/n^2,n}$.

2.1 Motivation The algorithm looks for the planted assignment (though it may come across another satisfying assignment). Thus, we say that a variable is wrongly assigned by some truth assignment τ, if $\tau(x)$ differs from the planted assignment of x.

Suppose that the planted assignment sets x to be true. In every chosen clause that contains x, x appears positively (namely, as the literal x) with probability $4/7$, and negatively (namely, as $\bar{x}$) with probability $3/7$. Therefore, by comparing for every variable the number of positive and negative appearances, and assigning it according to the majority, one expects to already have a good approximation of the planted assignment. Such a procedure is often called a *majority vote* (line 1 in the algorithm SAT). In fact, when $p \geq d \log n/n^2$, for a sufficiently large constant d, the majority vote suffices to reconstruct the planted assignment **whp**, see [5] and [23] for a complete discussion and proof. However, when $p = d/n^2$, the majority vote is wrong for some small fraction of the variables ($\exp\{-\Theta(n^2 p)\}$, which is constant in our case).

The next step is to distinguish between correctly and wrongly assigned variables. This is achieved by a careful unassignment procedure, in which the assignment of variables suspected to be wrong is peeled off, leaving us with a partial assignment (lines 3-5). Now we are in a better position as this partial assignment coincides **whp** with the planted one.

It remains to complete the assignment of the unassigned variables. Consider the formula induced by the unassigned variables. One can prove that this formula breaks down **whp** to mutually disjoint subformulas, each containing at most $O(\log n)$ variables. Therefore, using exhaustive search, separately in every subformula, the partial assignment is completed, if possible, to a satisfying one (line 7,8 and 13), while spending polynomial time.

This 3-step heuristic is very similar to the one described and analyzed in [30] (based on [19]). The heuristic finds **whp** a satisfying assignment for $\mathbb{P}_{d/n^2,n}$ but does not necessarily work for all instances. It might be the case, although rare, that the unassignment step didn't filter out the wrongly assigned variables. Therefore the exhaustive search might fail (as the formula induced by the unassigned variables may no longer be satisfiable, or the partial assignment may induce a $(F \vee F \vee F)$ clause). Since the algorithm is required to produce an answer for all instances, one needs to take care of this rare event as well. Suppose indeed that the exhaustive search fails. Our goal is to identify the (minimal) set of variables that survived the unassignment s.t. by flipping its current assignment (given by the majority vote), the partial assignment can be completed to a satisfying one. To this end, a controlled recovery step is employed (lines 9-11). Iteratively, one goes over all subsets Y of assigned variables (in ascending order of size). For each subset one tries all possible assignments $\pi(Y)$, looking for the aforementioned set and flip its assignment.

2.2 Notation Let $V = \{x_1, x_2, .., x_n\}$ be the set of all variables. For a set of variables $U \subseteq V$, let $G(U,E)$, the *residual graph*, be the graph defined as follows. U is the set of vertices. For every $x_i, x_j \in U$, $(x_i, x_j) \in E$ if there exists some clause in I containing both x_i, x_j (regardless of their polarity). A *partial truth assignment* is a vector of length n over $\{0, 1, *\}$, where $*$ stands for a variable which is not assigned. A variable x *supports* a clause C with respect to a partial assignment ψ, if it is the only variable to satisfy C under ψ, and the other two variables are assigned by ψ. E.g., if $C = (x \vee y \vee z)$ and $\psi(C) = (T \vee F \vee F)$, then x supports C w.r.t. ψ. By *simplify* I according to ψ, when ψ is a partial assignment, we mean: in every clause substitute every assigned variable with the value given to it by ψ. Next, if a clause contains a literal which evaluates to true, remove the clause. Otherwise, remove all literals which evaluate to false (the resulting instance is not necessarily in 3CNF form). Denote by $I|_\psi$ the 3CNF I simplified according to ψ. We say that $I|_\psi$ *induces an empty clause* if there exists a clause in which all three literals evaluate to false under ψ. For a set of variables $A \subseteq V$, denote by $I[A]$ the set of clauses in which all variables belong to A.

The algorithm as presented can be optimized using known procedures such as unit-clause-propagation or the pure-literal rule (which can be performed on $I|_{\psi \circ \pi(Y)}$ before the exhaustive search starts). These optimizations were not included in the description of the algorithm for simplicity of presentation and analysis.

Throughout the paper, ϵ is used to denote some small constant independent of d, say $\epsilon = 10^{-5}$. I denotes a random instance from $\mathbb{P}_{d/n^2,n}$, and φ its planted assignment. MAJ is the majority vote assignment of I.

Algorithm 1 : SAT(I)

1: MAJ $\leftarrow$ the majority vote assignment.
2: Set $i = 0$ and $\psi_0 \leftarrow$ MAJ.
3: **while** there exists a variable x whose support is less than $(1-\epsilon)d/2$ w.r.t. ψ_i **do**
4: $\quad \psi_{i+1} \leftarrow \psi_i$ with the variable x unassigned.
5: $\quad i \leftarrow i + 1$.
6: **end while**
7: Let ψ be the final partial assignment, and U be the set of unassigned variables in ψ.
8: Construct the residual graph $G(U,E)$ and find the set of connected components $\{\Gamma_i\}$.
9: **for** $y = 0$ to $|V \setminus U|$ **do**
10: $\quad$ **for all** $Y \subseteq V \setminus U$, $|Y| = y$ and every assignment $\pi(Y)$ to the variables of Y **do**
11: $\quad\quad$ Let $\psi \circ \pi(Y)$ be $\pi(x)$ if $x \in Y$, and $\psi(x)$ otherwise.
12: $\quad\quad$ **if** $I|_{\psi \circ \pi(Y)}$ doesn't induce an empty clause **then**
13: $\quad\quad\quad$ Independently in every Γ_i, using exhaustive search try to satisfy $I|_{\psi \circ \pi(Y)}[\Gamma_i]$.
14: $\quad\quad\quad$ **if** the search succeeded **return** the satisfying assignment **end if**
15: $\quad\quad$ **end if**
16: $\quad$ **end for**
17: **end for**
18: **return** I is not satisfiable.

The correctness of SAT is evident. If the algorithm returns an assignment (line 14), it satisfies I (by the correctness of the exhaustive search and the condition in line 12). If I is satisfiable, in the worst case, $y = |V \setminus U|$ is reached, all possible assignments to the assigned variables are examined, and at least for the one coinciding with some satisfying assignment, the exhaustive search succeeds. If I is not satisfiable, then at least one of the conditions in lines 12, 14 fails every iteration, line 18 is eventually reached and the algorithm answers "not satisfiable". It thus remains to analyze the expected running time over $\mathbb{P}_{d/n^2,n}$.

3 Properties of a Random Instance

In this section, a few properties of $\mathbb{P}_{d/n^2,n}$ are analyzed. These properties come useful when estimating the expected running time of the algorithm. Recall the definition of $\mathbb{P}_{d/n^2,n}$. First a truth assignment φ to the variables $V = \{x_1, x_2, ..., x_n\}$ is picked uniformly at random. Next, every clause satisfied by φ is included in the formula with probability $p = d/n^2$. Clearly I is satisfiable, since at least φ satisfies it. There are $(2^3-1)\cdot\binom{n}{3}$ clauses satisfied by φ. The expected size of I is then $p \cdot 7 \cdot \binom{n}{3} = 7dn/6 + o(n)$. As for the support, every variable x supports $\binom{n-1}{2}$ clauses w.r.t. φ. The expected support of a variable x in I w.r.t. φ is then $p \cdot \binom{n-1}{2} = d/2 + o(1)$. This explains the choice in line 3 of the algorithm.

The following inequality is used to bound binomial coefficients: for $k \in [1, n]$, $\binom{n}{k} \leq (ne/k)^k$. Unless stated otherwise, the basis of the logarithm is the natural basis. In the analysis, both d and n are assumed to be sufficiently large. No attempt was made to optimize the constants in the analysis. Throughout, we let $\alpha_0 = \exp\{-d/5000\}$, $\alpha_1 = \exp\{-\epsilon^2 d/40\}$.

3.1 Majority Vote and Support

PROPOSITION 3.1. *Let $\alpha \in [\alpha_0, \epsilon]$. Let F_{MAJ} be a random variable counting the number of variables in I on which the majority vote disagrees with the planted assignment. Then, $Pr[F_{MAJ} \geq \alpha n] \leq \exp\{-d/4000 \cdot \alpha n\}$.*

There are some delicate technicalities in the proof of this proposition due to dependency issues. The full proof is given in the appendix, here only some intuition is offered. For a variable x, the number of clauses it appears in positively, and the ones it appears in negatively are two independent binomially distributed **rv** (random variables) with a gap of about $d/2$ in their expectations. Let F_x be the event $\{MAJ(x) \neq \varphi(x)\}$. Applying standard bounds on the tail of the binomial distribution (e.g. Chernoff bounds), $Pr[F_x] \leq \exp\{-\Theta(d)\}$. Using the union bound,

$$Pr[F_{MAJ} \geq \alpha n] \leq \binom{n}{\alpha n} Pr[F_{x_1} \wedge F_{x_2} \wedge \ldots \wedge F_{x_{\alpha n}}].$$

The difficulty lies in estimating the latter, as the F_{x_i}'s are dependent.

PROPOSITION 3.2. *Let $\alpha \in [\alpha_1, 1]$. Let X_ϵ be a **rv** counting the number of variables whose support w.r.t. φ is less than $(1-\epsilon)d/2$. Then, $Pr[X_\epsilon \geq \alpha n] \leq \exp\{-\epsilon^2 d/20 \cdot \alpha n\}$*

Proof. The support of a variable x is a binomially distributed **rv** with expectation $d/2 + o(1)$. Let F_x be the event that x supports less than $(1-\epsilon)d/2$ clauses with respect to φ. Applying the Chernoff bound gives $Pr[F_x] \leq \exp\{-\epsilon^2 d/16\}$. Further, the clauses that x supports are all different than the ones $y \neq x$ supports. Hence, the events F_x, F_y are independent. Therefore, in the case of the support, one can easily perform the calculations sketched in Proposition 3.1. In particular,

$$Pr[F_{x_1} \wedge F_{x_2} \wedge \ldots \wedge F_{x_{\alpha n}}] = (Pr[F_x])^{\alpha n}.$$

Using standard calculations, the proposition then follows.

3.2 Dense Subformulas The next property to prove is analogous to a property proved in [1] for random graphs. Loosely speaking, [1] prove that **whp** a random graph doesn't contain a small induced subgraph with a large average degree. Formally, for 3SAT instances,

PROPOSITION 3.3. *For a set of variables $Y \subseteq V$, denote by $F(Y) \subseteq I$ the set of clauses in I containing at least two variables from Y. Let $\rho \in (0,1]$, and $\alpha \in (0, (\rho/16)^4]$. Then, $Pr[\exists Y \subseteq V, |Y| = \alpha n, \exists F(Y) \subseteq I, |F(Y)| \geq \rho d \cdot \alpha n] \leq \exp\{-(\rho d/4 \cdot \log\frac{1}{\alpha}) \cdot \alpha n\}$*

Proof. For fixed Y, there are $\binom{\alpha n}{2}(n-2)2^3 \leq 4\alpha^2 n^3$ clauses containing at least two variables from Y. Each is included in the formula with probability d/n^2. Therefore,

$$\begin{aligned}
&Pr[\exists Y \subseteq V, |Y| = \alpha n, \exists F(Y) \subseteq I, |F(Y)| \geq \rho d \cdot \alpha n] \\
&\leq \binom{n}{\alpha n}\binom{4\alpha^2 n^3}{\rho d \cdot \alpha n}(d/n^2)^{\rho d \cdot \alpha n} \\
&\leq \left(\frac{en}{\alpha n}\right)^{\alpha n}\left(\frac{e \cdot 4\alpha^2 n^3}{\rho d \cdot \alpha n}\right)^{\rho d \cdot \alpha n} \cdot \left(\frac{d}{n^2}\right)^{\rho d \cdot \alpha n} \\
&\leq \left(\frac{e}{\alpha}\right)^{\alpha n}\left(\frac{12\alpha}{\rho}\right)^{\rho d \cdot \alpha n} \\
&\leq \exp\{\rho d \cdot \alpha n \cdot (4 - \log\frac{1}{\alpha}/2 + \log\frac{1}{\rho})\} \\
&\underbrace{\leq}_{\alpha \leq (\rho/16)^4} \exp\{-(\rho d/4 \cdot \log\frac{1}{\alpha}) \cdot \alpha n\}.
\end{aligned}$$

3.3 Expanding Set of Variables We describe a subset of the variables, denoted throughout by H, which plays a crucial role in the analysis. The set H has a good expansion property in the following sense: every variable in H supports at least $(1-\epsilon)d/2$ clauses in $I[H]$ w.r.t. both the planted assignment and the majority vote. H is defined using the following procedure:

Let W be the set of variables whose majority vote disagrees with φ.

Let B be the set of variables whose support w.r.t. φ is less than $(1-\epsilon/2)d/2$.

Let $H \subset V$ be constructed as follows:

1. *Let $H_0 = V \setminus (B \cup W)$.*

2. *While there exists a variable $a_i \in H_i$ which supports less than $(1-\epsilon)d/2$ clauses in $I[H_i]$, define $H_{i+1} = H_i \setminus \{a_i\}$.*

3. *Let a_m be the last variable removed at step 2. Define $H = H_{m+1}$.*

PROPOSITION 3.4. *Let $\bar{H} = V \setminus H$, let $\alpha \in [4\alpha_1, (\epsilon/128)^4]$. Then, $Pr[|\bar{H}| \geq \alpha n] \leq \exp\{-\epsilon^2 d/100 \cdot \alpha n\}$.*

Proof. Partition the variables in $\bar{H}$ into variables that belong to $B \cup W$, and variables that were removed in the iterative step, $\bar{H}^{it} = H_0 \setminus H$. If $|\bar{H}| \geq \alpha n$, then at least one of $B \cup W$, $\bar{H}^{it}$ has cardinality at least $\alpha n/2$. Consequently,

$$Pr[|\bar{H}| \geq \alpha n] \leq \underbrace{Pr[|B \cup W| \geq \alpha n/2]}_{(1)} + \underbrace{Pr[|\bar{H}^{it}| \geq \alpha n/2 \mid |B \cup W| \leq \alpha n/2]}_{(2)}.$$

Propositions 3.1, 3.2 are used to bound (1). To bound (2), observe that every variable that is removed in iteration i of the iterative step (step 2), supports at least $\epsilon/2 \cdot d/2$ clauses in which at least another variable belongs to $\{a_1, a_2, ..., a_{i-1}\} \cup B \cup W$. Consider iteration $\alpha n/2$. Assuming $|B \cup W| \leq \alpha n/2$, by the end of this iteration there exists a set containing at most αn variables, and there are at least $(\epsilon/8) \cdot d \cdot \alpha n$ clauses containing at least two variables from it (note that no clause was counted twice as the supporter of a clause is unique). Plugging $\epsilon/8$ in Proposition 3.3, (2) is bounded. The proposition then follows from standard calculations.

Bounding the probability of $\bar{H}$ being too large, plays a crucial role in the proof of the following property.

PROPOSITION 3.5. *Let $\beta \in [\Theta(\log n/(n\epsilon^6 d)), 1]$. Let $G(\bar{H}, E)$ be the residual graph induced by $\bar{H}$. The probability there exists a connected component of size βn in $G(\bar{H}, E)$ is at most $\exp\{-\Omega(\epsilon^6 d) \cdot \beta n\}$.*

The proof of this proposition is very similar to the one given in [19], [30]. Here only some intuition is offered. A detailed, though not complete, proof is given in the appendix. Recall the well known $G_{n,p}$ distribution for random graphs, where each edge is included independently of the others with probability p. If $p < \frac{1}{n}$ then **whp** the largest connected component in G is of size at most $O(\log n)$ (see e.g. [3] for a complete discussion). Now consider a random subgraph of size αn of $G_{n,d/n}$, this subgraph is exactly $G_{\hat{n},\alpha d/\hat{n}}$, where $\hat{n} = \alpha n$. If $\alpha d < 1$, **whp** the largest connected component in $G_{\hat{n},\alpha d/\hat{n}}$ is of size $O(\log \hat{n})$. If on the other hand $\alpha d > 1$, in our setting that would imply that $\bar{H}$ is a large set. This however, happens with small probability (Proposition 3.4). In our case, the residual graph $G(\bar{H}, E)$ is not a truly random graph, and $\bar{H}$ is not a truly random subset of variables. Hence the proof is more delicate.

4 Analysis of the Expected Running Time

To begin with, some intuition is given. As mentioned above, the set H plays a crucial role in the analysis. The majority vote sets H correctly (by the construction of H). Also, H survives the unassignment step (due to its expansion properties w.r.t. MAJ). Therefore, by the end of the unassignment step, $I[H]$ is surely satisfied by ψ (the partial assignment defined in line 7 of SAT). It remains to satisfy, if not yet satisfied by ψ, $I \setminus I[H]$. Normally, $I[H]$ contains almost all clauses (Proposition 3.4). Therefore **whp**, $I \setminus I[H]$ breaks down to small-size mutually-disjoint subformulas, allowing the exhaustive search to run efficiently (more precisely, in expected polynomial time). Unfortunately, the unassignment step does not necessarily screen out exactly the set H. There may be additional variables, outside of H, that also survived the unassignment step. Normally, their assignment coincides with φ. Although rare, this may not be the case, possibly preventing the exhaustive search from ending up successfully. Let Y_0 be the set of variables which prevent the exhaustive search from ending up successfully. If Y_0 is non-empty (which normally doesn't happen), we shall prove that both the majority vote is wrong for Y_0 and Y_0 is a dense set in the sense of Proposition 3.3. Both however, happen with small probability. Carrying out all the calculations, one proves that in expectation only a constant number of recovery iterations are required to find Y_0 and correct its assignment, allowing the algorithm to run in expected polynomial time.

Formalizing the above, let Y_0 be the set of variables used in the iteration when the algorithm finally finds a satisfying assignment for I, and $\pi(Y_0)$ be the assignment given to Y_0. We introduce some **rv** that help clarify the description and analysis of the running time: R_{EXH} measures the maximal (over all sets $Y \subseteq V$) running time of the exhaustive search (line 13); $R_{SIMPLIFY}$ measures the time it takes to calculate $I|_{\psi\circ\pi(Y)}$ and $\{I|_{\psi\circ\pi(Y)}[\Gamma_i]\}$ (lines 12-13); R_{MAIN} measures the running time of the majority vote, the unassignment step and the decomposition of $G(U, E)$ to connected components (lines 1-8); R_{REC} (REC for recovery) is the number of times the iteration in lines 9-10 is performed.

Finally, let $R_{\mathcal{A}}$ be a **rv** measuring the running time of the algorithm. The probability of all **rv** is taken over the coin flips in the choice of I. Using the above notations, $R_{\mathcal{A}} \leq R_{MAIN} + R_{REC} \cdot (R_{EXH} + R_{SIMPLIFY})$. Our goal is to prove that $E[R_{\mathcal{A}}] \leq n^c$ for some constant $c \geq 1$. In particular we shall prove $c = 1 + \Theta(1/d)$.

PROPOSITION 4.1. $E[R_{\mathcal{A}}] \leq E[R_{MAIN}] + E[R_{REC} \cdot R_{EXH}] + E[R_{REC}] \cdot E[R_{SIMPLIFY}]$.

Proof. The proposition follows from linearity of expectation and the fact that $R_{SIMPLIFY}$ and R_{REC} are independent **rv** (the simplification's result does depend on the set Y, but the time it takes to simplify I can be bounded using a function of $|I|$, which is independent of R_{REC}). Unfortunately, R_{EXH} and R_{REC} are dependent.

PROPOSITION 4.2. *Both* $E[R_{MAIN}] = O(dn)$ *and* $E[R_{SIMPLIFY}] = O(dn)$.

Let m be the number of clauses in I. It is easily seen that $E[R_{SIMPLIFY}], E[R_{MAIN}] \leq O(m \cdot n)$, since the most naive way of performing lines 1-8, and calculating $I|_{\psi\circ\pi(Y)}$, $\{I|_{\psi\circ\pi(Y)}[\Gamma_i]\}$ takes at most $O(m \cdot n)$. To prove $E[R_{SIMPLIFY}], E[R_{MAIN}] = O(dn)$ it suffices to show how to perform these procedures in time $O(m + n)$. Since $E[O(m + n)] = O(dn)$ the proposition follows. This requires some extra, however very standard, work. Details omitted.

PROPOSITION 4.3. $E[R_{REC}] \leq O(1)$.

Before proving this proposition we make the following key observation.

PROPOSITION 4.4. *Let* $Y_0, \pi(Y_0)$ *be as above. If* $|Y_0| = y_0 > 0$ *then* (a) *the majority vote is wrong for at least* y_0 *variables* (b) *there are at least* $d \cdot y_0/3$ *clauses containing at least two variables from* Y_0.

Proof. To prove (a), suppose by contradiction that MAJ is wrong for less than y_0 variables. Let Y' be the set of assigned variables on which MAJ is wrong. Consider the iteration in which $Y = Y'$ and $\pi(Y') = \overline{MAJ}$ (line 10). In this iteration, $\psi \circ \pi(Y')$ agrees with φ on all assigned variables. Therefore, the exhaustive search (line 13) succeeds (since $\psi \circ \pi(Y')$ can be at least completed to φ) and the algorithm stops, contradicting the choice of Y_0 ($|Y'| < y_0$, therefore $Y' \neq Y_0$).
To prove (b), observe that by the algorithm $Y_0 \subseteq V \setminus U$. Further, $\pi(Y_0)$ must disagree with MAJ on Y_0 (otherwise define Y' to be Y_0 minus the variables on which $\pi(Y_0)$ and MAJ agree. The algorithm performs with Y' exactly as it would with Y_0, but $|Y'| < y_0$).

Now, if $x \in Y_0$, since $x \in V \setminus U$, by the algorithm, x must support at least $(1-\epsilon)d/2 > d/3$ clauses in which all variables are assigned. $\pi(Y_0)$ flips the assignment of x (w.r.t. MAJ), causing all the clauses it supports to become not satisfied. Therefore, in every such clause there must be at least another variable $z \in Y_0$ (or else, $I|_{\psi\circ\pi(Y_0)}$ induces an empty clause, and the algorithm could not have ended successfully with $Y = Y_0$). Since every clause is counted once (unique support) property (b) follows.

Proof. (Proposition 4.3) For every y (line 9), the number of iterations performed (line 10) is $\binom{n}{y}2^y$. Hence,

$$E[R_{REC}] \leq \sum_{y=0}^{n} Pr[|Y_0| = y] \cdot \sum_{y'=0}^{y} \binom{n}{y'} 2^{y'}.$$

To bound this expression, we decompose it according to the different values of $y = \alpha n$. If $|Y_0| = \alpha n \geq \alpha_0 n$, Propositions 3.1 and 4.4.a are employed to bound

$$Pr[|Y_0| = \alpha n] \leq \exp\{-d/4000 \cdot \alpha n\}.$$

Otherwise, Propositions 3.3 with $\rho = 1/3$ and 4.4.b give,

$$Pr[|Y_0| = \alpha n] \leq \exp\{-(d/12 \cdot \log \frac{1}{\alpha}) \cdot \alpha n\}.$$

For $y \leq n/2$,

$$\sum_{y'=1}^{y} \binom{n}{y'} 2^{y'} \leq y \binom{n}{y} 2^y \leq \exp\{y \cdot (\log(n/y) + 3)\}.$$

In any case,

$$\sum_{y'=1}^{y} \binom{n}{y'} 2^{y'} \leq 3^n.$$

Finally, we are ready to bound $E[R_{REC}]$:

$$E[R_{REC}] = \sum_{y=0}^{n} Pr[|Y_0| = y] \sum_{y'=0}^{y} \binom{n}{y'} 2^{y'} \leq$$

$$\leq \underbrace{O(1)}_{\alpha=0}$$

$$+ \sum_{\alpha=1/n}^{\alpha_0} \exp\{(\log \frac{1}{\alpha} + 3)\alpha n\} \cdot \underbrace{\exp\{-(d/12 \cdot \log \frac{1}{\alpha})\alpha n\}}_{\text{Proposition 3.3}}$$

$$+ \sum_{\alpha_0}^{\epsilon} \exp\{(\log \frac{1}{\alpha} + 3) \cdot \alpha n\} \cdot \underbrace{\exp\{-d/4000 \cdot \alpha n\}}_{\text{Proposition 3.1}}$$

$$+ \sum_{\alpha=\epsilon}^{1} 3^n \cdot \underbrace{\exp\{-\epsilon d/4000\}}_{\text{Proposition 3.1}} = O(1).$$

PROPOSITION 4.5. $E[R_{REC} \cdot R_{EXH}] = n^{1+\Theta(1/d)}$.

Before proving this proposition we make the following observation.

LEMMA 4.1. *The set H defined in subsection 3.3 is assigned when lines 9-17 are executed.*

Proof. φ and MAJ coincide on H (by the construction of H). Thus, every variable $x \in H$ supports at least $(1-\epsilon)d/2$ clauses w.r.t. to MAJ (the clauses which prevented the removal of x from H in step 2 in the construction of H). Hence H survives the first round of unassignment (lines 3-5). This invariant is kept throughout the unassignment step to prove that H remains assigned at its end. Since this (lines 3-5) is the only place where unassignment is performed, the lemma follows.

Returning to Proposition 4.5. Let K be a **rv** measuring the size of the largest connected component in $G(\bar{H}, E)$. Lemma 4.1 implies that $U \subseteq \bar{H}$. Therefore, $R_{EXH} \leq n \cdot 2^K \cdot O(K^3)$; n upper bounds the number of connected components, 2^K is the number of possible assignments $\tau(\Gamma_i)$, and $O(K^3)$ is the time it takes to check for every $\tau(\Gamma_i)$ whether it satisfies $I|_{\psi\circ\pi(Y)}[\Gamma_i]$ (the number of clauses in $I|_{\psi\circ\pi(Y)}[\Gamma_i]$ is $O(K^3)$). We introduce some notations to simplify the presentation.

$$f(k,\alpha) = n \cdot 2^k \cdot O(k^3) \cdot \binom{n}{\alpha n} \cdot 2^{\alpha n}.$$

$$g(k,\alpha) = Pr[K = k, |Y_0| = \alpha n].$$

$$E[R_{REC} \cdot R_{EXH}] \leq \sum_{k,\alpha} f(k,\alpha) \cdot g(k,\alpha).$$

Similarly to the proof of Proposition 4.3, the proposition follows by breaking the sum according to the different values of α and k. Details omitted. Combining Propositions 4.1, 4.2, 4.3, and 4.5 one obtains $E[R(\mathcal{A})] \leq \mathrm{n}^{1+\Theta(1/\mathrm{d})}$ as desired. This also finishes the proof of Theorem 1.1.

5 Discussion

To summarize, we presented an algorithm that solves 3SAT. Of course, one cannot expect the algorithm to be polynomial, however, over the planted distribution with $p \geq d/n^2$, the algorithm has an expected polynomial running time. Previous work on planted 3SAT present algorithms that work **whp**. Our result extends the latter by presenting an expected polynomial time algorithm. An expected polynomial time algorithm also works **whp**, but in addition it works for all instances, while keeping an overall average polynomial running time. A few interesting issues are open for future research.

5.1 The k-opt heuristic In [18], a different algorithm, based on the k-opt heuristic, is suggested. A succinct description follows. The algorithm can be viewed as performing a search in the graph whose vertices are all 2^n assignments, and two assignments are connected if they differ by the assignment of exactly one variable. The algorithm starts at the vertex corresponding to the majority vote on I. The goal is to reach a vertex corresponding to a satisfying assignment. We say that an assignment ψ' improves ψ if the set of clauses satisfied by ψ' contains the set of clauses satisfied by ψ. While not at a satisfying assignment, in every iteration the algorithm checks all possible improving assignments at distance at most k. If such is found, the algorithm moves to that vertex. Otherwise, the algorithm has reached a local optimum, and the algorithm fails (or in a different version, k is increased by 1 and the algorithm continues with the new k). It is shown in [18] that an improving assignment at distance at most k, if such exists, can be found in time $O(n2^k)$. Thus, by demanding $k = O(\log n)$, the algorithm is polynomial. As the analysis in [18] shows, for $\mathbb{P}_{d/n^2,n}$, the algorithm finds a satisfying assignment **whp** when $k = \log n$.

An interesting question is whether one can transform the k-opt version to run in expected polynomial time. The difficulty lies in the fact that no explicit distinction between correctly and wrongly assigned variables is done.

5.2 Simplifying the coloring algorithm for k-colorable graphs Let us compare our algorithm with the one presented in [8] for coloring planted k-colorable graphs in expected polynomial time. The coloring algorithm is arguably more complicated than the satisfiability one. However, this is not the case when comparing [1] to [19]. Why is the satisfiability algorithm simpler? We derive the first approximation to the planted solution by the majority vote. The coloring algorithm approximates the planted solution using SDP. Analogously to the set H, one can define such a set for planted k-colorable graphs. The role of the set H is crucial in the analysis of both algorithms. One first proves that **whp** the algorithm sets H according to the planted solution, allowing H to survive the unassignment step. Further, one proves that **whp** H is sufficiently large so that the residual graph $G(\bar{H}, E)$ has no connected component of size larger than $\log n$. Therefore, an exhaustive search can be employed to complete the solution. The analysis in [8] doesn't show that the SDP approximation sets H correctly with sufficiently high probability to allow an expected polynomial time algorithm. Hence, another refinement step precedes the unassignment. In our case,

H is set correctly by definition, and the refinement step can be skipped. This simplifies greatly the algorithm and the analysis. Further, the majority vote is much simpler to analyze and implement compared to SDP. An interesting research question is replace the SDP step in the coloring algorithm with a simpler and more natural step analogous to the majority vote.

Acknowledgements The second author would like to thank Prof. Uriel Feige for introducing him to the intriguing world of random structures.

References

[1] N. Alon and N. Kahale. *A spectral technique for coloring random 3-colorable graphs.* SIAM J. Comput., 26 (1997), pp. 1733–1748.

[2] N. Alon, M. Krivelevich, and B. Sudakov. *Finding a large hidden clique in a random graph.* Random Structures and Algorithms, 13 (1998), pp. 457–466.

[3] N. Alon and J. H. Spencer. *The probabilistic method.* Wiley-Interscience Series in Discrete Mathematics and Optimization. Wiley-Interscience [John Wiley & Sons], New York, second edition, 2000.

[4] R. Beier and B. Vöcking. *Random knapsack in expected polynomial time.* J. Comput. and Syst. Sci. 69 (2004), pp. 306–329.

[5] E. Ben Sasson, Y. Bilu, and D. Gutfreund. *Finding a randomly planted assigment in a random 3CNF.* manuscript, 2002.

[6] E. A. Bender and H. S. Wilf. *A theoretical analysis of backtracking in the graph coloring problem.* J. of Algorithms, 6 (1985), pp. 275–282.

[7] A. Blum and J. Spencer. *Coloring random and semi-random k-colorable graphs.* J. of Algorithms, 19 (1995), pp. 204–234.

[8] J. Böttcher. *Coloring sparse random k-colorable graphs in polynomial expected time.* In Proc. 30th International Symp. on Mathematical Foundations of Computer Science. Lecture Notes in Comput. Sci. 3618 (2005), pp. 156–167.

[9] B. Cha and K. Iwama. *Performance test of local search algorithms using new types of random CNF formulas.* In Proc. 14th International Joint Conference on Artificial Intelligence, pp. 304–311, 1995.

[10] A. Coja-Oghlan. *Coloring semirandom graphs optimally.* In Proc. 31st International Colloquium on Automata, Languages, and Programming, pp. 383–395, 2004.

[11] A. Coja-Oghlan. *Finding large independent sets in polynomial expected time.* In Proc. 20th Symp. on Theoretical Aspects of Comp. Science. Lecture Notes in Comput. Sci. 2607 (2003), pp. 511–522.

[12] A. Coja-Oghlan, A. Goerdt, A. Lanka, and F. Schädlich. *Techniques from combinatorial approximation algorithms yield efficient algorithms for random 2k-SAT.* Theoret. Computer Sci., 329 (2004), pp. 1–45.

[13] A. Coja-Oghlan and A. Taraz. *Colouring random graphs in expected polynomial time.* In Proc. 20th Symp. on Theoretical Aspects of Comp. Science. Lecture Notes in Comput. Sci. 2607 (2003), pp. 487–498.

[14] S. A. Cook. *The complexity of theorem-proving procedures.* In Proc. 3rd ACM Symp. on Theory of Computing, pp. 151–158, 1971.

[15] J. M. Crawford and L. D. Auton. *Experimental results on the crossover point in random 3-SAT.* Artificial Intelligence, 81 (1996), pp. 31–57.

[16] M. Dyer and A. Frieze. *The solution of some random NP-hard problems in polynomial expected time.* J. of Algorithms, 10 (1989), pp. 451–489.

[17] U. Feige and R. Krauthgamer. *Finding and certifying a large hidden clique in a semirandom graph.* Random Structures and Algorithms, 16 (2000), .pp 195–208.

[18] U. Feige and D. Vilenchik. *A local search algorithm for 3SAT.* Technical report, The Weizmann Institute of Science, 2004.

[19] A. Flaxman. *A spectral technique for random satisfiable 3CNF formulas.* In Proc. 14th ACM-SIAM Symp. on Discrete Algorithms, pp. 357–363, 2003.

[20] E. Friedgut. *Sharp thresholds of graph properties, and the k-sat problem.* J. Amer. Math. Soc., 12 (1999), pp. 1017–1054.

[21] A. Frieze and C. McDiarmid. *Algorithmic theory of random graphs.* Random Structures and Algorithms, 10 (1997), pp. 5–42.

[22] M. Fürer, C. R. Subramanian, and C. E. Veni Madhavan. *Coloring random graphs in polynomial expected time.* In Proc. 4th Annual International Symposium on Algorithms and Computation. Lecture Notes in Comput. Sci., 762 (1993), pp. 31–37.

[23] I. Gent. *On the stupid algorithm for satisfiability.* Technical report, Strathclyde University, 1998.

[24] A. C. Kaporis, L. M. Kirousis, and E. G. Lalas. *The probabilistic analysis of a greedy satisfiability algorithm.* In Proc. 10th Annual European Symposium on Algorithms. Lecture Notes in Comput. Sci., 2461 (2002), pp. 574–585.

[25] A. C. Kaporis, L. M. Kirousis, Y. C. Stamatiou, M. Vamvakari, and M. Zito. *Coupon collectors, q-binomial coefficients and the unsatisfiability threshold.* In Proc. Theoretical Computer Science: 7th Italian Conference. Lecture Notes in Comput. Sci., 2202 (2001), pp. 328–338.

[26] E. Koutsoupias and C. H. Papadimitriou. *On the greedy algorithm for satisfiability.* Info. Process. Letters, 43 (1992), pp. 53–55. 1992.

[27] M. Krivelevich. *Deciding k-colorability in expected polynomial time.* Info. Process. Letters, 81 (2002), pp. 1–6.

[28] M. Krivelevich and V. H. Vu. *Approximating the independence number and the chromatic number in expected polynomial time.* J. Comb. Optim., 6 (2002), pp. 143–155.

[29] H. J. Prömel and A. Steger. *Coloring clique-free graphs*

in linear expected time. Random Structures and Algorithms, 3 (1992), pp. 375–402.

[30] D. Vilenchik. *Finding a satisfying assignment for semi-random satisfiable 3CNF formulas.* Master's thesis, The Weizmann Institute of Science, Rehovot, Israel, January 2004.

[31] H. S. Wilf. *Backtrack: an $O(1)$ expected time algorithm for the graph coloring problem.* Info. Process. Letters, 18 (1984), pp. 119–121.

A Proof of Proposition 3.1

For a variable x, let F_x be the event $\{MAJ(x) \neq \varphi(x)\}$. Assume w.l.o.g that $\varphi(x) = T$. Let P_x (resp. N_x) be a **rv** counting the number of clauses in which x appears positively (resp. negatively). It is readily seen that

$$P_x \sim Binom(7\binom{n-1}{2}, 4/7 \cdot d/n^2)$$

$$N_x \sim Binom(7\binom{n-1}{2}, 3/7 \cdot d/n^2).$$

Therefore,

$$E[P_x] = 2d + o(1), E[N_x] = 3d/2 + o(1).$$

Namely there is a gap of $d/2$ in the expectations. If the majority is wrong for x then at least one of N_x, P_x has deviated from its expectation by at least $d/4$. Following the intuition given in Proposition 3.1, our first goal is to bound $Pr[F_{x_1} \wedge F_{x_2} \wedge ... \wedge F_{x_{\alpha n}}]$ for $\alpha \in [\alpha_0, \epsilon]$.

PROPOSITION A.1. *Let $T = \{x_1, x_2, .., x_{\alpha n}\}$ be a fixed set of variables. $Pr[F_{x_1} \wedge F_{x_2} \wedge ... \wedge F_{x_{\alpha n}}] \leq exp\{-d/2000 \cdot \alpha n\}$*

Before proving this proposition, some more observations are due. Let $f(T)$ be the number of clauses in I containing at least two variables from T. It is easily seen that

$$E[f(T)] \leq 7 \cdot \binom{|T|}{2} \cdot n \cdot d/n^2 \leq 7\alpha d/2 \cdot \alpha n.$$

LEMMA A.1. *For T as above, $Pr[f(T) \geq d/60 \cdot \alpha n] \leq \exp\{-d/80 \cdot \alpha n\}$.*

The proof of this Lemma consists of standard probabilistic arguments (Chernoff bound for example). Assume indeed that $f(T) \leq d/60 \cdot \alpha n$. We say that a variable in T is *T-influenced* if it appears in more that $d/10$ clauses with at least another variable from T. Let $Y \subseteq T$ be the set of T-influenced variables. It holds that $(|Y| \cdot d/10)/3 \leq f(T)$ (since every clause was counted at most 3 times). Put differently, $|Y| \leq 30f(T)/d \leq \alpha n/2$ (by our assumption on $f(T)$). Now we can approach the proof of Proposition A.1.

Proof. (Proposition A.1) Assume that $f(T) \leq d/60 \cdot \alpha n$. W.l.o.g, let $x_1, .., x_{\alpha n/2}$ be a set of non-T-influenced variables (namely a subset of $T \setminus Y$). Let $\hat{P}_{x_i}$ (resp. $\hat{N}_{x_i}$) be a **rv** counting the positive (resp. negative) appearances of x_i when only clauses where x_i is the only variable from T are counted. Assuming $\varphi(x_i) = T$,

$$E[\hat{P}_{x_i}] = 4 \cdot \binom{(1-\alpha)n}{2} \cdot d/n^2 = (1-\alpha)^2 2d + o(1)$$

$$E[\hat{N}_{x_i}] = (1-\alpha)^2 3d/2 + o(1).$$

Since $\alpha \leq \epsilon$, it follows then that

$$|E[\hat{P}_{x_1}] - E[\hat{N}_{x_1}]| > d/3.$$

For $i \in [1, \alpha n/2]$, if F_{x_i} occurs, then at least one of $\hat{P}_{x_i}, \hat{N}_{x_i}$ has deviated from its expectation by more than $(d/3 - d/10)/2 > d/10$ (otherwise the majority is not wrong, since x_i appears in at most $d/10$ "misleading" clauses with variables from T). Using the Chernoff bound once again,

$$\begin{aligned} &Pr[F_{x_i}] \leq \\ &\quad Pr[|\hat{P}_{x_i} - E[\hat{P}_{x_i}]| \geq d/10] + \\ &\quad Pr[|\hat{N}_{x_i} - E[\hat{N}_{x_i}]| \geq d/10] \\ &\quad \leq \exp\{-d/900\}. \end{aligned}$$

Observe that all the $\hat{P}_{x_i}, \hat{N}_{x_i}$, $i \in [1, \alpha n]$ are mutually independent, as they correspond to disjoint sets of clauses (by their definition). For brevity, Let $F^j = F_{x_1} \wedge F_{x_2} \wedge ... \wedge F_{x_j}$. To conclude,

$$\begin{aligned} &Pr[F^{\alpha n}] \leq \\ &\quad \leq Pr[F^{\alpha n/2} | f(T) \leq d/60 \cdot \alpha n] \cdot Pr[f(T) \leq d/60 \cdot \alpha n] \\ &\quad + Pr[f(T) \geq d/60 \cdot \alpha n] \\ &\quad \leq \exp\{-d/900 \cdot \alpha n/2\} \cdot (1 - \exp\{-d/80 \cdot \alpha n\}) \\ &\quad + \exp\{-d/80 \cdot \alpha n\} \\ &\quad \leq \exp\{-d/2000 \cdot \alpha n\} \end{aligned}$$

To conclude the proof of Proposition 3.1, taking the union bound over all possible sets T,

$$\begin{aligned} &Pr[F_{MAJ} \geq \alpha n] \leq \binom{n}{\alpha n} \cdot \exp\{-d/2000 \cdot \alpha n\} \\ &\quad \leq (e/\alpha)^{\alpha n} \cdot \exp\{-d/2000 \cdot \alpha n\} \\ &\quad \leq \exp\{\alpha n(1 + \log \frac{1}{\alpha} - d/2000) \leq \exp\{-d/4000 \cdot \alpha n\} \end{aligned}$$

The last inequality is due to $\alpha \geq \alpha_0 = \exp\{-d/5000\}$.

B proof of Proposition 3.5

Proof. Our strategy is to show that **whp**, the largest tree in $G(\bar{H}, E)$ is of size βn. Let T be a fixed tree on βn variables, and let T' be a fixed collection of clauses such that each edge of T is induced by some clause of T'. Let $V(T)$ be the set of vertices of T, and $E(T)$ be the set of edges of T. We say that an edge of T is *covered* by T' if there exists a clause in T' that contains both its endpoints. We say that T' is *minimal* if by deleting a clause from T', T is not covered by T' anymore. By the definition of minimality, $|T'| \le |E(T)| = |V(T)| - 1$ (T is a tree). Our first goal is to bound

$$Pr[T' \subseteq I \text{ and } V(T) \cap H = \emptyset].$$

Both $Pr[T' \subseteq I]$ and $Pr[V(T) \cap H = \emptyset]$ are easily calculated (for fixed T, T'). However, since H is not a truly random subset of V, the events $\{T' \subseteq I\}$ and $\{V(T) \cap H = \emptyset\}$ are not independent. To decouple the dependency, we introduce two new sets, J and H', replacing $V(T)$ and H.

The analysis follows closely the one given in [19], [30]. However, since we describe an expected polynomial time algorithm, a more tight analysis is due. We omit most of the proofs, given fully in [19], [30] and only point out the differences.

Let $J \subseteq V(T)$ be a set of variables which appear in at most 6 clauses of T'. For a set of clauses I, let $|I|$ denote the number of clauses in I.

Proposition B.1. $|J| \ge \frac{|V(T)|}{2}$

Similarly to the aforementioned set H, we define H' by the following iterative procedure:

Let V denote the set of all variables.

Let W be the set of variables for which the majority vote is wrong or for which the majority is right with advantage smaller than 7.

Let B be the set of variables which support less than $(1 - \epsilon/2)d/2$ clauses w.r.t. φ.

Let $H' \subset V$ be constructed as follows:

1. *Let $H'_0 = V \setminus \{W \cup B \cup (V(T') \setminus J)\}$.*
2. *While there is a variable $a_i \in H'_i$ which supports less than $(1 - \epsilon)d/2$ clauses with only variables of H'_i, define $H'_{i+1} = H'_i \setminus \{a_i\}$.*
3. *Let a_m be the last variable removed at step 2. Define $H' = H'_{m+1}$.*

Proposition B.2. *Let $\bar{H}' = V \setminus H'$, let $\beta \le \alpha \in [4\alpha_1, (\epsilon/128)^4]$. $Pr[\bar{H}' \ge 4\alpha n] \le \exp\{-\epsilon^2 d/100 \cdot \alpha n\}$.*

The following observation is the main key for the entire discussion.

Proposition B.3. *$Pr[T' \subset I$ and $V(T) \cap H = \emptyset] \le Pr[T' \subset I] \cdot Pr[J \cap H' = \emptyset]$.*

We are now ready to bound the probability of a tree of size βn in $G(\bar{H}, E)$.

Proposition B.4. *Let $\rho \in [\beta, 1]$ be such that $|\bar{H}'| = \rho n$. Then, $Pr[T' \subset I$ and $V(T) \cap H = \emptyset] \le (6\rho)^{|T'|/2}(d/n^2)^{|T'|}$*

Lemma B.1. *Let T be a fixed tree with k vertices. The number of minimal sets of clauses that span T is at most*

$$\sum_{s=0}^{k/2} \binom{k}{2} 7^{k-s-1} n^{k-2s-1}$$

Theorem B.1. *(Cayley) The number of spanning trees of K_n is n^{n-2}*

Proposition B.5. *The probability of a tree of size βn, $\beta n = \Omega(\log n / n\epsilon^6 d)$, in $G(\bar{H}, E)$ is at most $\exp\{-\Omega(\epsilon^6 d) \cdot \beta n\}$*

Proof. Set $\alpha_2 = 4(\epsilon/128)^4$. Let B be the event $\{\exists T' \subset I, |V(T)| = \beta n$ and $V(T) \cap H = \emptyset\}$ or namely, there exists a tree of size βn in $G(\bar{H}, E)$.

$$Pr[B] = \sum_{\rho=\beta}^{1} Pr[B \mid |\bar{H}'| = \rho n] \cdot Pr[|\bar{H}'| = \rho n]$$

$$\underbrace{\le}_{\text{union bound}} \sum_{\rho=\beta}^{1} Pr[|\bar{H}'| = \rho n] \sum_{s=0}^{\beta n/2} \underbrace{\binom{n}{\beta n} \cdot (\beta n)^{\beta n - 2}}_{\text{number of trees of size } \beta n} \cdot$$

$$\underbrace{\cdot\, 7^{\beta n - s - 1} \cdot n^{\beta n - 2s - 1} \binom{\beta n}{2}}_{\text{extend to a minimal set}} \underbrace{\cdot (d/n^2)^{\beta n - s - 1} \cdot (6\rho)^{\beta n/2}}_{\text{Proposition B.4}} \le$$

$$\le \sum_{\rho=\beta}^{\max(16\alpha_1,\beta)} (6\rho)^{\beta n/2}(e \cdot d)^{\beta n} \beta n/2$$

$$+ \sum_{\rho=\max(16\alpha_1,\beta)}^{1} Pr[|\bar{H}'| = \rho n] \le$$

$$\underbrace{\le}_{\text{Proposition B.2}} \sum_{\rho=\beta}^{\max(16\alpha_1,\beta)} (6\rho)^{\beta n/2}(e \cdot d)^{\beta n} \beta n/2$$

$$+ \sum_{\rho=\max(16\alpha_1,\beta)}^{\max(\alpha_2,\beta)} \exp\{-\epsilon^2 d/100 \cdot \rho n\}$$

$$+ \sum_{\rho=\max(\alpha_2,\beta)}^{1} \exp\{-\Omega(\epsilon^6 d) n\} \le \exp\{-\Omega(\epsilon^6 d) \cdot \beta n\}.$$

This finishes the proof of Proposition 3.5.