Finding a satisfying assignment for planted NAE-E3-SAT using a voting style algorithm

by

Siavash Bolourani
Honours Bachelor of Science - Computer science , University of Toronto, 2005
Honours Bachelor of Science - Mathematics - applications, University of Toronto, 2005

a Thesis submitted in partial fulfillment
of the requirements for the degree of
Master of Science
in the School
of
Computing Science

APPROVAL

Name:	Siavash Bolourani
Degree:	Master of Science
Title of Thesis:	Finding a satisfying assignment for planted NAE-E3-SAT using a voting style algorithm

Examining Committee: Dr. Ted Kirkpatrick
Chair

Dr. Andrei Bulatov, Senior Supervisor

Dr. David Mitchell, Senior Supervisor

Dr. Petra Berenbrink, SFU Examiner

Date Approved: 30 June 2010

Abstract

A random planted formula is constructed by, first, fixing a certain, planted assignment to the variables, and then adding only clauses that are satisfied by the planted assignment. The purpose of such approach is to generate a random-like formula that is guaranteed to have a satisfying assignment. The random planted 3-SAT has received significant attention. In particular, it has been shown through different approaches that when containing sufficiently many clauses such a problem is solvable with high probability in polynomial time.

In this work we obtain a similar result for the random planted Not-All-Equal-SAT problem, which is defined in the same way except that the clauses added are the Not-All-Equal clauses. We follow one of the aforementioned approaches by Krivelevich and Vilenchik. Their algorithm first obtains an approximation of the planted solution by counting the number of occurrences of each variable positively and negatively, then unassigning 'unreliable' variables and searching an assignment for then by brute force. In the case of NAE-SAT the first step, voting, makes no sense. We show that the same result can be achieved by solving the MAX-CUT problem in the co-occurrence graph of the formula.

To Maman, Baba, and Solmaz.

"There is no branch of mathematics, however abstract, which may not some day be applied to phenomena of the real world"

— Nikolai Ivanovich Lobachevsky

Acknowledgments

Publication of this thesis was an interesting journey. It is my honor to thank those whom I was fortunate to work with in my MSc studies. First of all, I would like to that Andrei Bulatov and David Mitchell for being great supervisors and mentors. David has been strong guidance through out my MSc studies. He was patient with the unconventional research avenues I took. Andrei was an idle supervisor who made sure that I focused on the problem and gave invaluable insight that shaped my approach to the problems that arose in my research. He gave me the freedom to develop my own direction and writing style while teaching me a great deal of his own. I was fortunate doing my thesis under their supervision.

I was also very fortunate to learn from and work with Eugenia Ternovska. During my conversations with her, I was introduced to Finite Model games, which contributed a lot to my understanding of the essence of expressiveness. Her support throughout my research, intellectually and financially, served to carry my work farther than would have been possible otherwise.

Many other people have taught me and steered me to the right direction with their discussion (in person or by email). My discussions with Evgeny Skvortsov about the power of local search made the initial steps possible. Neil Immerman showed me that the approach I took to the initial problem I was working on was flawed and needed tweaking. Gabor Tardos, with his strong insight in graph theory, pointed me in the tools I needed to approach my problems.

There are of course many people at SFU research community that influenced me academically. I'd particularly like to thank Valentine Kabanets, Sasha Fedorova, Petra Berenbrink, and Joseph Peters.

I wish to express my gratitude to my friends, colleagues and administrative staff at computing science department as SFU: Faraz Hach, Raheleh Mohebali, Tiffany Nguyen, Banasheh Noohi, Funda Ergun, Gerdi Snyder, and Val Galat.

Finally, I want to thank my family for their love and support. Without them, this could not have been accomplished.

Contents

Chapter 1

Introduction

The Boolean satisfiability problem (SAT) is one of the most studied problems in computer science. It has attracted much attention in recent years. Nowadays, more problems are being solved faster by SAT solvers than any of its competitors (for example, integer and linear programming solvers). Many problems originating from artificial intelligence, computer engineering, software verification, graph theory, operations research, and scheduling have various translations to SAT [53, 40, 39, 38, 54]. Due to this flexibility, many mathematical tools developed in these areas are now available to further improve the performance of SAT-solvers. SAT is by far the most natural of mathematical models for operations done with computers. Modeling problems with SAT has caused many advancements in SAT to be translated to other areas such as QBF reasoning and Pseudo-Boolean solvers [7, 17]. Many advancements in other fields like statistical physics [48, 55] are also being incorporated to improve the efficiency of SAT-solvers.

1.1 History and Development

Development of SAT is best understood with respect to its logic roots. Logic, as a science, was first invented by Aristotle (384-322 B.C.). One of the applications of SAT is to show that propositions $p_1, \cdots, p_n$, with certain restrictions, do or do not logically imply proposition q. Consider the statement:

$$\text{Some } A \text{ is } B \wedge \text{Some } C \text{ is } B \rightarrow \text{every } A \text{ is } C.$$

Aristotle would give an example to show that the implication is false. For example, statements "some student is computer savvy" and "some professor is computer savvy" are both true, but every student is a professor is false. In the language of Logic, we say that the set

$$\{\text{some } A \text{ is } B,\ \text{some } C \text{ is } B,\ \neg(\text{every } A \text{ is } C\ )\ \},$$

is satisfiable. Stoics (300s-200s BC) developed a sophisticated language using symbols similar to $\wedge$, $\vee$, and $\neg$. In the language developed by the Stoics, a proposition is the meaning of a sentence that expresses it. For example, the sentence "some student is computer savvy" is a proposition. Using this language, they discovered propositional inferences that have been part of Logic ever since.

Before the 19th century, many logicians, among which Descartes, Leibniz, and Boole are most notable, tried to develop algebraic languages to express reasoning. However, the semantics of these languages were imprecise; Therefore rigorous reasoning systems could not be developed through them. Some mathematicians were relatively successful. For example, Giuseppe Peano axiomatized part of set theory, which we now call Peano arithmetic. It was not until mid 19th century that a mathematician by the name of Gottlob Frege (1848-1925) tried to expand the meaning of reasoning. He started a project called logicism. Its primary goal was devising a mechanism by which one could deduce not only Logic but "all of mathematics". The realms of mathematics he was concerned with were set theory, number theory, and analysis. The project was proven to be a failure. However, the contributions made delineating the boundaries of "formal reasoning" through this project were grand [5].

Formal reasoning can be thought of as a set of rules that will decide based on a given set of assumptions, whether a logical statement is true or false. A classic example of logical syllogism concerning Socrates can be found in the following deduction:

$$\text{All men are mortal} \wedge \text{Socrates is a man} \rightarrow \text{Socrates is mortal}$$

Here, the assumptions are: All men share a common property of being mortal, and Socrates is a member of the set of all men. From this, we concluded that Socrates is mortal. The set of rules that decides whether Socrates is mortal from those assumptions can be thought of as a computer's reasoning. Being able to come up with a finite set of rules that can potentially deduce all of mathematics is what logicism tried to accomplish. Any hope for logicism was shattered when Kurt Gödel astounded philosophers, logicians, and mathematicians by proving that there are logically determined outcomes that are not derivable by any finite set of rules. However, as it was proven by Gödel himself, for a subset of mathematics, first order logic, any logically determined outcome is derivable from a finite set of rules.

1.1.1 Modern definition of satisfiability and its algorithms

By definition, decidable problems are problems that can be solved by a sequence of instructions in a finite number of steps. As Logic developed in the 20th century, the importance of effective decidability increased. The importance of the decidability of a problem was becoming apparent, and the lack of any formal definition for "process" was unsettling. Defining a process itself became a research goal and motivated the invention of the Turing machine as a mathematical model for solving problems. Around the same time, Alfred Tarski enunciated the notion of satisfiability for first-order logic [50, 51]. Tarski coined the term *satisfaction* to refer to the truth relative to an interpretation of Boolean variables. We will use the term assignment instead of interpretation, and we will use the terms Boolean variables and propositions interchangeably. Essentially, Tarski defined a "satisfiable propositional formula" to be a formula for which there is an assignment to variables that makes it true. This was the first definition of SAT in its modern form.

In 1937, Claude Shannon defended his master's thesis titled *A Symbolic Analysis of Relay and Switching Circuits* [49]. Shannon provided many logical statements and stated that they can be used to achieve minimal circuit representations for a problem he was considering with regard to switches on communication channels. But he did not offer a systematic way of doing this. While SAT was historically seen as a systematic way of using information to deduce statements, Shannon saw it as a possibility for modeling communication channels and simplifying them. Howard Gardner, a prominent psychologist and founder of the theory of multiple intelligence, calls this thesis "possibly the most important, and also the most famous, master's thesis of the century" [5]. This is partly due to the fact that it was the first time a practical application of SAT was discovered. This breakthrough led researchers to focus on efficiently solving large SAT instances.

Meanwhile, computers were developing rapidly, and research in the field of automated deduction was on the rise. In the beginning, researchers did not consider the amount of time and space their proposed algorithms for automated deduction required to solve complex problems, and this lead to huge failures [5]. An improvement was introduced when Davis and Putnam proposed using CNF, *conjunctive normal form*, for satisfiability testing [13]. CNF, over set of variables $\{x_1, \cdots, x_n\}$ is an encoding of SAT where formulas are conjunction of clauses $C_1, \cdots, C_m$, where each clause C_i where $1 \leq i \leq m$ is a disjunction of *literals*. A literal is either a variable x_i for $1 \leq i \leq n$, or its negation (i.e. $\neg x_i$). This encoding of SAT is what we use in this work. In their manuscript [13], Davis and Putnam provided all the essentials for modern DPLL SAT solvers [14, 12].

As heuristics were being developed, due to the availability of the Turing machine as a mathematical framework, mathematicians were working on theoretically efficient algorithms in various problems. In 1960, Jack Edmonds coined the term "good algorithm" to mean an algorithm that executes a polynomial number of steps with respect to the size of the input [16]. A problem is said to be in NP, if and only if there is a nondeterministic Turing machine that halts in a polynomial number of steps with respect to the size of every instance of the problem, and solves that problem. Stephen Cook showed that SAT can capture the hardness of all problems in NP [10]. That is, every instance of a problem P in NP can be reduced with a polynomial number of steps to a SAT instance with size polynomial in the size of the original instance of P. The problems that capture the complexity of the class NP are referred to as NP-complete. A year after Stephen Cook published his landmark paper on the NP-completeness of SAT, Karp showed there are other NP-complete problems by demonstrating that SAT is reducible to them in polynomial time [31]. Among these problems was the version of SAT in which every clause contains only 3 literals. This problem is called *3-SAT*. Some years later, Garey and Johnson also published their famous book [21] that contained other NP-complete problems. Among the problems mentioned by Garey and Johnson was the version of 3-SAT in which a satisfying assignment should set at least one literal in each clause to false, i.e. a satisfying assignment should set at least one literal to true, and at least one literal to false in each clause. This problem is called *NAE-3-SAT*. The problem that is the focus of this work is NAE-E3-SAT. In this problem, variables appear only positively (unnegated) in clauses, and an assignment to variables in every clause should satisfy the conditions of NAE-3-SAT. This problem is also NP-complete since NAE-3-SAT is reducible to it within polynomial number of steps (see, e.g. [23]).

1.1.2 Complete Algorithms

One class of algorithms for SAT is concerned with the guarantee of an answer (satisfiable or unsatisfiable) upon completion. Different approaches are considered within this class of algorithms. One approach is a systematic search in the space of possible truth assignments. That is, checking every possible set of assignments to variables in a systematic way to find a satisfying one. The dominating class of solvers that use this approach are DPLL based solvers. Another approach is to apply a simple, and complete set of inference rules on the SAT instance until either contradiction is reached and we conclude that the instance is unsatisfiable, or no application of these inference rules is possible and we conclude the instance is satisfiable. The last approach discussed here is the combination of search and inference, which is a fundamental part of all modern complete SAT-solvers.

The SAT-solvers that are called DPLL solvers are descendants of the algorithm devised by Davis, Putnam, Logemann, and Loveland [12, 13]. Imagine a search tree that is specified by a decision for a variable at every branch point, and consequently its leaves are truth assignments. It is obvious that every leaf can be reached from the root by only one sequence of decisions, and the number of these decisions, i.e. the depth of the tree, is the same as the number of variables. In order to optimize our search for a solution we should consider the fact that after each decision, the CNF has the potential to be simplified by either removing a clause, which means fewer conditions to consider, or reducing the search space; meaning that one clause is found to be unsatisfied by the partial assignment and therefore that branch is eliminated. After every decision, some clauses have the potential of becoming unit (i.e. only one literal left undecided in that clause, and the rest are decided to be false) and the literal in a unit clause should be assigned true. Therefore, we build the search tree as we go along, and the decisions are made according to which clause has become unit and what decision at that branch point produces a partial assignment that satisfies the unit clause. This technique is called “unit resolution” or “unit clause propagation”, and is one of the most important techniques behind systematic search-based algorithms. Another technique to add to the systematic search based SAT-solvers toolbox, which can also be used in inference based algorithms, is pure literal elimination. This is essentially eliminating unipolar variables from the instance. Pure literal elimination along with unit clause propagation are among the most effective techniques used in DPLL SAT-solvers.

The idea behind the inference based approach is *resolution*. It was first introduced by Blake in 1937 [6] and completed by Quine in 1955 [52]. Let x be a Boolean variable, and suppose that ϕ is a CNF formula which contains clauses C_i and C_j. If C_i contains x and C_j contains $\neg x$, we can derive the clause $C = C'_i \vee C'_j$, where C'_i is a clause produced by removing x from C_i and C'_j is a clause produced by removing $\neg x$ from C_j. The formula ϕ' is a new CNF formula with C added. If an empty clause is obtained, the CNF is not satisfiable. Unfortunately, resolution is not complete, i.e., it is not guaranteed to derive every clause that is implied by ϕ. However, resolution is *refutation complete*, i.e., it derives the empty clause if the given ϕ is unsatisfiable. The idea behind inference based algorithms is to recursively apply resolution until no application of resolution is possible, in which case we can conclude the instance of SAT is satisfiable, or an empty clause is reached, and we can conclude that the instance of SAT is unsatisfiable.

Modern complete SAT-solvers incorporate the ideas of search based solvers and inference based solvers to achieve ideal results. One of the most important ideas involving the combination of these approaches is *clause learning*. The idea came from the realization that certain partial assignments are sufficient to cause a conflict. By remembering these assignments, the solver would not fall back to the same conflict. It has been shown that clause learning reduces the search time by an exponential factor [26]. By remembering a new clause, we essentially add to the space complexity of the problem. The question researchers are trying to answer is when to add each clause to optimize this trade off [45].

One of the motivating factors for researchers to look beyond DPLL based algorithms came from the work of Mitchell et al. [37]. They observed that DPLL based solvers perform quite poorly on certain randomly generated instances. They also observed that a key trait that would characterize this hardness is clause to variable ratio, typically denoted by α. They demonstrated a now well known easy-hard-easy transition of randomly generated SAT instances with respect to α. The hard region has shown experimentally to be around $\alpha = 4.26$ [36]. Discovery of this apparent hard region initiated a research on both *phase transition phenomena* (see Section 1.2) and looking at new approaches of solving SAT, i.e. incomplete algorithms. We review briefly what these approaches are, and refer the reader to Cook and Mitchell [11] for a detailed survey on early works on incomplete algorithms.

1.1.3 Incomplete Algorithms

It is apparent from the name *incomplete algorithms* that these methods provide no guarantee of a satisfying assignment or a report of unsatisfiability. Clearly, they are biased toward satisfiable side because they will run for a limited time and either they reach a solution and or report failure. They never output unsatisfiable as they lack the ability to reach that conclusion. These algorithms are generally based on stochastic local search (SLS). There have also been attempts to combine DPLL and SLS approaches to get the best of both worlds. We do not consider these techniques. The reader is referred to [24, 35, 42, 43] for further reading on the subject.

Two methods that played a key role in the development of SLS solvers were GSAT and WalkSAT. The GSAT algorithm was first introduced by Selman et al [47]. The idea behind GSAT is iteratively improving the assignment. The improvement is usually the best flip, i.e. flipping the assignment to the propositional variable that will cause the maximum number of satisfied clauses. There are also "sideway" moves, i.e. moves that retain the number of satisfied clauses. The algorithm terminates either by reaching a satisfying assignment, or by reaching the maximum number of steps. This was a huge success as it outperformed the best backtracking search algorithm of the time. However, experiments showed that GSAT spends most of its time on *plateaus* [20], i.e. sequence of truth assignments that are connected together by a sequence of possible sideways moves. Some subtle modifications to GSAT were introduced to create some noise that would eliminate plateaus, leading to the development of WalkSAT [46]. The algorithm WalkSAT first picks a variable at random among unsatisfied clauses. If flipping a variable does not turn any of the satisfied clauses to an unsatisfied one, it flips this variable. Otherwise, it flips a random literal. This means that every variable, even a variable that reduces the number of satisfied clauses if flipped, has the potential of being chosen to be flipped. This seemingly simple idea turned out to be very beneficial [28]. Further modifications to WalkSAT, and an adaptive noise mechanism was later introduced by Holger Hoos in the algorithm Adaptive Novelty+ [27]. Adaptive Novelty+ is among the best algorithms for solving satisfiable random SAT instances.

1.2 Random SAT

We say that a 3-SAT formula ϕ has size m iff there are m clauses in formula ϕ. There are two main distributions we consider for generating random 3-SAT formulas. In the first, the formula is chosen uniformly from all 3-SAT formulae of size αn, where n is the number of variables and α is a positive number called the *order* of the formula. This distribution is denoted by $\mathbb{U}_{\alpha,n}$. In the second distribution, the formula is generated by including every 3-SAT clause over n variables with probability p in it. The constant p is called the *density* of the formula. We will focus more on former in this section.

1.2.1 Thresholds

Going back to the results of Mitchell et al [37] in early 90s, we mentioned that it instigated the research on the phenomenon of phase transitions. One of the Random 3-SAT distributions they looked at was $\mathbb{U}_{\alpha,n}$. Mitchell et al. provided experimental evidence that solving randomly generated 3-SAT is relatively easy in the region up to $\alpha = 4.26$ threshold, and then it becomes sharply harder at the threshold, quickly becoming easier at larger densities. Mitchell et al. conjectured that there is a threshold c, such that for every $\alpha > c$, the probability that there is a solution for a randomly generated SAT formula by distribution $\mathbb{U}_{\alpha,n}$ goes to 0 as $n \to \infty$. In this case, we say that *with high probability*, there is no solution to randomly generated SAT formula. Furthermore, they conjectured that with high probability there is a solution for randomly generated 3-SAT instances with $\alpha < c$. Usually, c is referred to as the satisfiability threshold. Finally, Friedgut proved that a sharp satisfiability threshold exists for randomly generated SAT (k-SAT) formula, i.e. there is a c such that for every $\epsilon > 0$, formulas generated by $\mathbb{U}_{c+\epsilon,n}$ have no solution with high probability and formulas generated by $\mathbb{U}_{c-\epsilon,n}$ have a solution with high probability. He thereby confirmed the conjecture made by Mitchell et al. [37]. However, Friedgut's result left open the possibility that satisfiability threshold is a function of n that tends to a constant limit as $n \to \infty$. The problem of finding the satisfiability threshold is still open. For now, the focus of research is proving tighter upper and lower bounds for the satisfiability threshold.

One way of getting close to the satisfiability threshold is by proving upper bounds, i.e. finding a value for α, such that with high probability a randomly generated SAT formula is unsatisfiable. The most popular way of proving upper bounds is by resolution, i.e. by proving that the probability that the shortest resolution proof is polynomially bounded goes to 0 as $n \to \infty$. The first of these results was obtained by Franco and Paull [19]. They showed that a random k-SAT formula is unsatisfiable with high probability if $\alpha > -\frac{\ln(2)}{\ln(1-2^{-k})}$. This translates to $\alpha > 5.19089307$ for 3-SAT [19]. The upper bound was further improved to 4.598 [32] and later to 4.506 [15].

Another way is to improve the lower bound on the threshold. These results are obtained by devising algorithms for SAT that would find a satisfying assignment for a CNF of certain α ratio with high probability. The first of these results was Chao and Franco's algorithm in 1986 called UC (unit clause propagation) [8]. They showed that UC finds an assignment with high probability given $\alpha < 8/3$. Combined with voting (discussed in section 1.2.2) and choosing two variables at a time, Achlioptas improved the lower bound to $\alpha = 3.145$ [1]. The rest of the results included incorporation of sophisticated spectral and greedy techniques. The best known lower bound is currently $\alpha = 3.52$ [25, 29, 30].

1.2.2 Hard satisfiable instances and voting

As far back as the early 90s, the idea of generating hard satisfiable instances has attracted much attention. Among the distributions considered by researchers were the uniform distribution $\mathbb{S}_{\alpha,n}$, where a satisfiable formula of size αn is chosen at random, and the uniform distribution $\mathbb{P}_{p,n}$ (or planted 3-SAT) where an assignment τ is chosen at random. A satisfiable formula is constructed by including every clause satisfied by τ in the formula with probability p. There is no theoretical complexity result connecting the unsatisfiability threshold to hardness. But how do we prove that the satisfiable formulas whose density surpasses the satisfiability threshold are not hard; How do we solve these hard instances? One idea is to give each variable a certain value by looking at the polarity of its appearances. If the majority of its appearances are positive, then it is voted to be true, and it is voted to be false if the majority of its appearances are not positive. We call this process *voting*. Building on the idea of voting, Ben-Sasson, Bilu, and Gutfreund [4] proved that an instance generated according to distribution $\mathbb{S}_{\alpha,n}$ where $\alpha \geq \Theta(n \log n)$, can be solved by a simple voting algorithm with high probability in polynomial time.

Another approach to solving hard satisfiable instances was introduced by Flaxman [18]. He looked at a more general distribution, $\mathbb{I}_{n,p_1,p_2,p_3}$, where an assignment is chosen at random and then a random formula constructed by including each that has i of its literals satisfied in the formula with probability p_i. Note that by setting $p = p_1 = p_2 = p_3$, the distribution $\mathbb{I}_{n,p_1,p_2,p_3}$ is the same as $\mathbb{P}_{p,n}$. Another example is a random distribution for NAE-3-SAT problem by setting $p_3 = 0$, and $p_1 = p_2$. In 1992, Papadimitriou and Koutsoupias conjectured that there is an algorithm that solves instances from $\mathbb{P}_{n,p}$ with high probability in polynomial time, where $p \geq \frac{d}{n^2}$ for some large constant d [33]. Flaxman devised a highly sophisticated algorithm using spectral techniques and proved that this algorithm finds a satisfying assignment for the formula generated by $\mathbb{I}_{n,p_1,p_2,p_3}$ with high probability in polynomial time, given some constraints on p_1, p_2, and p_3. The restriction $p_1 = p_2 = p_3 = \frac{d}{n^2}$ satisfies the assumptions of the proof provided d is large enough. Therefore, Flaxman's result proved the conjecture made by Papadimitriou and Koutsoupias.

The idea of voting for lower densities $p = \frac{d}{n^2}$ was considered by Krivelevich and Vilenchik.

They showed that highly sophisticated spectral techniques are not required to solve a planted 3-SAT with high probability in polynomial time. They came up with a simple algorithm [34] based on voting that would achieve Flaxman's result for distribution $\mathbb{P}_{n,p}$, where $p \geq \frac{d}{n^2}$ for some large constant d.

1.3 Results

Planted 3-SAT and similar planted problems have an inherent asymmetry in them. In the case of 3-SAT, the asymmetry is due to the difference in the expectation of number of positive and negative appearances of variables given their planted value. For the planted 3-SAT, a simple counting argument shows that if x is set to true in the planted assignment, the probability that it appears positively in a random satisfied clause is $4/7$, and the probability that it appears negatively is $3/7$. Therefore, we expect to get a "good" (close enough to the planted one) assignment by voting. This asymmetry is why voting works very well on 3-SAT.

What if we consider a problem that is completely symmetric in its planted assignment? NAE-E3-SAT is inherently symmetric since for any assignment with variable x set to true, there is an assignment with variable x set to false (the complement of the planted assignment). Answering this question is the concern of this work.

An important contribution of this thesis is introducing a new way to break the symmetry in inherently asymmetric problems. The word symmetric is relative in a sense that it corresponds to a harmonious behavior on some specific property of the solution. For example, in NAE-3-SAT, there is a symmetry in solution space corresponding to a single variable, i.e. looking at a single variable, there is a symmetry in its value with respect to the solution. Therefore, we cannot give preference to $\top$ (true) or $\bot$ (false) assignments no matter what the NAE-E3-SAT formula is. In other words, we cannot use the structure of the instance to get information on whether the value of a single variable is true or false in the planted solution. Note that we only talk about the assignment of a single variable in the solution. Any other trait of the solution may have asymmetry in it. To get some intuition, in the 3-Colorability problem, while it is true that for any planted color assignment that has node x set to Green, there is an assignment that has x set to Blue (or Red), if we look at two nodes x and y depending on the structure of graph (instance), we can get information on whether the two colors are the same or different. For example if they are adjacent we know that their colors should be different.

The problem that we tackle in this work is NAE-E3-SAT. As for NAE-3-SAT, the symmetry in the problem is that for every planted assignment π that assigns the variable x to $\top$, there is a satisfying assignment π' that sets x to $\bot$. The way this symmetry is broken is by considering two variables together. The number of appearances of two variables in the same clause give us information on the equality or inequality of the two variables. In order to clearly comprehend this information, we need to introduce the formal definition of NAE-E3-SAT problem.

Definition 1 (NAE-clause, NAE-formula, NAE-E3-SAT problem)**.** *A NAE-clause C over set of variables $V = \{x_1, x_2, \ldots, x_n\}$ is a triple of form (x_i, x_j, x_k) where $x_i, x_j, x_k \in V$ and it evaluates to true with respect to assignment π if and only if $\pi(x_i), \pi(x_j)$, and $\pi(x_k)$ are not all equal.*
NAE-formula ϕ over set of variables $V = \{x_1, x_2, \ldots, x_n\}$ is conjunction of NAE-clauses $C_1, C_2, \ldots C_m$, $C_i \neq C_j$ for $1 \leq i, j \leq m$, and it evaluates to true with respect to assignment π if and only if every clause C_i, $1 \leq i \leq m$, evaluates to true with respect to π.
NAE-E3-SAT(ϕ) is the problem of deciding whether there is an assignment π such that NAE-formula ϕ evaluates to true with respect to π.

A planted NAE-E3-SAT instance ϕ over distribution $\mathbb{P}^{\mathsf{NAE}}_{p,n}$ is generated by first picking at random a truth assignment τ to n variables $x_1, x_2, \ldots, x_n$. Next, each NAE-clause that evaluates to true with respect to τ is included in the formula with probability p.

Using the same approach as Ben-Sasson et al [4], we devise an algorithm for solving (i.e. finding a satisfying Boolean assignment for) planted NAE-E3-SAT problem instances over distribution $\mathbb{P}^{\mathsf{NAE}}_{p,n}$, where $p \geq \frac{d \log n}{n}$. This is made possible due to the fact that the expected number of appearances of two variables in clauses, given that in a planted assignment they are assigned to the same value, is at most $\frac{2}{3}pn$, and the expected number of appearances of two variables in clauses, given that in a planted assignment they are assigned to a different value, is pn. This discrepancy means we can vote a pair of variables to be equal or not equal depending on the number of their appearances together. We show that with high probability every relationship assigned by our devised algorithm to pairs of variables is correct. It is easy to see that every inequality is equivalent to $(\neg x \vee \neg y) \wedge (x \vee y)$, which are two 2-SAT clauses. The same can be said about equalities. The 2-SAT formula can therefore be constructed from the relationship given to pairs. We can conclude that any assignment that satisfies the 2-SAT clause satisfies the inequality (or equality) and vice versa. Therefore, w.h.p, the algorithm generates a 2-SAT formula that has the same satisfying assignment as the NAE-E3-SAT instance. The process of generating the 2-SAT formula from the NAE-E3-SAT resembles voting. We call this process pseudo-voting. From here on, we abbreviate $\mathbb{P}^{\mathsf{NAE}}_{p,n}$ to $\mathbb{P}_{p,n}$ in this work.

Theorem 2. *A pseudo-voting algorithm solves NAE-E3-SAT in polynomial time with high probability over distribution $\mathbb{P}_{p,n}$ where $p \geq \frac{d \log n}{n}$, and d is a sufficiently large constant.*

For the case of $p \geq \frac{d}{n^2}$, using a similar approach to Krivelevich and Vilenchik [34], we start with an initial assignment and improve it to get the satisfying assignment to the NAE-E3-SAT formula. Due to the small density, the difference between the expected number of appearances of two variables in clauses, given that in the planted assignment they are assigned to the same value, and expected number of appearances of two variables in clauses, given that in the planted assignment they are assigned to different values, asymptotically (as $n \to \infty$) approaches 0. Krivelevich and Vilenchik used voting to get an initial assignment, and improved it to get a satisfying assignment. For low density NAE-E3-SAT, $p \geq \frac{d}{n^2}$, the expected number of appearances of any pair approaches 0 whether they are assigned to the same value by the planted assignment or not. Therefore, pseudo-voting alone is not effective. An important contribution of this work is devising an algorithm that uses approximation algorithms to generate an initial assignment.

Approximation algorithms are algorithms that find approximate solutions to optimization problems. Unlike heuristics, approximation algorithms have a proven efficient running time and an *approximation factor*. The approximation factor associated with an approximation algorithm is a guarantee that the solution is optimal up to that factor. An algorithm that has an approximation factor a is called an a-approximation algorithm. To understand this better, let us consider NAE-E3-SAT; As we mentioned, NAE-E3-SAT is an NP-complete problem. An optimization version of NAE-E3-SAT is the problem of finding an assignment to a NAE-formula, that satisfies greater than or equal to the number of clauses that any other assignment satisfies. We call this problem MAX-NAE-E3-SAT (ϕ). The best known approximation factor for MAX-NAE-E3-SAT is $\frac{3}{2\pi} \arccos(\frac{-1}{3}) \simeq 0.91226$ [56], i.e., if the maximum number of clauses satisfiable by any assignment is max, then the algorithm guarantees that it will find an assignment that satisfies greater than or equal to $0.91226 \times \max$ clauses. Building on the idea of voting, we use an algorithm that guarantees a slightly weaker approximation factor for MAX-NAE-E3-SAT, i.e. approximation factor of 0.8785. This approximation is obtained by applying the approximation algorithm of Goemans and Williamson [22] to the translation of the NAE-E3-SAT instance to a MAX-CUT instance. We show that the initial assignment obtained has the potential to improve to the planted solution.

Theorem 3. *There is an algorithm that is a modified version of pseudo-voting and solves NAE-E3-SAT with high probability, and runs in polynomial time over distribution $\mathbb{P}_{p,n}$ where $p \geq \frac{d}{n^2}$ and d is a sufficiently large constant.*

Two different algorithms are used to prove Theorem 2 and Theorem 3. However, for obvious reasons the algorithm used for Theorem 2 is faster and easier to analyze.

1.4 Preliminaries

Before we start proving Theorem 2 and Theorem 3, it is best to introduce several standard tools that we use throughout this work:

Definition 4 (With high probability (whp))**.** *Let n be the size of the random instance. An event A occurs* with high probability *(whp) iff*

$$\Pr[A] = f(n), \ \textit{and}$$
$$\lim_{n->\infty} f(n) = 1,$$

where $f(n)$ is some function of n.

In probability theory, the union bound says that for a set of events, the probability that at least one of the events in the set happens is at most the sum of the probabilities of individual events. More formally:

Theorem 5 (Union bound, also known as Boole's inequality [?])**.** *Let $S = \{A_1, \ldots, A_m\}$ be the finite set of events A_i, for $1 \leq i \leq m$, and $m \geq 1$. We have*

$$\Pr\left[\bigcup_{A_i \in S} A_i\right] \leq \sum_{A_i \in S} \Pr[A_i].$$

Chernoff bounds, named after Herman Chernoff [9], are extremely useful inequalities in computer science. They are applied to a class of random variables and give an exponential fall-off of probability with distance from the expectation. We need the following definition to understand the class of random variables Chernoff bounds apply to:

Definition 6 (Indicator random variable)**.** *An* indicator random variable *associated with an event A, X_A, is a random variable that takes the value* 1 *if the event A occurs and takes the value* 0 *otherwise.*

There are many versions of Chernoff bounds. We use the following:

Theorem 7 (Chernoff bounds)**.** *Let $X_1, \ldots, X_n$ be independent indicator random variables such that $\Pr[X_i = 1] = p$, for some constant $0 < p \leq 1$. Let random variable $X = \sum_{1 \leq i \leq n} X_i$, and $E[X] = \mu$. The following inequalities hold:*

$$\Pr(X \geq (1+\delta)\mu) \leq e^{-\delta^2 \mu}, \quad \delta > 0,$$
$$\Pr(X \leq (1-\delta)\mu) \leq e^{-\delta^2 \mu}, \quad 0 < \delta < 1.$$

Chapter 2

Density of at least $p \geq \frac{d \log n}{n}$

Before we start analyzing the algorithm that we used to prove Theorem 2, we offer some intuition on how the algorithm was designed and define the terminology needed. In this section we assume instances are generated by $\mathbb{P}_{p,n}$ where $p \geq \frac{d \log n}{n}$.

Algorithm LDensity constructs a "2-SAT" formula that has a boolean assignment that is whp the planted assignment of the NAE-E3-SAT formula given as an input to the algorithm. To accomplish this, the algorithm adds a relationship (inequality or equality) to the "2-SAT" formula for every pair of variables. This relationship, as described in previous section, is set by the number of occurrences of those variables together.

Algorithm 1 LDensity(ϕ)

```
1: $\psi = \emptyset$
2: for all $x, y \in$ Variables do
3:     set $Y_{x,y}$ = number of co-occurrences of $x$ and $y$ in formula $\phi$
4:     if $Y_{x,y} \geq \frac{5}{6}pn$ then
5:         $\psi = \psi \wedge (x \neq y)$
6:     else
7:         $\psi = \psi \wedge (x = y)$
8:     end if
9: end for
10: $\tau$ = 2-SAT($\Psi$)
11: return $\tau$
```

2.1 Intuition

The algorithm's ultimate goal is to find an assignment that satisfies all of the clauses whp. Pairs of variables that have different truth assignments appear in more clauses together than pairs of variables with the same truth assignments. In fact the expectation of the number of their appearances together differs by at least $\frac{1}{3}pn$. For a random assignment whp the number of variables assigned true is almost the same as the number of variables assigned false. Therefore, by counting the number of their appearances together in NAE-clauses we can decide whether they have the same or different values with high probability. By considering every pair of variables, the algorithm constructs a 2-SAT formula that has the same solution as the planted NAE-E3-SAT formula. This is very similar to the algorithm proposed by Ben-Sasson et el [4]. The important distinction here is that since voting (considering the number of positive occurrences of a variable in contrast to its negative occurrences) has no meaning here, we cannot rely on voting to find the planted solution whp. We apply voting on pairs of variables and instead of true and false, we give the pairs $=$ and $\neq$ values.

2.2 Correctness

In order to prove Theorem 2, we need to show that all of the relationships that the algorithm assigns within pairs are true with respect to the planted solution to NAE-E3-SAT formula ϕ. Furthermore, since the algorithm includes a relationship for every pair, there can only be one pair of assignments (a solution and its complement are both solutions) that satisfies all the relationships. Therefore, it is in fact the solution to NAE-E3-SAT. By showing every "2-SAT" relationship derived evaluates to true with respect to the planted assignment, we can conclude that the solution to the "2-SAT" formula constructed is also a solution to the NAE-E3-SAT formula. The relationships that the algorithm assigns to variables are not really 2-SAT clauses. They are equalities of the form $(x = y)$, or inequalities of the form $(x \neq y)$. The following translation would change them into a set of 2-SAT clauses:

$$(\tau(x) \neq \tau(y)) \equiv (x \vee y) \wedge (\neg x \vee \neg y)$$

$$(\tau(x) = \tau(y)) \equiv (x \vee \neg y) \wedge (\neg x \vee y).$$

If we prove that the 2-SAT formula has the same solution as planted NAE-E3-SAT formula, since 2-SAT has a polynomial time algorithm [41], Theorem 2 follows.

Proposition 8. *For constant $d \geq 109$ and $p \geq \frac{d \log n}{n}$, algorithm LDensity finds a solution to the random formula ϕ chosen by distribution $\mathbb{P}_{p,n}$ whp.*

Before proving the proposition, we need to show that the number of variables assigned $\top$ is almost the same as the number of variables assigned $\bot$. The following lemma proves this. Let τ be the planted solution and let X be a random variable counting the number of variables x such that $\tau(x) = \top$

Lemma 9. *For every $0 < \epsilon \leq \frac{1}{2}$, whp, $(\frac{1}{2} - \epsilon)n \leq X \leq (\frac{1}{2} + \epsilon)n$.*

Proof. Given that

$$E[X] = \frac{n}{2}.$$

By Chernoff bound,

$$\Pr[X > (\frac{1}{2} + \epsilon)n] \leq e^{-2\epsilon n},$$

$$\Pr[X < (\frac{1}{2} - \epsilon)n] \leq e^{-2\epsilon n}.$$

□

Since we have proven that the number of $\top$s and $\bot$s are distributed almost evenly between variables, we are now able to bound the expected number of times the variables appear together given that they have the same truth assignment. We will show that the expected number of times two variables with different truth assignment appear together is significantly larger than the expected number of times two variables with the same truth assignments appear together in clauses. Given this difference, the algorithm can distinguish between equal and unequal pairs, and Proposition 8 follows.

Proof. (of Proposition 8) Let $Y_{x,y}$ be the number of co-occurrences of x, y in clauses of NAE-E3-SAT instance ϕ. The conditional expectation of $Y_{x,y}$, given that $\epsilon \leq \frac{1}{6}$ is as follows:

$$E[Y_{x,y} | \tau(x) \neq \tau(y)] = pn - o(1),$$

$$E[Y_{x,y} | \tau(x) = \tau(y)] \leq (\frac{1}{2} + \epsilon)pn \leq \frac{2}{3}pn.$$

The algorithm LDensity may work incorrectly if for some variables x,and y such that $\tau(x) = \tau(y)$, it decides that the variables have different values, or when $\tau(x) \neq \tau(y)$ it decides that they have the same value. We will show that this does not happen whp. Let R denote the event that the algorithm makes a wrong decision about some pair of variables. We show that $\Pr[R] \leq n^{-1/54}$.

Let $R_{x,y}$ be the event that we choose a wrong relationship for x and y (equality or inequality) and consequently wrong clauses to add to ψ. Assume first that x and y have the same assignment in the planted solution τ. By Chernoff bound we have

$$\begin{aligned}\Pr[R_{x,y}] \leq \Pr[Y_{x,y} \geq \frac{5}{6}pn] &\leq \exp\left\{-\left(\frac{1}{6}\right)^2 \frac{2}{3}pn\right\}\\ = \quad \exp\left\{-\frac{1}{54}pn\right\} &\leq \exp\left\{-\frac{1}{54}109\log n\right\} = n^{-109/54}.\end{aligned}$$

If x and y have different values in τ, the probability of $R_{x,y}$ can be evaluated in the same way.

By the union bound $\Pr[R] \leq \sum_{x,y} \Pr[R_{x,y}] \leq n^2 \times n^{-109/54} = n^{-1/54}$. We have assumed that $\epsilon \leq \frac{1}{6}$, and by Lemma 9 we can now assume $(\frac{1}{2}-\epsilon)n \leq X \leq (\frac{1}{2}+\epsilon)n$. Therefore, $\Pr[R \text{ or } \big((\frac{1}{2}-\epsilon)n > X \text{ or } X > (\frac{1}{2}+\epsilon)n\big)]$ is greater than the probability that the algorithm does not find a correct 2-SAT formula.

$$\begin{aligned}\Pr[R \text{ or } X > \big((\tfrac{1}{2}-\epsilon)n \text{ or } X > (\tfrac{1}{2}+\epsilon)n\big)] &\leq \Pr[R] + \Pr[\big((\tfrac{1}{2}-\epsilon)n > X \text{ or } X > (\tfrac{1}{2}+\epsilon)n\big)]\\ &\leq n^{-1/54} + 2e^{-\frac{1}{3}n}.\end{aligned}$$

As n grows, both $e^{-\frac{1}{3}n}$ and $n^{-1/54}$ approach 0. $\square$

There are at most n^2 pairs of variables x, y. Therefore the algorithm runs in polynomial time. The probability that we get any of the relationships within pairs wrong approaches 0 as n goes to infinity. This concludes the proof of the Theorem 2.

Chapter 3

Density of at least $p \geq \dfrac{d}{n^2}$

In this section we assume that an instance is chosen from the distribution $\mathbb{P}_{p,n}$. The algorithm LDensity works quite well with denser instances. However, we can easily conclude from the previous section that for density $p < \frac{\log n}{n}$ algorithm LDensity may wrongly determine values of variables. Moreover if $p \leq \frac{1}{n}$, the expected value of $Y_{x,y}$, the number of co-occurrences of x, y in clauses of NAE-E3-SAT instance ϕ is as follows:

$$E[Y_{x,y}|\tau(x) \neq \tau(y)] \geq (n-2) \times \frac{d}{n^2},$$

$$E[Y_{x,y}|\tau(x) = \tau(y)] \leq (\frac{1}{2} + \epsilon)n\frac{d}{n^2}.$$

This means that whp, the two variables x and y do not both appear in any clause. Therefore, counting the number of appearances of x and y does not make sense. We will discuss in reasonable detail how we can change the algorithm to work for smaller density.

Before we introduce the algorithm, we need some definitions. In this section we introduce some well known problems and their connection to NAE-E3-SAT.

Definition 10 (Neighbor, support). *A variable y is a* neighbor *of x iff $y \neq x$ and it appears with x in a clause. A variable x supports clause C with respect to partial assignment π iff all three variables in C are assigned and assignment of x, $\pi(x)$, is different from that of the other 2 variables in a C.*

The notion of support is a way of measuring the correctness of an assignment to a particular variable. The following definitions are well known graph theoretic concepts that we will use in this work:

Definition 11 (MAX-CUT, Max-CUT-formula)**.** *Let* $G = (V, E)$ *be an undirected graph. A cut is a partition of the set of vertices* V *into two sets* S *and* T*. Any edge* $(u, v) \in E$ *with* $u \in S$ *and* $v \in T$ *is an edge that is* crossing the cut*. The* size *of a cut is the total number of edges crossing that cut. A* maximum cut *is a cut that has a size greater than or equal to any other cut. The problem of finding a maximum cut in a graph is known as the* MAX-CUT *problem.*

Let V *be the set of Boolean variables. A* cut-clause *is a clause of the form* $(x \neq y)$*, where* x *and* y *are variables in* V *and a* cut-formula *is a conjunction of cut-clauses. For a cut-formula* ϕ*, the problem of finding an assignment to Boolean variables in* V*, such that the maximum number of cut-clauses are satisfied is called MAX-CUT-formula problem.*

We can see now that the problem MAX-CUT-formula is equivalent to MAX-CUT.

Lemma 12. *Any* a*-approximation algorithm for MAX-CUT, can be transformed to an* a*-approximation for MAX-CUT-formula.*

Proof. Suppose we have an algorithm that finds an a-approximation for MAX-CUT. For any cut-formula with set of variables $V = \{x_1, \dots x_n\}$ and m cut-clauses, we build a graph $G = (V', E)$ where $V' = \{x'_1 \dots x'_n\}$ and $(x'_i, x'_j) \in E$ if and only if there is a cut-clause $x_i \neq x_j$ in cut-formula. The size of both instances is the same (m cut-clauses vs m edges). Now any cut to $G = (V', E)$ would be a partition of V' into two disjoint subsets S', T'. Let π be the assignment to V such that $\pi(x_i) = \top$ if and only if $x'_i \in S'$ and $\pi(x_i) = \bot$ if and only if $x'_i \in T'$. The size of the cut S', T' is K, if and only if there are K cut-clauses in the cut-formula that are satisfied by π. Therefore the algorithm can also solve MAX-cut-formula with approximation a. □

Theorem 13 (Goemans and Williamson [22])**.** *There is a* 0.8785*-approximation algorithm for MAX-CUT.*

Given the Theorem 13 and Lemma 12, we have a following corollary:

Corollary 14. *There is a* 0.8785*-approximation algorithm for MAX-CUT-formula.*

From here on, we know that a variation of Goemans-Williamson algorithm can solve MAX-CUT-formula with an approximation ratio of 0.8785. The algorithm we use to prove Theorem 3, algorithm SDensity, is similar to the algorithm proposed by Krivelevich and

Algorithm 2 SDensity(ϕ)

```
if $p \geq \frac{1}{n^{3/2}}$ then
    $p' = \frac{\frac{d}{n^{3/2}}}{p}$
    $\theta =$ empty formula
    for all clause $C \in \phi$ do
        Include $C$ in $\theta$ with probability $p'$
    end for
else
    $\theta = \phi$
end if
$G_\theta = (V, E)$, $E = \emptyset$
for all $x, y \in V$ do
    if $Y_{x,y} \geq 3/4pn$ then
        $E = E \cup (x, y)$
    end if
end for
$\pi =$GOEMANS-WILLIAMSON($G_\theta$)
for all $x \in V$ do
    $N^x =$ number of neighbors of $x$
    $N^x_= =$ number of neighbors of $x$ that are same as $x$ by $\pi$
    if $N^x_= > \frac{1}{2}N^x$ then
        $\pi(x) = \neg\pi(x)$
    end if
end for
$\sigma = \pi$
while there is $x$ with support less than $0.99pn^2$ clauses do
    $\sigma = \sigma - x$
end while
Let $U$ be the set of unassigned variables in $\sigma$
Construct the residual graph $G$ and find the set of its connected components $\{\Gamma_i\}$
$\iota = \sigma$
for all $\Gamma_i$ do
    Let $U_i$ be the set of variables in $\Gamma_i$
    Let $\iota_i$ be the satisfying assignment for residual formula of $\Gamma_i$ found by trying every
    combination of values
    $\iota = \iota + \iota_i$
end for
return $\iota$
```

Vilenchik [34]. The algorithm SDensity constructs a cut-formula by assigning equalities and inequalities to pairs of variables. If $\tau(x) \neq \tau(y)$, the expected number of times variables x, y appear together in a same clause is pn. On the other hand if $\tau(x) = \tau(y)$, the expected number of times variables x, y appear together in a same clause is roughly $\frac{1}{2}pn$. Therefore the algorithm regards $\frac{3}{4}pn$ as "sufficiently many" times that two variables should appear together in order consider them unequal. By finding these inequalities, the algorithm builds a cut-formula G_θ. We will show that whp almost every NAE-clause in θ adds three cut-clauses to G_θ, two of which are satisfied by the planted solution. Note that the planted solution satisfies a $\frac{2}{3}$ portion of cut-clauses. Therefore whp any $\frac{2}{3} \times \gamma$-approximation solution to G_θ, is a γ-approximation to θ. That is, any solution that satisfies $\frac{2}{3}\dot{\gamma}$ portion of cut-clauses in G_θ will satisfy γ portion of NAE-clauses in θ. We have such an approximation algorithm from [22] where $\gamma = 0.8785$. Throughout this work, we will refer to this algorithm as Goemans-Williamson. This approximation gives us a solution that can be improved. There is a systematic way of deciding which of the variables are assigned correctly (whp). The algorithm finds these variables and unassigns the rest. Before we discuss the rest, we need following definitions:

Definition 15. *The* graph *of an NAE-formula ϕ with set of variable V is graph $G = (V, E)$ where $(x_i, x_j) \in E$ iff there is a clause in ϕ containing both x_i and x_j. The* graph *of G induced by $V' \subseteq V$ is the graph $G' = (V', E')$ where $(x, y) \in E'$ iff $x, y \in V'$ and $(x, y) \in E$. The* residual formula *w.r.t. V' is a sub formula ϕ, where only clauses that contain at least one variable from V are included.*

The next step of the algorithm SDensity is the unassignment of the variables that are assigned incorrectly. Consider the graph induced by set of unassigned variables. If we prove that this graph contains connected components of size $\Theta(\log n)$ (whp), we can perform brute force search to get the correct assignment.

3.1 An Overview of the algorithm

The algorithm works in 5 stages:

1. If the density of the formula ϕ is $p > \frac{1}{n^{3/2}}$, build the formula θ with density $p = \frac{d}{n^{3/2}}$ that has the same solutions as ϕ and behaves as though it is chosen from $\mathbb{P}_{n, \frac{d}{n^{3/2}}}$ and set $\theta = \phi$ otherwise (line 1 to 9).

2. Build a cut-formula from θ and approximate its maximum cut using Goemans-Williamson (line 10 to 16).

3. For every variable check the assignments of its neighbors and decide if the assignment of the variable should be changed because it is connected to too many variables assigned the same value (line 17 to 23).

4. Unassign the variables that do not support enough clauses (line 25 to 27)

5. Construct the graph induced by the set of unassigned variables and find connected components of the graph and the corresponding residual formulas of connected components. Then try every combination of assignments on each residual formula to find a satisfying assignment with respect to that residual formula. This can be done efficiently because whp, the connected components have size $\Theta(\log n)$ (line 28 to 36).

The proof of Theorem 3 is divided into 4 parts. In the first part we prove that in fact the formula θ has the same solution(s) as ϕ, and we can assume that $p \leq \frac{d}{n^{3/2}}$. Then we prove that after applying the neighbor counting algorithm (line 17-23) on the approximate solution of Goemans-Williamson we get a significant improvement on our closeness to the planted solution compared to the solution without neighbor counting. This improvement is significant enough so that whp only a small fraction of variables are assigned a different value in the planted assignment τ. Furthermore, we prove that there is a small fraction of variables that do not support "enough" clauses (i.e. less than $(1-\epsilon)d/9$ clauses) with respect to the planted assignment τ, and if in our assignment a variable does not support enough clauses, it is probably assigned wrong and those supporting many clauses are right. Finally, we prove that the variables that we keep after unassigning stage (line 25-27) are a "good" set of variables, i.e. whp they are correctly assigned, and the rest of the variables (unassigned variables) are grouped into some small (of size $\Omega(\log n)$) isolated connected components and applying brute force on each of those isolated components completes the assignment τ whp.

3.2 Property of planted assignment

One of the properties of a random planted solution is that the number of variables assigned 'true' is almost the same as the number of variables assigned 'false'. Let A denote the total number of clauses satisfied by a random assignment. For propositional variable x, let N^x be the random variable denoting the number of neighbors of x in a random planted formula, $N^x_=$ be the random variable denoting the number of neighbors of x that have the same assignment as x in the planted assignment, and $N^x_{\neq}$ be the random variable denoting the number of neighbors of x that have different assignment from that of x in the planted assignment. Let ζ be the fraction of variables assigned false in the planted solution. The following proposition shows that the number of true and false variables are almost the same. But in contrast with the previous section, the numbers need to be exact (not just ϵ away) because it would be much harder to work with given that ϵ would appear in every statement and we have to bound it. Moreover, since the random selection happens in two stages we can consider $E[N^x]$ to be a random variable depending on the portion of variables set to true.

Proposition 16. *The probability that any of the following does not happen is less that* $\exp\{-5 \times 10^{-8} n\}$*:*

(i) $0.748pn^2 \leq E[N^x] \leq 0.752pn^2$

(ii) $n^3/9 \leq A < n^3/8$

(iii) $0.499 \leq \zeta \leq 0.501$

Proof. First we prove that $0.499 \leq \zeta \leq 0.501$ with high probability. The rest of the statements follow. Given that the value for each variable is picked independently,

$$E[\zeta n] = \frac{1}{2}n.$$

By Chernoff bound,

$$\Pr[|\zeta n - \frac{1}{2}n| \geq 0.001n] \leq \exp\{-5 \times 10^{-8} n\}.$$

Given that $0.499 \leq \zeta \leq 0.501$, we can bound A as follows:

$$\begin{aligned} \binom{0.501n}{2}\binom{0.499n}{1} + \binom{0.501n}{1}\binom{0.499n}{2} \leq \quad & A \quad \leq 2\binom{n/2}{2}\binom{n/2}{1} \\ n^3/9 \leq \quad & A \quad \leq n^3/8. \end{aligned}$$

Let A^x be the total number of clauses with x in them. In NAE-clauses, one variable's assignment should be different than the other two in order for the clause to be satisfied (i.e. 2 variables are true variables and 1 is false or vice versa). Therefore, we can bound A^x by:

$$\binom{0.499n}{2} + \binom{0.499n}{1}\binom{0.501n-1}{1} \leq \quad A^x \quad \leq \binom{0.501n}{2} + \binom{0.499n-1}{1}\binom{0.501n}{1}.$$

Considering the expectation of N^x given that $A^x = m$, we have that:

$$E[N^x|A^x = m] = 2pm - o(1).$$

From this we get that whp:

$$\begin{aligned} 2 \times \left[\binom{0.499n}{2} + \binom{0.499n}{1}\binom{0.501n-1}{1}\right] p \leq \quad & E[N^x] \quad \leq 2 \times \left[\binom{0.501n}{2} + \binom{0.499n-1}{1}\binom{0.501n}{1}\right] p - o(1) \\ 0.748pn^2 \leq \quad & E[N^x] \quad \leq 0.752pn^2. \end{aligned}$$

□

From here on, we are going to assume that $0.499 \leq \zeta \leq 0.501$, $n^3/9 \leq A < n^3/8$, and $0.748d \leq E[N^x] \leq 0.752d$. The probability that these inequalities are false does not depend on d. Therefore, we can not bound it by making d larger. So we assume that n is large enough so that this probability is tiny.

3.3 Decreasing the density of the formula

We would like to prove that our Goemans-Williamson 0.8786-approximation to the cut-formula produced from NAE-E3-SAT instance would result in 0.8786-approximation to NAE-E3-SAT. If no pair appeared more than once, then every NAE-clause corresponds to 3 cut-clauses in the cut-formula, and 0.8786-approximation to the cut-formula would result in the 0.8786-approximation algorithm to NAE-E3-SAT instance. However, there are some pairs that appear more than once in the NAE-SAT instance. Let N be the size of cut-formula, and N^R be the total number of repetitions of pairs. In order to show that a 0.8786-approximation to the cut-formula is a 0.8786-approximation to the NAE-E3-SAT instance, it is sufficient to show $N^R \in o(N)$. Therefore any 0.8786-approximation to the cut-formula is a $0.8786 - o(1)$-approximation to the NAE-E3-SAT instance. This condition ($N^R \in o(N)$) only holds whp when $p \in o(\frac{1}{n})$. From the assumption of the algorithm, the density of the NAE-formula is $p \geq \frac{d}{n^2}$ for some large d. We picked the density $p = \frac{d}{n^{3/2}}$ so that it satisfies both criteria. Therefore, what we need is cut formula θ chosen from $\mathbb{P}_{n,\frac{d}{n^{3/2}}}$. In line 1 - 9 of the algorithm $\mathsf{SDensity}$, we make sure that the density of the formula ϕ is $p \leq \frac{d}{n^{3/2}}$ for some large constant d. The algorithm accomplishes this by choosing each clause from ϕ with probability $p' = \frac{\frac{d}{n^{3/2}}}{p}$ and places it in θ. Since $p' \times p = \frac{d}{n^{3/2}}$, θ behaves as though it was chosen from $\mathbb{P}_{n,\frac{d}{n^{3/2}}}$.

If we show that for θ chosen from $\mathbb{P}_{n,p}$ where $p \geq \frac{d}{n^{3/2}}$ whp only 1 pair of solutions (the solution and its complement) exists, we can conclude that this solution is also the solution of ϕ and therefore whp we have not lost any information with regards to solution of ϕ by constructing θ. We prove this using in Lemma 17. It will be convenient to treat the number of variables on which two assignments τ and η disagree as the Hamming distance between two strings of values, and denote it by $\mathsf{Ham}(\tau, \eta)$.

Lemma 17. *If $p \geq \frac{d}{n^{3/2}}$, then whp there is only 1 pair of solutions to a planted NAE-E3-SAT formula.*

Proof. Let A^k be the event that some assignment η, $k = \mathsf{Ham}(\tau, \eta)$, satisfies all the clauses in ϕ. Note that if $k > n/2$ then $\mathsf{Ham}(\bar{\tau}, \eta) < n/2$, where $\bar{\tau}$ is the complement of the planted solution. Since $\bar{\tau}$ is also a solution to the NAE-E3-SAT instance, we assume that $k \leq n/2$. Let N^k denote the total number of NAE-clauses (not just those in the formula) that are satisfied by τ, but not satisfied by η. A clause that is satisfied by τ is not satisfied by η, if the variable that is assigned a different value than the other two variables in the clause is among the k variables that are different in τ and η while the other two variables in that clause are not. Note that a clause is also satisfied by τ and not by η if the variable that is assigned a different value than the other two variables in the clause is not among the k variables that are different in τ and η while the other two variables in that clause are. However, we neglect these clauses in our calculation of N^k since the number of these clauses is negligible and will only strengthen the inequality 3.1. Let s be the number of variables x that are different in τ and η, where $\tau(x) = 0$. Note that $k - s$ is the number of variables x

that are different in τ and η, where $\tau(x) = 1$. By Proposition 16, whp we have

$$\begin{aligned}
N^k &\geq (k-s)\binom{0.499n}{2} - (k-s)\binom{s}{1}\binom{0.501n}{1} + s\binom{0.499n}{2} - (s)\binom{k-s}{1}\binom{0.501n}{1} \\
&\geq k\binom{0.499n}{2} - 2s(k-s)\binom{0.501n}{1} \\
&\geq k(0.499^2 \times n^2 - \frac{0.501nk}{2}) \\
&\underbrace{\geq}_{k \leq n/2} -k(0.499^2 \times n^2 - \frac{0.501n^2}{4}) \\
&\geq k \times \frac{n^2}{10}.
\end{aligned} \tag{3.1}$$

We need to show

$$\sum_{1 \leq k \leq n/2} \binom{n}{k} \Pr[A^k] = o(1).$$

Each clause is chosen by probability $p \geq \frac{d}{n^{3/2}}$. Therefore, we have

$$\begin{aligned}
\sum_{1 \leq k \leq n/2} \binom{n}{k} \Pr[A^k] &\leq \sum_{1 \leq k \leq n/2} \binom{n}{k} (1 - \frac{d}{n^{3/2}})^{N^k} \\
&\leq \sum_{1 \leq k \leq n/2} \binom{n}{k} (1 - \frac{d}{n^{3/2}})^{\frac{kn^2}{10}} \\
&\leq \sum_{1 \leq k \leq n/2} n^k \exp(-\frac{dk}{10} n^{1/2}) \\
&\leq \sum_{1 \leq k \leq n/2} \exp(-\frac{dk}{20} n^{1/2}) \\
&\leq \frac{n}{2} \times \exp(-\frac{d}{20} n^{1/2}) \\
&\leq \exp(-\frac{d}{30} n^{1/2}).
\end{aligned}$$

□

3.4 MAX-CUT approximation

In this section, we consider the assignment π produced by the Goemans-Williamson algorithm. We know that the Goemans-Williamson approximate solution satisfies more than

0.8786 of all clauses. The following proposition proves that this solution is in fact close to the planted solution.

First, we consider the assignment π produced by Goemans-Williamson algorithm.

Proposition 18. $\Pr[\mathsf{Ham}(\pi,\tau) \geq 0.15n] \leq \exp\left\{-0.00001 \cdot 0.8726\frac{pn^3}{16}\right\}$

Proof. We start by showing that the number of cut-clauses contained in the cut-formula is in the order of $3N$ where N is the size of formula θ (number of cut-clauses). We know that only one occurrence of a pair in the NAE-E3-SAT formula appears in the cut-formula. Therefore, if a pair appears $k+1$ times, k of the appearances are not included in the cut-formula. The probability that a pair is repeated more than k times is $p^{k+1}n^{k+1}$ and there are at most n^2 pairs appearing in clauses. Therefore, the expected number of pairs that are repeated more than k times is at most $p^{k+1}n^{k+3}$. Let N^R denote the total number of repetitions of all pairs in the NAE-formula. By Lemma 17, we can assume that the density of the formula is $p \leq \frac{d}{n^{3/2}}$. We have

$$N^R = \sum_{k=1,\dots,n} p^{k+1}n^{k+3} \leq 2p^2n^4 \underbrace{\in}_{p \leq \frac{d}{n^{3/2}}} o(pn^3)$$

If N is the number of clauses in θ then G_θ contains $3N - N^R \geq 3N - o(N)$ edges.

Among the 3 edges that arise from each clause at most 2 can be included into a cut. On the other hand, the cut corresponding to the planted solution contains $2N$ edges, and therefore is almost maximum one. Thus if an assignment λ does not satisfy αN clauses, it gives rise to a cut that is smaller than the maximum one by a factor of $\alpha - o(1)$. Let λ be a random assignment, and let A_λ be the event that $\mathsf{Ham}(\lambda,\tau) > 0.15n$.

$$\begin{aligned} E[Y_\lambda | A_\lambda] &\leq \underbrace{\left(0.85^3 + 0.15^3 + 2(0.85)^2(0.15) + 2(0.85)(0.15)^2\right)}_{\substack{\text{probability that a clause is satisfied} \\ \text{by random } \lambda \text{ with } \mathsf{Ham}(\lambda,\tau) < 0.85n}} \times p \times \frac{n^3}{8} \\ &\leq 0.8726\frac{pn^3}{8}. \end{aligned}$$

By Chernoff bound,

$$\begin{aligned}
\Pr[Y_\lambda \geq 0.8786\frac{pn^3}{8}|A_\lambda] &= \Pr[Y_\pi \geq (1+0.006)0.8726\frac{pn^3}{8}|A_\lambda] \\
&\leq \left(\frac{\exp\{0.006\}}{(1.006)^{1.006}}\right)^{0.8726\frac{pn^3}{8}} \\
&= \exp\left\{0.8726\frac{pn^3}{8}(0.006 - 1.006\log 1.006)\right\} \\
&\leq \exp\left\{-0.00001 \cdot 0.8726\frac{pn^3}{8}\right\}.
\end{aligned}$$

Moreover, if d is sufficiently large whp no assignment λ exists such that $Y_\lambda \geq 0.8786$, and A_λ. Indeed, applying the union bound we have

$$\begin{aligned}
\Pr[\mathsf{Ham}(\lambda,\tau) \geq 0.15n] &\leq \Pr[\exists\lambda|Y_\lambda \geq 0.8778\frac{pn^3}{8}|A_\lambda] \\
&\leq 2^n \exp\left\{-0.00001 \cdot 0.8726\frac{pn^3}{8}\right\} \\
&\leq \exp\left\{-0.00001 \cdot 0.8726\frac{pn^3}{16}\right\}.
\end{aligned}$$

Finally, since the assignment π found by the Goemans-Williamson algorithm satisfies at least the fraction 0.8786 of all clauses, whp $\mathsf{Ham}(\pi,\tau) < 0.15n$. □

3.5 Neighbor counting

Ultimately, we would like to count the neighbors of x that are assigned the same value as x in our Goemans-Williamson solution and then unassign the x's that do not support enough clauses. Before we get to the unassigning stage, we need to see why the set of variables that the algorithm SDensity assigns wrong is small. We prove this in 2 stages. Let σ be the assignment that the algorithm finds after the neighbor counting stage. First we prove that the number of variables x such that $\boxed{|N^x_{\neq} - N^x_{=}| \leq \frac{13pn^2}{100}}$ is small. In the next stage we prove that the set of variables whose number of neighbors differ in the planted solution and σ by more than $\frac{12pn^2}{100}$, is also small. Therefore, the union of these two sets is a small subset of variables.

Proposition 19. *Let F^{NC} be the number of variables x that satisfy the following inequality*

$$N^x_{\neq} - N^x_{=} \leq \frac{13pn^2}{100}.$$

Let α *be a constant such that* $\exp\{-\frac{d}{4\times 10^5}\} \leq \alpha \leq \frac{1}{3000}$. *For* $p \geq \frac{d}{n^2}$ *and* $d \geq 3000$ *we have*

$$\Pr[F^{NC} \geq \alpha n/2] \leq \exp\{-\frac{d}{4\times 10^5}\alpha n\}$$

Proof. The problem with calculating the expected value of F^{NC} is that the random variables $N^x_{\neq}$ are not independent. Therefore we need to introduce another set of random variables, $\hat{N}^x_{\neq}$, that are independent from each other and serve our purpose. This would help us calculate the expected value of F^{NC}.

Let $T = \{x_1, ..., x_{\alpha n/2}\}$ be a random subset of variables in V. A variable from T is called T-influenced iff it shares more than $d/1000$ neighbors with other variables of T. Let $Y \subseteq T$ be the set of all T-influenced variables of T, F^{x_i} be the random variable counting the number of neighbors that x_i has in common with other variables of T, and F^{x_i,x_j} be the random variable counting the number of neighbors x_i and x_j have in common. Considering the expectation of F^{x_i} and F^{x_i,x_j} we have:

$$\begin{aligned} E[F^{x_i}] &\leq E\big[\sum_{j=0}^{\alpha n/2} F^{x_i,x_j}\big] \\ &= \sum_{j=0}^{\alpha n/2} E[F^{x_i,x_j}] \\ &\leq \frac{\alpha n}{2}p + o(1) \\ &\underbrace{\leq}_{\alpha\leq 1/3000} \frac{pn^2}{2000}. \end{aligned}$$

By Chernoff bound, we have

$$\Pr[F^{x_i} \geq pn^2/1000] \leq \exp\{-pn^2/2000\}.$$

Calculating the expected size of Y we obtain the following:

$$E[|Y|] \leq \sum_{i=1}^{\alpha n/2} \Pr[F^{x_i} \geq pn^2/1000] \leq \exp\{-pn^2/2000\}\alpha n/2.$$

By Chernoff bound, since $d > 3000$ ($d/2500 > 1$), we have:

$$\begin{aligned}\Pr[|Y| > \alpha n/4] &\leq \exp\{-(\exp\{pn^2/2500\})^2 \exp\{-pn^2/2000\}\frac{\alpha n}{2}\}\\ &\leq \exp\{-(\exp\{pn^2/1250\})\exp\{-pn^2/2000\}\frac{\alpha n}{2}\}\\ &\leq \exp\{-\exp\{pn^2/30000\}\alpha n/2\}\\ &\underbrace{\leq}_{d\geq 3000} \exp\{-d/300\frac{\alpha n}{2}\}.\end{aligned}$$

Therefore we can assume that $|Y| \leq \alpha n/4$. Without loss of generality, assume that $x_1, ..., x_{\alpha n/4}$ are non-T-influenced variables of T. Let $\hat{N}_{=}^{x_i}$ be the number of neighbors of x_i that have the same truth value as x_i but are not neighbors of any other variable in T. Considering the expectation of $\hat{N}_{=}^{x_i}$, we obtain the following:

$$\begin{aligned}E[\hat{N}_{=}^{x_i}] &= \sum_{m=0}^{n} E[\hat{N}_{=}^{x_i}|N^{x_i}=m]\Pr[N^{x_i}=m]\\ &= \sum_{m=0}^{n}\Pr[N^{x_i}=m]\sum_{l=0}^{n}E[\hat{N}_{=}^{x_i}|N^{x_i}=m, N_{=}^{x_i}=l]\Pr[N_{=}^{x_i}=l|N^{x_i}=m]\\ &= \sum_{m=0}^{n}\Pr[N^{x_i}=m]\sum_{l=0}^{n}(1-\alpha/2)l\Pr[N_{=}^{x_i}=l|N^{x_i}=m]\\ &= (1-\alpha/2)\sum_{m=0}^{n}\Pr[N^{x_i}=m]\underbrace{E[N_{=}^{x_i}|N^{x_i}=m]}_{\frac{m}{3}}\\ &= \frac{1}{3}(1-\alpha/2)E[N^{x_i}].\end{aligned}$$

Similarly,

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

Therefore, since $\alpha \leq 1/3000$ and by Proposition 16 we have:

$$E[\hat{N}_{\neq}^{x_i} - \hat{N}_{=}^{x_i}] \geq \frac{1}{3}(1-\frac{\alpha}{2})E[N^{x_i}] \geq \frac{1}{3}(1-\frac{\alpha}{2})0.748pn^2 \geq 0.249pn^2.$$

By the definition of $\hat{N}_{\neq}^{x_i}$ and $N_{\neq}^{x_i}$, we know that for $i \in [1, \alpha n/4]$,

$$N_{\neq}^{x_i} - N_{=}^{x_i} \leq \hat{N}_{\neq}^{x_i} - \hat{N}_{=}^{x_i} - \frac{d}{1000}.$$

This is due to the fact x_i can only have at most $pn^2/1000$ misleading neighbors (neighbors that also have neighbors in T) . Therefore, for $N_{\neq}^{x_i} - N_{=}^{x_i}$ to be less than $\frac{13pn^2}{100}$, it suffices that $\hat{N}_{\neq}^{x_i} - \hat{N}_{=}^{x_i} < \frac{131pn^2}{1000}$. If $\hat{N}_{\neq}^{x_i} - \hat{N}_{=}^{x_i} < \frac{131pn^2}{1000}$, then at least one of $\hat{N}_{\neq}^{x_i}$ or $\hat{N}_{=}^{x_i}$ has deviated from its expectation by $\frac{0.249-0.131}{2}pn^2 \geq 0.01pn^2$. Estimating the probability of this event, we obtain

$$\begin{aligned}
\Pr[N_{\neq}^{x_i} - N_{=}^{x_i} < \frac{13pn^2}{100}] &\leq \Pr\left[|\hat{N}_{\neq}^{x_i} - E[\hat{N}_{\neq}^{x_i}]| \geq 0.01pn^2\right] + \Pr\left[|\hat{N}_{=}^{x_i} - E[\hat{N}_{=}^{x_i}]| \geq 0.01pn^2\right] \\
&\leq 2\exp\{-2 \times 0.0001 \times (0.249)pn^2\} \\
&\leq \exp\{-\frac{pn^2}{100000}\}.
\end{aligned}$$

Let F^T be the event that every variable x in T has the property $N_{\neq}^{x} - N_{=}^{x} < \frac{13pn^2}{100}$ and let $F^{T/2}$ be the event that from some variable $x_i \in \{x_1, \cdots, x_{\alpha n/4}\}$ has the property $N_{\neq}^{x_i} - N_{=}^{x_i} < \frac{13pn^2}{100}$. Since the random variables $\hat{N}_{\neq}^{x_i}$ (and random variables $\hat{N}_{=}^{x_i}$) are independent, they correspond to disjoint set of variables or neighbors. We conclude:

$$\begin{aligned}
\Pr[F^T] &\leq \Pr[F^{T/2} \big| |Y| < \alpha n/4] \Pr[|Y| < \alpha n/4] + \Pr[|Y| \geq \alpha n/4] \\
&\leq \exp\{-\frac{pn^2}{10^5}\alpha n/4\}(1 - \exp\{-\frac{pn^2}{300} \times \frac{\alpha n}{2}\}) + \exp\{-\frac{pn^2}{300} \times \frac{\alpha n}{2}\} \\
&\leq \exp\{-\frac{d}{2 \times 10^5}\alpha n\}.
\end{aligned}$$

Furthermore, by the union bound,

$$\begin{aligned}
\Pr[F^{NC} \geq \alpha n/2] &\leq \Pr[\exists T, |T| = \alpha n/2 \text{ and } F^T] \\
&\leq \binom{n}{\alpha n/2} \exp\{-\frac{d}{2 \times 10^5}\alpha n\} \\
&\leq \left(2e/\alpha\right)^{\frac{\alpha n}{2}} \exp\{-\frac{d}{2 \times 10^5}\alpha n\} \\
&\leq \exp\{\frac{\alpha n}{2}(\log 2 + \log(1/\alpha) - \frac{d}{10^5})\} \\
&\underbrace{\leq}_{\alpha > \exp\{-\frac{d}{4 \times 10^5}\}} \exp\{-\frac{d}{4 \times 10^5}\alpha n\}.
\end{aligned}$$

□

We have proven that whp the number of neighbors of a variable x that in the planted solution τ have a different value than $\tau(x)$ is significantly larger that the number of neighbors that have the same value in the planted solution for most variables. Furthermore, we have proven that whp Goemans-Williamson algorithm gives us a solution that is close to the planted solution ($\mathsf{Ham}(\pi, \tau) < 0.15n$). Now, we would like to show that after the neighbors counting step whp only a sublinear number of wrongly assigned variables remains.

Proposition 20. *Let σ be the assignment obtained after line 25 of the algorithm SDensity. Then for any constants α and d such that* $\max\{\exp\{-d/4800\}, \frac{3000}{d}\} < \alpha < \frac{1}{500}$ *and* $d \geq 70000$ *(recall that we assume $p \geq \frac{d}{n^2}$),*

$$\Pr[\mathsf{Ham}(\sigma,\tau) \geq \alpha n] \leq \exp\{-d \times 10^{-10}\alpha n\}.$$

We prove the proposition in the rest of this section. Let π be the assignment generated in line 16 of the algorithm. Recall that $N^x_=$ and $N^x_{\neq}$ denote the number of neighbors y of the variable x such that $\pi(y) = \pi(x)$, and the number of neighbors of x such that $\pi(y) \neq \pi(x)$, respectively. Let $N^{x,NC}_=$ and $N^{x,NC}_{\neq}$ be the numbers of neighbors y of variable x such that $\rho(y) = \tau(y)$ and $\rho(y) \neq \tau(y)$, respectively. Let also D^x_{NC} denote the number of neighbors of x that are different in their truth assignment in τ and π. Variable x gets the right assignment if $D^x_{NC} \leq 0.12pn^2$ and $N^x_{\neq} - N^x_= \leq 0.13pn^2$. Since

$$|N^{x,NC}_{\neq} - N^{x,NC}_=| \geq |N^x_{\neq} - N^x_=| - \frac{12pn^2}{100} \geq \frac{pn^2}{100} > 0,$$

if x is has the wrong value, then either $D^x_{NC} > 0.12pn^2$ or $N^x_{\neq} - N^x_= < 0.13pn^2$. Let Y be the set of variables in which $D^x_{NC} > 0.12pn^2$, and Y' be the set of variables such that $N^x_{\neq} - N^x_= < 0.13pn^2$. We have

$$\begin{aligned} \Pr[\mathsf{Ham}(\sigma,\tau) \geq \alpha n] &\leq \Pr[|Y \cup Y'| \geq \alpha n] \\ &\leq \Pr[|Y| \geq \alpha n/2] + \Pr[|Y'| \geq \alpha n/2]. \end{aligned} \tag{3.2}$$

We have already shown that $\Pr[|Y'| \geq \alpha n/2] = \Pr[F^{NC} \geq \alpha n/2] \leq \exp\{-\frac{d}{4\times 10^5}\alpha n\}$. It remains to bound $\Pr[|Y| \geq \alpha n/2]$. We do this in the following lemma.

Lemma 21. *Let π be the assignment returned in line 16 of the algorithm. Let D_{NC} denote the number of variables x in which $D^x_{NC} \geq 0.12pn^2$. For constants α and d such that* $\max\{\exp\{-d/4800\}, \frac{3000}{d}\} < \alpha < \frac{1}{500}$ *and* $d \geq 70000$ *(recall that $p \geq \frac{d}{n^2}$), we have*

$$\Pr[D_{NC} \geq \alpha n/2] \leq \exp\{-10^{-8}\alpha n\}.$$

Proof. By Proposition 18 we can assume that $\mathsf{Ham}(\pi,\tau) < 0.15n$. Let $T = \{x_1, ..., x_{\alpha n/2}\}$ be a random set of variables, and let *T-weak* influenced variables be variables that share more than $pn^2/200$ neighbors with other variables of T. Let $Y \subset T$ be the set of all T-weak influenced variables of T. Recall from Proposition 19 that F^{x_i} is the random variable counting the number of neighbors that x_i has in common with variables from T. Observe, however, that now the set T is different. Also F^{x_i,x_j} is the random variable counting the number of neighbors that x_i and x_j have in common. We have

$$E[F^{x_i}] \leq E\big[\sum_{j=0}^{\alpha n/2} F^{x_i,x_j}\big] \leq \sum_{j=0}^{\alpha n/2} E[F^{x_i,x_j}] \leq \frac{\alpha n}{2}np + o(1) \underbrace{\leq}_{\alpha \leq 1/500} \frac{pn^2}{400}.$$

Bounding the probability of $F^{x_i} \geq \frac{pn^2}{200}$ we get

$$\Pr[F^{x_i} \geq pn^2/200] \leq \exp\{-pn^2/500\}.$$

Therefore,

$$E[|Y|] \leq \sum_{i=1}^{\alpha n/2} \Pr[F^{x_i} \geq d/200] \leq \exp\{-pn^2/500\}\frac{\alpha n}{2}.$$

From this we can calculate the chances that $|Y| \geq \alpha n/4$

$$\begin{aligned}
\Pr[|Y| > \alpha n/4] &\leq \exp\{-(\exp pn^2/600)^2 \exp\{-pn^2/500\}\frac{\alpha n}{2}\} \\
&\leq \exp\{-\exp\{pn^2/600\}\exp\{-pn^2/30000\}\frac{\alpha n}{2}\} \\
&\leq \exp\{-\exp\{pn^2/6000\}\frac{\alpha n}{2}\} \\
&\underbrace{\leq}_{pn^2 \geq 70000} \exp\{-pn^2/60\frac{\alpha n}{2}\} \leq \exp\{-d/60\frac{\alpha n}{2}\}.
\end{aligned}$$

Therefore we can assume that $|Y| \leq \alpha n/4$. Without loss of generality assume that $\{x_1, ..., x_{\alpha n/4}\}$ the set of non-T-weak influenced variables of T. Let π be a random assignment such that $\mathsf{Ham}(\pi, \tau) < 0.15n$, as the Goemans-Williamson solution agrees whp with more that $0.85n$ of the assignments of the planted solution. Let $D_\pi^{x_i}$ be defined as the random variable counting the number of neighbors of x_i that are assigned different values in π and τ. Let $\hat{D}_\pi^{x_i}$ be the number of neighbors of variables x_i that are different in π from planted solution τ where we only count the neighbors that are not neighbors of other variables in T. For $i \in [1, \alpha n/4]$, by definition of $D_\pi^{x_i}$ and $\hat{D}_\pi^{x_i}$ we know that $D_\pi^{x_i} \leq \hat{D}_\pi^{x_i} + \frac{pn^2}{200}$ because it can only have $pn^2/200$ misleading neighbors (neighbors that also have neighbors in T). Note that for $D_\pi^{x_i} \geq 0.12pn^2$, it is necessary that $\hat{D}_\pi^{x_i} \geq 0.115pn^2$. For $i \in [1, \alpha n/4]$

the following holds

$$\begin{aligned} E[\hat{D}_{\pi}^{x_i}] &= \sum_{m=0}^{n} E[\hat{D}_{\pi}^{x_i} | N^{x_i} = m] \Pr[N^{x_i} = m] \\ &\underbrace{\leq}_{\pi \text{ is random}} 0.15 E[N^{x_i}] \\ &\leq 0.15 \times 0.752 pn^2 \leq 0.1128 pn^2. \end{aligned}$$

Which implies

$$\Pr[\hat{D}_b^{x_i} \geq 0.115pn^2] \leq \exp\{-(0.019)^2(0.1128)pn^2\} \leq \exp\{-pn^2/30000\}.$$

Let D_{π}^{T} be the event that $D_{\pi}^{x} \geq 0.12pn^2$ for every variable x in T. Let $T/2 \subseteq T$ denote the first $\alpha n/4$ of the variables of T and let $D_{\pi}^{T/2}$ be the event that for all $x \in T/2$, $D_{\pi}^{x} \geq 0.08pn^2$. Now, since $\hat{D}_{\pi}^{x_i}$ and $\hat{D}_{\pi}^{x_j}$ for $i \neq j$ are independent,

$$\begin{aligned} \Pr[D_{\pi}^{T}] &\leq \Pr[D_{\pi}^{T/2} || Y| < \alpha n/4] \times Pr[|Y| < \alpha n/4]] + \Pr[|Y| \geq \alpha n/4] \\ &\leq \exp\left\{-(pn^2/30000)\frac{\alpha n}{4}\right\}(1 - \exp\{-\frac{d}{60} \times \frac{\alpha n}{2}\}) + \exp\{-\frac{d}{60} \times \alpha n/2\} \\ &\underbrace{\leq}_{p \geq \frac{d}{n^2}} \exp\{-\frac{d}{500} \times \alpha n\}. \end{aligned}$$

This inequality holds for a random assignment π and random subset T of size $\alpha n/2$. By the union bound,

$$\begin{aligned} \Pr[D_{NC} \geq \alpha n/2] &\leq \Pr[\exists \pi \exists T D_{\pi}^{T}] \\ &\leq \binom{n}{0.2n}\binom{n}{\alpha n/2} \exp\{-\frac{d}{500} \times \alpha n\} \\ &\leq (5e)^{n/5}(\frac{2e}{\alpha})^{\alpha n} \exp\{-\frac{d}{500} \times \alpha n\} \\ &\underbrace{\leq}_{\alpha \geq \frac{3000}{d}} (\frac{2e}{\alpha})^{\alpha n} \exp\{-\frac{d}{600} \times \alpha n\} \\ &\underbrace{\leq}_{d > 1800} \exp\{-\frac{d}{800} \times \alpha n\}. \end{aligned}$$

By Proposition 18 we can assume that $D < 0.15n$. Then

$$\Pr[D_{NC} \geq \alpha n/2 | D < 0.15n] \leq \exp\{-d/800\alpha n\}.$$

Making use of Proposition 18 we obtain:

$$\begin{aligned}
&\Pr[D_{NC} \geq \alpha n/2] \\
&\qquad \leq \Pr[D_{NC} \geq \alpha n/2 | D < 0.15n] \times \Pr[D < 0.15n] + \Pr[D \geq 0.15n] \\
&\qquad \leq \exp\{-\frac{d}{800}\alpha n\} \times (1 - \exp\{-d \times 10^{-9}\alpha n\}) + \exp\{-d \times 10^{-8}\alpha n\} \\
&\qquad \leq \exp\{-d \times 10^{-9}\alpha n\}.
\end{aligned}$$

□

Considering equation (3.2), by Proposition 21 and 19 we have

$$\begin{aligned}
\Pr[\mathsf{Ham}(\sigma, \tau) \geq \alpha n] &\leq \exp\{-d \times 10^{-9}\alpha n\} + \exp\{-\tfrac{d}{4\times 10^5}\alpha n\} \\
&\leq \exp\{-d \times 10^{-10}\alpha n\},
\end{aligned}$$

Proposition 20 is proved.

3.6 Support

Now we estimate the number of variables supporting insufficiently many clauses.

Proposition 22. *For $0 \leq \epsilon < 1$, let X_ϵ be the random variable counting the number of variables that support less than $(1-\epsilon)pn^2/9$ clauses with respect to τ. For $\alpha \in [\exp\{-\frac{\epsilon^2 d}{90}\}, 1]$, we have*

$$\Pr[X_\epsilon \geq \alpha n] \leq \exp\{-\frac{\epsilon^2 d}{10}\alpha n\}.$$

Proof. Let S^x_τ be the number of clauses that x supports w.r.t τ. Then

$$E[S^x_\tau] \geq p\binom{0.499n}{2} \geq \frac{pn^2}{9}.$$

Let F_x be the event that x supports less that $(1-\epsilon)pn^2/9$. As it is easily seen,

$$\Pr[F_x] \leq \exp\{-\epsilon^2 pn^2/9\}.$$

For any variables x and y, the clauses that x supports are disjoint from the clauses supported by y, and clauses are chosen independently. Let $\{x_1, ..., x_{\alpha n}\}$ be a random set of variables,

$$\Pr[F_{x_1} \wedge F_{x_2} \wedge ... \wedge F_{x_{\alpha n}}] \leq \Pr[F_{x_1}] \times \Pr[F_{x_1}] \times ... \times \Pr[F_{x_{\alpha n}}] \leq \exp\{-\frac{1}{18}\epsilon^2 pn^3\alpha\}.$$

Therefore, for any set $\{x_1, ..., x_{\alpha n}\}$ of size αn, it follows that

$$\begin{aligned}\Pr[X_\epsilon \geq \alpha n] &\leq \Pr[\exists\{x_1, ..., x_{\alpha n}\}, F_{x_1} \wedge F_{x_2} \wedge ... \wedge F_{x_{\alpha n}}] \\ &\leq \binom{n}{\alpha n} \exp\{-\frac{1}{18}\epsilon^2 p n^3 \alpha\} \leq \exp\{-\frac{\epsilon^2 d}{36}\alpha n\},\end{aligned}$$

where the last inequality follows from $\alpha > \exp\{-\frac{\epsilon^2 d}{90}\}$ and $p \geq \frac{d}{n^2}$. □

3.7 Unassigning variables

In this section, we consider the partial assignment obtained after the unassigning step (lines 25-27) of the algorithm. The main goal is to prove that the number of variables unassigned is small (sublinear), and that the residual graph induced by the set of such variables has very small connected components.

First we show that whp the algorithm does not produce a large number of unassigned variables. Let Z denote the set of variables remaining assigned after the unassigning step (line 28 of the algorithm).

Proposition 23. *Let $\alpha < (1/10)^4$. If $|Z| = k$ for some $k \in [1, \alpha n]$, then* $\Pr[\exists Z] \leq \exp\{-\frac{dk}{40}\log(n/k)\}$.

Before proving this proposition, we start with a different proposition that proves that whp there is no large set Y with the following property: there are "many" clauses that have at least 2 variables from Y. Note that "many" actually still means a set much smaller than the set of all variables. This proposition will be used when estimating the number of variables unassigned in line 26, as well as a model for proving Proposition 23. We will show that unassigning variables that support only a few clauses does not create too many new variables that have the mentioned property. Observe that if another variable with low support is created this means that it occurs in many clauses together with unassigned variables; and the probability of this happening is very small.

Proposition 24. *Let $Y \subseteq V$ and $f(Y)$ denote the set of clauses in formula ϕ such that least 2 of their variables belong to Y. Let $\gamma \in (0, 1/2]$, $\gamma \geq 2/d$, and $\alpha \in (0, \gamma^4]$. Then,*

$$\Pr[\exists Y, |Y| = \alpha n, |f(Y)| \geq \gamma \alpha p n^3] \leq \exp\left\{\frac{1}{4}\gamma d \alpha n \log \alpha\right\}.$$

Proof. For a fixed Y, let C_Y be the total number of clauses satisfied by τ containing at least two variables from Y (not necessarily belonging to ϕ). Then

$$C_Y \leq \binom{\alpha n}{2}(n-2) \leq \alpha^2 n^3.$$

Therefore,

$$\begin{aligned}
&\Pr[\exists Y, |Y| = \alpha n, |f(Y)| \geq \gamma\alpha p n^3] \\
\leq \quad & \binom{n}{\alpha n}\binom{\alpha^2 n^3}{\gamma\alpha p n^3} p^{\gamma\alpha p n^3} \\
\leq \quad & (\frac{en}{\alpha n})^{\alpha n}(\frac{e\alpha^2 n^3}{\gamma\alpha p n^3})^{\gamma\alpha p n^3} p^{\gamma p \alpha n^3} \\
\leq \quad & (\frac{e}{\alpha})^{\alpha n}(\frac{e\alpha}{\gamma})^{\gamma\alpha p n^3} \\
\leq \quad & \exp\{\gamma\alpha p n^3(1 - \log\gamma + \log\alpha)\}\exp\{\alpha n(1 - \log\alpha)\} \\
\underbrace{\leq}_{d \geq 2/\gamma, p \geq d/n^2} \quad & \exp\{\gamma d\alpha n(\frac{3}{2} - \log\gamma + \frac{1}{2}\log\alpha)\} \\
\underbrace{\leq}_{\alpha \leq \gamma^4} \quad & \exp\{(-\frac{1}{4}\gamma d \log\alpha \times \alpha n\}.
\end{aligned}$$

□

Before proving Proposition 23 we need an auxiliary lemma.

Lemma 25. *Let Z, $|Z| = k$, be the set of variables that survived the unassignment stage and π is different than the planted solution τ on those variables. Then there are at least $\frac{pn^2k}{10}$ clauses that contains at least 2 variables from Z.*

Proof. For $x \in Z$, since x survived the unassignment stage, it must support at least $\frac{(1-\epsilon)pn^2}{9} \geq \frac{pn^2}{10}$ clauses in which all the variables are assigned. Since π is wrong on x, this means that the true value of x would make the clauses it supports ($\frac{pn^2}{10}$ of them) to become unsatisfied. Therefore there must be another variable with a wrong assignment of π in those clauses. Since that variable also survived the unassignment stage, it is also in Z. Therefore there are $\frac{pn^2}{10}$ clauses that x supports with respect to π and has another variable from Z in them. Since the set of clauses that x supports is disjoint from the set of clauses that y, a different variable in Z supports. There are at least $\frac{pn^2k}{10}$ of these clauses. □

Now, we are ready to prove Proposition 23.

Proof. (of Proposition 23) Given that we know by Lemma 25 there are at least $\frac{pn^2k}{10}$ clauses that contain at least 2 variables from Z, it suffices to plug in k instead of αn and $1/2$ instead of γ in Proposition 24. □

Now we turn to the last part of the proof.

3.8 Small connected components

We have proven that by the time we arrive to line 28 of the algorithm, we have a partial assignment that matches the planted assignment on the variables that it has assigned. In this section, we will show that the remaining variables constitute a set of isolated components of size $\Omega(\log n)$ in the graph of the formula ϕ. We will accomplish this by the following Proposition:

Proposition 26. *Let d and ϵ be such that $d \geq 70000$ and $\epsilon \in (0, 10^{-10}]$. For any $\alpha \in [4\exp\{-\frac{\epsilon^2 d}{90}\}, (\frac{\epsilon}{72})^4]$. Let $1 \geq \beta \geq c \cdot \frac{\log n}{(n\epsilon^6 d)}$ for some constant $c > 0$. Let $G = (Y, E')$ be the residual graph induced by Y. There is a constant $s > 0$, such that the probability that there is a connected component of size βn in G is at most $\exp\{-s \cdot (\epsilon^6 d)\beta n\}$.*

Before we prove this proposition, we need some preliminaries.

Definition 27 ([34])**.** *Let $\alpha < 1$ be some small constant independent of d and let σ be an assignment of variables of ϕ. We say that a set of variables C is a core of ϕ with respect to σ, if the following holds:*

1. *$|V|/|C| \leq \alpha$;*
2. *Every variable in C supports at least $(1-\epsilon)pn^2/9$ (for some small ϵ to be chosen later) clauses with respect to π in which the other two variables are also in C.*

Observe that the set of variables that remain assigned after lines 17 and 18 is a core set. The set H we construct below is a maximal core set, later we show that this is exactly the set constructed in lines 17 and 18. Let W be the set of variables in which σ disagrees with the planted solution τ, let B be the set of variables that support less than $(1 - \epsilon/2)pn^2/9$ clauses, and let $H \subseteq V$ be constructed as follows:

1. $H_0 = V \backslash (B \cup W)$;

2. while there exists a variable $x_i \in H_i$ which supports less than $(1-\epsilon)pn^2/9$ clauses in the subformula induced by H_i, define $H_{i+1} = H_i \setminus \{x_i\}$;
3. let x_m be the last variable removed at step 2; then set $H = H_{m+1}$.

The second property of a core set for H follows from the construction, the first property is proven by the following lemma:

Lemma 28. *Let* $\bar{H} = V \setminus H$. *Let* d *and* ϵ *be such that* $d \geq 70000$ *and* $\epsilon \in (0, 10^{-10}]$. *For any* $\alpha \in [4\exp\{-\frac{\epsilon^2 d}{90}\}, (\frac{\epsilon}{72})^4]$,

$$\Pr[|\bar{H}| \geq \alpha n] \leq \exp\left\{-\frac{\epsilon^2 d}{42}\alpha n\right\}.$$

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

$$\Pr[|\bar{H}| \geq \alpha n] \leq \Pr[|B \cup W| \geq \alpha n/2] + \Pr[\bar{H}^{it} \geq \alpha n/2 \big| |B \cup W| \leq \alpha n/2].$$

By Proposition 20 and 22 (here we assume $\alpha/2 > \exp\{-d/4000\}$) we have

$$\begin{aligned}\Pr[|B \cup W| \geq \alpha n/2] &\leq \Pr[|W| \geq \alpha n/4] + \Pr[|B| \geq \alpha n/4] \\ &\leq \exp\{-d/2 \times 10^{-10}\alpha n\} + \exp\{-\epsilon^2(d/40)\alpha n\} \\ &\underbrace{\leq}_{\epsilon \leq 10^{-10}} \exp\{-\frac{\epsilon^2 d}{41}\alpha n\}.\end{aligned}$$

To bound $\Pr[\bar{H}^{it} \geq \alpha n/2 \big| |B \cup W| \leq \alpha n/2]$, observe that every variable that is removed in iteration i (removing x_i) supports at least $\epsilon/2 \times d/9$ clauses in which at least one variable belongs to $\{x_1, x_2, ..., x_i\} \cup B \cup W$. Consider iteration $\alpha n/2$; note that we reach this iteration only if $\bar{H} \geq \alpha n$. By the end of this iteration, there is a set $\{x_1, x_2, ..., x_{\alpha n/2}\} \cup B \cup W$ of size at least αn such that its variables support at least $\frac{\epsilon}{2} \times \frac{pn^2}{9} \times \frac{\alpha n}{2} \geq \frac{\epsilon}{36} \times pn^2 \times \alpha n$ clauses with respect to the planted solution τ containing another variable from this set. By Proposition 24, using $\epsilon/36$ instead of ρ and $\alpha < (\epsilon/72)^4$, we have

$$\Pr[\bar{H}^{it} \geq \alpha n/2 \big| |B \cup W| \leq \alpha n/2] \leq \exp\{(-\frac{\epsilon}{36} \times pn^2 \times \log\frac{1}{4\alpha} \times \alpha n\} \leq \exp\{\epsilon d\alpha n\}.$$

Combining the two inequalities we get

$$\Pr[|\bar{H}| \geq \alpha n] \leq \exp\{-\epsilon d\alpha n\} + \exp\{-\epsilon^2(d/41)\alpha n\} \leq \exp\{-\epsilon^2(d/42)\alpha n\}.$$

The proposition is proved. □

Next, 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. We say that an edge of T is *covered* by T' if there is a clause in T' that contains both its endpoints. T' is a *minimal* no proper subset of T' covers T. By minimality,

$$|T'| \leq |V(T)| - 1. \tag{3.3}$$

Let I be the set of clauses in the formula ϕ. First, we bound the following probability $\Pr[T' \subseteq I \wedge V(T) \cap H = \emptyset]$. If this probability is small for every tree of size βn and every minimal set, it would provide a bound for the probability of the existence of a connected component of size βn in $G(\bar{H}, E)$. However, $T' \subseteq I$ and $V(T) \cap H = \emptyset$ are not independent events because H is not a random subset of V. Therefore instead of H and $V(T)$, we introduce different sets, H' and J, such that $\bar{H}'$ is small enough and J is big enough, so that

$$\Pr[T' \subseteq I \wedge V(T) \cap H = \emptyset] \leq \Pr[T' \subseteq I] \Pr[J \cap H' = \emptyset].$$

Let $J \subseteq V(T)$ be a set of variables of T which appear in at most 6 clauses of T'. Let $V(T')$ be the set of variables appearing in clauses of T'. Let H' be the set constructed by the following process:

- set W' to be the set of variables x such that $N^x_{\neq} - N^x_{=} < 13$;
- set B to be the set of variables that support less than $(1 - \epsilon/2)pn^2/9$ clauses in I;
- let $H' \subseteq V$ be constructed as follows:
    1. $H'_0 = V \setminus (B \cup W' \cup (V(T') \setminus J))$;
    2. while there exists a variable $x_i \in H'_i$ which supports less than $(1-\epsilon)pn^2/9$ clauses in $I[H'_i]$, define $H'_{i+1} = H'_i \setminus \{x_i\}$;
    3. let x_m be the last variable removed at step 2 and set $H' = H'_{m+1}$.

The bound 13 here is explained by the choice of set J. If a variable x appear in at most 6 clauses of T' (it is the definition of J) it means that for any set $F \subseteq I$, the number of neighbors of x in $F \cup T'$ is at most 6×2 greater than the number of neighbors of x in F. If the deviation is at least 13, adding T' to set of clauses would not change the neighborhood structure of x, i.e., if $N^x_{\neq} - N^x_{=} > 13$, then after adding T' the deviation is $N^x_{\neq} - N^x_{=} > 13 - 12 > 0$.

Lemma 29. *Let $\bar{H}'$ be as described above, let α, d, and ϵ be such that $\alpha \in [4\exp\{-\epsilon^2 d/4\}, (\epsilon/10)^4]$, and $d \geq 70000$, and $\epsilon \in (0, 10^{-10}]$. Let β be a constant such that $\beta \leq \alpha$ where βn is the size of $V(T)$. Then* $\Pr[|\bar{H}| \geq 4\alpha n] \leq \exp\{-(\epsilon^2 pn^2/24)\alpha n\}$.

Proof. Note that $|V(T') \setminus J| \leq |V(T')| \leq 3|V(T)| \leq 3\beta n \leq 3\alpha n$. Therefore, this lemma is essentially Lemma 28 with αn replaced with $3\alpha n$. If $|\bar{H}'| \geq 4\alpha n$, we can show that at least one of $W' \cup B$ and H'^{it} is of size at least $(4\alpha n - 3\alpha n)/2 = \alpha n/2$ and the proof of Proposition 28 is repeated. In proof of Proposition 20, we know that the difference between $N^x_{\neq} - N^x_{=}$ with respect to π is at least $13pn^2/100 - 13pn^2/100 = pn^2/100 \geq d/100 > 13$ (d is large enough). Therefore it would not change anything when replacing W' with W. □

Now we have to prove that J is big enough.

Proposition 30. $J \geq V(T)/2$.

Proof. For the sake of contradiction, assume that $J < V(T)/2$. This means there are more than $V(T)/2$ variables that appear in more than 6 clauses. Therefore, they appear in at least $\frac{1}{3} \times V(T)/2 \times 6 = V(T)$ clauses. But by definition of minimality, this cannot happen, since $V(T') \leq V(T) - 1$. The proposition is proved. □

Let F be any set of clauses, and let $H(F)$ (respectively, $H'(F)$) denote the set defined analogously to the set H (respectively, H') for the formula induced by F. It is not necessarily true that $H(F) \subseteq H(F \cup T)$, because some variable appearing in $H(F)$ might not be in $H(F \cup T)$ due to a change of the fraction of neighbors of those variables in F. However, it is always true that $H'(F) \subseteq H(F \cup T')$. This would help us bound the probability $\Pr[T' \subseteq I \wedge V(T) \cap H = \emptyset]$ by $\Pr[T' \subseteq I] \times \Pr[J \cap H' = \emptyset]$.

Lemma 31. $H'(F) \subseteq H(F \cup T')$

Proof. We will prove this lemma by induction on H_i and H'_i as they are defined above. First we show that $H'_0(F) \subseteq H_0(F \cup T')$. If $x \in H'_0(F)$ then $N^x_{\neq} - N^x_{=} > 13$, and x supports at least $(1-\epsilon)pn^2/9$ clauses. Since x appears in at most 6 clauses from T' and each clause adds at most 2 neighbors, the number of neighbors of x that T' can add is at most 12. Since $N^x_{\neq} - N^x_{=} > 13 - 12 \geq 1$, variable x is in H. Note the number of clauses x supports can only increase.

Suppose $H'_i(F) \subseteq H_i(F \cup T')$. If x supports at least $(1-\epsilon)pn^2/9$ clauses that contains only variables from $H'_i(F)$, then, as $H_i(F) \subseteq H_i(F \cup T')$ and $F \subseteq F \cup T'$, it supports the same clauses from $F \cup T'$ with only variables of $H_i(F \cup T')$. Hence if it is in $H'_{i+1}(F)$ it is also in $H_{i+1}(F \cup T')$. □

Lemma 32. $\Pr\left[(T' \subseteq I) \wedge (V(T) \cap H = \emptyset)\right] \leq \Pr[T' \subseteq I]\Pr[J \cap H' = \emptyset]$.

Proof. Since $J \subseteq V(T)$, we have:

$$\begin{aligned}\Pr\left[(T' \subseteq I) \wedge (V(T) \cap H = \emptyset)\right] &\underbrace{\leq}_{J \subseteq V(T)} \Pr\left[(T' \subset I) \wedge (J \cap H = \emptyset)\right] \\ &\leq \Pr[J \cap H = \emptyset | T' \subseteq I]\Pr[T' \subseteq I].\end{aligned}$$

Therefore, we only need to show that

$$\Pr[J \cap H = \emptyset | T' \subseteq I] \leq \Pr[J \cap H' = \emptyset].$$

Let $F' = F \setminus T'$ and let $F'' = F \cap T'$. We start with the right hand side:

$$\begin{aligned}\Pr[J \cap H' = \emptyset] &= \sum_{F : J \cap H'(F) = \emptyset} \Pr[I = F] \\ &\underbrace{\geq}_{H'(F) \subseteq H(F \cup T')} \sum_{F : J \cap H(F \cup T') = \emptyset} \Pr[I = F] \\ &\geq \sum_{F' : F' \cap T' = \emptyset, F'' \subseteq T', J \cap H'\left((F' \cup F'') \cup T'\right) = \emptyset} \Pr[I \backslash T' = F' \wedge I \cap T' = F''] \\ &\geq \sum_{F' : F' \cap T' = \emptyset, F'' \subseteq T', J \cap H'(F' \cup T') = \emptyset} \Pr[I \backslash T' = F' \wedge I \cap T' = F''].\end{aligned}$$

We can now split the sum into two parts:

$$\geq \sum_{F' : F' \cap T' = \emptyset, J \cap H'(F' \cup T') = \emptyset} \sum_{F'' \subseteq T'} \Pr[I \backslash T' = F' \wedge I \cap T' = F''].$$

where $I \cap T'$ and $I \cap T$ are disjoint sets. The clauses are chosen independently, therefore the event $I \backslash T' = F'$ is independent of the event $I \cap T' = F''$. Hence,

$$\begin{aligned}&\geq \sum_{F' : F' \cap T' = \emptyset, J \cap H'(F' \cup T') = \emptyset} \sum_{F'' \subseteq T'} \Pr[I \backslash T' = F'] \times \Pr[I \cap T' = F''] \\ &\geq \sum_{F' : F' \cap T' = \emptyset, J \cap H'(F' \cup T') = \emptyset} \Pr[I \backslash T' = F'] \underbrace{\sum_{F'' \subseteq T'} \Pr[I \cap T' = F'']}_{1} \\ &\geq \sum_{F' : F' \cap T' = \emptyset, J \cap H'(F' \cup T') = \emptyset} \Pr[I \backslash T' = F'].\end{aligned}$$

Again, $I \cap T'$ and $I \cap T'$ are disjoint sets, and the clauses are chosen independently, therefore the event $I \setminus T' = F'$ is independent of the event $T' \subseteq I$, and $\Pr[I \setminus T' = F'] = \Pr[I \setminus T' = F' | T' \subseteq I]$. We can rewrite the last expression as:

$$\geq \sum_{F':F' \cap T' = \emptyset, J \cap H'(F' \cup T') = \emptyset} \Pr[I \backslash T' = F' | T' \subseteq I]. \tag{3.4}$$

Now let us consider $\Pr[J \cap H = \emptyset | T' \subseteq I]$. We have

$$\Pr[J \cap H = | T' \subseteq I] = \sum_{F : J \cap H(F) = \emptyset} \Pr[I = F | T' \subseteq F]$$

$$= \sum_{F', F'' : J \cap H(F' \cup T') = \emptyset, F' \cup T' = \emptyset, F'' \subseteq T'} \Pr[I \backslash T' = F' \wedge I \cap T' = F'' | T' \subseteq F' \cup F''].$$

We have $F'' \subseteq T'$, $T' \subseteq F' \cup F''$ and $I \backslash (T' \cup F') = \emptyset$. From these we can conclude that it can only be $F'' = T'$ so we replace it with T' and get

$$= \sum_{F' : F' \cap T' = \emptyset, J \cap H'(F' \cup T') = \emptyset} \Pr[I \backslash T' = F' | T' \subseteq I]. \tag{3.5}$$

The concluding expression in (3.5) is the same as (3.4). Therefore,

$$\Pr[J \cap H = \emptyset | T' \subseteq I] \leq \Pr[J \cap H' = \emptyset].$$

□

The following lemma proves bounds which are needed to prove of Proposition 26.

Lemma 33. *(1) Let $\gamma \in [0,1]$ and $k < n/2$. Then $\binom{\gamma n}{k} / \binom{n}{k} \leq (6\gamma)^k$.*

(2) Let $\gamma \in [\beta, 1]$ and $\beta \leq \alpha < \frac{1}{2}$ such that $|H'| = (1-\gamma)n$. Then $\Pr[J \cap H' = \emptyset] \leq (6\gamma)^{\beta n/2}$.

(3) Let γ be as in (2). Then, $\Pr[T' \subset I \wedge V(T) \cap H = \emptyset] \leq (6\gamma)^{|T'|/2} p^{T'}$.

(4) Let T be a fixed tree with k vertices. The number of minimal sets of NAE-clauses that span T is at most $\sum_{s=0}^{k/2} \binom{k}{2} n^{k-2s-1}$.

Proof. (1) First observe that $\binom{\gamma n}{k} \leq (e\gamma n/k)^k$ and $\binom{n}{k} \geq (\frac{n-k}{k})^k \geq (n/2k)^k$. Therefore, $(e\gamma n/k)^k / (n/2k)^k \leq (6\gamma)^k$.

(2) Since H' is independent of J we have

$$\Pr[J \cap H' = \emptyset] \leq \frac{\binom{n-|J|}{|H'|}}{\binom{n}{|H'|}} = \frac{\binom{n-|H'|}{|J|}}{\binom{n}{|J|}} \leq (6\gamma)^{\beta n/2}.$$

(3) By Lemma 32, $\Pr[J \cap H = \emptyset | T' \subseteq I] \leq \Pr[T' \subseteq I] \times \Pr[J \cap H' = \emptyset]$. By part (2) $\Pr[J \cap H' = \emptyset] \leq (6\gamma)^{|T'|/2}$. Since the clauses are chosen independently, $\Pr[T' \subseteq I] \leq p^{|T'|}$.

(4) Let T' be a minimal set. We showed that a minimal set can contain at most $k-1$ clauses (inequality (3.3)). Any clause may cover up to 2 edges (otherwise T is not a tree). Let us say we have s clauses covering 2 edges and $k-1-2s$ clauses covering 1 edge. The total number of clauses is $k-s-1$. For every clause that covers 1 edge, there are at most $n-2$ ways to choose the third variable. The clause that covers 2 edges is of course has a only 1 way. We have at most $\binom{k}{2}$ ways to form the clauses that cover 2 edges. The variable s can range from 0 to $k/2$, as otherwise it covers more than k nodes. □

Recall that the number of spanning trees of an n-vertex clique K_n is n^{n-2} ([2]).

We are now ready to prove Proposition 26. Before we start, note that a sum like $\sum_0^{\beta/2}$ is always understood as a sum over all rational numbers ξ that are between 0 and $\beta/2$ of the form $\xi = \lceil \frac{s}{\beta n/2} \rceil$ where s is an integer.

Proof. (Proposition 26) It suffices to prove that the probability that there is a tree of size

βn in the residual graph G induced by $\bar{H}$ is small.

$$\Pr[\exists T,\ T \text{ is a tree of size } \beta n \in G] \leq \Pr[\exists T' \subseteq I, |V(T)| = \beta n, V(T) \cap H = \emptyset]$$

$$= \sum_{\rho=\beta}^{1} \Pr[\exists T' \subseteq I, |V(T)| = \beta n \wedge V(T) \cap H = \emptyset || \bar{H}'| = \rho n] \cdot \Pr[|\bar{H}'| = \rho n]$$

$$\leq \sum_{\rho=\beta}^{1} \Pr[|\bar{H}'| = \rho n] \sum_{s=0}^{\beta n/2} \underbrace{\binom{n}{\beta n} \cdot (\beta n)^{\beta n-2}}_{\text{number of trees}} \cdot \underbrace{n^{\beta n-2s-1}\binom{\beta n}{2}}_{\text{minimal set}} \cdot \underbrace{p^{\beta n-s-1} \cdot (6\rho)^{\beta n/2}}_{\Pr[T' \subset I \wedge V(T) \cap H = \emptyset]}$$

$$\leq \sum_{\rho=\beta}^{\max(16\exp\{-\epsilon^2 d/4\},\beta)} (6\rho)^{\beta n/2}(e\cdot pn^2)^{\beta n}\beta n/2 + \sum_{\rho=\max(16\exp\{-\epsilon^2 d/4\},\beta)}^{1} \Pr[|\bar{H}'| = \rho n]$$

$$\underbrace{\leq}_{\text{Lemma 29}} \sum_{\rho=\beta}^{\max(16\exp\{-\epsilon^2 d/4\},\beta)} (6\rho)^{\beta n/2}(e\cdot pn^2)^{\beta n}\beta n/2 + \sum_{\rho=\max(16\exp\{-\epsilon^2 d/4\},\beta)}^{\max(\epsilon/72)^4,\beta} \exp\{(\epsilon^2 pn^2/24)\rho n\}$$

$$+ \sum_{\rho=\max(\epsilon/72)^4,\beta}^{1} \exp\{-\Omega(\epsilon^6 d)n\}$$

$$\leq \sum_{\rho=\beta}^{\max(16\exp\{-\epsilon^2 d/4\},\beta)} (6\rho)^{\beta n/2}(e\cdot pn^2)^{\beta n}\beta n/2 + \sum_{\rho=\max(16\exp-\epsilon^2 d/4,\beta)}^{\max(\epsilon/10)^{12},\beta} \exp\{-(\epsilon^2 pn^2/9)\beta n\}$$

$$+ \sum_{\rho=\max\{(\epsilon/10)^{12},\beta\}} \exp\{-\Omega(\epsilon^6 pn^2)\beta n\}$$

$$\leq n\cdot\exp\{-\Omega(\epsilon^6 pn^2)\beta n\} \leq \exp\{-\Omega(\epsilon^6 pn^2)\beta n + \log n\}$$

$$\underbrace{\leq}_{\beta\in[\Theta(\log n/(n\epsilon^6 pn^2)),1],p\geq\frac{d}{n^2}} \exp\{-\Omega(\epsilon^6 d)\beta n\}.$$

The proposition is proved. □

Proposition 26 proves that the probability that there is a connected component of size greater than $\frac{c}{\epsilon^6 d}\log n$ is at most $\exp\{-c\cdot s\cdot\log n\}$. Indeed, assume that the size of the largest connected component is less than $\frac{c}{\epsilon^6 d}\log n$. The number of combinations the algorithm has to try for each connected components is at most $2^{\frac{c}{\epsilon^6 d}\log n} \leq n^{\frac{c}{\epsilon^6 d}}$ and the number of connected component is at most n so the algorithm runs in time $O(n^{\frac{c}{\epsilon^6 d}+1})$ after the unassigning step (line 28 to line 36) which is polynomial. It follows that the algorithm SDensity is also polynomial. Theorem 3 is proved.

Bibliography

[1] Dimitris Achlioptas. Lower bounds for random 3-SAT via differential equations. *Theor. Comput. Sci.*, 265(12):159–185, 2001.

[2] Martin Aigner and Günter M. Ziegler. *Proofs from the book.* Springer, 2004.

[3] Noga Alon and Nabil Kahale. A spectral technique for coloring random 3-colorable graphs (preliminary version). In *STOC '94: Proceedings of the twenty-sixth annual ACM symposium on Theory of computing*, pages 346–355, New York, NY, USA, 1994. ACM.

[4] Eli Ben-sasson, Yonatan Bilu, and Danny Gutfreund. Finding a randomly planted assignment in a random 3-CNF. In *In preparation*, 2002.

[5] Armin Biere, Marijn J. H. Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, page 980. IOS Press, February 2009.

[6] Archie. Blake. Canonical expressions in boolean algebra. *dissertation, Dept. of Mathematics*, 1937.

[7] Marco Cadoli, Marco Schaerf, Andrea Giovanardi, and Massimo Giovanardi. An algorithm to evaluate quantified boolean formulae and its experimental evaluation. *J. Autom. Reason.*, 28(2):101–142, 2002.

[8] Ming-Te Chao and John Franco. Probabilistic analysis of two heuristics for the 3-satisfiability problem. *SIAM J. Comput.*, 15(4):1106–1118, 1986.

[9] H. Chernoff. A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations. *Annals of Math. Stat.*, 23:493–509, 1952.

[10] S.A. Cook. The complexity of theorem-proving procedures. In *STOC '71: Proceedings of the third annual ACM symposium on Theory of computing*, pages 151–158, New York, NY, USA, 1971. ACM.

[11] Stephen A. Cook and David G. Mitchell. Finding hard instances of the satisfiability problem: A survey. pages 1–17. American Mathematical Society, 1997.

[12] Martin Davis, George Logemann, and Donald Loveland. A machine program for theorem-proving. *Commun. ACM*, 5(7):394–397, 1962.

[13] Martin Davis and Hilary Putnam. Computational methods in the propositional calculus. *Unpublished Report*, 1958.

[14] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *J. ACM*, 7(3):201–215, 1960.

[15] Olivier Dubois, Yacine Boufkhad, and Jacques Mandler. Typical random 3-sat formulae and the satisfiability threshold. In *SODA '00: Proceedings of the eleventh annual ACM-SIAM symposium on Discrete algorithms*, pages 126–127, Philadelphia, PA, USA, 2000. Society for Industrial and Applied Mathematics.

[16] Jack Edmonds. Paths, trees, and flowers. *Canadian Journal of Mathematics*, 17:449–467, 1965.

[17] Niklas Eén and Niklas Sörensson. Translating pseudo-boolean constraints into sat. *JSAT*, 2(1-4):1–26, 2006.

[18] Abraham Flaxman. A spectral technique for random satisfiable 3cnf formulas. In *SODA '03: Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 357–363, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics.

[19] J. Franco and M. Paull. Probabilistic analysis of the davis putnam procedure for solving the satisfiability problem. *Discrete Appl. Math.*, 5:7787, 1983.

[20] Jeremy D. Frank, Peter Cheeseman, and John Stutz. When gravity fails: Local search topology. *Journal of Artificial Intelligence Research*, 7:249–281, 1997.

[21] Michael R. Garey and David S. Johnson. *Computers and Intractability; A Guide to the Theory of NP-Completeness*. W. H. Freeman & Co., New York, NY, USA, 1990.

[22] Michel X. Goemans and David P. Williamson. Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *J. ACM*, 42(6):1115–1145, 1995.

[23] Venkatesan Guruswami. Inapproximability results for set splitting and satisfiability problems with no mixed clauses. In *APPROX '00: Proceedings of the Third International Workshop on Approximation Algorithms for Combinatorial Optimization*, pages 155–166, London, UK, 2000. Springer-Verlag.

[24] Djamal Habet, Chu Min Li, Laure Devendeville, and Michel Vasquez. A hybrid approach for sat. In *CP '02: Proceedings of the 8th International Conference on Principles and Practice of Constraint Programming*, pages 172–184, London, UK, 2002. Springer-Verlag.

[25] MohammadTaghi Hajiaghayi and Gregory B. Sorkin. The satisfiability threshold of random 3-SAT is at least 3.52. Technical Report, 2003.

[26] Philipp Hertel and Toniann Pitassi. Exponential time/space speedups for resolution and the pspace-completeness of black-white pebbling. In *FOCS '07: Proceedings of the 48th Annual IEEE Symposium on Foundations of Computer Science*, pages 137–149, Washington, DC, USA, 2007. IEEE Computer Society.

[27] Holger H. Hoos. A mixture-model for the behaviour of sls algorithms for sat. In *Eighteenth national conference on Artificial intelligence*, pages 661–667, Menlo Park, CA, USA, 2002. American Association for Artificial Intelligence.

[28] David J. Johnson and Michael A. Trick. *Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, Workshop, October 11-13, 1993.* American Mathematical Society, Boston, MA, USA, 1996.

[29] Alexis C. Kaporis, Lefteris M. Kirousis, and Efthimios G. Lalas. Selecting complementary pairs of literals. *Electronic Notes in Discrete Mathematics*, 16:1–24, 2004.

[30] Alexis C. Kaporis, Lefteris M. Kirousis, and Efthimios G. Lalas. The probabilistic analysis of a greedy satisfiability algorithm. *Random Struct. Algorithms*, 28(4):444–480, 2006.

[31] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[32] Lefteris M. Kirousis, Evangelos Kranakis, Danny Krizanc, and Yannis C. Stamatiou. Approximating the unsatisfiability threshold of random formulas. pages 253–269, 1996.

[33] Elias Koutsoupias and Christos H. Papadimitriou. On the greedy algorithm for satisfiability. volume 43, pages 53–55, Amsterdam, The Netherlands, The Netherlands, 1992. Elsevier North-Holland, Inc.

[34] M. Krivelevich and D. Vilenchik. Solving random satisfiable 3cnf formulas in expected polynomial time. In *SODA '06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 454–463, New York, NY, USA, 2006. ACM.

[35] Bertrand Mazure, Lakhdar Saïs, and Éric Grégoire. Boosting complete techniques thanks to local search methods. *Annals of Mathematics and Artificial Intelligence*, 22(3-4):319–331, 1998.

[36] Stephan Mertens, Marc Mézard, and Riccardo Zecchina. Threshold values of random k-sat from the cavity method. *Random Struct. Algorithms*, 28(3):340–373, 2006.

[37] David G. Mitchell, Bart Selman, and Hector J. Levesque. Hard and easy distributions of sat problems. In *AAAI*, pages 459–465, 1992.

[38] G. Nam, K. Sakallah, and R. Rutenbar. A boolean satisfiability-based incremental rerouting approach with application to FPGAs. In *DATE '01: Proceedings of the conference on Design, automation and test in Europe*, pages 560–565, Piscataway, NJ, USA, 2001. IEEE Press.

[39] Gi-Joon Nam, Fadi Aloul, Karem Sakallah, and Rob Rutenbar. A comparative study of two boolean formulations of FPGA detailed routing constraints. In *ISPD '01: Proceedings of the 2001 international symposium on Physical design*, pages 222–227, New York, NY, USA, 2001. ACM.

[40] Gi-Joon Nam, Karem A. Sakallah, and Rob A. Rutenbar. Satisfiability-based layout revisited: detailed routing of complex FPGAs via search-based boolean sat. In *FPGA '99: Proceedings of the 1999 ACM/SIGDA seventh international symposium on Field programmable gate arrays*, pages 167–175, New York, NY, USA, 1999. ACM.

[41] Christos M. Papadimitriou. *Computational complexity.* Addison-Wesley, Reading, Massachusetts, 1994.

[42] Steven Prestwich. Local search and backtracking vs non-systematic backtracking. In *In AAAI 2001 Fall Symposium on Using Uncertainty within Computation*, pages 109–115. AAAI Press, 2001.

[43] Irina Rish and Rina Dechter. To guess or to think? hybrid algorithms for SAT (extended abstract). In *In Proceedings of the International Conference on Principles and Practice of Constraint Programming (CP96)*, 1996.

[44] Gian-Carlo Rota and Kenneth Baclawski. *Introduction to Probability and Random Processes.* 1979.

[45] Tian Sang, Paul Beame, and Henry Kautz. Heuristics for fast exact model counting. In *In Proc. 8th International Conference on Theory and Applications of Satisfiability Testing*, pages 226–240, 2005.

[46] Bart Selman, Henry A. Kautz, and Bram Cohen. Noise strategies for improving local search. In *AAAI '94: Proceedings of the twelfth national conference on Artificial intelligence (vol. 1)*, pages 337–343, Menlo Park, CA, USA, 1994. American Association for Artificial Intelligence.

[47] Bart Selman, Hector Levesque, and David Mitchell. A new method for solving hard satisfiability problems. *AAAI*, pages 440–446, 1992.

[48] Guilhem Semerjian. On the freezing of variables in random constraint satisfaction problems. *Journal of Statistical Physics*, 130(2), 2007.

[49] Claude Elwood Shannon. A symbolic analysis of relay and switching circuits. *Masters thesis*, 1940.

[50] Alfred Tarski. On undecidable statements in enlarged systems of logic and the concept of truth. *J. Symb. Log.*, 4(3):105–112, 1939.

[51] Alfred Tarski. A problem concerning the notion of definability. *Journal of symbolic logic*, 13(2):107–111, 1948.

[52] Willard. Van Orman Quine. A proof procedure for quantification theory. *J. Symbolic Logic*, 20:141149, 1955.

[53] R. Glenn Wood and Rob A. Rutenbar. FPGA routing and routability estimation via boolean satisfiability. In *FPGA '97: Proceedings of the 1997 ACM fifth international symposium on Field-programmable gate arrays*, pages 119–125, New York, NY, USA, 1997. ACM.

[54] Hui Xu, Rob A. Rutenbar, and Karem Sakallah. sub-sat: a formulation for relaxed boolean satisfiability with applications in routing. In *ISPD '02: Proceedings of the 2002 international symposium on Physical design*, pages 182–187, New York, NY, USA, 2002. ACM.

[55] Lintao Zhang, Conor F. Madigan, Matthew H. Moskewicz, and Sharad Malik. Efficient conflict driven learning in a boolean satisfiability solver. In *ICCAD '01: Proceedings of the 2001 IEEE/ACM international conference on Computer-aided design*, pages 279–285, Piscataway, NJ, USA, 2001. IEEE Press.

[56] Uri Zwick. Approximation algorithms for constraint satisfaction problems involving at most three variables per constraint. In *SODA '98: Proceedings of the ninth annual ACM-SIAM symposium on Discrete algorithms*, pages 201–210, Philadelphia, PA, USA, 1998. Society for Industrial and Applied Mathematics.