THEORETICAL ANALYSIS OF PRACTICAL HEURISTICS FOR SATISFIABILITY

by

Evgeny Skvortsov

B.Sc., Ural State University, 2001

M.Sc., Ural State University, 2003

A THESIS SUBMITTED IN PARTIAL FULFILLMENT
OF THE REQUIREMENTS FOR THE DEGREE OF
DOCTOR OF PHILOSOPHY
in the School
of
Computing Science

APPROVAL

Name:	Evgeny Skvortsov
Degree:	Doctor of Philosophy
Title of Thesis:	Theoretical Analysis of Practical Heuristics for Satisfiability

Examining Committee: Dr. Richard Vaughan
Chair

Dr. Andrei Bulatov, Senior Supervisor

Dr. David Mitchell, Supervisor

Dr. Funda Erugn, SFU Examiner,

Dr. Evgeny Dantsin, External Examiner,
Professor of Computing Science,
Roosevelt University

Date Approved: 6th Nov, 2009

Abstract

The SATISFIABILITY problem (SAT) is one of the central subjects of research in modern computing science. This problem provides a rich language to model practical problems. Moreover modern SAT solvers are capable of solving large instances of SAT and can be successfully applied to industrial scale problems. Since SAT is NP-complete the success of the practical SAT algorithms apparently contradicts the belief that NP-complete problems are hard to solve. One of the most promising approaches to explaining this phenomenon is the typical case analysis.

Local search principles are actively used for practical SAT solving, as well as for many other important problems. In this work we perform a typical case analysis of the basic Local Search algorithm on Random Planted 3-SAT. We show that a phase transition of the effectiveness of the Local Search on Random Planted 3-SAT occurs at density $\rho = \frac{7}{6}\ln n$. That is, for any constant ε the algorithm with high probability solves instances of density $(\frac{7}{6}+\varepsilon)\ln n$ and with high probability fails for instances of density $(\frac{7}{6}-\varepsilon)\ln n$.

The first successful practical algorithm based on local search principles, GSAT, was proposed in 1991 by Selman, Levesque and Mitchell. In fact this algorithm is nothing more than basic Local Search enhanced with plateau moves. At the time the algorithm was proposed, it was outperforming state of the art systematic search solvers and it continues serving a basis for development of efficient local search algorithms. We analyze GSAT theoretically on Random Planted 3-SAT and show that it can solve Random Planted 3-SAT of any density ρ such that $\rho > \kappa \ln n$ for some constant κ. This theoretical result agrees with, and partially explains, the empirical observation that adding plateau moves dramatically improves Local Search.

Finally we propose a Weighted Random Walk algorithm. The algorithm is obtained by adding a simple weighting scheme to the well known Random Walk algorithm. We prove that Weighted Random Walk with high probability gives a good approximation. Moreover, in experiments this simple algorithm solves Random Planted 3-SAT for any constant density.

To my grandmother Galina Efimovna and grandfather Evgeny Emeljanovich.

Acknowledgments

I would like to thank my senior supervisor Dr. Andrei Bulatov for the exciting joint work, continuous support of the research and strong encouragement to pay attention to a physical activity as well as to mental. I am also grateful to my supervisor Dr. David Mitchell for multiple fruitful discussions of the topic of the thesis and encouragement to pursue research in the area of heuristics analysis.

I owe particular thanks for Dr. Funda Ergun, Dr. Evgeny Dantsin and Dr. Richard Vaughan, for their questions and comments that helped me to look at the problem in a more broad context.

I am grateful to Ehsam Amiri for the fruitful interesting joint work and many discussions on the topic of the thesis.

It is a pleasure to thank those who helped me to keep connection to the practical areas of computer science while working on the thesis: Duncan Phillips, Ian Andrew Bell, Derek Ferguson and Nick Arden.

I thank all my friends for their support, especially Yaroslav Litus, Bradley Coleman and Javier Thaine for numerous discussions of the work.

Special thanks are owned to my wife Natalia Skvortsova and my parents Irina Skvortsova and Sergej Skvortsov for their continuous support and encouragement.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Historical Perspective

The SATISFIABILITY (or SAT) problem is important from both theoretical and practical points of view. In this problem we are given a Boolean formula in CNF and the question is whether there is an assignment of values to variables that satisfies the formula. SAT is one of the first problems that was proven to be NP-complete [14]. On the other hand, its practical importance is also unquestionable since this problem is actively used as a framework to model many practical problems, such as hardware and software verification [13, 51, 9], scheduling [52], some problems arising in bioinformatics [47, 10] etc.

SAT is a decision problem, so a SAT algorithm must either prove that a solution exists or prove that the CNF is unsatisfiable. But if there is no satisfying assignment one may wish to find an assignment that satisfies the maximum number of clauses. The optimization problem of finding such an assignment is called MAXIMUM SATISFIABILITY or MAX-SAT. In case we know that a solution exists and want to find one, it may be convenient to consider the SAT problem as MAX-SAT, since then optimization techniques can be applied to it. Algorithms that work for only satisfiable instances are called *incomplete*.

A k-CNF is a CNF formula in which each clause has at most k literals, SAT restricted to k-CNFs is called k-SAT. Since for $k \geq 3$ a CNF can be easily transformed into an equivalent k-CNF, by introducing a linear number of auxiliary variables, k-SAT for $k \geq 3$ is also an NP-complete problem. In practice it is often the case that the length of clauses in the CNF is bounded above by some small constant, so it is natural to model such classes of problems by k-SAT rather than general SAT. Consequently k-SAT, and its simplest NP-complete subclass 3-SAT, are being extensively studied in the worst and typical cases.

1.1.1 Solving SATISFIABILITY

Great progress has been made in practical solving SATISFIABILITY problems in the last two decades. Some problems that were considered to be intractable in the early 1990's can be solved in less than a second by modern solvers [51, 7].

There are two major classes of algorithms used in SAT solving: *systematic search* and *local search*. Systematic search algorithms explore the whole solution space by starting with all variables unassigned and then assigning them one by one. If at some point no variable can be assigned without making some clause false the algorithm backtracks. Systematic search algorithms are described most naturally using recursion. The basic systematic search algorithm works as follows. Given a boolean formula φ it picks a random variable x_i, and obtains two smaller formulas φ_0 and φ_1 by assigning x_i to 0 and 1 respectively. For instance when we assign $x_i = 1$ we can remove from φ all clauses that contain x_i and we remove all literals $\neg x_i$. If an empty clause appears after such an assignment then it means that the formula is made false and further exploration of the case $x_i = 1$ is not required. Then the algorithm is recursively applied to φ_0 and φ_1. If, say, φ_0 happens to be satisfiable then the satisfying assignment for φ can be obtained from the satisfying assignment of φ_0 and $x_i = 0$. If both φ_0 and φ_1 are unsatisfiable then φ is also unsatisfiable.

Modern systematic search algorithms are based on the Davis-Putnam-Logemann-Loveland (DPLL) procedure [18]. *Pure literal heuristic* and *unit propagation* were used in DPLL to improve the performance of the systematic search. Modern solvers use *clause learning* for further speed up. We say that a literal x_i (or $\neg x_i$) is a pure literal in φ if no clause in φ contains literal $\neg x_i$ (x_i respectively). It is obvious that if we have a pure literal in φ then we can assign its variable to satisfy this literal without risk of loosing solutions. So that is what the pure literal heuristic does.

Unit propagation looks for a clause in φ that contains only one literal (*a unit clause*). If such clause is found then the only variable occurring in this clause is assigned the value to satisfy the literal (and the clause). Again we are guaranteed that this action will not make us lose any satisfying assignments. It may happen that after pure literal rule or unit propagation were executed another pure literal or unit clause are generated and we have a kind of a chain reaction that simplifies the formula substantially.

Clause learning works as follows. When a contradiction is reached and the algorithm needs to backtrack, it tries to find a small set of variables which assignment made the rest of the formula unsatisfiable and records them as a new clause. For instance if assigning $x_1 = 1, x_2 = 1, x_3 = 0$ leads to a sequence of unit propagations and pure literal assignments that results in a generation of the empty clause then we can conclude that no assignment that contains $x_1 = 1, x_2 = 1, x_3 = 0$ can satisfy the formula. Thus we can add the clause $(\neg x_1 \wedge \neg x_2 \wedge x_3)$ to φ. Adding new clauses in turn increases chances that unit propagation will be triggered more often and so fewer cases have to be considered.

Local search algorithms are inherently incomplete. They explore the solution space partially and aim at finding a satisfying assignment, without trying to prove that such an assignment does not exist. The basic Local Search algorithm (also known as Iterative Improvement [32]) starts with a random assignment and then works as follows. At each step it computes for each variable what will be the change in the number of satisfied clauses if this variable is *flipped* (i.e. the value of the variable is changed). Let U be a set of all variables for which flipping increases the number of satisfied clauses. The algorithm picks a variable uniformly at random from U, flips it and goes to the next step.

Local Search will fail to find a satisfying assignment if it gets trapped in a local maximum. That is, where the current assignment is such that flipping any variable can only decrease the number of satisfied clauses, or leave it unchanged. *Plateau moves* is a natural strategy that can be used to find a better assignment after a local maximum is reached. If there is no variable that we can flip to increase the number of satisfied clauses then we can consider the set of variables that does not change this number and pick a variable to flip from it. It may happen that after some steps we come to an assignment that will not be a local maximum. Enhancement of the basic Local Search algorithm with plateau moves results in algorithms that are called GSAT and CSAT.

The GSAT algorithm is widely known for being the first successful practical heuristic developed for SAT that was based on local search principles. It was proposed in the early 90s by Selman, Levesque and Mitchell [54]; the name GSAT stands for "**G**reedy algorithm for **SAT**". This algorithm starts with a random assignment and then tries to improve the assignment greedily. That is, at every step it flips one of the variables that give the maximum increase in the number of satisfied clauses (the maximum possible increase can happen to be zero or even negative). If a satisfying assignment is not found after a certain number of steps then the algorithm restarts. The number of steps to be performed before a restart and the number of restarts to be done are two parameters of the GSAT algorithm that can be tuned to improve performance on a specific class of problems one is interested in. It is demonstrated in the paper where GSAT is introduced [54] that this algorithm outperforms state-of-the-art systematic search algorithms of that time. Extensive empirical analysis of GSAT was carried out by Gent et al. [24].

Later many algorithms were built on the basis of GSAT. In particular Gent and Walsh have experimentally demonstrated [26, 25] that greediness is not very important for the success of GSAT. The algorithm they designed, CSAT, works as the basic Local Search at assignments that are not local maxima and switches to plateau moves when a local maximum is reached.

Local Search, GSAT and CSAT explore the solution space by making moves that do not decrease the number of satisfied clauses[1]. A quite different approach is used in the Random Walk [50] algorithm that was proposed by Papadimitriou [50]. This algorithm flips a variable to satisfy one of the unsatisfied clauses even if that causes a substantial decrease in the total number of satisfied clauses. While the basic Local Search algorithm is generic and can be applied to optimization of any value function, Random Walk was developed specifically for SAT. Random Walk also starts from a random assignment and then at every step an unsatisfied clause is selected uniformly at random, a literal is selected uniformly at random from the clause and the variable corresponding to the literal is flipped. The Random Walk algorithm was used as a basis for the practical WalkSat algorithm by Kautz and Selman [53]. WalkSat picks an unsatisfied clause uniformly at random and then for each of the variables in the clause it computes how many clauses will become unsatisfied if this variable is flipped. It flips the variable that will cause the least number of clauses to become unsatisfied. WalkSat happened to be very successfully at planning problems. WalkSat successfully solves SAT encodings of hard planning problems faster than the best domain specific algorithms that existed when WalkSat was introduced.

Many local search algorithms have been built upon GSAT and WalkSat (see Chapter 6 of [8] for survey). Those algorithms are more sophisticated and more powerful. We believe that theoretical analysis of these algorithm is crucial for further progress in solving SAT. And yet there is still lack of theoretical understanding of the success even of the most simple Local Search algorithms: GSAT and WalkSat.

1.1.2 Algorithms Analysis

Worst-case analysis of SAT algorithms deals with proving polynomial upper bounds for the algorithms applied to subclasses of SAT, exponential bounds and approximation results. Since SAT is NP-complete there is not much hope of designing a polynomial (or even sub-exponential [33]) algorithm solving it in the general case.

The best known algorithm [17] for solving the general case of SAT takes $2^{n-o(n)}$ polynomial time steps. Finding an algorithm that can solve SAT exponentially faster than in 2^n steps or proving

[1]Though flips that decrease the number of satisfied clauses can be done by GSAT they do not play substantial role in the success of GSAT [26].

that such algorithm does not exist is a well known open problem. Upperbounds of the form $c^n, c < 2$, were obtained for many important large subclasses of SAT. For instance, algorithms working within such bounds were found for k-SAT [16], problems with bounded ratio of the number of clauses to the number of variables [42], problems with bounded number of positive (or negative) occurrences of variables [34] etc. These results are proven by providing a *splitting algorithm*, i.e. an algorithm that reduces a SAT instance to subproblems of smaller size. Depending on the size of these subproblems the number of such recursive calls required to solve the problems differs. There are techniques [44] that allow bounding this number by analyzing the sizes of the subproblems. To apply these techniques, size of a formula does not have to be defined as the number of clauses or number of variables, but can be an arbitrary positive function of the formula.

A great amount of work [62, 35, 6] has been done in the area of approximating the solution of the MAX-SAT problem. The best algorithm for the general MAX-SAT problem so far, allows to get a 0.77 approximation of the optimum [6]. The case of 3-SAT is especially interesting in the context of approximating the solution. It is not hard to see that the expected number of clauses satisfied by a random assignment equals $\frac{7}{8}$ of the total number of clauses in the formula. On the other hand Håstad's celebrated result [29] states that, assuming $P \neq NP$, there is no deterministic algorithm that could be guaranteed to achieve a better approximation of 3-SAT. This surprising result was obtained using the notion of *Probabilistically Checkable Proof (PCP)* and the famous PCP-theorem [5]. Thus the question of optimal polynomial time approximability of 3-SAT in the worst case is essentially closed. And consequently we have that for general SAT the approximation ratio that can be achieved in polynomial time is bounded between 0.77 and $\frac{7}{8} = 0.875$.

The worst case performance of GSAT has been studied in two contexts. Let us recall that the algorithm can be tuned by setting the number of steps done before a restart and the number of restarts. So the algorithm can be used as a polynomial time algorithm as well as exponential. Worst case efficiency of GSAT in polynomial settings applied to the k-SAT problem is not better than the performance of the basic Local Search algorithm. It guarantees only $\frac{k}{k+1}$-th fraction of clauses to become satisfied [28, 46], which is also always achievable by the basic Local Search. The worst case efficiency of GSAT and CSAT as exponential algorithms for formulas of bounded clause to variable ratio was studied by Hirsch. In particular it was shown that if the number of restarts is $2^{cn}, c < 1$, and each restart makes n steps then CSAT succeeds as a Monte-Carlo randomized algorithm. This upper bound was obtained for its execution without use of plateau moves, that is essentially the bound was proven for the Local Search with restarts.

This work is devoted to the **typical case analysis** of SAT algorithms. In this approach we assume that problem instances are generated according to some probabilistic distribution. Then we can try to prove that *with high probability* (i.e. with probability that tends to 1 as the size of the problem goes to infinity), the algorithm solves the problem or gives a good approximation of the optimal solution. Performance of the algorithm with respect to the typical case can be much better than it is in the worst case.

Until the early 1990's it appeared that SAT is a very easy problem in the typical case and several very simple algorithms were proposed for it (see for example [27, 31, 37]). In 1992, it was argued by Selman, Levesque and Mitchell [48, 55] that SAT formulas that are hard on average, can be generated by picking some natural distributions and parameter values. According to the proposed model *random k-SAT of fixed density* (or simply *Random k-SAT*), the problem is sampled as follows: given a *density* ρ and a number of variables n, a formula is picked uniformly at random among all CNFs containing ρn k-clauses over the n variables, where a k-clause is a clause containing exactly k literals corresponding to k distinct variables.

A randomly generated 3-SAT problem may happen to have a solution or to be unsatisfiable. Thus it is a natural question to ask what is the probability that a Random 3-SAT of density ρ is satisfiable. It was experimentally shown that there is a threshold $\rho_0 \approx 4.25$, such that if $\rho < \rho_0$, then with high probability the problem has a solution and otherwise with high probability there is no solution. More experimental evidence of this phenomenon was published shortly thereafter, see e.g. [15, 45]. In 1994 Kirkpatrik and Selman noted that this threshold behavior of the system is known in statistical physics as a *phase transition*.

The term "phase transition" in statistical physics generalizes the phase transition of medium between solid, liquid, gas and plasma states. Intuitively speaking, a system demonstrates the phase transition phenomenon with respect to parameter p if for some values small change of p leads to a dramatic change of the behavior of the system. Statistical physics has certain techniques for experimental data analysis that indicate whether a system demonstrates the phase transition at some point or not. Those techniques were used by Kirkpatrik and Selman in their experimental research of 3-SAT.

In 1999 Freidgut [22] succeeded in rigorously proving the existence of a function $\rho_0^k = \rho_0^k(n)$ such that for $\rho < \rho_0^k(n)$ a Random k-SAT has a solution with high probability and for $\rho > \rho_0^k(n)$ it does not. Plenty of results were bounding bounding the range where the transition happens in 3-SAT from below [57, 56, 11, 1, 38] and above [36, 59, 39, 19]. The best lower bound at the moment [38] is 3.520 and the best upper bound [19] is 4.506.

All lower bounds for the phase transition were obtained by providing an algorithm and then proving that with high probability it solves the problems of density below the bound. A breakthrough in this area is due to Achlioptas in 2000. He unified all previously known algorithms in a so called *Card Game* framework [2]. These algorithms are essentially different versions of DPLL algorithms without backtracking. Since there is no backtrack, once a variable gets its value it never changes it. Thus once a variable is assigned the formula can be simplified by removing unsatisfied literals corresponding to the variable and clauses that became satisfied by that assignment. In the Card Game framework a state of the run of the algorithm is represented as a layout of cards. Each clause is represented as a column of cards, each of which corresponds to a literal. So when the algorithm decides what variable to assign a value for, it can either pick the variable uniformly at random among those which are still unassigned, or pick a size of the clause and then pick a variable that corresponds to a random literal in a random clause of this size. Once the variable is assigned the cards that correspond to unsatisfied literals and columns that correspond to satisfied clauses are removed. If the algorithm is acting within the Card Game framework its state can be described as $k + 2$ numbers. Namely numbers of l-clauses for $1 \leq l \leq k$, the number of satisfied clauses and the number of unsatisfied clauses. This framework is somewhat restricting but still allows a wide range of algorithms to be implemented. For instance the Pure Literal heuristic does not fit into this framework, but the Unit Propagation algorithm does.

To analyze an algorithm represented in the Card Game framework Achlioptas used Wormald's theorem. This theorem allows to make the transition from a discrete stochastic process to a deterministic system of differential equations. The obtained system of differential equations can be studied by numerical methods. The Wormald's theorem was originally applied in the analysis of algorithms on random graphs and the Card Game framework made it possible to use it for SAT algorithms. The intuitive idea behind the theorem is that discrete parameters of a large system behave almost as if they were continuous. For instance if we plot a graph of the number of satisfied clauses versus time for Local Search, then as the number of the variables grows the graph will look more and more like a graph of a smooth function. Wormald's theorem provides a rigorous basis for using this phenomenon.

When it was shown that Random 3-SAT for densities close to 4.25 was hard for existing algorithms it became a natural problem to find out whether an efficient algorithm exists for this distribution of 3-SAT or not. Statistical physics analysis of the 3-SAT solution space resulted in development of the *Survey Propagation* [51] algorithm that is highly efficient on random 3-SAT instances in experiments. The algorithm is capable of finding solutions of large (around 10^6 variables) instances of random 3-SAT with density close to the threshold ρ_0, but is yet to be rigorously analyzed and that seems to be very hard.

A *random 3-SAT of fixed density with a planted solution* (or simply *Random Planted 3-SAT* [3]) distribution is easier to solve and it provides a model for overconstrained [60] practical problems. To generate a k-CNF according to this distribution one first picks some assignment of values to the variables. Then a formula of *density* ρ is sampled uniformly at random from all formulas with ρn clauses that are satisfied by that assignment. Experimental data suggest that random planted 3-SAT is easier than random 3-SAT: modern practical algorithms are capable of solving it for any density.

It was first shown by Flaxman in 2003 that Random Planted 3-SAT of high constant density can be solved in polynomial time [21]. A specific algorithm, named *Spectral Heuristic* was developed to prove that. The algorithm deals with a graph G of co-occurenses of literals, i. e. a graph in which vertices are literals and two literals are connected by an edge if they occur in the same clause. It was observed that graph G has negative eigenvalues and that the eigenvectors corresponding to the most negative eigenvalue of G can be used to get a solution of the problem.

Later Feige and Vilenchik [58] developed another algorithm with similar performance on Random planted 3-SAT. The algorithm consists of two stages. First a value for each variable is selected so as to satisfy as many literals as possible. That is, the variable is assigned 1 or 0 based on whether it occurs more often in positive or negative literals respectively. Next a specific kind of a Local Search algorithm called the *k-opt heuristic* is executed. At each step the algorithm has an assignment $\vec{v}$ and computes the set S of all clauses of the formula that are satisfied by $\vec{v}$. Then all subsets of variables of size at most logarithm of the total number of variables are considered. A variable is flipped if that results in an assignment that satisfies a set of clauses S' such that $S' \supset S$.

The k-opt heuristic algorithm is more intuitive than the spectral heuristic, but still uses quite different ideas than the practical algorithms. Next we discuss typical case analysis of practical algorithms. We focus on two heuristics used in practical local search solvers to escape local maxima and boost algorithm performance: plateau moves and flipping of a variable occurring in an unsatisfied clause.

The typical case performance of the GSAT algorithm for random planted 3-SAT was studied by Koutsoupias and Papadimitriou [40]. It was shown that for linear density $\rho = \kappa n$ GSAT succeeds with high probability. As in the proof of the upper bound of the time complexity in [30] plateau and downward moves were not used in the analysis. Also the authors conjectured that their upper bound is not tight and that GSAT can solve random planted 3-SAT for a large constant density.

Later Gent [23] adapted techniques of Koutsoupias and Papadimitriou [40] to show that even for the *Stupid Algorithm* (i. e. an algorithm that assigns 1 or 0 to a variable depending on whether this variable occurs more often in positive or negative literals) there exists a large constant κ_0 such that this algorithm can solve Random Planted 3-SAT for density $\kappa \ln n$, for any $\kappa > \kappa_0$.

The Random Walk algorithm was initially proposed by Papadimitriou [50] and in the same paper it was also shown that it can find solution for any 2-SAT problem in expected quadratic time. The algorithm was also analyzed theoretically in the typical case and tested for 3-SAT by Alekhnovich and Ben-Sasson [3]. They proved that the algorithm solves Random 3-SAT in linear time for densities lower than 1.6 and there is experimental evidence that the algorithm succeeds for densities up to 2.7. On the other hand they proved that there exists a constant c such that RW does not solve instances with planted solution of density greater than c.

1.2 Definitions

1.2.1 Basic Notions

A 3-CNF is a conjunction of *3-clauses* i.e. clauses with exactly 3 literals. As we consider only 3-CNFs, we will always call them just clauses. Depending on the number of negated literals, we distinguish 4 types of clauses: $(-,-,-)$, $(+,-,-)$, $(+,+,-)$, and $(+,+,+)$. If φ is a 3-CNF over variables $x_1,\ldots,x_n$, an *assignment* for these variables is a Boolean n-tuple $\vec{u} = (u_1,\ldots,u_n)$, so the value of x_i is u_i. Let $\vec{v}_i'$ be a vector obtained from $\vec{v}$ by flipping the i-th coordinate. Statements "c is a clause in φ" and "clause c is satisfied by $\vec{v}$" we denote by $c \in \varphi$ and $c(\vec{v})$ respectively.

The *density* of a 3-CNF φ is the number $\frac{m}{n}$ where m is the number of clauses, and n is the number of variables in φ. The *uniform* distribution of 3-CNFs of density ρ (density may be a function of n), $\Phi(n, \rho n)$ is the set of all 3-CNFs containing n variables and ρn clauses together with the uniform probability distribution on this set. To sample a 3-CNF according to $\Phi(n, \rho n)$, one chooses uniformly and independently ρn clauses out of the $2^3\binom{n}{3}$ possible clauses. Thus, we allow repetitions of clauses, but not repetitions of variables within a clause. *Random 3-SAT* is the problem of deciding the satisfiability of a 3-CNF randomly sampled accordingly to $\Phi(n, \rho n)$. For short, we will call such a random formula a 3-CNF from $\Phi(n, \rho n)$.

The *uniform planted* distribution of 3-CNF of density ρ is constructed as follows. First, choose at random a Boolean n-tuple $\vec{u}$, the *planted* satisfying assignment. Then let $\Phi^{\mathsf{plant}}(n, \rho n, \vec{u})$ be the uniform probability distribution over the set of all 3-CNFs over variables $x_1,\ldots,x_n$ with density ρ and such that $\vec{u}$ is a satisfying assignment. For our purposes we can always assume that $\vec{u}$ is the all-ones tuple, that is a 3-CNF belongs to $\Phi^{\mathsf{plant}}(n, \rho n, \vec{u})$ if and only if it contains no clauses of the type $(-,-,-)$. We also simplify the notation $\Phi^{\mathsf{plant}}(n, \rho n, \vec{u})$ by $\Phi^{\mathsf{plant}}(n, \rho n)$. To sample a 3-CNF according to $\Phi^{\mathsf{plant}}(n, \rho n)$ one chooses uniformly and independently ρn clauses out of $7\binom{n}{3}$ possible clauses of types $(+,-,-),(+,+,-)$, and $(+,+,+)$. *Random Planted 3-SAT* is the problem of deciding the satisfiability of a 3-CNF from $\Phi^{\mathsf{plant}}(n, \rho n)$.

The problems *Random MAX-3-SAT* and *Random Planted MAX-3-SAT* are the optimization versions of Random 3-SAT and Random Planted 3-SAT. The goal in these problems is to find an assignment that satisfies as many clauses as possible. In particular we study Local Search and its practical modification GSAT algorithm. Local Search terminates when it reaches a *local maximum* of the function

$$\mathcal{V}^{\varphi}(\vec{x}) = |\{c \in \varphi \mid c(\vec{x})\}|.$$

That is when the assignment $\vec{v}$ under consideration is such that there is no variable which can be flipped to increase the number of satisfied clauses. In this case we will just say that $\vec{v}$ is a local maximum of φ. The function $\mathcal{V}^{\varphi}$ is easy to compute and we will use it in the pseudocode.

Below we define several notions assuming that φ is sampled according to probability distribution $\Phi(n, \rho n)$. The definitions are analogous for $\varphi \in \Phi^{\mathtt{plant}}(n, \rho n)$. By saying that a statement $\mathcal{E}(\varphi)$ is true *with high probability* for $\varphi \in \Phi(n, \rho n)$ we mean that probability of event $\mathcal{E}(\varphi)$ tends to 1 for $\varphi \in \Phi(n, \rho n), n \to \infty$. We shall also use the standard acronym "whp". For arbitrary functions $f(n), g(n)$ we denote equality $f(n) = g(n) + o(n)$ by $f(n) \approx g(n)$ and we write $f(n) \lesssim g(n)$ if inequality $f(n) \leq g(n)$ holds for all large enough n.

Let φ be sampled according to probability distribution $\Phi(n, \rho n)$ and let $\mathcal{E}$ be some event. We say that the probability distribution $\Phi(n, \rho n)$ demonstrates a *phase transition with respect to* $\mathcal{E}$ if there is a function $\rho_0(n)$ such that for any $\varepsilon > 0$, for $\rho > \rho_0(1+\varepsilon)$ we have $\mathbf{P}(\mathcal{E}) \longrightarrow 1$, as n goes to infinity, and for $\rho < \rho_0(1-\varepsilon)$ we have $\mathbf{P}(\mathcal{E}) \longrightarrow 0$. In the 3-SAT phase transition we have $\mathcal{E} =$ "formula φ is unsatisfiable". The function $\rho_0(n)$ is called *the phase transition threshold*. Note that the phase transition threshold is not necessarily a constant. We also apply the notion of phase transition to the performance of Local Search on Random Planted 3-SAT, and in this case we have $\mathcal{E} =$ "Local Search solves φ".

1.2.2 Classical Local Search algorithms

Local Search, GSAT, CSAT. A formal description of the Local Search algorithm (LS) is given in Fig. 1.1. Observe that LS stops when it reaches a local maximum in the number of unsatisfied clauses. We shall also study One Pass Local Search (OLS), a simplified version of LS. Like LS, OLS flips variables that give an increase in the number of satisfied clauses, but it considers any variable only once (see Fig. 1.2).

```
INPUT: 3-SAT formula $\varphi$ over variables $x_1, \ldots, x_n$.
OUTPUT: Boolean $n$-tuple $\vec{u}$, which is a local maximum of $\varphi$.
METHOD:
  pick uniformly at random a Boolean $n$-tuple $\vec{u}$
  let $U = \{x_i \mid \mathcal{V}(\vec{u}_i') > \mathcal{V}(\vec{u})\}$
  while $U$ is not empty
    pick uniformly at random a variable $x_j$ from $U$
    change the value of $x_j$ in $\vec{u}$
    recompute $U$
  return $\vec{u}$
```

Figure 1.1: The basic Local Search algorithm (Iterative Improvement)

```
INPUT: 3-SAT formula $\varphi$ over variables $x_1, \ldots, x_n$.
OUTPUT: Boolean $n$-tuple $\vec{u}$, which is a local maximum of $\varphi$.
METHOD:
  pick uniformly at random a Boolean $n$-tuple $\vec{u}$
  let $U = \{x_i \mid \mathcal{V}(\vec{u}_i') > \mathcal{V}(\vec{u})\}$
  for $i$ in $1, \ldots n$
    if $x_i$ belongs to $U$
    change the value of $x_i$ in $\vec{u}$
    recompute $U$
  return $\vec{u}$
```

Figure 1.2: One Pass Local Search algorithm

Given an assignment $\vec{u}$ and a clause c it will be convenient to say that c *votes* for a variable x_i to have value 1 if c contains literal x_i and its other two literals are unsatisfied. In other words if either (a) c is not satisfied by $\vec{u}$, and it will be satisfied if the value of x_i is changed, or (b) the only literal in c satisfied by $\vec{u}$ is x_i. Similarly, we say that c votes for x_i to have value 0 if c contains the negation of x_i and its other two literals are not satisfied. Using this terminology we define the set U (see Fig. 1.1, 1.2) as the set of all variables such that the number of votes received to change the current value is greater than the number of those to keep it.

The GSAT algorithm is presented in Fig. 1.3. The CSAT algorithm is similar to GSAT, but without greediness as shown in Fig. 1.4. In this work we are interested in these algorithms as decision algorithms rather than optimization algorithms, so to keep the pseudocode simple we return *fail* rather then remembering and returning the best assignment that was considered.

```
INPUT: A 3-CNF $\varphi$, integers MAXTRIES, MAXFLIPS
OUTPUT: fail or an assignment $\vec{v}$ that satisfies $\varphi$
METHOD:
  do MAXTRIES times
    pick uniformly at random a Boolean $n$-tuple $\vec{v}$
    do MAXFLIPS times
      if $\vec{v}$ satisfies $\varphi$ then return $\vec{v}$
      pick a variable $x_i$ such that $\mathcal{V}(\vec{v}'_i)$ is maximal uniformly at random
      let $\vec{v} = \vec{v}'_i$
  return fail
```

Figure 1.3: The GSAT algorithm

Random Walk. Random Walk (see Fig. 1.5) is a very simple algorithm proposed by Papadimitriou [50]. It starts with a random assignment and at every step an unsatisfied clauses is picked uniformly at random. Then a literal in the clause is selected uniformly at random and the corresponding variable is flipped.

1.3 Main Results

In this work we make a contribution into understanding the effectiveness of practical local search heuristics. Our first result is a typical case study of the performance of the basic Local Search for Random 3-SAT and Random Planted 3-SAT. For Random 3-SAT of arbitrary constant density ρ we show that basic Local Search does not find an optimal solution but returns an assignment that satisfies $c\rho n$ clauses. The constant c is less than $\frac{7}{8}$. Thus Local Search typically achieves a result which is NP-hard to achieve in the worst case. While Random 3-SAT becomes unsatisfiable when ρ is greater than the phase transition threshold (≈ 4.25) the Random Planted 3-SAT gets easier since counting the number of positive and negative occurrences of variables can help with assigning them correct values. We discovered that basic Local Search does not solve Random Planted 3-SAT for any constant density. But for the density of the form $\rho = \kappa \ln n$ basic Local Search has a phase transition at $\kappa = 7/6$. Namely for constant $\kappa < 7/6$ basic Local Search whp does not solve the Random 3-SAT and for $\kappa > 7/6$ whp it does.

We generalize the Card Game [2] to model One-Pass Local Search (OLS), a restricted version of the Local Search (LS) algorithm that considers each variable only once, and prove the following

```
INPUT: 3-SAT formula $\varphi$ over variables $x_1, \dots, x_n$, integers MAXFLIPS, MAXTRIES
OUTPUT: Boolean $n$-tuple $\vec{v}$, which is a local maximum of $\varphi$.
METHOD:
  let $U = \{x_i \mid \mathcal{V}(\vec{v}_i') > \mathcal{V}(\vec{v})\}$
  let $U_0 = \{x_i \mid \mathcal{V}(\vec{v}_i') = \mathcal{V}(\vec{v})\}$
  do MAXTRIES times
    pick uniformly at random a Boolean $n$-tuple $\vec{v}$
    do MAXFLIPS times
      if $U \cup U_0$ is empty
        return $fail$
      if $U$ is not empty
        pick uniformly at random a variable $x_j$ from $U$
      else
        pick uniformly at random a variable $x_j$ from $U_0$
      change the value of $x_j$ in $\vec{v}$
      recompute $U$ and $U_0$
      if $\vec{v}$ satisfies $\varphi$
        return $\vec{v}$
  return $fail$
```

Figure 1.4: The CSAT algorithm.

Theorem 1 *For any positive ρ there is a constant ω such that for a random 3-CNF $\Phi(n, \rho n)$ whp the OLS algorithm finds an assignment such that the number of satisfied clauses equals $\omega n + o(n)$.*

Then we build a more sophisticated model to be able to get a system of differential equations describing the work of the Local Search algorithm. To apply Worlmald's theorem to that model we rely on a certain assumption (See Assumption 1 in Section 3.1). Intuitively the assumption states that at each step of the algorithm the pair of the formula and the assignment remains random given the parameters of the process we are tracking.

We use this assumption in the following theorem.

Theorem 2 *If Assumption 1 is true then for any positive ρ there is a constant ω such that for a random 3-CNF $\Phi(n, \rho n)$ almost surely the LS algorithm finds an assignment such that the number of satisfied clauses equals $\omega n + o(n)$.*

The existence of a phase transition in performance of the Local Search applied to Random Planted 3-SAT is formally stated in the following theorem:

Theorem 3 *(1) Let $\rho \geq \kappa \cdot \ln n$, and $\kappa > \frac{7}{6}$. Then Local Search whp finds a solution of Random Planted 3-SAT of density ρ.*

```
INPUT: A CNF $\varphi$ containing $n$ variables, integers MAXFLIPS
OUTPUT: An assignment $\vec{x}$
METHOD:
  let $\vec{x}$ be a random vector.
  for step from 1 to MAXFLIPS do
    pick a random unsatisfied clause $C$ in $\varphi$
    for each variable $x_j$ in $C$ do
      let $x_j = \neg x_j$
  return $\vec{x}$
```

Figure 1.5: The Random Walk Algorithm

(2) Let $c \leq \rho \leq \kappa \cdot \ln n$, where c is an arbitrary positive constant, and $0 < \kappa < \frac{7}{6}$. Then Local Search whp does not find a solution of Random Planted 3-SAT of density ρ.

Next we move to the analysis of the Local Search enhanced with plateau moves. All our proofs work for both GSAT and CSAT, and for simplicity we formulate theorems for GSAT. It follows from Corollary 1 (see Section 4.2) that for any finite density ρ the GSAT algorithm satisfies $c_1 n, c_1 > 0$ more clauses than LS. The gain in performance from plateau moves is more impressive for Random Planted 3-SAT. While LS has a phase transition at $\frac{7}{6} \ln n$, GSAT solves Random Planted 3-SAT for any logarithmic density. This is stated formally in the following theorem.

Theorem 4 *For any $\kappa > 0$ GSAT with settings* MAXFLIPS$= n^{60/\kappa+3}$, MAXTRIES$= 1$ *finds a solution for $\varphi \in \Phi^{\texttt{plant}}(n, \rho n), \rho = \kappa \ln n$ whp.*

The second heuristic we analyze in this work uses dynamic weights on variables. We introduce and study Weighted Random Walk, a modification of the well known Random Walk algorithm. In this algorithm we favor selection of variables for a flip that were flipped recently. To do that we assign each variable a positive integer weight and flip a variable only if its weight is one. Similarly to the Random Walk we pick an unsatisfied clause uniformly at random. For each variable in the clause if weight is strictly greater than one we decrease the weight by one and if the weight is exactly one we flip the variable. Every step two variables are selected uniformly at random and have their weights increased by one. Pseudocode for the algorithm is given in Fig. 1.6. Weighted Random Walk with setting MAXWEIGHT$= 1$ turns into Random Walk. We increase weights of two variables at each step and the number 2 appears to be arbitrary. We could make the number of variables that have their weight increased to be a tuning parameter of the algorithm, but the analysis is the easiest if the number is two.

```
INPUT: A CNF $\varphi$ containing $n$ variables, integers MAXFLIPS, MAXWEIGHT
OUTPUT: An assignment $\vec{x}$
METHOD:
  let $\vec{x}$ be a random vector.
  let $w(i) = 1$, for $i \in \{1, \ldots, n\}$
  for step from 1 to MAXFLIPS do
     pick a random unsatisfied clause $C$ in $\varphi$
     for each variable $x_j$ in $C$ do
        let $w(j) = w(j) - 1$
        if $w(j) = 0$:
          let $x_j = \neg x_j, w(j) = 1$
     pick two random variables, and for each of them do
        if its weight is less than MAXWEIGHT then increase it by one
  return $\vec{x}$
```

Figure 1.6: The Weighted Random Walk Algorithm

Our experiments suggest that Weighted Random Walk (WRW) performs well at finding solutions of random planted instances of 3-SAT of any fixed density. This is in contrast with Alekhnovich and Ben-Sasson's exponential lower bound for the running time of the standard random walk algorithm for solving random planted 3-SAT of density larger than a constant [3]. Random Planted 3-SAT is not a hard distribution for modern practical algorithms, but Weighted Random Walk is much simpler and is easier to analyze theoretically.

We prove that Weighted Random Walk solves the Full CNF (CNF consisting of all clauses that are satisfied by the planted solution) and for random CNFs with planted solution of unbounded densities $\rho = \rho(n) \underset{n\longrightarrow\infty}{\longrightarrow} \infty$ for any $\varepsilon > 0$ whp it finds an assignment that differs from the planted solution on at most ε-fraction of all variables.

Theorem 5 *Let φ be a random 3-CNF with a planted solution $\vec{r}$ of density $\rho = \rho(n) \underset{n\longrightarrow\infty}{\longrightarrow} \infty$, and let $\varepsilon > 0$ be some constant. With high probability WRW with* MAXWEIGHT ≥ 5 *finds a vector that differs from $\vec{r}$ in at most εn coordinates.*

Chapter 2

Probabilistic tools

In this chapter we formulate several statements regarding properties of stochastic processes that will be used throughout the work.

2.1 Chernoff Bounds

Let $B(p, n)$ be random variable, that is the number of successes in n independent trials. If p is the probability of success in each trial, then the following inequality is known as Chernoff Bound [49]

$$\mathbf{P}\left(\left|\frac{B(p,n)}{n} - p\right| \leq \varepsilon\right) \leq 2e^{-\frac{\varepsilon^2 n}{3p}}.$$

Next we prove a lemma that generalizes the Chernoff bound to linear combinations of binomial random variables.

Lemma 1 *Let r, s be integers, $\theta < 1$ a positive real, and let $\alpha_1, \ldots, \alpha_r, \beta_1, \ldots, \beta_s$ be some real constants. There are constants λ and C such that we have*

$$\mathbf{P}\left(X > Y\right) < Ce^{-\lambda \mathbf{E}(Y)} \tag{2.1}$$

for any random variables X and Y such that $\mathbf{E}\left(X\right) < \theta \mathbf{E}\left(Y\right)$ and $X = \sum\limits_{i=0}^{r} \alpha_i X_i, Y = \sum\limits_{i=0}^{s} \beta_i Y_i$ for some binomial random variables $X_1, \ldots, X_r, Y_1, \ldots, Y_s$.

Proof. Let $\xi = \frac{1-\theta}{(r+s)\max(\max(\alpha_i),\max(\beta_i))}$. It is easy to see that event $X > Y$ implies occurrence of at least one of the events from the set

$$\mathcal{S} = \{\{X_i \geq \mathbf{E}\left(X_i\right) + \xi \mathbf{E}\left(Y\right)\}_{i\in\{0,\ldots,r\}}, \{Y_i \leq \mathbf{E}\left(Y_i\right) - \xi \mathbf{E}\left(Y\right)\}_{i\in\{0,\ldots,s\}}\}.$$

Indeed, inequality $X < Y$ follows from inequalities

$$\begin{aligned} X_i &< \mathbf{E}(X_i) + \xi \mathbf{E}(Y)_{i \in \{0,\dots,r\}} \\ Y_i &> \mathbf{E}(Y_i) - \xi \mathbf{E}(Y)_{i \in \{0,\dots,s\}} \\ \mathbf{E}(X) &< \theta \mathbf{E}(Y). \end{aligned}$$

Application of Chernoff bound to each of X_i and Y_i gives us inequalities

$$\begin{aligned} \mathbf{P}(|X - \mathbf{E}(X_i)| > \xi \mathbf{E}(Y)) &< e^{-\mathbf{E}(X_i)\xi^2 \left(\frac{\mathbf{E}(Y)}{\mathbf{E}(X_i)}\right)^2/3} \le e^{-\xi^2 \mathbf{E}(Y)\theta^{-2}/3}, \\ \mathbf{P}(|Y - \mathbf{E}(Y_i)| > \xi \mathbf{E}(Y)) &< e^{-\mathbf{E}(Y_i)\xi^2 \left(\frac{\mathbf{E}(Y)}{\mathbf{E}(Y_i)}\right)^2/3} \le e^{-\xi^2 \mathbf{E}(Y)/3}. \end{aligned}$$

Thus if we set $\lambda = \xi^2/3$ and $C = r + s$ then using the union bound we can conclude that inequality (2.1) holds. □

2.2 Azuma's Inequality

We say that a sequence of random variables $X_0, X_1, \dots$ is a *supermartingale* (or *submartingale*) if for any k we have $E(X_{k+1}|X_1, \dots, X_k) < X_k$ (or $E(X_{k+1}|X_1, \dots, X_k) > X_k$ respectively).

Let $X_0, X_1, \dots$ be a submartingale and c, Δ constants such that for each k, $|X_k - X_{k-1}| \le c, \mathbf{E}(X_k - X_{k-1}) \ge \Delta$. Then according to Azuma's inequality [49] for all t and any λ we have

$$\mathbf{P}(X_t - X_0 \le \lambda) \le 2e^{-\frac{(t\Delta - \lambda)^2}{2tc^2}}.$$

The following Observation can be easily done using the Azuma's inequality for supermartingales (see Lemma 1 from [61]).

Observation 1 *(1) Let Y_t be a supermartingale such that $\mathbf{E}(Y_{t+1}|Y_t) \le Y_t$ and $|Y_{t+1} - Y_t| < c$ for some c. Then $\mathbf{P}(Y_t - Y_0 \ge bc) \le e^{-\frac{b^2}{2t}}$, for any $b > 0$.*

(2) This inequality implies that if $\mathbf{E}(Y_{t+1}|Y_t) < Y_t - d$ and $|Y_{t+1} - Y_t| < c \le 1$ then the process $Z_t = Y_t - dt$ is a supermartingale and we have the following inequality

$$\mathbf{P}(Y_t - Y_0 \ge bc) = \mathbf{P}\left(Z_t - Z_0 \le \left(b + \frac{dt}{c}\right)\right) \le e^{-\frac{(b+dt)^2}{2tc^2}} \le e^{-bd}. \tag{2.2}$$

2.3 Random Graphs

We use some standard probabilistic tools of random graph analysis which can be found in the book [4].

Let φ be a 3-CNF with variables $x_1,\ldots,x_n$. The *primal graph* $G(\varphi)$ of φ is the graph with vertex set $\{x_1,\ldots,x_n\}$ and edge set $\{x_ix_j \mid \text{literals containing } x_i,x_j \text{ appear in the same clause}\}$. The *hypergraph* $H(\varphi)$ *associated with* φ is a hypergraph, whose vertices are the variables of φ and edges are the 3-element sets of variables belonging to the same clause. Note that if $\varphi\in\Phi^{\mathsf{plant}}(n,\rho n)$ or $\varphi\in\Phi(n,\rho n)$ then $H(\varphi)$ is sampled uniformly at random among all 3-hypergraphs with n vertices and ρn edges.

We will need the following properties that are possessed by graphs of logarithmic and smaller density.

Lemma 2 *Let $\rho<\kappa\ln n$ for a certain constant κ, and let $\varphi\in\Phi^{\mathsf{plant}}(n,\rho n)$.*

(1) For any $\alpha<1$, whp all the subgraphs of $G(\varphi)$ induced by at most $O(n^\alpha)$ vertices have average degree less than 5.

(2) The probability that $G(\varphi)$ has a vertex of degree greater than $\ln^2 n$ is $o(n^{-3})$.

Proof. (1) This part of the lemma is very similar to Proposition 13 from [20], and is proven in a similar way. Let U be a fixed set of variables with $|U|=\ell$. The number of 3-element sets of variables that include 2 variables from U is bounded from above by

$$\binom{\ell}{2}(n-2)\le\frac{1}{2}\ell^2n.$$

For each of them the probability that this set is the set of variables of one of the random clauses chosen for φ (we ignore the type of the clause) equals

$$\frac{\kappa n\ln n}{\binom{n}{3}}=\frac{6\kappa\ln n}{(n-1)(n-2)}.$$

Thus, the probability that 2ℓ of them are included as clauses is at most

$$\binom{\frac{1}{2}\ell^2n}{2\ell}\left(\frac{6\kappa\ln n}{(n-1)(n-2)}\right)\le\left(3e\kappa\cdot\frac{\ell\ln n}{n}\right)^{2\ell}.$$

Let $d=e(3e\kappa)^2$. Using the union bound, the probability that there exists a required set U with at most n^α variables is at most

$$\begin{aligned}
&\sum_{\ell=2}^{n^\alpha} \binom{n}{k} \left(\sqrt{\frac{d}{e}}\frac{\ell \ln n}{n}\right)^{2\ell} \\
&\quad\leq \sum_{\ell=2}^{n^\alpha} \left(\frac{ne}{\ell}\cdot\frac{d}{e}\cdot\frac{\ell^2 \ln^2 n}{n^2}\right)^{\ell} \\
&\quad\leq \sum_{\ell=2}^{n^\alpha} \left(d\frac{n^\alpha \ln^2 n}{n}\right)^{\ell} \\
&\quad= (dn^{\alpha-1}\ln^2 n)^2 \frac{1-(dn^{\alpha-1}\ln n)^{\ell-1}}{1-dn^{\alpha-1}\ln n} \\
&\quad= O(n^{2\alpha-2}\ln^4 n).
\end{aligned}$$

(2) The probability that the degree of a fixed vertex is at least $\ln^2 n$ is bounded from above by

$$\left(\frac{1}{n}\right)^{\ln^2 n} \binom{3\kappa n \ln n}{\ln^2 n} \leq n^{-\ln^2 n} \left(\frac{3e\kappa n \ln n}{\ln^2 n}\right)^{\ln^2 n} = \left(\frac{3e\kappa}{\ln n}\right)^{\ln^2 n},$$

where $n^{-\ln^2 n}$ is the probability that some particular $\ln^2 n$ random clauses include x, and $\binom{3\kappa n \ln n}{\ln^2 n}$ is the number of $\ln^2 n$-element sets of clauses. Then it is not hard to see that

$$n\left(\frac{3e\kappa}{\ln n}\right)^{\ln^2 n} \longrightarrow 0,$$

as n goes to infinity. □

2.4 Wormald's Theorem

The key tool in our analysis in Chapters 3 and 6 is the theorem by Wormald [61] that allows one to replace probabilistic analysis of a combinatorial algorithm with analysis of a deterministic system of differential equations.

All random processes we consider are discrete time random processes. Such a process is a probability space Ω denoted by $(Q_0, Q_1, \ldots)$, where each Q_i takes values in some set S. Consider a sequence Ω_n, $n = 1, 2, \ldots$, of random processes. The elements of Ω_n are sequences $(q_0(n), q_1(n), \ldots)$ where each $q_i(n) \in S$. For convenience the dependence on n will usually be dropped from the notation. Asymptotics, denoted by the notation o and O, are for $n \to \infty$, but uniform over all other variables. For a random X, we say $X = o(f(n))$ *always* if $\max\{x | \mathbf{P}\,(X = x) \neq$

$0\} = o(f(n))$. We denote by S^+ the set of all $h_t = (q_0, \dots, q_t)$, each $q_t \in S$ for $t = 0, 1 \dots$. By H_t we denote the *history* of the processes, that is the $n \times (t+1)$-matrix with entries $Q_i(j)$, $0 \le i \le t, 1 \le j \le n$. A function $f(u_1, \dots, u_j)$ satisfies *Lipschitz condition* on $D \subseteq \mathbb{R}^j$ if a constant $L > 0$ exists with the property that

$$|f(u_1, \dots, u_j) - f(v_1, \dots, v_j)| \le L \sum_{i=1}^{j} |u_j - v_i|$$

for all $(u_1, \dots, u_j)$ and $(v_1, \dots, v_j)$ in D.

Theorem (Wormald, [61]) *Let k be fixed. For $1 \le \ell \le k$, let $y^{(\ell)} \colon S^+ \to \mathbb{R}$ and $f_\ell \colon \mathbb{R}^{k+1} \to \mathbb{R}$, such that for some constant C and all ℓ, $|y^{(\ell)}| < Cn$ for all $h_t \in S^+$ for all n. Suppose also that for some function $m = m(n)$:*

(i) for all ℓ and uniformly over all $t < m$, $\mathbf{P}\left(|Y_{t+1}^{(\ell)} - Y_t^{(\ell)}| > n^{1/5} \mid H_t\right) = o(n^{-3})$ always;

(ii) for all ℓ and uniformly over all $t < m$,
$\mathbf{E}(Y_{t+1}^{(\ell)} - Y_t^{(\ell)} \mid H_t) = f_\ell(t/n, Y_t^{(1)}/n, \dots, y_t^{(k)}/n) + o(1)$ always;

(iii) for each ℓ the function f_ℓ is continuous and satisfies a Lipschitz condition on D, where D is some bounded connected open set containing the intersection of $\{(t, z^{(1)}, \dots, z^{(k)}) \mid t \ge 0\}$ with some neighbourhood of $\{(0, z^{(1)}, \dots, z^{(k)}) \mid \mathbf{P}\left(Y_0^{(\ell)} = z^{(\ell)}n, 1 \le \ell \le k\right) \neq 0$ for some $n\}$.

Then:

(a) For $(0, \hat{z}^{(1)}, \dots, \hat{z}^{(k)}) \in D$ the system of differential equations
$\dfrac{dz_\ell}{ds} = f_\ell(s, z_1, \dots, z_k)$, $\ell = 1, \dots, k$, has a unique solution in D for $z_\ell \colon \mathbb{R} \to \mathbb{R}$ passing through $z_\ell(0) = \hat{z}^{(\ell)}$, $1 \le \ell \le k$, and which extends to points arbitrarily close to the boundary of D.

(b) Whp $Y_t^{(\ell)} = nz_\ell(t/n) + o(n)$ uniformly for $0 \le t \le \min\{\sigma n, m\}$ and for each ℓ, where $z_\ell(s)$ is the solution in (a) with $\hat{z}^{(\ell)} = Y_0^{(\ell)}/n$, and $\sigma = \sigma(n)$ is the supremum of those s to which the solution can be extended.

This theorem was proven by Wormald [61] and was originally used for typical case analysis of algorithms on random graphs. Later it was successfully used by Achlioptas [1] to prove lower bounds for the Random 3-SAT phase transition threshold.

Chapter 3

Local Search in Uniform Random 3-SAT

In this chapter we study performance of the basic Local Search algorithm and its simplified version One-Pass Local Search in application to the Random 3-SAT problem. We show that the processes of the execution of LS and OLS can be described by systems of differential equations. Though the algorithms are very similar their analysis differs substantially.

In this chapter we use the following trick to simplify the process of the SAT algorithm execution. To avoid keeping track of the assignment of values of variables we shall assume that the assignment is all-ones. Then when a variable x_t needs to be flipped we shall replace all literals x_t with $\neg x_t$ and vice versa.

3.1 One Pass Local Search

One Pass Local Search is a simplified version of the Local Search algorithm where each variable is considered only once (see fig. 1.2). In order to analyze the algorithm we are going to modify the Card Game framework [2]. In this section we develop a system of differential equations and prove that it describes behavior of the OLS.

3.1.1 Model

To analyze the OLS algorithm we use an extended version of the Card Game framework [2]. Every clause of CNF Φ is represented by three cards. At step t the intermediate opens all cards with x_t or $\neg x_t$ and also tells us the 'polarity' of the remaining literals in the clauses containing $x_t, \neg x_t$ (that is how many of them are negative). Then we compare the numbers $a(x_t)$ of clauses containing $\neg x_t$, the remaining literals of which are negative, and $b(x_t)$ of clauses containing x_t, the remaining literals of which are negative. If $a(x_t) > b(x_t)$ then we flip x_t replacing everywhere x_t with $\neg x_t$

and $\neg x_t$ with x_t. Finally we remove clauses containing x_t and remove $\neg x_t$. If in the latter case a clause becomes empty we count it as unsatisfied. Note that in contrast to the card games used in [2], in the described game we have some information on the unopened cards, and therefore the formula obtained on each step is not quite random. Thus a more thorough analysis is required.

Such an analysis can be done by monitoring the dynamics of eight sets of clauses that we define at each step of the algorithm. Let Φ_t denote the formula at the start of step t. Variables (and the corresponding literals) from the set $\{X_1, \dots, X_{t-1}\}$ will be called *processed* (they cannot change anymore), the remainining variables will be called *unprocessed*. We define the following 8 sets:

- $E_\varnothing$ is the set of all clauses in Φ_t that do no contain processed literals;
- E_1 is the set of all clauses in Φ_t containing a positive processed literal;
- E_0 is the set of all clauses in Φ_t that contain three negated processed literals;
- E_{++}, E_{--}, E_{+-} are the sets of all clauses in Φ_t that contain one negated processed literal and two positive, two negative, or a positive and negative unprocessed literals, respectively;
- E_+, E_- are the sets of all clauses in Φ_t containing two processed negative literals, and a positive, or a negative unprocessed literal, respectively.

We will denote the sizes of these sets by $e_\varnothing, e_1, e_0, e_{++}, e_{+-}, e_{--}, e_+, e_-$ respectively, and the vector $(e_\varnothing, e_1, e_0, e_{++}, e_{+-}, e_{--}, e_+, e_-)$ by $\mathbf{e}$. These numbers will be our random variables from Wormald's theorem. All these values depend on t, but we always refer to them at the current step t, and so drop t from the notation. We also use v to denote $n - t + 1$ (the number of steps remaining).

It is easy to see that clauses that once enter E_0 or E_1 never leave these sets, and that at each step for each clause that doesn't belong to $E_0 \cup E_1$ there is a chance to get to E_1. The other possible transitions of clauses between the sets are shown on Figure 3.1.

If $E_\diamond$ and $E_\star$ are some of the eight sets, then we will denote conditional probability for a clause to move from set $E_\diamond$ to $E_\star$ by $\mathbf{P}\left(E_\diamond \to E_\star\right)$, assuming a certain particular value of vector $\mathbf{e}$.

We will compute the probability that variable x_t is flipped at step t. This event happens when there are more unsatisfied clauses containing this variable (we denoted the set of such clauses by $A(x_t)$) than clauses that are satisfied only by x_t (we denoted this set by $B(x_t)$). Clauses from sets E_-, E_{--} and $E_\varnothing$ can fall into set $A(x_t)$, while clauses from sets E_+, E_{+-} and $E_\varnothing$ can fall into $B(x_t)$. The probability that a clause from E_- belongs to $A(x_t)$ equals $\frac{1}{v}$, this is the probability that x_t is written on the only card currently unrevealed in the clause. In a similar way we compute such

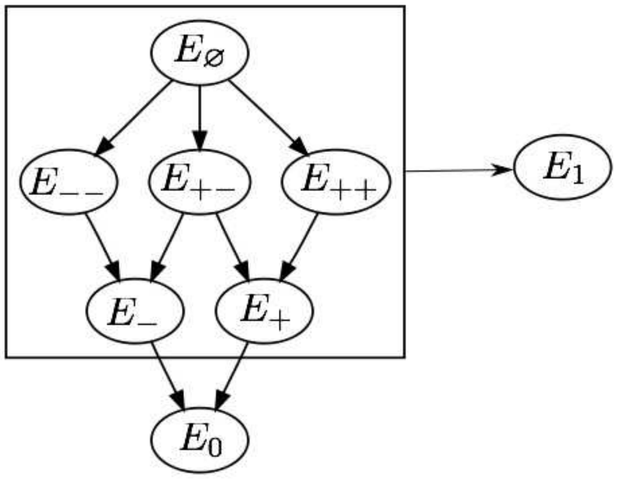


Figure 3.1: Flow diagram

probabilities for clauses from E_{--} and $E_\varnothing$, which are $\frac{2}{v}$ and $\frac{3}{8v}$ respectively. The probabilities that a clause from E_+, E_{+-}, and $E_\varnothing$ belongs to $B(x_t)$ equal $\frac{1}{v}$, $\frac{1}{v}$, and $\frac{3}{8v}$ respectively. Note that for different clauses the considered events are independent.

Now let $F(n_1, n_2, n_3, p_1, p_2, p_3)$ denote the event that exactly n_1, n_2, and n_3 clauses from E_-, E_{--}, and $E_\varnothing$ respectively belong to $A(x_t)$, and exactly p_1, p_2, and p_3 clauses from E_+, E_{+-} and $E_\varnothing$ belong to $B(x_t)$. By the Bernoulli formula we have

$$\begin{aligned}\mathbf{P}\left(F(n_1,n_2,n_3,p_1,p_2,p_3)\right) &= \binom{e_+}{p_1}\left(\frac{1}{v}\right)^{p_1}\binom{e_{+-}}{p_2}\left(\frac{1}{v}\right)^{p_2}\binom{e_\varnothing}{p_3}\left(\frac{3}{8v}\right)^{p_3}\\ \times &\quad \binom{e_-}{n_1}\left(\frac{1}{v}\right)^{n_1}\binom{e_{--}}{n_2}\left(\frac{2}{v}\right)^{n_2}\binom{e_\varnothing}{n_3}\left(\frac{3}{8v}\right)^{n_3}\end{aligned}$$

As n tends to infinity, the Bernoulli distribution tends to the Poisson distribution and we have

$$\begin{aligned}\mathbf{P}\left(F(n_1,n_2,n_3,p_1,p_2,p_3)\right) &= \left(\frac{e_+}{v}\right)^{p_1}\left(\frac{e_{+-}}{v}\right)^{p_2}\left(\frac{3e_\varnothing}{8v}\right)^{p_3}\left(\frac{e_-}{v}\right)^{n_1}\\ \times &\quad \left(\frac{2e_{--}}{v}\right)^{n_2}\left(\frac{3e_\varnothing}{8v}\right)^{n_3}\frac{e^{\frac{e_+}{v}+\frac{e_{+-}}{v}+\frac{3e_\varnothing}{8v}+\frac{e_-}{v}+\frac{2e_{--}}{v}+\frac{3e_\varnothing}{8v}}}{n_1!n_2!n_3!p_1!p_2!p_3!}+O\left(\frac{1}{n}\right)\end{aligned}$$

The probability that x_t is flipped can then be calculated as follows:

$$\begin{aligned}
&\mathbf{P}\left(x_t \text{ is flipped}\right) = \mathbf{P}\left(|A(x_t)| > |B(x_t)|\right) \\
&= \mathbf{P}\left(\bigvee_{p_1+p_2+p_3<n_1+n_2+n_3} |A(x_t)| = n_1+n_2+n_3 \;\&\; |B(t)| = p_1+p_2+p_3\right) \\
&= \sum_{n_1+n_2+n_3>p_1+p_2+p_3} \mathbf{P}\left(F(n_1,n_2,n_3,p_1,p_2,p_3)\right)
\end{aligned}$$

It will be convenient for us to denote the sum similar to that appearing in the last line of the equation above, but over $n_1, n_2, n_3, p_1, p_2, p_3$ satisfying a certain condition Ξ, by $S(\Xi)$. Using this notation the probability that variable x_t is flipped can be expressed as

$$\mathbf{P}\left(x_t \text{ is flipped}\right) = S(n_1+n_2+n_3 > p_1+p_2+p_3), \tag{3.1}$$

when the parameters are clear from the context we denote this value by S.

Now we compute probability $\mathbf{P}\left(E_\varnothing \to E_{--}\right)$. A clause goes from $E_\varnothing$ to E_{--} in two disjunct cases. Firstly, if a clause has only negative literals, one of them is $\neg x_t$, and x_t is not flipped. Secondly, if a clause has two negative literals, and one positive literal x_t, and x_t is flipped. The probability of the first event equals $\frac{3}{8v}$, and under this assumption the conditional probability that x_t is flipped equals $S(p_1+p_2+p_3 < n_1+n_2+n_3+1)$. The probability that a clause has two negative and one positive literal x_t equals $\frac{3}{8v}$ as well, and under this assumption the conditional probability that x_t flips equals $S(p_1+p_2+p_3+1 < n_1+n_2+n_3)$. We denote the two values specified in the last two sentences by S_+ and S_- respectively. Thus

$$\mathbf{P}\left(E_\varnothing \to E_{--}\right) = \frac{3}{8v}(S_+ + S_-) + o(\frac{1}{n}).$$

The other probabilities can be computed in a similar way:

$$\mathbf{P}\left(E_{\varnothing} \to E_{+-}\right) = \frac{6}{8v} + o\left(\frac{1}{n}\right), \quad \mathbf{P}\left(E_{\varnothing} \to E_{++}\right) = \frac{3}{8v} + o\left(\frac{1}{n}\right),$$
$$\mathbf{P}\left(E_{++} \to E_{+}\right) = \frac{2}{v}S + o\left(\frac{1}{n}\right), \quad \mathbf{P}\left(E_{+-} \to E_{+}\right) = \frac{1}{v}(1-S) + o\left(\frac{1}{n}\right),$$
$$\mathbf{P}\left(E_{+-} \to E_{-}\right) = \frac{1}{v}S_{-} + o\left(\frac{1}{n}\right), \quad \mathbf{P}\left(E_{--} \to E_{-}\right) = \frac{2}{v}(1-S_{+}) + o\left(\frac{1}{n}\right),$$
$$\mathbf{P}\left(E_{-} \to E_{0}\right) = \frac{1}{v}(1-S_{+}) + o\left(\frac{1}{n}\right), \quad \mathbf{P}\left(E_{+} \to E_{0}\right) = \frac{1}{v}S_{-} + o\left(\frac{1}{n}\right),$$
$$\mathbf{P}\left(E_{\varnothing} \to E_{1}\right) = \frac{6S}{8v} + \frac{7(1-S)}{8v} + \frac{3S_{+}}{8v} + \frac{3(1-S_{-})}{8v} + o\left(\frac{1}{n}\right),$$
$$\mathbf{P}\left(E_{++} \to E_{1}\right) = \frac{2}{v}(1-S) + o\left(\frac{1}{n}\right), \quad \mathbf{P}\left(E_{+-} \to E_{1}\right) = \frac{1}{v}(1-S_{-}+S) + o\left(\frac{1}{n}\right),$$
$$\mathbf{P}\left(E_{--} \to E_{1}\right) = \frac{2}{v}S_{+} + o\left(\frac{1}{n}\right), \quad \mathbf{P}\left(E_{+} \to E_{1}\right) = \frac{1}{v}(1-S_{-}) + o\left(\frac{1}{n}\right),$$
$$\mathbf{P}\left(E_{-} \to E_{1}\right) = \frac{1}{v}S_{+} + o\left(\frac{1}{n}\right).$$

The probabilities $P(E_{\diamond} \to E_{\star})$ that are not mentioned above equal zero.

We are ready to check that random process $(\mathbf{e}(1), \mathbf{e}(2), \mathbf{e}(3), \dots)$ satisfies conditions (i) - (iii) of Wormald's theorem.

(i) Let $e_{\diamond}$ be a component of $\mathbf{e}$. It is obvious that $|e_{\diamond}(t+1) - e_{\diamond}(t)|$ is less than the number of occurrences of x_t in Φ. The probability that x_t occurs in some clause equals $\frac{3}{n}$, therefore the probability that x_t occurs in k clauses equals $\binom{\rho n}{k}\left(\frac{3}{n}\right)^k$. So assuming that n is large enough we have

$$\mathbf{P}\left(x_t \text{ occurres in more that } n^{1/5} \text{ clauses}\right) = \sum_{k=n^{1/5}}^{n} \binom{\rho n}{k}\left(\frac{3}{n}\right)^k$$
$$= \sum_{k=n^{1/5}}^{n} \frac{\rho n(\rho n-1)\dots(\rho n-k+1)3^k}{k! n^k} \leq \frac{(3\rho)^{n^{1/5}} n}{n^{1/5}!} = o(n^{-3}).$$

(ii) Let $e_{\star}$ be a component of $\mathbf{e}$. Then we have

$$\begin{aligned} \mathbf{E}(e_{\star}(t+1) - e_{\star}(t) | H_t) &= \\ \sum_{E_{\diamond} \neq E_{\star}} \left(\sum_{c \in E_{\diamond}(t)} \mathbf{P}\left(c \in E_{\star}(t+1)\right) - \sum_{c \in E_{\star}(t)} \mathbf{P}\left(c \in E_{\diamond}(t+1)\right) \right) &= \\ \sum_{e_{\diamond} \neq e_{\star}} \left(e_{\diamond}\mathbf{P}\left(E_{\diamond} \to E_{\star}\right) - e_{\star}\mathbf{P}\left(E_{\star} \to E_{\diamond}\right)\right). & \end{aligned} \tag{3.2}$$

For arbitrary functions $f(n), g(n)$ we denote equality $f(n) = g(n) + o(n)$ by $f(n) \approx g(n)$ and we write $f(n) \lesssim g(n)$ if inequality $f(n) \leq g(n)$ holds for large enough n. Thus we set $s = \frac{t}{n}$, $f_\star(s) = \frac{1}{n}e_\star(t)$ (as is easily seen, $v \approx n(1-s)$) and

$$\begin{aligned} p(s, n_1, n_2, n_3, p_1, p_2, p_3) = {} & \left(\frac{3f_\varnothing(s)}{8}\right)^{p_3+n_3} \left(\frac{1}{1-s}\right)^{n_1+n_2+n_3+p_1+p_2+p_3} \\ \times \quad & f_+^{p_1}(s) f_{+-}^{p_2}(s) f_-^{n_1}(s)(2f_{--}(s))^{n_2} \frac{e^{\frac{e_+(s)+e_{+-}(s)+e_-(s)+2e_{--}(s)+3/4e_\varnothing(s)}{1-s}}}{n_1!n_2!n_3!p_1!p_2!p_3!}. \end{aligned}$$

Then set

$$\begin{aligned} s_0(s) &= \sum_{n_1+n_2+n_3 > p_1+p_2+p_3} p(s, n_1, n_2, n_3, p_1, p_2, p_3), \\ s_+(s) &= \sum_{n_1+n_2+n_3+1 > p_1+p_2+p_3} p(s, n_1, n_2, n_3, p_1, p_2, p_3), \\ s_-(s) &= \sum_{n_1+n_2+n_3+1 > p_1+p_2+p_3} p(s, n_1, n_2, n_3, p_1, p_2, p_3). \end{aligned}$$

Note that these functions are represented by series. Later we show that this does not cause any problems. Finally, the required system of differential equations can be obtained from equations (3.2) using $s_0(s), s_+(s), s_-(s)$ to compute the probabilities instead of S_0, S_+, S_-.

(iii) The functions constructed above have two substantial deficiencies: they are not defined when $s = 1$, and the series used to define them do not converge uniformly in the naturally defined set D. However, this can be overcome using a standard trick, namely, for each $\epsilon > 0$, define set D such that it includes only points with $s \leq 1 - \epsilon$. It is not hard to see that, as the series above are non-negative and bounded with 1, they converge uniformly in any closed set. Then we find the required value as the limit when $\epsilon \to 0$.

Applying Wormald's theorem we conclude that values of parameters $e_\star$ at step t whp can be expressed as

$$e_\star(t) = nz_\star(t/n) + o(n), \tag{3.3}$$

where collection of $z_\star$ is a solution of some system of differential equations.

Setting $t = n$ in (3.3) we get that whp

$$e_0(n) = \omega n + o(n),$$

where $\omega = nz_\star(t/n)$. Thus we have proven the following.

Theorem 1 *For any positive ρ there is a constant ω such that for a random 3-CNF $\Phi(n, \rho n)$ almost surely the OLS algorithm finds an assignment such that the number of satisfied clauses equals $\omega n + o(n)$.*

3.2 Local Search

To analyze the Local Search algorithm we use techniques similar to ones used for the One Pass Local Search. However, as every variable in this algorithm can be considered and flipped several times we cannot use the card game approach; instead we have to find quite a different set of random variables that represents properties of the problem crucial for the performance of the algorithm. Although we were unable to carry out a complete rigorous analysis, it turns out that such an analysis boils down to a certain simple assumption (see Assumption 1 below). This assumption looks very plausible, but we could not neither prove nor disprove it. Experiments show that our model is accurate enough, this is why we believe that either Assumption 1 is true, or it can be replaced with a property that gives rise to an equivalent model.

3.2.1 Model

We need some notation. Let φ be a 3-CNF and x_l a variable in φ. By $Q_{i\alpha}(x_l)$, where $i \in \{0,1,2\}$ and $\alpha \in \{-,+\}$, we denote the set of clauses c such that $x_l \in c$ if $\alpha = +$, $\neg x_l \in c$ if $\alpha = -$, and among the other two literals there are exactly i positive. If $c \in Q_{i\alpha}(x_l)$ we also say that c has *type* $i\alpha$ for x_l, and that variable x_l occupies *position* of type $i\alpha$ in the clause c. Let also $q_{i\alpha}(x_l)$ denote the size of $Q_{i\alpha}(x_l)$. By $E_{\overline{a}}$, $\overline{a} = (a_{0-}, a_{0+}, a_{1-}, a_{1+}, a_{2-}, a_{2+})$ we denote the set of all variables x_l of Φ such that $q_{i\alpha}(x_l) = a_{i\alpha}$ for all i and α. By $e_{\overline{a}}$ we denote the size of $E_{\overline{a}}$. As Φ is changing over time all these sets and numbers are actually functions of the number of steps made. Thus, sometimes we use notation $E_{\overline{a}}(t), e_{\overline{a}}(t)$. Functions $e_{\overline{a}}(t)$ will be the random variables required in Wormald's theorem. If $x_l \in E_{\overline{a}}$ then variable x_l is said to have type $\overline{a}$. Note that as n grows the number of different tuples $\overline{a}$ and therefore the number of random variables also grow. To overcome this problem we will consider only those variables that appear in at most M clauses for some fixed M. Clearly, this does affect the analysis, but in a certain controllable way, as we shall see.

Before checking conditions (i)–(iii) of Theorem (Wormald, [61]) we make a simple observation.

Lemma 3 *If Φ is a random 3-CNF of density ρ with n variables, then for each variable x_l*

$$\begin{aligned}
\mathbf{P}\left(q_{i\alpha}(x_l) = a\right) &= \frac{\left(\frac{3\rho}{8}\right)^a e^{3\rho/8}}{a!} + o(1) \quad \textit{if } i = 0,2, \\
\mathbf{P}\left(q_{i\alpha}(x_l) = a\right) &= \frac{\left(\frac{3\rho}{4}\right)^a e^{3\rho/4}}{a!} + o(1) \quad \textit{if } i = 1, \\
\mathbf{P}\left(x_l \in E_{\overline{a}}\right) &= \prod_{i,\alpha} \mathbf{P}\left(q_{i\alpha}(x_l) = a\right), \quad \mathbf{E}(e_{\overline{a}}) = n \cdot \mathbf{P}\left(x_l \in E_{\overline{a}}\right).
\end{aligned}$$

Lemma 4 *If Φ is a random 3-CNF of density ρ with n variables, then $\mathbf{P}\left(|e_{\overline{a}} - \mathbf{E}(e_{\overline{a}})| > n^{1/5}\right) = o(n^{-3})$.*

Lemma 4 provides the initial values for equations from Theorem (Wormald, [61]). Now we are verifying conditions (i)–(iii).

(i) Possible variations of random variables $e_{\overline{a}}$ are bounded by $2K$ where K is the degree of the flipped variable. Therefore condition (i) can be proven in the same way as for the OLS algorithm.

(ii) Suppose that on the current step t of LS the variable to flip is x_l. Since x_l is a variable picked uniformly at random from the set $B(t) = \bigcup_{\substack{\overline{a} \\ a_{0-}->a_{0+}}} E_{\overline{a}}(t)$, we have $\mathbf{P}\left(x_l \in E_{\overline{a}}(t)\right) = \frac{e_{\overline{a}}(t)}{b(t)}$, where $b(t) = |B(t)|$. Also we have

$$\mathbf{E}(q_{i\alpha}(x_l)) = \sum_{\substack{\overline{a} \\ a_{0-}->a_{0+}}} a_{i\alpha} \cdot \mathbf{P}\left(x_l \in E_{\overline{a}}(t)\right).$$

We say that tuples $\overline{a}, \overline{b}$ are *adjacent* if there are i, j, α such that $|j - i| = 1$, $a_{i\alpha} = b_{i\alpha} + 1$, $a_{j\alpha} = b_{j\alpha} - 1$, and $a_{i'\alpha'} = b_{i'\alpha'}$ in all other cases. Intuitively, adjacency means that if $x_l \in E_{\overline{b}}$ then it can be moved to $E_{\overline{a}}$ or vice versa by flipping one literal in one of the clauses containing x_l. Let also $\overline{a}'$ denote the tuple such that $a'_{i-} = a_{i+}$ and $a'_{i+} = a_{i-}$.

Set $E_{\overline{a}}$ changes in two ways. First, variable x_l can move to or from $E_{\overline{a}}$, in this case it moves from or to $E_{\overline{a}'}$. Second, x_l may happen to be in the same clause with some other variable, x_m, and then x_m can move to or from $E_{\overline{a}}$. Such a variable moves then from or to $E_{\overline{b}}$ for some $\overline{b}$ adjacent with $\overline{a}$.

Clearly, the expectation of change of the first type equals $\mathbf{P}\left(x_l \in E_{\overline{a}'}\right) - \mathbf{P}\left(x_l \in E_{\overline{a}}\right)$. Further calculation we carry out under the following assumption.

Assumption 1 *Assuming history H_t, for a random clause c of the current formula, any positions p, r, $p \neq r$, in c, any tuples $\overline{a}, \overline{b}$, and any variables $x_l \in E_{\overline{a}}, x_m \in E_{\overline{b}}$, the events "$x_l$ is in position p of clause c" and "x_m is in position r of clause c" are independent.*

Let us take a variable $x_m \in E_{\overline{a}}$ and calculate the probability of an event $\mathcal{G}^-$:"variable x_m moves from $E_{\overline{a}}$ to $E_{\overline{b}}$", where $\overline{b}$ is some tuple adjacent to $\overline{a}$ and $\overline{a}, \overline{b}$ differ in components $i\alpha$ and $j\alpha$. This happens if in some clause c containing both x_l and x_m some position occupied by x_m changes its type from $i\alpha$ to $j\alpha$. Obviously, depending on $i\alpha$ the type of the position occupied by x_l may vary. We use $\hat{i}\hat{\alpha}$ to denote the possible type of such a position. Simple case analysis shows that $\hat{i} = j$ if $j < i$ and $\alpha = -$, or if $j > i$ and $\alpha = +$, otherwise $\hat{i} = i$. Then $\hat{\alpha} = -$ if $j < i$ and $\hat{\alpha} = +$ if $j > i$.

Let $\theta_{\overline{a}\to\overline{b}}$ denote the number of positions of type $\hat{i}\hat{\alpha}$ in c except the one possibly occupied by x_m. It is easy to see that $\theta_{\overline{a}\to\overline{b}} = 1$ if $i = 1$, $\theta_{\overline{b}\to\overline{a}} = 1$ if $j = 1$, and $\theta_{\overline{a}\to\overline{b}} = 2$, $\theta_{\overline{b}\to\overline{a}} = 2$ otherwise. Thus, the number of positions in the clauses of Φ such that if x_l in such a position then $\mathcal{G}$ happens to some variable x_m equals $\theta_{\overline{a}\to\overline{b}} a_{i\alpha}$. Let also $k_{\hat{i}\hat{\alpha}}(t) = \sum_{\overline{a}} a_{\hat{i}\hat{\alpha}} \cdot e_{\overline{a}}(t)$ be the number of positions of type $\hat{i}\hat{\alpha}$ in the formula.

Suppose that variable x_l that is flipped belongs to $E_{\overline{c}}$. Then among all $k_{\hat{i}\hat{\alpha}}(t)$ positions of type $\hat{i}\hat{\alpha}$ we have $c_{\hat{i}\hat{\alpha}}$ positions occupied by x_l, and $\theta_{\overline{a}\to\overline{b}} a_{i\alpha}$ positions such that the presence of x_l in one of them makes the event $\mathcal{G}^-$ happen for some x_m. By Assumption 1, we have $\mathbf{P}\left(\mathcal{G}^- | x_l \in E_{\overline{c}}\right) = \frac{c_{\hat{i}\hat{\alpha}} \theta_{\overline{a}\to\overline{b}} a_{i\alpha}}{k_{\hat{i}\hat{\alpha}}(t)}$. Therefore,

$$\begin{aligned}
\mathbf{P}\left(\mathcal{G}^-\right) &= \sum_{\substack{\overline{c} \\ c_{0-} > c_{0+}}} P(x_l \in E_{\overline{c}}) \frac{c_{\hat{i}\hat{\alpha}} \theta_{\overline{a}\to\overline{b}} a_{i\alpha}}{k_{\hat{i}\hat{\alpha}}(t)} \\
&= \mathbf{E}(q_{\hat{i}\hat{\alpha}}(x_l) | q_{0-}(x_l) > q_{0+}(x_l)) \frac{\theta_{\overline{a}\to\overline{b}} a_{i\alpha}}{k_{\hat{i}\hat{\alpha}}(t)}
\end{aligned}$$

Similarly, the probability of an event $\mathcal{G}^+$:"variable x_m moves from $E_{\overline{b}}$ to $E_{\overline{a}}$", where $\overline{b}$ is some tuple adjacent to $\overline{a}$ and $\overline{a}, \overline{b}$ differ in components $i\alpha$ and $j\alpha$, equals

$$\mathbf{P}\left(\mathcal{G}^+\right) = \mathbf{E}(q_{\hat{j}\hat{\alpha}}(x_l) | q_{0-}(x_l) > q_{0+}(x_l)) \frac{\theta_{\overline{b}\to\overline{a}} b_{j\alpha}}{k_{\hat{j}\hat{\alpha}}(t)}$$

Observing that the expectations of the numbers of variables that move to and from $E_{\overline{a}}$ (excluding x_l) equal

$$e_{\overline{b}} \mathbf{P}\left(\mathcal{G}^+\right) \qquad \text{and} \qquad e_{\overline{a}} \mathbf{P}\left(\mathcal{G}^-\right),$$

respectively, we get

$$\begin{aligned}
&\mathbf{E}(e_{\overline{a}}(t+1) - e_{\overline{a}}(t) \mid H_t) \qquad (3.4) \\
&= \mathbf{P}\left(x_l \in E_{\overline{a}'}(t) \mid q_{0-}(x_l) > q_{0+}(x_l)\right) - \mathbf{P}\left(x_l \in E_{\overline{a}}(t) \mid q_{0-}(x_l) > q_{0+}(x_l)\right) \\
&+ \sum_{\substack{\overline{b} \text{ adjacent to } \overline{a} \\ i,j,\alpha}} \left(\frac{\theta_{\overline{b}\to\overline{a}} b_{j\alpha} e_{\overline{b}}(t)}{k_{\hat{j}\hat{\alpha}}(t)} - \frac{\theta_{\overline{a}\to\overline{b}} a_{j\alpha} e_{\overline{a}}(t)}{k_{\hat{i}\hat{\alpha}}(t)} \mathbf{E}(q_{\hat{i}\hat{\alpha}}(x_l) \mid q_{0-}(x_l) > q_{0+}(x_l)) \right).
\end{aligned}$$

Denoting $s = \frac{t}{n}$, $z_{\overline{a}}(s) = e_{\overline{a}}(sn)$ and

$$u(s) = \sum_{\substack{\overline{a} \\ a_{0-} > a_{0+}}} z_{\overline{a}}, \quad g_{i\alpha} = \sum_{\overline{a}} a_{i\alpha} z_{\overline{a}}, \quad h_{i\alpha} = \sum_{\substack{\overline{a} \\ a_{0-} > a_{0+}}} a_{i\alpha} \frac{z_{\overline{a}}}{u}$$

we get

$$\frac{dz_{\overline{a}}}{ds} = \frac{z_{\overline{a}'} - z_{\overline{a}}}{u} + \sum_{\substack{\overline{b} \text{ adjacent to } \overline{a} \\ i,j,\alpha}} \left(\frac{\theta_{\overline{b}\to\overline{a}} b_{j\alpha} z_{\overline{b}}}{g_{\hat{j}\hat{\alpha}}} - \frac{\theta_{\overline{a}\to\overline{b}} a_{j\alpha} z_{\overline{a}}}{g_{\hat{i}\hat{\alpha}}} h_{\hat{i}\hat{\alpha}} \right).$$

(iii) We are interested in the value of g_{0-} when $u(s)$ becomes 0 for the first time. Thus D can be chosen to be any open set with positive elements satisfying the condition $u > \epsilon$ for some $\epsilon > 0$. As before we can find the required value as limit as $\epsilon \to 0$.

Theorem 2 *If Assumption 1 is true then for any positive ρ there is a constant ω such that for a random 3-CNF $\Phi(n, \rho n)$ almost surely the LS algorithm finds and assignment such that the number of satisfied clauses equals $\omega n + o(n)$.*

Proof. Applying Wormald's theorem we get that, for any positive ρ and any M there is a constant ω' such that for a random 3-CNF $\Phi(n, \rho n)$ almost surely the SL algorithm finds and assignment such that the number of satisfied clauses equals $\omega' n + o(n)$ not containing variables of degree higher than M. We estimate how many clauses may contain a variable (or its negation) of degree higher than M. It is not hard to see that almost surely the number of such clauses is no more than

$$e^{\rho/2} \cdot \sum_{k>M} k \binom{\rho n}{k} \left(\frac{1}{2n-1} \right)^k,$$

which is $o(1) \cdot n$ where o means asymptotic as $M \to \infty$. □

3.2.2 Experiments

In this subsection we report on experiments aiming to estimate constant ω from Theorem 2 for different values of ρ. In order to do this we solve numerically the system of differential equations built in the previous subsection. Unfortunately, even for small M this system contains far too many equations. For example, if $M = 15$ then the number of equations exceeds one million. However, while conducting experiments we observed some properties of functions involved that allow us to decrease the number of equations without loss of accuracy. We state these properties later after proper definitions.

To simplify the system of equations we introduce new random variables

$$E_{ab}(t) = \bigcup_{\substack{\overline{a} \\ a_{0-}=a, a_{0+}=b}} E_{\overline{a}}(t), \quad e_{ab}(t) = \sum_{\substack{\overline{a} \\ a_{0-}=a, a_{0+}=b}} e_{\overline{a}}(t).$$

It is also clear that $\mathbf{E}(e_{ab}(t+1) - e_{ab}(t)) = \sum_{\substack{\overline{a} \\ a_{0-}=a, a_{0+}=b}} \mathbf{E}(e_{\overline{a}}(t+1) - e_{\overline{a}}(t))$. Along with $e_{ab}(t)$ we shall use the following random variables: $A(t), B(t), C(t)$, and $D(t)$ that are equal to the number of clauses with 0,1,2, and 3 positive literals, respectively. It is not hard to see that $A(t) = 1/3 \sum_{c,d} c \cdot e_{cd}(t)$, $B(t) = \sum_{c,d} d \cdot e_{cd}(t)$, and $D(t) = \rho n - (A(t) + B(t) + C(t))$. Thus, as a matter of fact, we need only one extra random variable, $C(t)$. Now we compute the sum in the right side of this equation accordingly to the three parts of the expression (3.4) for $\mathbf{E}(e_{\overline{a}}(t+1) - e_{\overline{a}}(t))$. The first part

$$\sum_{\substack{\overline{a} \\ a_{0-}=a, a_{0+}=b}} \left(\mathbf{P}\left(x_l \in E_{\overline{a}'}(t) \mid q_{0-}(x_l) > q_{0+}(x_l)\right) - \mathbf{P}\left(x_l \in E_{\overline{a}}(t) \mid q_{0-}(x_l) > q_{0+}(x_l)\right)\right)$$

can be converted into

$$\begin{aligned} &\mathbf{P}\left(x_l \in E_{ba} \mid q_{0-}(x_l) > q_{0+}(x_l)\right) - \mathbf{P}\left(x_l \in E_{ab} \mid q_{0-}(x_l) > q_{0+}(x_l)\right) \\ &\qquad = \begin{cases} \frac{e_{ba}(t)}{G(t)}, & \text{if } a < b, \\ -\frac{e_{ab}(t)}{G(t)} & \text{if } a > b, \end{cases} \end{aligned}$$

where $G(t) = \sum_{c>d} e_{cd}(t)$.

It is easier to compute the second and third parts from scratch. Compute first the third part. Function $e_{ab}(t)$ can be decreased if for some variable $x_m \in E_{ab}$ either (a) a certain clause of type $0-$ for x_m contains $\neg x_l$, or (b) a certain clause of type $1-$ contains x_l, or (c) a certain clause of type $0+$ contains $\neg x_l$, or (d) a certain clause of type $1+$ contains x_l. The probabilities of these events are

$$\begin{aligned} &\mathbf{P}\left(\neg x_l \in c \mid c \text{ of type } 0- \text{ for } x_m, q_{0-}(x_l) = K_1\right) = \frac{2K_1}{A(t)}, \\ &\mathbf{P}\left(x_l \in c \mid c \text{ of type } 1- \text{ for } x_m, q_{0+}(x_l) = K_2\right) = \frac{2K_2}{B(t)}, \\ &\mathbf{P}\left(\neg x_l \in c \mid c \text{ of type } 0+ \text{ for } x_m, q_{1-}(x_l) = K_3\right) = \frac{2K_3}{B(t)}, \\ &\mathbf{P}\left(x_l \in c \mid c \text{ of type } 1+ \text{ for } x_m, q_{1+}(x_l) = K_4\right) = \frac{2K_4}{C(t)}. \end{aligned}$$

By Assumption 1,

$$\begin{aligned}
P_1 = \mathbf{P}\left(\neg x_l \in c \mid c \text{ of type } 0- \text{ for } x_m\right) &= \sum_{K_1} \mathbf{P}\left(q_{0-}(x_l) = K_1\right) \frac{2K_1}{A(t)} = 2\frac{\mathbf{E}(q_{0-}(x_l))}{A(t)},\\
P_2 = \mathbf{P}\left(x_l \in c \mid c \text{ of type } 1- \text{ for } x_m\right) &= \sum_{K_2} \mathbf{P}\left(q_{0+}(x_l) = K_2\right) \frac{2K_2}{B(t)} = 2\frac{\mathbf{E}(q_{0+}(x_l))}{B(t)},\\
P_3 = \mathbf{P}\left(\neg x_l \in c \mid c \text{ of type } 0+ \text{ for } x_m\right) &= \sum_{K_3} \mathbf{P}\left(q_{1-}(x_l) = K_3\right) \frac{2K_3}{B(t)} = 2\frac{\mathbf{E}(q_{1-}(x_l))}{B(t)},\\
P_4 = \mathbf{P}\left(x_l \in c \mid c \text{ of type } 1+ \text{ for } x_m\right) &= \sum_{K_4} \mathbf{P}\left(q_{1+}(x_l) = K_4\right) \frac{2K_4}{C(t)} = 2\frac{\mathbf{E}(q_{1+}(x_l))}{C(t)}.
\end{aligned}$$

The expectations $\mathbf{E}(q_{0-}(x_l)), \mathbf{E}(q_{0+}(x_l))$ can be easily found, since

$$\mathbf{P}\left(q_{0-}(x_l) = K_1\right) = \frac{\sum_b e_{K_1 b}(t)}{G(t)}, \quad \mathbf{P}\left(q_{0+}(x_l) = K_2\right) = \frac{\sum_a e_{a K_2}(t)}{G(t)}.$$

The expectations $\mathbf{E}(q_{1-}(x_l)), \mathbf{E}(q_{1+}(x_l))$ we find using the following empirical observation.

Observation 2 *For a randomly chosen x_m and any i, α, $i \neq 0$, and a, b*

$$\mathbf{E}(q_{i\alpha}(x_m) \mid x_m \in E_{ab}) \approx \mathbf{E}(q_{i\alpha}(x_m)).$$

Thus, easy computation shows that $\mathbf{E}(q_{1-}(x_l)) = \frac{B(t)}{n}$ and $\mathbf{E}(q_{1+}(x_l)) = \frac{C(t)}{n}$. Then the expectation for the third part equals

$$\begin{aligned}
&e_{ab}(t)(P_1\mathbf{E}(q_{0-}(x_m)) + P_2\mathbf{E}(q_{1-}(x_m)) + P_3\mathbf{E}(q_{0+}(x_m)) + P_4\mathbf{E}(q_{1+}(x_m)))\\
&\quad= 2e_{ab}(t)\left(\frac{a\mathbf{E}(q_{0-}(x_l))}{A(t)} + \frac{B(t)}{n} + \frac{b}{n} + \frac{C(t)}{n}\right).
\end{aligned}$$

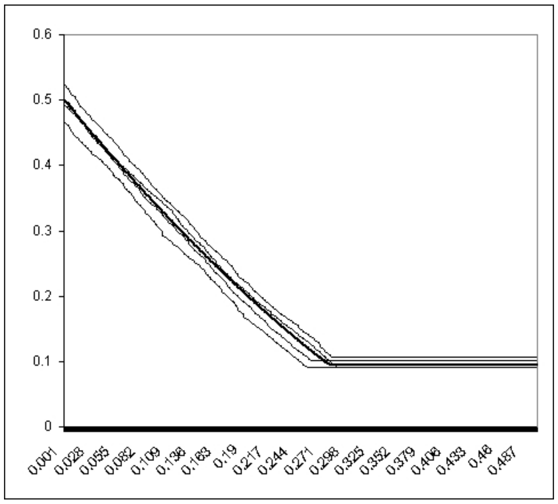
The second part of the expectation equals

$$2\frac{\mathbf{E}(q_{0+}(x_l))}{n} e_{(a-1)b} + 2\frac{\mathbf{E}(q_{0-}(x_l))(a+1)}{A(t)} e_{(a+1)b} + 2\frac{C(t)}{n^2} e_{a(b-1)} + 2\frac{b+1}{n} e_{a(b+1)}.$$

Similarly we have

$$\mathbf{E}(C(t+1) - C(t)) = \mathbf{E}(q_{1-}(x_l)) + \mathbf{E}(q_{2+}(x_l)) - \mathbf{E}(q_{1+}(x_l)) - \mathbf{E}(q_{2-}(x_l)).$$

Denoting $s = \frac{t}{n}, z_{ab}(s) = \frac{e_{ab}(sn)}{n}, p(s) = \frac{A(sn)}{n}, q(s) = \frac{B(sn)}{n}, r(s) = \frac{C(sn)}{n}, u(s) = \frac{D(sn)}{n}, g(s) = \frac{G(sn)}{n}$, $g_{ab} = z_{ba}$ if $a < b$ and $g_{ab} = -z_{ab}$ if $a > b$, and $h_1(s) = \frac{1}{g}\sum_{\substack{a,b\\ a>b}} a z_{ab}, \quad h_2(s) =$



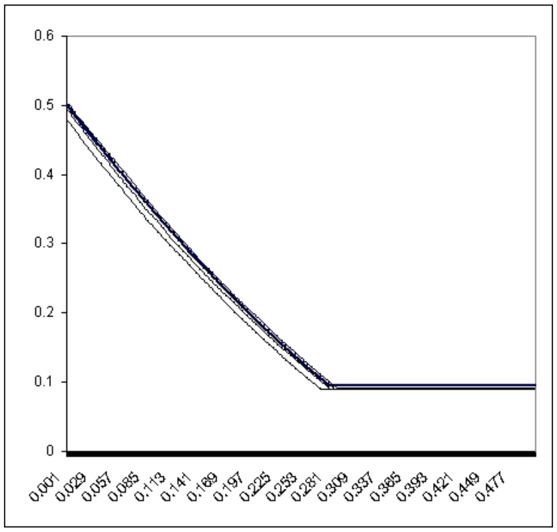


Figure 3.2: Empirical performance of LS and its prediction. The vertical axis shows the number of unsatisfied clauses divided by the number of variables. The horizontal axis shows the number of the step of the algorithm divided by the number of variables.

$\frac{1}{g}\sum_{\substack{a,b\\a>b}} bz_{ab}$ we get

$$\begin{aligned}
\frac{dz_{ab}}{ds} &= \frac{g_{ab}}{g} + 2\left(\frac{h_2}{g} z_{(a-1)b} + \frac{(a+1)h_1}{g} z_{(a+1)b} + r z_{a(b-1)} + (b+1) z_{a(b-1)}\right) \\
&- 2z_{ab}\left(\frac{ah_1}{p} + q + b + r\right), \\
\frac{dr}{ds} &= 2q + u - 3r. \qquad (3.5)
\end{aligned}$$

As the graphs in Fig. 4 show, these equations give a very good approximation for empirical results. The graphs show the evolution of $p(s)$ that is the relative number of unsatisfied clauses. Thin lines are values observed when running LS for particular random problems, and the thick lines are computed from a numerical solution of the system above. In the examples shown $\rho = 4, M = 30$, $n = 1000$ for the graph on the left and $n = 10000$ for the graph on the right.

The following table shows the dependance between ρ and the constant ω from Theorem 2 both empirical and predicted by the system (3.5). Experimental figures are average on 10 formulas with 1000 variables each.

ρ	2	3	4	4.5	5	6	7	10	15	20	25
c (experiment)	1.98	2.95	3.91	4.39	4.86	5.80	6.74	9.53	14.11	18.69	23.23
c (system (3.5))	1.98	2.95	3.91	4.38	4.85	5.80	6.73	9.52	14.14	18.74	23.32

Table 3.1: Dependence of the predicted and actual relative number of the satisfied clauses on the density of the problem.

Chapter 4

Phase transition of basic Local Search

We now move to the analysis of the performance of Local Search on Random Planted 3-SAT. We prove the following.

Theorem 3 *(1) Let $\rho \geq \kappa \cdot \ln n$, and $\kappa > \frac{7}{6}$. Then Local Search whp finds a solution of Random Planted 3-SAT of density ρ.*
(2) Let $c \leq \rho \leq \kappa \cdot \ln n$, where c is an arbitrary positive constant, and $0 < \kappa < \frac{7}{6}$. Then Local Search whp does not find a solution of Random Planted 3-SAT of density ρ.

Intuitively speaking, this theorem states that Local Search performance on Random Planted 3-SAT of logarithmic density demonstrates a phase transition phenomenon. The algorithm succeeds whp if the density of the problem is asymptotically greater than $7/6 \ln n$ and fails if the density is asymptotically less than $7/6 \ln n$.

4.1 Success of Local Search

In this section we prove that Local Search succeeds for Random Planted 3-SAT of density greater than $7/6 \ln n$ (see Theorem 3(1)). To prove this we need to show that if a 3-CNF has high density, that is, greater than $\kappa \log n$ for some $\kappa > \frac{7}{6}$ then whp all the local maxima that do not satisfy the CNF — we call such maxima *proper* — concentrate very far from the planted assignment. This is the statement of Proposition 1 below. Then we use Lemma 5 to prove that starting from a random assignment LS whp does not go to that remote region. Therefore the algorithm does not get stuck to a local maximum that is not a solution.

To prove Proposition 1 we use the following three lemmas. Recall that the planted solution is the all-ones one.

Lemma 5 *Let $\rho \geq \kappa \ln n$ for some constant $\kappa > 0$, and let constants q_0, q_1 be such that $0 \leq q_0 < q_1 \leq 1$. Whp any assignment with less than $q_0 n$ zeros satisfies more clauses than any assignment with more than $q_1 n$ zeros.*

Proof. Let $\vec{u}, \vec{v}$ be some vectors with less than $q_0 n$ and more than $q_1 n$ zeros, respectively. Let c be a random clause, then (a) with probability $\frac{1}{7}$ all its literals are positive, (b) with probability $\frac{3}{7}$ two literals are positive and similarly (c) with probability $\frac{3}{7}$ one literal is positive. The probabilities that the clause is not satisfied by $\vec{u}$ in these cases equals to $q_0^3, q_0^2(1-q_0)$ and $q_0(1-q_0)^2$, respectively. Hence the total probability that a clause is not satisfied by $\vec{u}$ equals $\frac{q_0^3+3q_0^2(1-q_0)+3q_0(1-q_0)^2}{7} = \frac{1-(1-q_0)^3}{7}$. This function for $0 \leq q_0 \leq 1$ monotonically increases. A similar result holds for $\vec{v}$. Thus the expectation of the number of clauses unsatisfied by $\vec{u}$ and $\vec{v}$ in a random formula is less than $\frac{1-(1-q_0)^3}{7}\kappa n \ln n$ and greater than $\frac{1-(1-q_1)^3}{7}\kappa n \ln n$ respectively. The random variable "the total number of clauses satisfied by $\vec{u}$" is a sum of binomial random variables "the number of clauses that contain i positive literals and are satisfied by $\vec{u}$", $i = 1, 2, 3$, so applying Lemma 1 we conclude that

$$\mathbf{P}\left(\ \vec{u} \text{ satisfies less clauses than } \vec{v}\right) < e^{-\lambda' n \ln n},$$

for some $\lambda' > 0$. There are less than 2^{n+1} pairs of assignments, hence, application of the union bound finishes proof of the lemma. $\square$

Lemma 6 *Let $\rho \geq \kappa \ln n$ for some $\kappa > 0$. There is $\alpha < 1$ such that for $\varphi \in \Phi^{\mathsf{plant}}(n, \rho n)$ whp for any proper local maximum $\vec{u}$ of φ the number of variables assigned to 0 by $\vec{u}$ is either less than n^α, or greater than $\frac{9n}{10}$.*

Proof. Let $M, |M| = \ell$, be the set of all variables that $\vec{u}$ assigns to 0. Let $\mathcal{B}_M^{each}$ be event "for every $x_i \in M$ the number of clauses voting for x_i to be 1 is less than or equal to the number of clauses voting for x_i to be 0". Since $\vec{u}$ is a local maximum, $\mathcal{B}_M^{each}$ is the case for $\vec{u}$. It is easy to see that event $\mathcal{B}_M^{each}$ implies event $\mathcal{B}_M^{all}$ = "the total number of votes given by clauses for variables in M to be 1 is less than or equal to the total number of votes given by clauses for variables in M to be 0". To bound the probability of $\mathcal{B}_M^{each}$ we will bound the probability of $\mathcal{B}_M^{all}$.

Let c be a random clause. It can contribute from 0 to 3 votes for variables in M to be one and 0 or 1 vote for them to remain zero. Let us compute, for example, the probability that it contributes exactly two votes for variables in M to become one. This happens if c is of type $(+,+,-)$, both its positive variables are in M and the negative variable is outside of M. The probability of this event is $\frac{3}{7}\ell^2 n^{-2}(1-\ell/n)$. So the expectation of the number of clauses voting for exactly 2 variables in

M to be 1 is $\frac{3}{7}\ell^2 n^{-1}(1-\ell/n)\kappa \ln n$. The expectations of the numbers of clauses voting for three and one variables to be 1 are $\frac{1}{7}\ell^3 n^{-2}\kappa\ln n$ and $\frac{3}{7}(1-\frac{\ell}{n})^2\ell\kappa\ln n$, respectively.

A clause votes for a variable in M to remain 0 if its type is $(+,-,-)$, one of its negative literals is not in M, and two other literals are in M, or if its type is $(+,+,-)$ and all the variables in it belong to M. Thus the expectation of the number of clauses voting for variables in M to remain 0 is $\frac{3}{7}\kappa\ln n\left(2\ell^2 n^{-1}(1-\ell/n)+\ell^3 n^{-2}\right)$.

Hence the expectation of the number of votes for variables in M to flip equals

$$\mathbf{E}\,(\text{votes for a flip}) = \kappa\ln n\times\left(3\cdot\frac{1}{7}\ell^3 n^{-2}+2\cdot\frac{3}{7}\ell^2 n^{-1}(1-\ell/n)+1\cdot\frac{3}{7}\ell(1-\ell/n)^2\right)$$

and the expectation of the number of votes for variables in M to remain 0 equals

$$\mathbf{E}\,(\text{votes for status quo}) = \kappa\ln n\times\left(\frac{6}{7}\ell^2 n^{-1}(1-\ell/n)+\frac{3}{7}\ell^3 n^{-2}\right).$$

If $\ell < \frac{9}{10}n$ then

$$\begin{aligned}\frac{\mathbf{E}\,(\text{votes for status quo})}{\mathbf{E}\,(\text{votes for a flip})} &= \frac{6\ell(n-\ell)+3\ell^2}{6\ell(n-\ell)+3\ell^2+3(n-\ell)^2} = 1-\frac{3(n-\ell)^2}{6\ell(n-\ell)+3\ell^2+3(n-\ell)^2}\\ &< 1-\frac{3\cdot\frac{1}{100}n^2}{12n^2} = 1-\frac{1}{400}.\end{aligned}$$

Therefore we can apply Lemma 1 to the votes for and against 0s and get the following bound $\mathbf{P}\left(\mathcal{B}_M^{all}\right) < e^{-\lambda\mathbf{E}(\text{votes for a flip})}$ for some $\lambda > 0$. Then we can bound number of votes for a flip from below by $\delta\ell\ln n$ for some constant δ and we can bound the number of sets M of size ℓ as

$$\#(\text{M of size }\ell) = \binom{n}{\ell} \le \left(\frac{ne}{\ell}\right)^{\ell} = e^{\ell\ln(n/\ell)+\ell}.$$

Therefore if

$$\ell\ln(n/\ell)+\ell < \delta\ell\ln n$$

then union bound implies that whp there is no set M such that $\mathcal{B}_M^{all}$ happens. It is easy to see that for $\ell > n^\alpha$ and α that is close enough to 1 the above inequality holds, which finishes the proof of the lemma. □

Now suppose that $\vec{u}$ is a proper local maximum of $\varphi\in\Phi^{\mathsf{plant}}(n,\rho n)$. There is a clause $c\in\varphi$ that is not satisfied by $\vec{u}$. Without loss of generality, let the variables in c be x_1, x_2, x_3, and let the variable assigned 0 be x_1. Thus, clause c votes for x_1 to be flipped to 1. Since $\vec{u}$ is a local maximum there must a clause that is satisfied, and that becomes unsatisfied should x_1 flipped. We call such a clause a *support* clause for the 0 value of x_1. In any support clause the supported variable is negated, and therefore any support clause has the type $(+,-,-)$ or $(+,+,-)$. A variable of a CNF is called k*-isolated* if it appears positively in at most k clauses of the type $(+,-,-)$. The *distance* between variables of a CNF φ is the length of the shortest path in $G(\varphi)$ connecting them.

Lemma 7 *If $\kappa > \frac{7}{6}$ and $\rho \geq \kappa \ln n$ then for any integers $d_1, d_2 \geq 1$ and for a random $\varphi \in \Phi^{\mathsf{plant}}(n, \rho n)$ whp there are no two d_1-isolated variables within distance d_2 from each other.*

Proof. Let x be some variable. The probability that it is d_1-isolated can be computed as

$$\begin{aligned}
\mathbf{P}\left(x \text{ is } d_1\text{-isolated}\right) &= d_1 \cdot \binom{\kappa n \ln n}{d_1} \left(1 - \frac{3}{7n}\right)^{\kappa n \ln n - d_1} \left(\frac{3}{7n}\right)^{d_1} \\
&\leq d_1 (\kappa n \ln n)^{d_1} \left(1 - \frac{3}{7n}\right)^{\kappa n \ln n} \left(1 - \frac{3}{7n}\right)^{-d_1} \left(\frac{7}{3} n\right)^{-d_1} \\
&\sim d_1 \left(1 - \frac{3}{7n}\right)^{-d_1} (\frac{7\kappa}{3} \ln n)^{d_1} e^{-\frac{3}{7}\kappa \ln n} \\
&= O(n^{-\frac{3\kappa}{7} + \varepsilon}),
\end{aligned}$$

for any $\epsilon > 0$.

By Lemma 2(2), the degree of every vertex of $G(\varphi)$ whp does not exceed $\ln^2 n$. Hence, there are at most $\ln^{2d_2} n$ vertices at distance at most d_2 from x. Applying the union bound we can estimate the probability that there is a d_1-isolated vertex at distance d_2 from x as $O(\ln^{2d_2} n \cdot n^{-\frac{3}{7}\kappa})$. Finally, taking into account the probability that x itself is d_1-isolated, and applying the union bound over all vertices of $G(\varphi)$ we obtain that the probability that two d_1-isolated vertices exist at distance d_2 from each other can be bounded from above by

$$n \cdot O(n^{-\frac{3\kappa}{7}}) \cdot O(\ln^{2d_2} n \cdot n^{-\frac{3}{7}\kappa}) = O(\ln^{2d_2} n \cdot n^{1-\frac{6}{7}\kappa}).$$

Thus for $\kappa > \frac{7}{6}$ whp there are no two such vertices. □

Proposition 1 *Let $\rho \geq \kappa \cdot \ln n$, and $\kappa > \frac{7}{6}$. Then whp proper local maxima of a 3-CNF from $\Phi^{\mathsf{plant}}(n, \rho n)$ have at most $\frac{n}{10}$ ones.*

Proof. Let $\varphi \in \Phi^{\mathsf{plant}}(n, \rho n)$ be a random planted instance. Suppose that $\vec{u}$ is a proper local maximum that has more than $\frac{n}{10}$ ones. We use the following observation. Let c be a clause not satisfied by $\vec{u}$. Then it contains at least one variable x_i that is assigned to zero by $\vec{u}$. The assignment

$\vec{u}$ is a local maximum, so there must be a clause c' that is satisfied only by x_i. Hence, c' is a support clause, and contains a variable x_j which is assigned to zero by $\vec{u}$. Variables x_i and x_j are at distance 1. Setting $d_1 = 11$ and $d_2 = 1$, by Lemma 7, we conclude that one of them is not 11-isolated.

Set $d_1 = 11$, $d_2 = 3$ and consider the set Z of all variables assigned to zero by $\vec{u}$ that are not 11-isolated. By the observation above this set is non-empty. On the other hand, by Lemma 6, $|Z|$ is $O(n^\alpha)$ for some $\alpha < 1$. Consider $x \in Z$. It appears positively in at least 10 clauses of the type $(+,-,-)$. Each of these clauses is either unsatisfied or contains a variable assigned to 0. Suppose there are k unsatisfied clauses among them. Since $\vec{u}$ is a local maximum, to prevent x from flipping, x must be supported by at least k support clauses, each of which contains a variable assigned to 0. Thus, at least 6 neighbors of x in $G(\varphi)$ are assigned to 0. Any two neighbors of x are at distance 2. By Lemma 7 at least 5 of the neighbors assigned to 0 are not 11-isolated, and therefore belong to Z. Thus the subgraph induced by Z in $G(\varphi)$ has average degree greater than 5, which is not possible by Lemma 2(1). □

Now we are in a position to prove statement (1) of Theorem 3.

Proof. [of Theorem 3(1)] By Lemma 5 for a $\varphi \in \Phi^{\mathsf{plant}}(n, \rho n)$ whp any assignment with dn variables equal to 1, where $\frac{1}{3} \le d \le \frac{2}{3}$, satisfies more clauses than any assignment with less than $\frac{n}{10}$ variables equal to 1. Then, whp a random initial assignment for LS assigns between $\frac{1}{3}$ and $\frac{2}{3}$ of all variables to 1. Therefore, whp LS never arrives to a proper local maximum with less than $\frac{n}{10}$ variables equal to 1, and, by Proposition 1, to any proper local maximum. □

4.2 Failure of Local Search

We now prove statement (2) of Theorem 3. The overall strategy is the following. First, we show (Proposition 2) that in contrast to the previous case there are many proper local maxima in close proximity of the planted assignment. Then we show (Proposition 3) that those local maxima are located so that they intercept almost every run of LS, and thus almost every run is unsuccessful.

A pair of clauses $c_1 = (x_1, \overline{x}_2, \overline{x}_3)$, $c_2 = (\overline{x}_1, \overline{x}_4, x_5)$ is called a *cap* if x_1, x_5 are 1-isolated, that is they do not appear in any clause of the type $(+,-,-)$ except for c_1 and c_2, respectively, and x_2, x_3 are not 0-isolated (see Figure 4.1(a)). We denote equality $f(n) = g(n)(1 + o(n))$ by $f(n) \sim g(n)$.

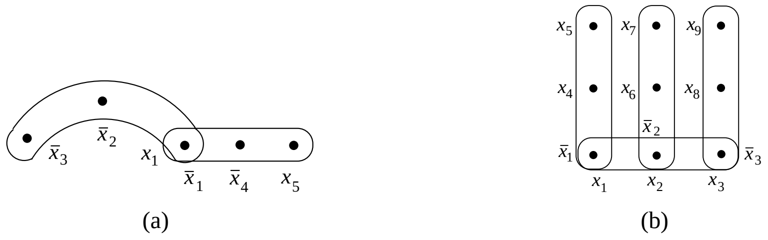


Figure 4.1: Caps and crowns

Lemma 8 *Let* $c > 0, 0 < \kappa < \frac{7}{6}$, *and density* ρ *be such that* $c < \rho \leq \kappa \cdot \ln n$. *There is* α, $0 < \alpha < 1$, *such that whp a random planted CNF* φ *from* $\Phi^{\mathsf{plant}}(n, \rho n)$ *contains at least* n^α *caps.*

Proof. The proof is fairly standard, see, e.g. the proof of Theorem 4.4.4 in [4]. We use the second moment method. The result follows from the fact that a cap has properties similar to the properties of *strictly balanced graphs*, see [4]. Take some n, and let X be a random variable equal to the number of caps in a 3-CNF $\varphi \in \Phi^{\mathsf{plant}}(n, \rho n)$. Straightforward calculation shows that the probability that a fixed 5-tuple of variables is a cap is $\sim \rho^4 n^{-4-\frac{6}{7}\frac{\rho}{\ln n}}$. Therefore $\mathbf{E}(X) \sim \rho^4 n^{1-\frac{6}{7}\frac{\rho}{\ln n}}$.

Let S be a fixed 5-tuple of variables, say, $S = (x_1, x_2, x_3, x_4, x_5)$, and A_S denote the event that S forms a cap. For any other 5-tuple T, the similar event is denoted by A_T, and we write $A_T \asymp A_S$ if these two events are not independent. By Corollary 4.3.5 of [4] it suffices to show that

$$\sum_{T \asymp S} \mathbf{P}(A_T \mid A_S) = o(\mathbf{E}(X)).$$

Let $T = (y_1, y_2, y_3, y_4, y_5)$. It is not hard to see that the only cases when A_T and A_S are not independent and the probability $\mathbf{P}(A_T \mid A_S)$ is significantly different from 0 is: $y_1 = x_1$ and $\{y_2, y_3\} = \{x_2, x_3\}$, or $y_1 = x_5$ and $\{y_2, y_3\} = \{x_1, x_4\}$, or $y_5 = x_1$ and $\{y_1, y_4\} = \{x_2, x_3\}$, or $y_5 = x_5$ and $\{y_1, y_4\} = \{x_1, x_4\}$. Then, as before, it can be found that in each of these cases $\mathbf{P}(A_T \mid A_S) = O(\rho^4 n^{-2-\frac{3}{7}\frac{\varrho}{\ln n}})$.

Finally,

$$\sum_{T \asymp S} \mathbf{P}(A_T \mid A_S) = n^2 \mathbf{P}(A_T \mid A_S) = n^2 \cdot O(\rho^4 n^{-2-\frac{3}{7}\frac{\varrho}{\ln n}}) =$$
$$O(\rho^4 n^{-\frac{3}{7}\frac{\varrho}{\ln n}}) = o(\mathbf{E}(X)).$$

We can choose $\alpha = 1 - \frac{6}{7}\kappa$ if $\rho \geq 1$, and $\alpha = 1 - 4\nu$ if $1 > \rho > n^{-\nu}$ for $\nu < \frac{1}{4}$. □

Proposition 2 *Let* $c > 0, 0 < \kappa < \frac{7}{6}$, *and density* ρ *be such that* $c < \rho \leq \kappa \cdot \ln n$. *Then there is* α, $0 < \alpha \leq 1$, *such that a 3-CNF from* $\Phi^{\mathsf{plant}}(n, \rho n)$ *whp has at least* n^α *proper local maxima.*

Indeed, let $c_1 = (x_1, \overline{x}_2, \overline{x}_3)$, $c_2 = (\overline{x}_1, \overline{x}_4, x_5)$ be a cap and $\vec{u}$ an assignment such that $u_3 = u_5 = 0$, and $u_i = 1$ for all other i. It is straightforward that $\vec{u}$ is a proper local maximum. By Lemma 8, there is α such that whp the number of such maxima is at least n^{α}.

Before proving Theorem 3(2), we note that a construction similar to caps helps evaluate the approximation rate of Local Search in the case of constant density on planted and also on arbitrary CNFs. A subformula consisting of clauses $c = (x_1, x_2, x_3), c_1 = (\overline{x}_1, x_4, x_5), c_2 = (\overline{x}_2, x_6, x_7), c_3 = (\overline{x}_3, x_8, x_9)$ is called a *crown* if the variables $x_1, \ldots, x_9$ do not appear in any clauses other than c, c_1, c_2, c_3 (see Fig. 4.1(b)). The crown is satisfiable, but the all-zero assignment is a proper local maximum. For a CNF φ and an assignment $\vec{u}$ to its variables, by $\mathsf{OPT}(\varphi)$ and $\mathsf{sat}(\vec{u})$ we denote the maximum number of simultaneously satisfiable clauses and the number of clauses satisfied by $\vec{u}$, respectively.

Corollary 1 *If density ρ is such that $c \le \rho \le \kappa \ln n$ for some $c > 0, 0 < \kappa < 1/27$, then there is $\gamma_\rho = \frac{1}{o(n)}$ such that whp Local Search on a 3-CNF $\varphi \in \Phi(n, \rho n)$ ($\varphi \in \Phi^{\mathsf{plant}}(n, \rho n)$) returns an assignment $\vec{u}$ such that $\mathsf{OPT}(\varphi) - \mathsf{sat}(\vec{u}) \ge \gamma_\rho \cdot n$, where $\mathsf{OPT}(\varphi)$ denotes the maximal number of clauses in φ that can be simultaneously satisfied and $\mathsf{sat}(\vec{u})$ denotes the number of clauses satisfied by $\vec{u}$.*

If ρ is constant then γ_ρ is also constant.

Proof of Corollary 1 is similar to that of Lemma 8. It can be shown that for ρ that satisfies conditions of this theorem there is $\gamma' = \frac{1}{o(n)}$ such that whp a random (random planted) formula has at least $\gamma' n$ crowns. If ρ is a constant, γ' is also a constant. For a random assignment $\vec{u}$, whp the variables of at least $\frac{\gamma'}{1024} n$ crowns are assigned zeroes. Such an all-zero assignment of a crown cannot be changed by Local Search.

Proposition 3 *Let $c > 0, 0 < \kappa < \frac{7}{6}$, and density ρ be such that $c < \rho \le \kappa \cdot \ln n$. Then Local Search on a 3-CNF from $\Phi^{\mathsf{plant}}(n, \rho n)$ whp ends up in a proper local maximum.*

In what follows we prove Proposition 3.

If $\rho = o(\ln n)$ then Proposition 3 follows from Corollary 1. So we assume that $\rho > \kappa' \cdot \ln n$. The main tool for the proof is coupling of Local Search (LS) with the algorithm STRAIGHT DESCENT (SD) that on each step chooses at random a variable assigned to 0 and changes its value to 1. Obviously SD is not a practical algorithm, since to apply it we need to know the solution. For the purposes of our analysis we modify SD as follows. At each step SD chooses a variable at random, and if it is assigned 0 changes its value (see Fig. 4.2(a)). The algorithm LS is modified in a similar way (see Fig. 4.2(b)).

```
INPUT: $\varphi \in \Phi^{\mathsf{plant}}(n, \rho n)$ with the all-ones
  solution, Boolean tuple $\vec{u}$,
OUTPUT: The all-ones Boolean tuple.
ALGORITHM:
while there is a variable assigned 0
  pick uniformly at random variable $x_j$ from
    the set of all variables
  if $u_j = 0$ then set $u_j = 1$
```

(a)

```
INPUT: 3-SAT formula $\varphi$, Boolean tuple $\vec{u}$,
OUTPUT: Boolean tuple $\vec{v}$, which is a local maximum of $\varphi$.
ALGORITHM:
while $\vec{u}$ is not a local maximum
  pick uniformly at random variable $x_j$ from the set of
  all variables
  if the number of clauses that can be made satisfied by
    flipping the value of $x_i$ is strictly greater than the
    number of those made unsatisfied
    then set $u_i = \overline{u}_i$
```

(b)

Figure 4.2: Straight Descent (a) and Modified Local Search (b)

We will frequently use the following two properties of the algorithm SD. Intuitively speaking the first one follows from the observation that the vector obtained by SD at step t does not depend on the formula.

Lemma 9 *If SD starts its work at a random vector with m_0 ones and after step t, $t \le n - m_0$, it arrives to a vector with m ones, then this vector is selected uniformly at random from all vectors with m ones.*

Proof. Let us denote the probability that at step t SD arrives to vector $\vec{u}$, conditional on it starting from a vector with m_0 ones, by $\mathbf{P}(\vec{u}, t, m_0)$. We prove by induction on t that $\mathbf{P}(\vec{u}_1, t, m_0) = \mathbf{P}(\vec{u}_2, t, m_0)$ for any $\vec{u}_1, \vec{u}_2$ with m ones. We denote this number by $\mathbf{P}(t, m, m_0)$. As the starting vector is random, it is obvious for $t = 0$. Then for $t > 1$ and any vector $\vec{u}$ with m ones we have

$$\begin{aligned}\mathbf{P}(\vec{u}, t, m_0) &= \mathbf{P}(\vec{u}, t-1, m_0) \cdot \frac{m}{n} + \sum_{\vec{u}'} \mathbf{P}(\vec{u}', t-1, m_0) \cdot \frac{1}{n} \\ &= \mathbf{P}(t-1, m, m_0) \cdot \frac{m}{n} + \mathbf{P}(t-1, m-1, m_0) \cdot \frac{m}{n},\end{aligned}$$

where n is the number of variables in the formula and $\vec{u}'$ goes over all vectors that can be obtained from $\vec{u}$ by flipping a one into zero. It does not depend on a particular vector $\vec{u}$. □

Lemma 10 *Whp the running time of SD does not exceed $2n \ln n$.*

Proof. For a variable x_i the probability that it is not considered for t steps equals $\left(1 - \frac{1}{n}\right)^t$. So for $t = 2n \ln n$ this probability equals $\left(1 - \frac{1}{n}\right)^{2n \ln n} \le e^{-2 \ln n} = n^{-2}$. Applying the union bound over all variables we obtain the required statement. □

Given 3-CNF φ and an assignment $\vec{u}$ we say that a variable x_i is *k-righteous* if the number of clauses voting for it to be one is greater by at least k than the number of clauses voting for it to be zero. Let $\varphi \in \Phi^{\mathsf{plant}}(n, \rho n)$ and $\vec{u}$ be a Boolean tuple. The *ball* of radius m with the center at $\vec{u}$ is the set of all tuples of the same length as $\vec{u}$ at Hamming distance at most m from $\vec{u}$. Let $f(n)$ and $g(n)$ be arbitrary functions and d be an integer constant. We say that a set S of n-tuples is *$(g(n), d)$-safe,* if for any $\vec{u} \in S$ the number of variables that are not d-righteous does not exceed $g(n)$. A run of SD is said to be *$(f(n), g(n), d)$-safe* if at each step of this run the ball of radius $f(n)$ with the center at the current assignment is $(g(n), d)$-safe.

For a proof of the following lemma we shall need the following observation that can be checked using the inequality $\binom{n}{\ell} \le \left(\frac{ne}{\ell}\right)^{\ell}$. For any n, γ, and α with $0 < \alpha < 1$

$$\binom{n}{\gamma n^{\alpha}} \le e^{(1-\alpha)\gamma n^{\alpha} \ln n - \gamma n^{\alpha} \ln \gamma + \gamma n^{\alpha}}. \tag{4.1}$$

Lemma 11 *Let $\rho > \kappa' \cdot \ln n$ for some $\kappa', \kappa' > 0$. For any positive constants γ and d there is a constant $\alpha_1 < 1$ such that, for any $\alpha > \alpha_1$, whp a run of SD on $\varphi \in \Phi^{\mathsf{plant}}(n, \rho n)$ is $(\gamma n^{\alpha}, n^{\alpha}, d)$-safe.*

Proof. Consider a run of SD on $\varphi \in \Phi^{\mathsf{plant}}(n, \rho n)$ with a random initial assignment. If SD starts its work at a tuple with m_0 ones, then at step t it has $m \le m_0 + t$ ones. Then by Lemma 9 if at step t the current assignment of SD has m ones then it is drawn uniformly at random from all vectors with m ones. Event *Unsafe* = "run of SD is not $(\gamma n^{\alpha}, n^{\alpha}, d)$-safe" is a union of events "at step t of SD's run the ball of radius γn^{α} with the center at the current assignment is not (n^{α}, d)-safe". We will use the union bound to show that probability of *Unsafe* is small.

Let $\vec{u}$ be a Boolean n-tuple having pn ones. Since whp the number of 1s in the initial assignment is at least $\frac{n}{3}$, for every step the number of 1s is at least $\frac{n}{3}$. Let M be an arbitrary set of variables with $|M| = n^{\alpha}$. We consider events $\mathcal{B}_M^{each}$ = "every variable $x_i \in M$ is not k-righteous" and $\mathcal{B}_M^{all}$ = "the total number of votes given by clauses for variables in M to be 1 does not exceed the total number of votes given by clauses for variables in M to be 0 plus $|M| \cdot k$."

The same technique as in Lemma 6 can be used to show that the probability of $\mathcal{B}_M^{all}$ and consequently the probability of $\mathcal{B}_M^{each}$ is bounded above by

$$e^{-\lambda' n^{\alpha} \ln n}$$

for some constant λ', not dependent on α. By inequality (4.1), there are at most $\gamma n^\alpha \cdot e^{\gamma(1-\alpha)n^\alpha \ln n\cdot(1+o(1))}$ distinct assignments in the γn^α-neighborhood of SD and $e^{n^\alpha(1-\alpha)\ln n(1+o(1))}$ distinct subsets of size n^α. So for α close to 1 the union bound implies that $\mathcal{B}_M^{each}$ whp does not take place for any tuple, any subset of variables at any step which completes the proof of the lemma. □

For CNFs ψ_1, ψ_2 we denote their conjunction by $\psi_1 \wedge \psi_2$.

We will use formulas that are obtained from a random formula by adding some clauses in an ‘adversarial’ manner. Following [41] we call distributions for such formulas *semi-random*. However, the type of semi-random distributions we need is different from that in [41]. Let $\eta < 1$ be some constant. A formula φ is sampled according to semi-random distribution $\Phi_\eta^{\mathtt{plant}}(n, \rho n)$ if $\varphi = \varphi' \wedge \psi$, where φ' is sampled according to $\Phi^{\mathsf{plant}}(n, \rho n)$ and ψ contains at most n^η clauses and is given by an adversary.

Corollary 2 *If $\varphi' \in \Phi_\eta^{\mathtt{plant}}(n, \rho n)$ then for any positive constants γ and d there is a constant $\alpha_2 < 1$ such that for any $\alpha > \alpha_2$ a run of SD on $\varphi' \wedge \psi$ is whp $(\gamma n^\alpha, 2n^\alpha, d)$-safe.*

Indeed, let α_1 be obtained by application of Lemma 11 to φ'. Let $\alpha_2 = \max(\alpha_1, \eta)$. Then for $\alpha > \alpha_2$ whp run of SD on φ' is $(\gamma n^\alpha, n^\alpha, d)$-safe. Since for n large enough ψ contains less than n^α variables a run of SD will be $(\gamma n^\alpha, 2n^\alpha, d)$-safe on $\varphi' \wedge \psi$.

Lemma 12 *Let $(D_0, \dots, D_l)$ be an integer random process, $d > 0$, and let L, H be integer constants such that (a) $D_0 = 0$, $0 < L < H$; (b) $|D_{\tau+1} - D_\tau| = 1$; (c) if $L \le D_\tau \le H$ the expectation of $D_{\tau+1}$ conditioned on D_τ satisfies the inequality $\mathbf{E}\,(D_{\tau+1}|D_\tau) < D_\tau - d$. Then the probability that there is τ such that $D_\tau > H$ is less than $l \cdot e^{-d\frac{H-L}{2}}$.*

Proof. We define a set of auxiliary processes D_τ^ξ:

$$D_\tau^\xi = \begin{cases} L, & \text{if } \tau < \xi, \\ D_\tau, & \text{if } (\tau \ge \xi),\ (D_\xi = L) \text{ and } (D_\zeta \ge L),\ \text{for all } \zeta \in \{\xi, \dots, \tau\}), \\ D_\zeta - d(\tau - \zeta), & \text{if } \tau > \xi,\, D_\xi = L, \text{ and } \zeta \in \{\xi, \dots, \tau\} \text{ is the least such that } D_\zeta < L, \\ L - d(\tau - \xi), & \text{otherwise, i.e., } D_\xi \ne L \text{ and } \tau \ge \xi. \end{cases}$$

The processes $D_\tau^0, \dots, D_\tau^l$ are designed so that every D_τ^ξ for $\tau \ge \xi$ satisfies inequality $\mathbf{E}\left(D_{\tau+1}^\xi | D_\tau^\xi\right) \le D_\tau^\xi - d$. Indeed, suppose that $\tau \ge \xi$. If $D_\xi \ne L$ then

$$\mathbf{E}\left(D_{\tau+1}^\xi | D_\tau^\xi\right) = L - d(\tau + 1 - \xi) = (L - (\tau - \xi) - d = D_\tau^\xi - d.$$

Let $D_\xi = L$. If $D_\zeta \geq L$ for all $\zeta\{\xi,\dots,\tau\}$ then $D^\xi_\tau = D_\tau$, $D^\xi_{\tau+1} = D_{\tau+1}$, and the result follows from the assumption $\mathbf{E}\left(D_{\tau+1}|D_\tau\right) < D_\tau - d$. If there is $\zeta \in \{\xi,\dots,\tau\}$ with $D_\zeta < L$ then

$$\mathbf{E}\left(D^\xi_{\tau+1}|D^\xi_\tau\right) = \mathbf{E}\left(D^\xi_{\tau+1}|D_\zeta\right) = D_\zeta - d(\tau+1-\zeta) = (D_\zeta - d(\tau-\zeta)) - d = D^\xi_\tau - d.$$

By Azuma's inequality (2.2) for each ξ the probability of the event "there exists τ such that $D^\xi_\tau = H$" is less than $e^{-(H-L)d}$.

On the other hand let $D_\tau > L$ and ξ be equal to the number of the most recent step for which $D_\xi = L$. It is easy to see that $D_\tau = D^\xi_\tau$. Thus if at some step $D_\tau = H$ then there is $\xi < \tau$ such that $D^\xi_\tau = H$. Using the union bound we get the required inequality. □

Lemma 13 *Let $\rho > \kappa' \cdot \ln n$ for some $\kappa', \kappa' > 0$. Let φ be a random 3-CNF sampled according to distribution $\Phi^{\texttt{plant}}_\eta(n, \varrho n)$ such that run of SD on φ is whp $(\gamma_1 n^\alpha, \gamma_2 n^\alpha, 1)$-safe for some positive constants γ_1, γ_2 with $\gamma_1 > 3\gamma_2$. Let $\vec{u}_d(m), \vec{u}_l(m)$ denote the pair of assignments produced by the pair of processes (SD,LS) on step m. For any t, whp the Hamming distance between $\vec{u}_d(t)$ and $\vec{u}_l(t)$ does not exceed $\gamma_1 n^\alpha$.*

Proof. Let N_t be the set of tuples at Hamming distance at most $\gamma_1 n^\alpha$ from $\vec{u}_d(t)$, and $\mathcal{E}$ be event "$\vec{u}_l(t) \notin N_t$ for some t". LS starts with the same initial assignment as SD and we will show that it does not leave N_t.

At some steps the distance between $\vec{u}_d(t)$ and $\vec{u}_l(t)$ remains the same, and at some it changes. Let $\vec{u}_d, \vec{u}_l$ be the assignments produced by the algorithms after τ changes have taken place, and D_τ be the distance between them. If $2\gamma_2 n^\alpha < D_\tau < \gamma_1 n^\alpha$ we have $\mathbf{E}\left(D_{\tau+1}|D_\tau\right) < D_\tau - \frac{1}{3}$. Indeed, the number of variables voted to be zero does not exceed $\gamma_2 n^\alpha$ and is at least twice less than the number of variables that differ in $\vec{u}_d(t)$ and $\vec{u}_l(t)$. Since any change in the distance between the assignments happens if and only if a variable voted to be 0 or a variable at which $\vec{u}_d(t)$ and $\vec{u}_l(t)$ are different is considered by SD, we have the required inequality. Now we can apply Lemma 12 for D setting $L = 2\gamma_2 n^\alpha, H = 3\gamma_2 n^\alpha, d = 1/3$ and get that probability of LS leaving N_t is less than $\varrho n e^{-n^\alpha/6}$. □

Corollary 3 *For $\varphi \in \Phi^{\texttt{plant}}_\eta(n, \varrho n)$ there is a constant $\alpha_3 < 1$ such that distance between $\vec{u}_d(t)$ and $\vec{u}_l(t)$ defined in Lemma 13 whp does not exceed n^{α_3}.*

We say that a variable *plays d-righteously in a run of LS* if every time it is considered for flipping it is d-righteous. Combining corollaries 2 and 3 we obtain the following

Lemma 14 *For any d there is $\alpha_4 < 1$ such that, for a run of LS on $\varphi \in \Phi_\eta^{\mathtt{plant}}(n, \varrho n)$ whp the number of variables that do not play d-righteously is bounded above by n^{α_4}.*

Proof. From Corollaries 2 and 3 it follows that whp at every step of LS the number of variables that are not d-righteous is less than $n^{\tilde{\alpha}}$, for some $\tilde{\alpha}$.

Therefore denoting the number of different assignments considered by LS by T (note that $T \le \rho n$) and observing that at each step the probability to consider a variable voted to be 0 is $n^{\tilde{\alpha}-1}$ we obtain the following upper bound for the expectation of the number of non-d-righteous variables throughout the run: $Tn^{\tilde{\alpha}-1} \le \kappa' n(\ln n)n^{\tilde{\alpha}-1} = \kappa' n^{\tilde{\alpha}} \ln n \le n^{\tilde{\alpha}+\varepsilon}$, for arbitrary ε with $\tilde{\alpha} + 2\varepsilon < 1$. We apply Markov inequality and obtain $\mathbf{P}\left(I > n^{\tilde{\alpha}+2\varepsilon}\right) \le n^{-\varepsilon}$, where I denotes the number of variables that do not play d-righteously. Now α_4 can be set to be $\tilde{\alpha} + 2\varepsilon$. $\square$

A clause $(\overline{x}, \overline{y}, z)$ is called a *cap support* if there are w_1, w_2 such that (x, w_1, w_2, y, z) is a cap in φ. For a formula ψ we denote the set of variables that occur in it by $\mathrm{var}(\psi)$. For a set of clauses K we denote by $\bigwedge K$ a CNF formula constructed by conjunction of the clauses. For the sake of simplicity we will write $\mathrm{var}(K)$ instead of $\mathrm{var}\left(\bigwedge K\right)$. In what follows it will be convenient to view a CNF as a sequence of clauses. Note that representation of a CNF is quite natural when we sample a random CNF by generating random clauses. This way every clause occupies certain position in the formula. For a set of positions P we denote the formula obtained from φ by removing all clauses except for occupying positions P by $\varphi\downarrow_P$. The set of variables occurring in the clauses in positions in P will be denoted by $\mathrm{var}(P)$.

We denote by $\mathcal{C}$ the set of all possible clauses over n variables. Let us fix a real constant $\nu < 1$. We will need the following notation:

– let $[k]$ denote the set of the first k positions of clauses in φ, V be the set of all variables in φ;

– let $S^{\varphi,\nu}$ be the set of positions from $[n^\nu]$ occupied by clauses that are cap supports in φ, and $L^{\varphi,\nu}$ be the set of variables that occur in clauses in positions $S^{\varphi,\nu}$;

– let $T^{\varphi,\nu}$ be set of positions of φ occupied by clauses containing a variable from $L^{\varphi,\nu}$;

– let $U^{\varphi,\nu}$ be the set of positions in φ occupied by clauses containing a variable from $\mathrm{var}\left(\varphi\downarrow_{[n^\nu]\setminus S^{\varphi,\nu}}\right)$;

– finally, let $M^{\varphi,\nu} = \mathrm{var}(T^{\varphi,\nu})$ and $N^{\varphi,\nu} = \mathrm{var}(U^{\varphi,\nu})$.

Fig. 4.3 pictures the notation just introduced.

Lemma 15 *Let $c > 0, 0 < \kappa < \frac{7}{6}$, and density ρ be such that $c < \rho \le \kappa \cdot \ln n$. Then there is $\mu_0 > 0$ such that for any $\mu < \mu_0$ there is $\nu < 1$ such that whp: (1) $|S^{\varphi,\nu}| \sim n^\mu$; (2) $M^{\varphi,\nu} \cap N^{\varphi,\nu} = \varnothing$,*

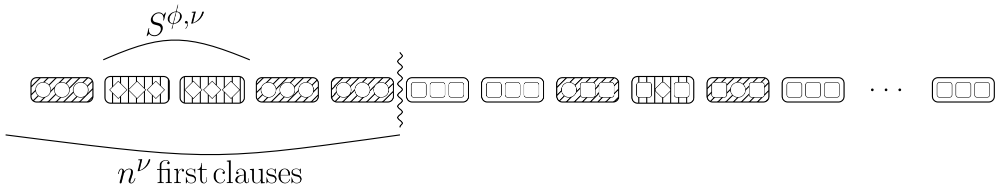


Figure 4.3: A scheme of a 3-CNF. Every clause is shown as a rectangle with its literals represented by squares inside the rectangle. Literals corresponding to variables from $L^{\phi\nu}$ and from var $\left(\varphi\downarrow_{[n^{\nu}]\setminus S^{\varphi,\nu}}\right)$ are shown as diamonds and circles, respectively. Shaded rectangles with vertical and diagonal lines represent clauses from $T^{\phi\nu}$ and $U^{\phi\nu}$, respectively.

i.e. variables from clauses in $U^{\varphi,\nu}$ do not appear in the same clauses with variables from $S^{\varphi,\nu}$; (3) $|M^{\varphi,\nu}| = 3|T^{\varphi,\nu}|$, that is no variable occurs twice in the clauses from $T^{\varphi,\nu}$.

Proof. It follows from Lemma 8 that for $\varrho \leq \kappa \ln n, \kappa < \frac{7}{6}$ there exists $\alpha, 0 < \alpha < 1$ such that the number of caps in the formula is $\sim n^{\alpha}$. We set

$$\mu_0 = \alpha/2, \qquad \nu = \mu + 1 - \alpha.$$

(1) For a subset R of all positions of clauses in ϕ let $\mathcal{C}_R$ denote event "R is exactly the set of positions occupied by cap supports". Obviously for any sets $R_1, R_2, |R_1| = |R_2|$ we have $\mathbf{P}(\mathcal{C}_{R_1}) = \mathbf{P}(\mathcal{C}_{R_2})$. Thus positions of the cap supports are selected uniformly at random without repetition. By straightforward computation we have expectation of the number of cap supports among first n^{ν} clauses equal approximately $n^{\alpha} \cdot n^{\nu-1} = n^{\mu+1-\alpha-1+\alpha} = n^{\mu}$ and variance is bounded above by the expectation, so it follows from Chebyshev inequality that random variable "number of cap supports among first n^{ν} clauses" is whp $\sim n^{\mu}$.

(2) By Lemma 2(2) whp there is no variable that occurs in more than $\ln^2 n$ clauses. Therefore $|M^{\varphi,\nu}| = O(n^{\mu} \ln^2 n)$ and $|N^{\varphi,\nu}| = O(n^{\nu} \ln^2 n)$. These sets are randomly chosen from an n-element set, and therefore the probability they have a common element is at most $n^{\mu+\nu-1} \ln^4 n$. By definition of μ and ν we have $\mu + \nu - 1 < \alpha/2 + \alpha/2 + 1 - \alpha - 1 = 0$.

(3) Since whp $|T^{\varphi,\nu}| = O(n^{\mu} \ln^2 n)$, the probability that two clauses from this set share a variable is bounded from above by $n^{2\mu-1} \ln^4 n$. We have $2\mu - 1 < \alpha - 1 < 0$ so this probability tends to 0. $\square$

Let n be the number of variables, let ρ be density, ν be a real constant such that $0 < \nu < 1$, T_0 and U_0 be subsets of $[\varrho n]$ such that $T_0 \cap U_0 = \varnothing$, $[n^{\nu}] \subseteq T_0 \cup U_0$ and let $S_0 = T_0 \cap [n^{\nu}]$. We denote by $H_{T_0U_0\nu}$ a hypothesis stating that φ is such that $S^{\varphi,\nu} = S_0$, $T^{\varphi,\nu} = T_0$, $U^{\varphi,\nu} = U_0$ and also $M^{\varphi,\nu} \cap N^{\varphi,\nu} = \varnothing$, $|M^{\varphi,\nu}| = 3\,|T^{\varphi,\nu}|$. The following lemma allows us to concentrate on small sets U_0, T_0.

Lemma 16 *Let $\phi \in \Phi_\eta^{\mathtt{plant}}(n, \varrho n)$, $\mu_0, \mu < \mu_0$ and ν be as in Lemma 15. If for an event E there is a sequence $\delta(n) \underset{n\longrightarrow\infty}{\longrightarrow} 0$ such that for all pairs (T_0, U_0), $|T_0 \cup U_0| < n^{2\nu}$ we have $\mathbf{P}\left(E|H_{T_0U_0\nu}\right) \leq \delta(n)$ then $\mathbf{P}\left(E\right) \underset{n\longrightarrow\infty}{\longrightarrow} 0$.*

Proof. We can bound probability of event E as

$$\begin{aligned}
\mathbf{P}\left(E\right) &\leq \sum_{T_0,U_0:|T_0\cup U_0|<n^{2\nu}} \mathbf{P}\left(E|H_{T_0U_0}\right)\mathbf{P}\left(H_{T_0U_0}\right) + \\
&+ \mathbf{P}\left(M^{\varphi,\nu} \cap N^{\varphi,\nu} \neq \varnothing \vee |M^{\varphi,\nu}| < 3\,|T^{\varphi,\nu}| \vee |T_0 \cup U_0| \geq n^{2\nu}\right) \\
&\leq \delta(n) + \mathbf{P}\left(M^{\varphi,\nu} \cap N^{\varphi,\nu} \neq \varnothing\right) + \mathbf{P}\left(|M^{\varphi,\nu}| < 3\,|T^{\varphi,\nu}|\right) + \mathbf{P}\left(|T_0 \cup U_0| \geq n^{2\nu}\right).
\end{aligned}$$

By Lemma 15 probabilities of events $M^{\varphi,\nu} \cap N^{\varphi,\nu} \neq \varnothing$ and $|M^{\varphi,\nu}| < 3\,|T^{\varphi,\nu}|$ tend to 0 as n approaches infinity. By Lemma 2 (2) we have $|T_0 \cup U_0| < n^{2\nu}$ whp. Thus we obtain the result. □

Observation 3 *If φ is selected according to $\Phi^{\mathsf{plant}}(n, \rho n)$ conditioned to $H_{T_0U_0\nu}$ then formula $\varphi\downarrow_{[\varrho n]\setminus(T_0\cup U_0)}$ has the same distribution as if it was generated by picking clauses from all clauses over variables $V \setminus \mathrm{var}([n^\nu])$ uniformly at random.*

Proof. [of Proposition 3] Let α_4 be the exponent corresponding to ρ and $d = 2$ by Lemma 14, let μ be such that $\alpha_4 + 2\mu < 1$ and let ν be taken by Lemma 15. We fix arbitrary pair (T_0, U_0) of subsets of $[\rho n]$ satisfying $T_0 \cap U_0 = \varnothing$, $[n^\nu] \subseteq T_0 \cup U_0, |T_0 \cup U_0| < n^{2\nu}$. We will bound the probability of success of Local Search under a hypothesis of the form $H_{T_0U_0\nu}$ and apply Lemma 16 to get the result.

Let $M = M^{\varphi,\nu}$ and $L = L^{\varphi,\nu}$. We split formula φ into $\varphi_1 = \varphi\downarrow_{T_0}$ and $\varphi_2 = \varphi\downarrow_{[\varrho n]\setminus T_0}$ and first consider a run of LS applied to φ_2 only. Formula φ_2 can in turn be considered as the conjunction of $\varphi_{21} = \varphi\downarrow_{U_0}$ and $\varphi_{22} = \varphi\downarrow_{[\varrho n]\setminus(T_0\cup U_0)}$. In Fig. 4.3 formula φ_1 consists of clauses shaded with vertical lines, formula φ_{21} of clauses shaded with diagonal lines and formula φ_{22} of clauses that are not shaded. By Observation 3 formula φ_{22} is sampled according to $\Phi^{\mathtt{plant}}(n - \delta_1(n), n\varrho - \delta_2(n))$ modulo names of variables where $\delta_1(n)$ and $\delta_2(n)$ are $o(n)$. So formula φ_2 is sampled according to $\Phi_{2\mu}^{\mathtt{plant}}(n - \delta_1(n), n\varrho - \delta_2(n))$. By Lemma 14 the number of variables that do not play 2-righteously during run of LS on φ_2 is bounded from above by n^{α_4} for a certain $\alpha_4 < 1$. We consider coupling $(LS_\varphi, LS_{\varphi_2})$ of runs of LS on φ and φ_2, denoting assignments obtained by the runs of the algorithm at step t by $\vec{u}_\varphi(t)$ and $\vec{u}_{\varphi_2}(t)$ respectively. Let K be the set of those variables which do not belong to L (squares and circles in Fig. 4.3). Formula φ_2 is a 3-CNF containing only variables from K. For an assignment of values of all variables $\vec{u}$ we will denote by $\vec{u}|_K$ its restriction onto variables from K. We make process LS_φ start with a random assignment $\vec{u}_\varphi(0) = \vec{u}^0_\varphi$ to all variables, and LS_{φ_2} with a random assignment $\vec{u}_{\varphi_2}(0) = \vec{u}^0_{\varphi_2}$ to variables in K, such that $\vec{u}^0_\varphi|_K = \vec{u}^0_{\varphi_2}$. Now the algorithms work as follows. At every step a random variable x_i is chosen. Process LS_φ makes its step, and process LS_{φ_2} makes its step if $x_i \in K$.

Whp LS_{φ_2} will run with at most n^{α_4} variables that do not play 2-righteously. Let W denote the set of such variables. Variables in formula φ_1 are selected uniformly at random so if $\alpha_4 + 2\mu < 1$ then whp set M does not intersect with W. Hence, every time LS_φ considers some variable from M it is 2-righteous in φ_2 and belongs to at most one clause of φ_1. Therefore such a variable is at least 1-righteous φ and is flipped to 1, or stays 1, whichever is to happen for LS_{φ_2}. Thus whp at every step of $(LS_\varphi, LS_{\varphi_2})$ we have $\vec{u}_\varphi(t)|_K = \vec{u}_{\varphi_2}(t)$. In the rest of the proof we consider only this highly probable case.

Consider some cap support $c_i = (\overline{x}_1, \overline{x}_4, x_5)$ occupying a position $i \in [n^\nu]$ and such that $x_1 = 0, x_4 = 1, x_5 = 0$ at time 0, and a set P_{c_i} of variables occurring in clauses that contain variables $\mathrm{var}(c_i)$ (obviously $\mathrm{var}(c_i) \subseteq P_{c_i}$). Let c_j be the clause that forms a cap with c_i. We say that a variable is *discovered* at step t if it is considered for the first time at step t. Let $p_1, \ldots, p_k$ be an ordering of elements of P_{c_i} according to the step of their discovery. In other words if variable p_1 is the first variable from P_{c_i} that is discovered, p_k was the last. In the case some variables are not considered at all, we place them in the end of the list in a random order. Observe that all variables that play at least 1-righteously are discovered at some step. All orderings of variables are equiprobable, hence, the probability of variables $\mathrm{var}(c_i)$ to occupy places p_{k-2}, p_{k-1} and p_k equals $3!/k(k-1)(k-2)$. We will call this ordering *unlucky*.

Let us consider what happens if the order of discovery of P_{c_i} is unlucky. All variables in $P_{c_i} \setminus \mathrm{var}(c_i)$ play 1-righteously, therefore once they are discovered by LS_φ they equal to 1. Thus when x_1, x_4, x_5 are finally considered all clauses they occur in are satisfied, except for c_j. So variables x_1, x_4, x_5 do not change their values and the clause c_j remains unsatisfied by the end of the work of LS_φ.

By Lemma 2(2) whp no vertex has degree greater than $\ln^2 n$, so the size of the set P_{c_i} is bounded above by $3\ln^2 n$. Thus the probability of event $Unluck(i) =$"order of discovery of P_{c_i} is unlucky" is greater than $\frac{1}{\ln^6 n}$. Thus, the expectation of $|\{i | Unluck(i)\}|$ equals $\frac{|S_0|}{\ln^6 n} = \frac{n^\mu}{\ln^6 n}$. By definition of $H_{T_0 U_0 \nu}$ any variable whp occurs in clauses from $T^{\varphi,\nu}$ at most once, hence there is no variable that occurs in the same clause with a variable from c_{i_1} and a variable from c_{i_2} for $i_1, i_2 \in S_0$, $i_1 \neq i_2$. This implies that events of the form $Unluck(i)$ are independent. Therefore random variable $|\{i|Unluck(i)\}|$ is Bernoulli and, as its expectation tends to infinity, the probability that it equals to 0 goes to 0. Since unlucky ordering of at least one cap support leads to failure of the LS this proves the result. $\square$

Chapter 5

On the Power of Plateau Moves

5.1 Main Result and General Intuition

The main result of this chapter is the following

Theorem 4 *For any $\kappa > 0, \rho = \kappa \ln n$ GSAT with settings* MAXFLIPS$= n^{60/\kappa+3}$, MAXTRIES$= 1$ *finds a solution for φ from $\Phi^{\texttt{plant}}(n, \rho n)$ whp.*

After $O($MAXFLIPS$\times$MAXTRIES$)$ steps that did not lead to a solution of the problem GSAT fails so Theorem 4 implies that GSAT finds solution in a polynomial number of steps.

It is sufficient to prove the result for $\kappa < 2$, since by Theorem 1 of [12] for $\kappa > \frac{7}{6}$ GSAT will find solution without even switching to plateau moves stage.

Lemmas 17-19 describe relations between variables that are assigned to 0 in a local maximum and variables that occur in few $(+,-,-)$ clauses. These lemmas lead to a proof of Lemma 20 that will be the key instrument to prove Theorem 4. This lemma is formulated in terms of a graph of co-occurrences. In terms of the original formula φ Lemma 20 states that when a local maximum of the number of satisfied clauses is reached we have the following picture. Clauses containing variables that are still assigned incorrectly fall apart into several sub-formulas $\varphi_1, \ldots, \varphi_t$. Formulas $\varphi_1, \ldots, \varphi_t$ are pairwise disjoint, that is no φ_i, φ_j contain a common clause. Moreover these sub-formulas are disjoint with respect to variables, that is no φ_i, φ_j refer to a common variable. The lemma also states that any such sub-formula φ_i that contains an unsatisfied clause contains an incorrectly assigned variable that can be flipped by GSAT. In the proof of Theorem 4 we apply Lemma 20 and observe that the solution space is with high probability such that from any proper local maximum there is a finite path along a plateau that leads to higher ground. The path is finite meaning that its length can be bounded by some constant ℓ that does not depend on n. So with constant probability among the next ℓn^{ℓ} steps there will be ℓ steps that will be made along such a path and a better assignment will be reached.

We will discuss intuition behind Lemmas 17-19 directly before stating them.

5.2 Proof of the Main Result

Let G^{φ} be the graph of co-occurences of variables in φ. That is, vertices of G^{φ} are variables; x_i and x_j are connected if x_i and x_j occur in the same clause in φ. We shall need several notions from graph theory that we are going to apply to the graph G^{φ}. For a graph $G = (V, E)$ and a subset X of its vertices we denote by $G|_X$ the subgraph induced by X, i.e. graph $(X, E \cap X^2)$. Let $l \in \mathbb{N}$ and E^l be the set of pairs of vertices that are connected by paths of length at most l in G. We denote graph (V, E^l) by G^l. We denote by $N^{G,d}(X)$ the d-th neighborhood of X in G, i.e.

$$N^{G,d}(X) = \{y \mid \text{there exists } x \in X \text{ such that } (x, y) \in E^d\}.$$

Observation 4 *For any $\rho \leq 2 \ln n$ whp no two variables occur together in more that 2 clauses of φ from $\Phi^{\mathtt{plant}}(n, \rho n)$.*

Proof. Indeed if we fix two variables and three clauses then the probability of the variables to occur together in these three clauses is $O(n^{-6})$. There are $O(n^2)$ pairs of variables and $O(m^3)$ triples of clauses so applying union bound we conclude that the probability that there exist such a pair and triple is less than $O(n^{-4}m^3)$ which tends to zero for $m \leq 2n \ln n$. $\square$

Let us recall that a variable is called Δ-isolated in φ if it occurs positively in fewer than Δ clauses of type $(+, -, -)$. From now on we shall denote the set of all Δ-isolated variables by I_{Δ}.

Intuitively, our interest in Δ-isolated variables comes from the fact that in the case of logarithmic density of the formula extremely few clauses with two or three positive literals are unsatisfied in a local maximum. This happens because almost all variables are assigned correctly and the probability that a clause has two incorrect variables is very small. Thus many unsatisfied clauses in a local maximum are $(+, -, -)$ and variables that are assigned wrong tend to be Δ-isolated variables. In the following Lemma we show that Δ-isolated variables do not “flock together” so with high probability you do not find many of them close to each other.

Lemma 17 *For any $\kappa \in \mathbb{R}$ there is a constant $l \in \mathbb{N}$ such that for any constants $\Delta \in \mathbb{N}, d \in \mathbb{N}$ and $\rho = \kappa \ln n, \varphi$ from $\Phi^{\mathtt{plant}}(n, \rho n)$ whp any connected component of $G^d|_{I_{\Delta}}$ contains fewer than l variables.*

Proof. Fix an arbitrary constant r. Let M be a set of r variables. Obviously

$$\mathbf{P}\left(\text{all variables in } M \text{ are in } I_\Delta\right)$$

is less than

$$\mathbf{P}\left(\#(\text{positive occurrences of variables from } M \text{ in } (+,-,-)\text{-clauses}) \le |M| \times \Delta\right).$$

The latter probability can be bounded above by

$$\begin{aligned} \sum_{k=0}^{r\Delta} \binom{\rho n}{k} \left(\frac{r}{n}\right)^k \left(1-\frac{3r}{7n}\right)^{\rho n - k} &\le \\ r\Delta \binom{\rho n}{r\Delta} \left(\frac{r}{n}\right)^{r\Delta} \left(1-\frac{3r}{7n}\right)^{\rho n - r\Delta} &\lesssim \\ r\Delta \rho^{r\Delta} \Delta^{-r\Delta} e^{-3/7\rho r}/((r\Delta)!) &\lesssim e^{-3\rho r/14}. \end{aligned}$$

The number of connected sets of variables in G^d can be bounded above by the number of subgraphs of G^d isomorphic to trees. Since whp the maximum degree of a variable is bounded above by $\ln^2 n$ the number of subgraphs of size r isomorphic to trees can be bounded above by $n(r\ln^{2d} n)^{r-1} \lesssim n^2$ whp.

Now we apply union bound to the probability of an event $\mathcal{E}$ = “there exists a connected set M of vertices of G^d of size r such that all variables in M are in I_Δ” getting the upper bound

$$\mathbf{P}(\mathcal{E}) \le n^2 e^{-3\kappa r \ln n/14} = e^{\ln n(2-3\kappa r/14)}.$$

Therefore if we set $l = 10/\kappa$ then for any constant $r > l$ we have the probability $\mathbf{P}(\mathcal{E})$ tending to 0. □

For an assignment $\vec{v}$ we denote by $W_{\vec{v}}$ a set of all variables assigned by $\vec{v}$ incorrectly. I. e.

$$W_{\vec{v}} = \{x_i \mid v_i = 0\}.$$

In Lemma 18 we show that in a local maximum any connected component of a graph of co-occurrences of variables assigned incorrectly needs a constant fraction of its members to belong to I_Δ.

Lemma 18 *Let $\kappa \in \mathbb{R}$, Δ be odd and $\Delta \ge 11$, $\rho = \kappa \ln n$ and φ from $\Phi^{\texttt{plant}}(n, \rho n)$. Then whp for any local maximum $\vec{v}$ any connected component C of $G|_{W_{\vec{v}}}$ contains at least $\frac{|C|(\Delta-9)}{\Delta+1}$ variables from I_Δ.*

Proof. By Lemma 6 there exists α such that $0 < \alpha < 1$ and any local maximum contains less than n^{α} zeros. Consider an arbitrary local maximum $\vec{v}$ and a connected component C of $G|_{W_{\vec{v}}}$.

Let x_i be such that $x_i \in C \setminus I_{\Delta}$. Variable x_i occurs positively in at least Δ clauses of type $(+,-,-)$. If clause c is of type $(+,-,-)$, variable x_i occurs positively in c and it is the only variable in c that is assigned to 0 then c is not satisfied and will become satisfied if x_i is flipped. But $\vec{v}$ is a local maximum and flipping x_i should not increase the number of satisfied clauses. If x_i has no neighbors assigned to 0 by $\vec{v}$ then after flipping x_i the number of satisfied clauses will increase by at least Δ. A neighbor x_j of x_i may decrease this advantage by making one of $(+,-,-)$ clauses that refer to x_i satisfied or by making some other clause where x_i occurs negatively unsatisfied. But each neighbor x_j assigned to 0 can not decrease the advantage of flipping x_i by more than the number of co-occurences of x_i and x_j in clauses of φ. By Observation 4 whp no two variables occur together in more than 2 clauses. So if x_i has t neighbors assigned to 0 then advantage of flipping x_i will be at least $\Delta - 2t$. And since the advantage must be a non positive and integer we have $t \geq (\Delta+1)/2$.

Therefore x_i must have at least $(\Delta+1)/2$ neighbors in G that are assigned to zero by $\vec{v}$. Obviously all these variables are in C, so the degree of x_i in $G|_{W_{\vec{v}}}$ is at least $(\Delta+1)/2$. We can bound the average degree of C from below by $\left(|C \setminus I_{\Delta}| \cdot (\Delta+1)/2\right)/|C|$. Since $|C| < n^{\alpha}$ by Lemma 2(1) it follows that the average degree of C is less than 5. Thus we have

$$\left(|C \setminus I_{\Delta}| \cdot (\Delta+1)/2\right)/|C| < 5$$

and consequently

$$\begin{aligned} |C \setminus I_{\Delta}| &< \frac{10|C|}{\Delta+1}, \\ |C \cap I_{\Delta}| &> \frac{|C|(\Delta-9)}{\Delta+1}. \end{aligned} \tag{5.1}$$

Since inequality (5.1) was shown for an arbitrary connected component of $G|_{W_{\vec{v}}}$ the lemma is proven. □

We say that a variable x_i is *potentially wrong* if there is an assignment $\vec{v}$ such that it is a local maximum and $v_i = 0$. The set of all potentially wrong variables is denoted by W, that is

$$W = \bigcup_{\vec{v} \text{ is a local maximum}} W_{\vec{v}}.$$

In the following lemma we show that for large enough Δ the set of Δ-isolated variables is dense in the set of potentially wrong variables. Namely that any potentially wrong variable must have at least one Δ-isolated variable within a finite (bounded by a predefined constant) distance.

Lemma 19 *Let $\kappa \in \mathbb{R}$, Δ be odd and $\Delta \geq 11$. Then there is a constant $r \in \mathbb{N}$ such that whp any $x_i \in W$ has a Δ-isolated variable at distance less than r.*

Proof. Let l be the number corresponding to κ to satisfy the conditions of Lemma 17, let $r = 7l$ and let

$$M = \{x|x \text{ is connected in } G^r \text{ to some } y \in I_\Delta\} = N^{G,r}(I_\Delta).$$

Consider an arbitrary local maximum $\vec{v}$. To prove the lemma we must show that

$$W_{\vec{v}} \subseteq M.$$

To derive a contradiction let us assume that

$$\text{there exists } x_j \in W_{\vec{v}} \setminus M. \tag{5.2}$$

Let C be a connected component of $G|_{W_{\vec{v}}}$ containing x_j. By Lemma 18 set C contains at least $\frac{|C|(\Delta-9)}{\Delta+1}$ variables from I_Δ. For $\Delta \geq 11$ we have

$$|C \cap I_\Delta| \geq 1/6|C| \tag{5.3}$$

and consequently nonempty C must contain at least 1 variable form I_Δ. Now we take arbitrary $x_k \in I_\Delta \cap C$ and consider a connected component C' of $G^{2r}|_{I_\Delta}$ that contains x_k. Note that Lemma 17 implies

$$|N^{G,r}(C') \cap I_\Delta \cap C| \leq |N^{G,r}(C') \cap I_\Delta| \leq l.$$

On the other hand since $x_j \notin M \supseteq N^{G,r}(C')$ and x_j is connected to an element of C' we have

$$|N^{G,r}(C') \cap C| \geq r = 7l.$$

By definitions of $N^{G,d}$ and G^d any distinct connected components C_1 and C_2 of $G^{2r}|_{I_\Delta}$ satisfy

$$N^{G,r}(C_1) \cap N^{G,r}(C_2) = \varnothing.$$

Therefore if $s > 0$ is the number of connected components of $G^{2r}|_{I_\Delta}$ that intersect with C we have bounds

$$|I_\Delta \cap C| \leq sl \tag{5.4}$$

and

$$|C| \geq 7sl. \tag{5.5}$$

Conjunction of (5.4) and (5.5) contradicts (5.3), therefore assumption (5.2) was false and the lemma is proven. □

Now we are in a position to prove the key lemma of the proof of the main result. As it was discussed in the beginning of the section the intuitive meaning of Lemma 20 is that once a local maximum is reached we observe the following. Incorrectly assigned variables split into several finite connected components. Moreover each component that contains a variable occurring in an unsatisfied clause contains also a variable that can be flipped without reducing the number of satisfied clauses.

Lemma 20 *Let $\kappa \in \mathbb{R}$, $\rho = \kappa \ln n$ and $\varphi \in \Phi^{\mathtt{plant}}(n, \rho n)$. Then there is $s \in \mathbb{N}$ such that whp for any local maximum $\vec{v}$ and any connected component C of $G|_{W_{\vec{v}}}$ the following statements is true:*

- *C contains less than s elements,*
- *if there is an unsatisfied clause containing a variable from C then there is a variable $x_j \in C$ such that the number of unsatisfied clauses where x_j occurs equals to the number of clauses that are satisfied only by x_j.*

Proof. We fix some local maximum $\vec{v}$ and a connected component C of $G|_{W_{\vec{v}}}$.

By Lemma 19 for $\Delta = 11$ there is a constant r such that whp for every variable $x_i \in C$ there is a variable $x_j \in I_\Delta$ such that distance between x_i and x_j is less than r. Let us denote such x_j by $x_i\!\uparrow$. It is easy to see that set $\{x_i\!\uparrow |x_i \in C\} \cup (C \cap I_\Delta)$ is connected in $G^{2r+1}|_{I_\Delta}$ and by Lemma 17 its size can not be greater than some constant l. Thus $|C \cap I_\Delta| < l$ and by Lemma 18 for $\Delta = 11$ we have $|C| < 6l$. So we set $s = 6l$ and have the first statement of the lemma proven. Note that in the proof of Lemma 18 we set $l = 10/\kappa$ so here we have $s = 60/\kappa$.

To prove the second statement of the lemma we consider an arbitrary variable x_j in C and a clause c where x_j occurs.

Observation 5 *For c to be satisfied only by x_j in $\vec{v}$ the following two conditions are necessary: (a) x_j occurs in c negatively, (b) there is a variable x_i that occurs in c positively and such that $v_i = 0$.*

Consider a directed graph

$$\mathcal{S}_C = (C, \{(x_i, x_j)|\text{there is a clause } c \in \varphi \text{ containing literals } x_i \text{ and } \neg x_j\}).$$

Assume for the sake of contradiction that for any variable $x_j \in C$ the number of clauses that are satisfied only by x_j is strictly greater than the number of unsatisfied clauses where x_j occurs. Then for each $x_j \in C$ there is at least one clause that is satisfied only by x_j. This by Observation 5 means that the in-degree of every vertex in $\mathcal{S}_C$ is at least 1. Set C contains a variable x_k that occurs in some unsatisfied clause c. So there must be at least two clauses that are satisfied only by x_k. Thus the in-degree of x_k in $\mathcal{S}_C$ is at least 2 and $\mathcal{S}_C$ contains at least $|C| + 1$ edges. If variables are connected in $\mathcal{S}_C$ they are connected in $G|_C$ and we have that $G|_C$ contains at least $|C| + 1$ edges.

We finish the proof by showing that for any constants h and q such that $h > q$ whp there is no set of variables C such that $|C| = q$ and the graph $G|_C$ contains h edges. Indeed there are $\binom{n}{q}$ sets of variables of size q and $\binom{m}{h}$ sets of clauses of size h. For a given set of clauses of size h the probability to have $2h$ positions to be occupied by variables from a given set $C, |C| = q$ can be bounded from above by $(3h)^{2h} n^{-2h}$. Applying union bound we have that the probability under consideration is less than $(3h)^{2h} m^h n^q n^{-2h}$ which tends to 0 for any fixed κ if $h > q$.

$\square$

We are now in a position to prove the main result.

Proof of Theorem 4. By Lemma 20 once GSAT reaches a local maximum $\vec{v}$ set $W_{\vec{v}}$ falls apart into several connected components of size at most s. If there are no more unsatisfied clauses left then the problem is solved and GSAT returns a satisfying assignment. Otherwise let us consider a connected component C of $G|_{W_{\vec{v}}}$ that contains a variable x_i that occurs in an unsatisfied clause.

We show that with probability at least n^{-s} after s steps GSAT will be at some assignment $\vec{u}$ that satisfies more clauses than $\vec{v}$. By Lemma 20 there is a variable $x_{i_1} \in C$ such that the number of clauses that are satisfied only by x_{i_1} equals the number of clauses that contain x_{i_1} and are not satisfied. Thus with probability $1/n$ variable x_{i_1} will be the next variable that is flipped by GSAT. If it happens then for an assignment $\vec{v}'$ obtained at the next step there are two possibilities: 1) $\vec{v}'$ is not a local maximum or 2) $\vec{v}'$ is still a local maximum. In case 1) $GSAT$ will increase the number of satisfied clauses at the next step. In case 2) let C_1 be a subset of $C \setminus \{x_{i_1}\}$ that is a connected component of $G|_{W_{\vec{v}}}$ and contains variable that occurs in an unsatisfied clause. We have $|C_1| \leq |C| - 1$ and with probability $1/n$ a variable from C_1 will be flipped by GSAT at the next step, which will either lead to an increase of the number of satisfied clauses or to a new set $C_2, |C_2| \leq |C| - 2$, etc. Size of C is at most s so with probability greater than n^{-s} after s steps the number of satisfied clauses will be increased.

Therefore if GSAT is at a local maximum then in sn^{s+1} steps it will increase the number of satisfied clauses with probability at least $1-(1-n^{-s})^{n^{s+1}} \approx 1-e^{-n}$. So once the local maximum is reached for the first time the problem will be solved after sn^{s+2} steps with probability at least $1-ne^{-n}$.

□

5.3 Discussion

Note that we never used greediness of GSAT and all the reasoning would go through in the very same way for CSAT. Therefore, we have the analogous result for CSAT.

Corollary 4 *The CSAT algorithm with settings* MAXFLIPS$= n^{60/\kappa+3}$, MAXTRIES$= 1$ *solves random planted 3-SAT of logarithmic density whp.*

Comparing Corollary 4 with Theorem 3 and noting that CSAT is the basic Local Search enhanced with plateau moves and restarts we can conclude that adding plateau moves to Local Search increases its power substantially. The intuitive essence of this conclusion is by no means novel but now we have it rigorously proven within the context of random planted 3-SAT.

We believe that the analysis of the landscape of the solution space of random planted 3-SAT carried out in our work gives more general intuitive understanding of the process of the execution of the local search algorithms.

Chapter 6

WRW: A Candidate to Solve Random Planted 3-SAT

In the last chapter we present the Weighted Random Walk algorithm (see Fig. 1.6). This algorithm is obtained by a simple modification of the Random Walk which results in a substantial increase in efficiency. We present experimental data and prove that the algorithm reaches a good approximation of the solution of Random Planted 3-SAT of high density.

6.1 Theoretical Analysis of the Approximation Ratio

We first analyze the behavior of the algorithm given a formula φ that has all clauses that are satisfied by $\vec{1} = (1, \ldots, 1)$. Then we show that if the density of a formula is high enough then we get the same result.

Let $\vec{x}$ be some assignment of boolean values to the variables. Let A be the set of variables assigned to 1, and B be the set of variables assigned to 0. We denote by $\mathcal{C}_{AAA}, \mathcal{C}_{AAB}, \mathcal{C}_{ABB}, \mathcal{C}_{BBB}$ sets of clauses containing three, two, one and no variables from A respectively.

6.1.1 Formula with All Clauses

In this section we analyze the performance of WRW on a 3-SAT formula φ that contains all clauses that are satisfied by the all-ones assignment, i.e. all clauses that have at least one positive literal.

Let $\vec{x}$ and $w(\cdot)$ be a vector of values of the variables and weight function at a step of WRW, and let $\vec{x}', w'(\cdot)$ be a vector and weight function which are obtained from $\vec{x}, w(\cdot)$ by performing one step of the algorithm. Let A_i be the set of all variables that have value 1 and weight i, B_i be the set of all variables that have value 0 and weight i, and let $a_i = \frac{|A_i|}{n}, b_i = \frac{|B_i|}{n}, a = \sum_{i=1}^{K} a_i, b = 1 - a$.

We consider function $V(\vec{x}) = \sum_{i=1}^{K} ia_i - \sum_{i=1}^{K}(i-1)b_i$, which is obviously bounded by K. We will show that there exists a positive constant δ such that for any $\vec{x}$ we have

$$\mathbf{E}\left(V(\vec{x}') - V(\vec{x})\right) > \delta, \tag{6.1}$$

which by the Azuma's inequality implies that whp after $O(n)$ steps $V(\vec{x})$ becomes equal to K, and consequently the process stops.

Lemma 21 *Let $\vec{x}$ be an assignment and X be some variable from A, and let Y be some variable from B. If C is an unsatisfied clause, chosen uniformly at random, then the probability that X occurs in C is $\frac{3(1-a^2)}{(1-a^3)n}$ and the probability that Y occurs in C is $\frac{3}{(1-a^3)n}$.*

Proof. Let C be a clause chosen uniformly at random (not necessary unsatisfied). Then

- $\mathbf{P}(C \in \mathcal{C}_{AAA}) = a^3, \mathbf{P}(X \in C | C \in \mathcal{C}_{AAA}) = \frac{3}{an}$,
  $\mathbf{P}(Y \in C | C \in \mathcal{C}_{AAA}) = 0$,
- $\mathbf{P}(C \in \mathcal{C}_{ABB}) = 3a^2b, \mathbf{P}(X \in C | C \in \mathcal{C}_{AAB}) = \frac{2}{an}$,
  $\mathbf{P}(Y \in C | C \in \mathcal{C}_{AAB}) = \frac{1}{bn}$,
- $\mathbf{P}(C \in \mathcal{C}_{AAB}) = 3ab^2, \mathbf{P}(X \in C | C \in \mathcal{C}_{ABB}) = \frac{1}{an}$,
  $\mathbf{P}(Y \in C | C \in \mathcal{C}_{ABB}) = \frac{2}{bn}$,
- $\mathbf{P}(C \in \mathcal{C}_{BBB}) = b^3, \mathbf{P}(X \in C | C \in \mathcal{C}_{AAA}) = 0$,
  $\mathbf{P}(Y \in C | C \in \mathcal{C}_{BBB}) = \frac{3}{bn}$.

In the first case the clause will definitely be satisfied and in each of the latter three the probability that the clause is unsatisfied equals $\frac{1}{7}$. So we have $\mathbf{P}(\neg C(\vec{x})) = \frac{1}{7}(3a^2b + 3ab^2 + b^3) = \frac{1}{7}(1-a^3)$.

Now we compute

$$\begin{aligned}\mathbf{P}(X \in C \,\&\, \neg C(\vec{x})) &= \mathbf{P}(X \in C \,\&\, \neg C(\vec{x}) | C \in \mathcal{C}_{AAB})\,\mathbf{P}(C \in \mathcal{C}_{AAB}) + \\ &\quad \mathbf{P}(X \in C \,\&\, \neg C(\vec{x}) | C \in \mathcal{C}_{ABB})\,\mathbf{P}(C \in \mathcal{C}_{ABB}) + \\ &\quad \mathbf{P}(X \in C \,\&\, \neg C(\vec{x}) | C \in \mathcal{C}_{BBB})\,\mathbf{P}(C \in \mathcal{C}_{BBB}).\end{aligned}$$

Events $X \in C$ and $\neg C(\vec{x})$ are independent under the conditions, so we have

$$\begin{aligned}\mathbf{P}(X \in C \,\&\, \neg C(\vec{x})) &= \frac{1}{7}\Big(\frac{2}{an}3a^2b + \frac{1}{an}3ab^2 + 0b^3\Big) = \\ &= \frac{3}{7n}(2ab + b^2) = \frac{3}{7n}(1-a^2).\end{aligned}$$

We plug the obtained expression into the definition of conditional probability and get the desired expression $\mathbf{P}\left(X \in C | \neg C(\vec{x})\right) = \frac{3(1-a^2)}{(1-a^3)n}$.

The probability $\mathbf{P}\left(Y \in C | \neg C(\vec{x})\right)$ is computed similarly. □

Now let a_i, b_i correspond to $\vec{x}$ and a'_i, b'_i to $\vec{x}'$. We are interested in $\mathbf{E}\left(V(\vec{x}') - V(\vec{x})\right)$. We can express the change in V as

$$V(\vec{x}') - V(\vec{x}) = \sum_{i=1}^{K} i(a'_i - a_i) - \sum_{i=1}^{K} (i-1)(b'_i - b_i) \tag{6.2}$$

so to compute $\mathbf{E}\left(V(\vec{x}') - V(\vec{x})\right)$ we need to compute $\mathbf{E}\left(a'_i - a_i\right)$ and $\mathbf{E}\left(b'_i - b_i\right)$.

The numbers a_i and b_i are changed similarly. The set A_i changes because some variables leave it and some arrive into it. Now it is convenient to denote $c_i = a_i$ and $c_{-(i-1)} = b_i$. Let $q_{c_i \to c_{i\pm1}}$ be the number of variables that leave A_i and arrive into $A_{i\pm1}$. None of the variables can change its weight by more than one in one step, so we have

- $c'_i = c_i - q_{c_i \to c_{i-1}} - q_{c_i \to c_{i+1}} + q_{c_{i-1} \to c_i} + q_{c_{i+1} \to c_i}$, for all i, except $-K+1$ and K,
- $c'_K = c_K - q_{c_K \to c_{K-1}} + c_{b_{K-1} \to b_K}$, and similarly for c_{-K+1}.

Variables go from A_i to A_{i+1} when the weights of two variables are increased, so

$$\mathbf{E}\left(q_{a_i \to a_{i+1}}\right) = 2a_i. \tag{6.3}$$

Variables go from A_i to A_{i-1} and from A_1 to B_1 when three variables of an unsatisfied clause decrease weights/flip. Applying lemma 21 we get

$$\mathbf{E}\left(q_{a_i \to a_{i-1}}\right) = \frac{3(1-a^2)}{(1-a^3)n} a_i n = \frac{3(1-a^2)}{(1-a^3)} a_i, \mathbf{E}\left(q_{a_1 \to b_1}\right) = \frac{3(1-a^2)}{(1-a^3)} a_1. \tag{6.4}$$

Analogously we get

$$\mathbf{E}\left(q_{b_i \to b_{i-1}}\right) = \frac{3}{(1-a^3)} b_i, \mathbf{E}\left(q_{b_1 \to a_1}\right) = \frac{3}{(1-a^3)} b_1, \mathbf{E}\left(q_{b_i \to b_{i+1}}\right) = 2b_i. \tag{6.5}$$

Substituting the expressions for a'_i into (6.2) we obtain

$$V(\vec{x}') - V(\vec{x}) = \tag{6.6}$$
$$\underbrace{\sum_{i=0}^{K-1} q_{a_i \to a_{i+1}}}_{\Psi_1} \underbrace{- \sum_{i=1}^{K} q_{a_i \to a_{i-1}} - q_{a_1 \to b_1}}_{\Psi_2} \underbrace{- \sum_{i=0}^{K-1} q_{b_i \to b_{i+1}}}_{\Psi_3} \underbrace{+ \sum_{i=1}^{K} q_{b_i \to b_{i-1}} + q_{b_1 \to b_a}}_{\Psi_4}.$$

Using equations (6.3) - (6.5) we get $\mathbf{E}(\Psi_1) = 2a - 2a_K, \mathbf{E}(\Psi_2) = -\frac{3a(1-a^2)}{1-a^3}, \mathbf{E}(\Psi_3) = -2b + 2b_K, \mathbf{E}(\Psi_4) = \frac{3b}{1-a^3}$, thus

$$\mathbf{E}\left(V(\vec{x}') - V(\vec{x})\right) = 2a - 2a_K - \frac{3a(1-a^2)}{1-a^3} - 2b + 2b_K + \frac{3b}{1-a^3} = \tag{6.7}$$

$$\frac{4a^3 - a^2 - a + 1}{1 + a + a^2} - 2a_K + 2b_K. \tag{6.8}$$

Using the standard method for finding local maxima by analysis of the first derivative it can easily be shown that $\frac{4a^3-a^2-a+1}{1+a+a^2} > 0.42$, so if we could bound $2a_K$ by $c < 0.42$ then we would be able to conclude that $\mathbf{E}(V(\vec{x}') - V(\vec{x})) \geq 0.42 - c > 0$.

Next we argue that $a_K \leq \frac{1}{K}$. Thus, taking $K \geq 5$ we obtain inequality (6.1) with $\delta = 0.02$.

Lemma 22 *For any natural number K and for any $T = O(n)$ whp at any step of the WRW before step T we have $a_K \leq \frac{1}{K}$.*

Proof.

We will use Wormald's theorem to prove that the system behaves close to solutions of a system of differential equations and then argue that the variable corresponding to a_K never becomes greater than $\frac{1}{K}$. Below we check conditions (i)-(iii).

(i) As at every step only one variable is flipped we have inequalities

$$max|a'_l - a_l| \leq 1, max|b'_l - b_l| \leq 1$$

true with probability one.

(ii) This condition follows from equations (6.3) - (6.5), when we set $f_{a_i}(a_0, \ldots, b_K) = \mathbf{E}\left(q_{a_{i+1}\to a_i}\right) + \mathbf{E}\left(q_{a_{i-1}\to a_{i+1}}\right) - \mathbf{E}\left(q_{a_i\to a_{i+1}}\right) - \mathbf{E}\left(q_{a_i\to a_{i-1}}\right)$ and use obtained expressions for all $\mathbf{E}(q_\diamond)$ for $i > 0$, and similarly for f_{b_i}, f_{a_1}.

(iii) The functions $f_\diamond$ are Lipschitz, because they have finite first derivative.

Thus we get the equations

$$\begin{cases} \frac{du_l}{dx} = f_{a_l}(u_1, \ldots, u_K) \\ \frac{dv_l}{dx} = f_{b_l}(u_1, \ldots, u_K) \end{cases}, \tag{6.9}$$

and initial conditions $u_0(0) = v_0(0) = \frac{1}{2}$, for $0 < i \leq K$, $u_i(0) = v_i(0) = 0$. Almost surely $a_l(t) = u_l(t/n) + o(1)$, $b_l(t) = v_l(t/n) + o(1)$.

Thus to finish the proof of the lemma we show that the solution of system (6.9) satisfies $u_K(x) < \frac{1}{K}$.

We use induction to show the following:

Claim 1 *For any* $0 \le R < K$*, if* s *is such that* $\sum_{l=R+1}^{K} u_l(s)$ *is maximal and equals* $\alpha(K-R)$ *then* $u_R(s) > \alpha$*, which in particular means that the maximum value of* $\sum_{l=R}^{K} u_l(s)$ *is greater than* $\alpha(K-R+1)$*.*

Proof. We denote $\sum_{l=R+1}^{K} u_l(s)$ by $\mathcal{I}_{R+1}(s)$. First note that $\frac{\mathrm{d}\mathcal{I}_{R+1}(s)}{\mathrm{d}s} = q_{u_R \to u_R+1} - q_{u_{R+1} \to u_R}$, which follows directly from the definition of f, so if $\frac{\mathrm{d}\mathcal{I}_R(s)}{\mathrm{d}s} = 0$ then $u_R \ge u_{R+1}$. Indeed we have

$$q_{u_R \to u_R+1} - q_{u_R+1 \to u_R} = 2u_R - \frac{3(1-u^2)}{(1-u^3)} u_{R+1}$$

and the expression $\frac{3(1-u^2)}{(1-u^3)}$ equals 2 if $u = 1$ and is smaller if $u < 1$. Thus $u_R < u_{R+1}$ would imply $\frac{\mathrm{d}\sum_{l=R+1}^{K} u_l(s)}{\mathrm{d}s} < 0$.

The induction proof will go from $R = K-1$ to $R = 0$. The base of induction $R = K-1$ follows from the fact that $\frac{du_K}{ds} = 0$ implies $u_{K-1} \ge u_K$.

Induction step:

Consider s_0 such that $\mathcal{I}_{R+1}(s_0)$ is maximum and equals $\alpha(K-R)$. We have $u_R(s_0) \ge u_{R+1}(s_0)$. Assume that $u_R(s_0) < \alpha_1$, which implies $u_{R+1}(s_0) < \alpha_1$. Then

$$\mathcal{I}_{R+2}(s_0) = \mathcal{I}_{R+1}(s_0) - u_{R+1}(s_0) > (K-R)\alpha_1 - \alpha_1,$$

which leads to a contradiction as $(K-R-1)\alpha_1$ is the maximum value of $\mathcal{I}_{R+2}$. □

It follows from the Claim that if the maximum value of $\mathcal{I}_R(s)$ is $\alpha(K-R)$ then the maximum value of $\mathcal{I}_{R-1}(s)$ is at least $\alpha(K-R+1)$. Thus if the maximum value of $u_K(s)$ is α then the maximum value of $\mathcal{I}_0(s)$ is at least $K\alpha$. As $\mathcal{I}_0(s)$ cannot be greater than 1 we get the inequality $u_K(s) < \frac{1}{K}$. Thus almost surely we have $a_K(t) < \frac{1}{K}$. □

So if $K \ge 5$ then there is a constant $c > 0$ such that at every step of the WRW the expectation of the amount of change of $V(\vec{x})$ is greater than c. The value of $V(\vec{x})$ cannot change by more than 5 at every step so by Azuma's inequality we have

$$\mathbf{P}\left(V(\vec{x}^t) \le nK\right) \le 2e^{-\frac{(tc-nK)^2}{2 \cdot 25t}}.$$

The right hand side of the above equation starts to decrease rapidly when tc becomes greater than nK. This proves the following

Lemma 23 *If φ is a 3-CNF with all possible clauses that are satisfied by assignment $\vec{r}$ then almost surely WRW with $K \geq 5$ weights finds $\vec{r}$ in $O(nK)$ steps.*

6.1.2 Random Formula

In this subsection we prove

Theorem 5 *Let φ be a random 3-CNF with a planted solution $\vec{x}_0$ of density $\rho = \rho(n) \underset{n \longrightarrow \infty}{\longrightarrow} \infty$, and $\varepsilon > 0$ be some constant. With high probability WRW with more than 5 weights finds a vector that differs from $\vec{x}_0$ in at most εn coordinates.*

Let φ be a random 3-CNF with a planted solution $\vec{1}$. For $A, B, A_1 \subseteq A$ we denote by $\mathcal{C}_{AA_1B}$ the set of all unsatisfied clauses that have one variable from A_1, one variable from $A \setminus A_1$ and one from B. By $\mathcal{C}^{\varphi}_{AA_1B}$ we denote the set of clauses in φ that have this property. Analogously we define $\mathcal{C}_{AA_iB}, \mathcal{C}_{AAB_i}$, etc. We define the set of all unsatisfied clauses by $\mathcal{C}_u$ and all unsatisfied clauses in φ by $\mathcal{C}^{\varphi}_u$.

For a formula with all clauses the expectation of the number of variables that at a given step go from A_1 to B_1 equals

$$\frac{|\mathcal{C}_{AA_1B}| + 2|\mathcal{C}_{A_1A_1B}|}{|\mathcal{C}^u|}, \tag{6.10}$$

while for formula φ it is

$$\frac{|\mathcal{C}^{\varphi}_{AA_1B}| + 2|\mathcal{C}^{\varphi}_{A_1A_1B}|}{|\mathcal{C}^{\varphi}_u|}. \tag{6.11}$$

In the next lemma we show that $\frac{\mathcal{C}^{\varphi}_{AA_1B}}{\rho n}$ is close to $6aa_1b$ whp, which equals $\frac{\mathcal{C}_{AA_1B}}{\rho n}$. The same techniques can be used to show that other members of equation (6.11) divided by ρn are close to respective members of equation (6.10) divided by ρn. Under the conditions of theorem 5 we have $\frac{\mathcal{C}_u}{\rho n} = b^3 > \varepsilon^3$, thus the denominator of (6.10) is separated from zero, so expression (6.11) is close to (6.10).

Lemma 24 *Let $\rho(n)$ tend to infinity and let b be a constant greater than 0. Let also $\vec{x}$ be a Boolean assignment and $B, A, A_1 \subseteq A$ be arbitrary subsets of variables such that $\frac{|B|}{n} = b$ and variables from A equal to 1 in $\vec{x}$ and all variables from B equal to 0 in $\vec{x}$. Then the following inequality holds whp: $\left|\frac{\mathcal{C}_{AA_1B}}{\rho n} - \frac{3}{7}a(a - a_1)b\right| \leq o(1)$.*

Proof.

By simple counting it can be shown that if C is chosen uniformly at random then $\mathbf{P}\left(C \in \mathcal{C}_{AA_1B}\right) = \frac{3}{7}a_1(a-a_1)b$. Let $\mathcal{C}^{\varphi}_{AA_1B}$ be the multiset of clauses in φ that belong to $\mathcal{C}_{AA_1B}$. In total ρn clauses are chosen to be in φ, so the expectation of the size of $\mathcal{C}^{\varphi}_{AA_1B}$ equals $\frac{3}{7}a_1(a-a_1)b\rho n$. Using Chernoff bound we get

$$\mathbf{P}\left(\left|\frac{|\mathcal{C}^{\varphi}_{AA_1B}|}{\rho n} - \frac{3}{7}a_1(a-a_1)b\right| > \epsilon\right) < 2e^{-\frac{\epsilon^2\rho n}{\frac{3}{7}a_1(a-a_1)b}}.$$

We will say that φ is ϵ-*bad* if there exists an assignment, and subsets of variables $A, B, A_1 \subseteq A$ for which inequality $\left|\frac{|\mathcal{C}^{\varphi}_{AA_1B}|}{\rho n} - \frac{3}{7}a_1(a-a_1)b\right| > \epsilon$ is true.

Now we put to use the fact that the probability of a union of events is less than or equal to the sum of the probabilities of the events to estimate the probability of φ being ϵ-bad.

There are 2^n boolean assignments, 2^n ways to select A and B, and at most 2^n ways to select $A_i \subseteq A$, so we have

$$\mathbf{P}\left(\varphi \text{ is bad}\right) \leq 2e^{-\frac{\epsilon^2\rho n}{\frac{3}{7}a_1(a-a_1)b}}2^{3n} = e^{-n(\gamma_1\epsilon^2\rho-\gamma_2)}, \tag{6.12}$$

where γ_1, γ_2 are constants. We can choose $\epsilon = \rho^{-1/3} = o(1)$ so as $\rho \to \infty$ the function in the right hand side of the equation (6.12) is $o(1)$, which completes the proof. □

Thus for random CNFs with planted solution $\vec{1}$ and vectors with more than εn zeros, WRW acts as it does for the Full CNF, that is it tends to get closer and closer to $\vec{1}$. So with high probability an assignment with more than $(1-\varepsilon)n$ ones will be found.

6.1.3 Discussion

The obtained theoretical results provide intuition on the reasons of the algorithm's success. When standard Random Walk starts with a random assignment there are more occurrences of variables assigned zero in the unsatisfied clauses, so the number of zero variables decreases. But when the golden ratio conjugate is reached the numbers of occurrences of zeros and ones become equal and progress stops. For the same reasons, when WRW starts, the number of zero variables is decreased. Once the assignment contains fewer zeros than ones, the ones start benefiting from increasing weight of randomly picked variables. The problem one might expect here is that weights of some one variables grow infinitely (or up to a maximum allowed size), while other variables still stay zero. Lemma 2 shows that this is not the case with WRW: the set of one variables with maximum possible weight stays reasonably bounded, and thus the added weight is used in the 'struggle' between one and zero. Formally it is shown via use of a potential function V.

6.2 Experiments

In this section we describe experiments done with the WRW algorithm. Our experiments are done on random and random planted 3-SAT.

With regard to random 3-SAT, our experiments show that WRW works in linear time for formulas with density 3.9. We studied the running time of the algorithms on formulas with 10000 variables and then increased the number of variables in steps of 1000 until 50000. For each fixed density we ran the algorithm on 100 random instances. For each run we were waiting until the algorithm finds solution. The result of this experiment shows linear running time of WRW for density 3.9 when $K = 4$. This is interesting when compared with the empirical evidence that standard random walk requires exponential time for densities higher than 2.7.

Other experiments that we report here are for random planted 3-SAT. In the first set of experiments we try to determine for which densities WRW can solve random planted 3-SAT in a reasonable timebound. It turns out that for any fixed density WRW works reasonably fast. Our experiment was done on formulas with 10000 variables. The density started from 3 and increased to 10 in steps of 0.1. For each fixed density the algorithm ran on 100 random instances and we looked at the average running time on these 100 instances. The results of this experiment are summarized in Fig. 6.1. As it is seen, the hardest instances are those with density around 5. When the density is below 3, the formula has too many solutions and it is easy to find one. When the density is higher than 10, intuitively speaking, the formula contains a lot of information about the planted solution and this information guides the algorithm toward it.

The next set of experiments was aimed at figuring out the running time of WRW on random planted instances of a fixed density. Our observations in this part were surprising. For density 10, we ran the algorithm on instances with 10000 to 100000 variables. The running time was the highest for instances with 10000 variable and then it reduced and converged to a fixed value and remained steady. We believe that this is because this number of variables is not large enough to allow us to see the asymptotic behavior of the algorithm. We observed a similar behavior when the density was set to 4.5.

The last set of experiments was done to check how the number of variables of specific weight changes during the course of the algorithm. For this experiment K is set to 5, so there are ten

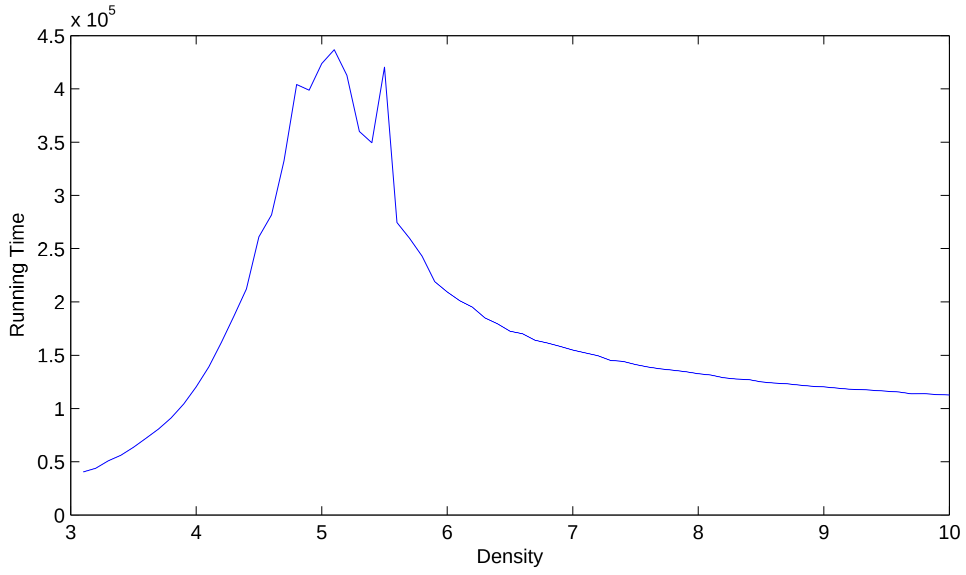


Figure 6.1: Running time vs. density.

classes of variables. The experiment was done on formulas with 10000 variables and density 30. The results are summarized in the Fig. 6.2. Each curve shows c_l, i.e. the total number of variables with a specific weight in 1000 experiments. The solid lines correspond to $c_1, \dots, c_K$, the dashed to $c_0, \dots, c_{-K+1}$ starting with the upmost ones and going down. Since all experiments were finished before 90000 steps, when time approaches to 90000 in the graph, all lines become straight. Planted solution was chosen to be all ones. Firstly, we see that for all $l, 1 \leq l < K$ we have $c_l > c_l + 1$, which agrees with Claim 1. Secondly, when $a \longrightarrow 1$ we have the value b_i/b_{i+1} growing, while the value a_i/a_{i+1} decreases, as it could have been predicted by (6.3), (6.4) and (6.5). This intuitively means that the weights are more and more evenly distributed over ones, while there are more zeros with small weights than with bigger weights.

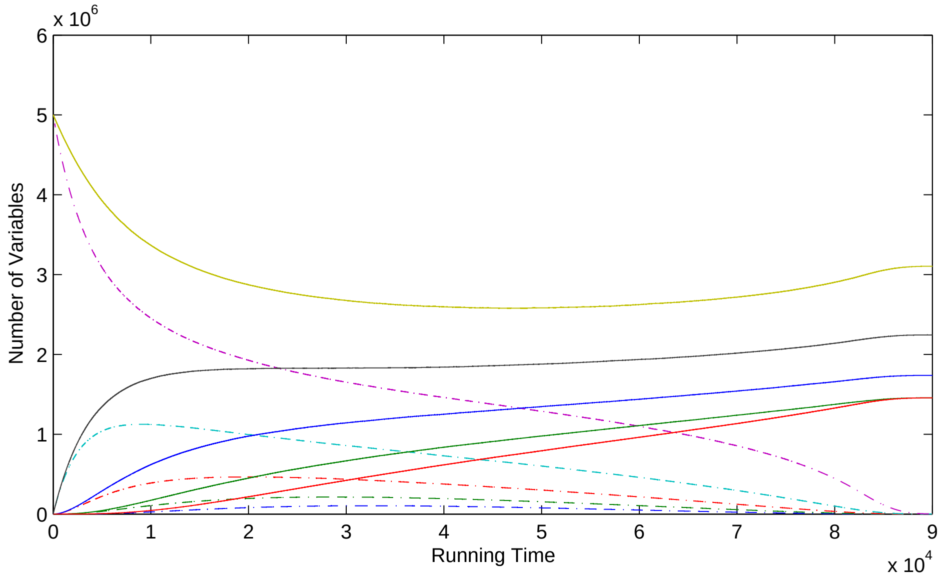


Figure 6.2: Number of variables with different weights vs. time.

Bibliography

[1] Dimitris Achlioptas. Setting 2 variables at a time yields a new lower bound for random 3-sat (extended abstract). In *STOC*, pages 28–37, 2000.

[2] Dimitris Achlioptas. Lower bounds for random 3-sat via differential equations. *Theor. Comput. Sci.*, 265(1-2):159–185, 2001.

[3] Mikhail Alekhnovich and Eli Ben-Sasson. Analysis of the random walk algorithm on random 3-CNFs, technical report ECCC TR04-016, 2002.

[4] Noga Alon and Joel Spencer. *The Probabilistic Method*. John Wiley, 1992.

[5] Sanjeev Arora and Shmuel Safra. Probabilistic checking of proofs: a new characterization of np. *J. ACM*, 45(1):70–122, 1998.

[6] Takao Asano, Takao Ono, and Tomio Hirata. Approximation algorithms for the maximum satisfiability problem. *Nordic J. of Computing*, 3(4):388–404, 1996.

[7] Armin Biere. A short history on sat solver technology and what is next? In *SAT*, 2007.

[8] 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*. IOS Press, February 2009.

[9] Armin Biere and Wolfgang Kunz. SAT and ATPG: Boolean engines for formal hardware verification. In Lawrence T. Pileggi and Andreas Kuehlmann, editors, *ICCAD*, pages 782–785. ACM, 2002.

[10] Maria Luisa Bonet and Katherine St. John. Efficiently calculating evolutionary tree measures using sat. In Kullmann [43], pages 4–17.

[11] Andrei Z. Broder, Alan M. Frieze, and Eli Upfal. On the satisfiability and maximum satisfiability of random 3-cnf formulas. In *SODA*, pages 322–330, 1993.

[12] Andrei A. Bulatov and Evgeny S. Skvortsov. Phase transition for local search on planted SAT. *CoRR*, abs/0811.2546, 2008.

[13] Edmund M. Clarke, Daniel Kroening, Natasha Sharygina, and Karen Yorav. SATABS: SAT-based predicate abstraction for ANSI-C. In Nicolas Halbwachs and Lenore D. Zuck, editors, *TACAS*, volume 3440 of *Lecture Notes in Computer Science*, pages 570–574. Springer, 2005.

[14] Stephen A. Cook. The complexity of theorem-proving procedures. In *STOC*, pages 151–158. ACM, 1971.

[15] James M. Crawford and Larry D. Auton. Experimental results on the crossover point in random 3-sat. *Artif. Intell.*, 81(1-2):31–57, 1996.

[16] Evgeny Dantsin, Andreas Goerdt, Edward A. Hirsch, Ravi Kannan, Jon M. Kleinberg, Christos H. Papadimitriou, Prabhakar Raghavan, and Uwe Schöning. A deterministic $(2-2/(k+1))^n$ algorithm for k-sat based on local search. *Theor. Comput. Sci.*, 289(1):69–83, 2002.

[17] Evgeny Dantsin, Edward A. Hirsch, and Alexander Wolpert. Clause shortening combined with pruning yields a new upper bound for deterministic SAT algorithms. In *CIAC*, pages 60–68, 2006.

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

[19] Olivier Dubois, Yacine Boufkhad, and Jacques Mandler. Typical random 3-SAT formulae and the satisfiability threshold. In *SODA*, pages 126–127, Philadelphia, PA, USA, 2000. Society for Industrial and Applied Mathematics.

[20] Uriel Feige, Elchanan Mossel, and Dan Vilenchik. Complete convergence of message passing algorithms for some satisfiability problems. In Josep Díaz, Klaus Jansen, José D. P. Rolim, and Uri Zwick, editors, *APPROX-RANDOM*, volume 4110 of *Lecture Notes in Computer Science*, pages 339–350. Springer, 2006.

[21] Abraham Flaxman. A spectral technique for random satisfiable 3cnf formulas. In *SODA*, pages 357–363, 2003.

[22] Ehud Freidgut. Necessary and sufficient conditions for sharp thresholds of graph properties and the k-problem. *Journal of AMS*, 12:1017–1054, 1999.

[23] Ian P. Gent. On the stupid algorithm for Satisfiability. *APES Report APES-03-1998. APES Research Group*, 1998.

[24] Ian P. Gent, South Bridge, and Toby Walsh. An empirical analysis of search in GSAT. *Journal of Artificial Intelligence Research*, 1:47–59, 1993.

[25] Ian P. Gent and Toby Walsh. The enigma of SAT hill-climbing procedures. Technical report, Department of AI, University of Edinburgh, 1992.

[26] Ian P. Gent and Toby Walsh. Towards an understanding of hill-climbing procedures for SAT. In *AAAI*, pages 28–33, 1993.

[27] Allen Goldberg. On the complexity of the satisfiability problem. *Fourth Workshop on Automated Deduction, Austin, TX*, pages 1–6, 1979.

[28] Pierre Hansen and Brigitte Jaumard. Algorithms for the maximum satisfiability problem. *Computing*, 44:279–303, 1990.

[29] Johan Håstad. Some optimal inapproximability results. *J. ACM*, 48(4):798–859, 2001.

[30] Edward A. Hirsch. Sat local search algorithms: Worst-case study. *J. Autom. Reasoning*, 24(1/2):127–143, 2000.

[31] John N. Hooker. Resolution vs. cutting plane solution of inference problems: Some computational experience. *Operations Research Letter, 7(1)*, pages 1–7, 1988.

[32] Holger Hoos and Thomas Sttzle. *Stochastic Local Search: Foundations & Applications*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004.

[33] Russell Impagliazzo and Ramamohan Paturi. On the complexity of k-sat. *J. Comput. Syst. Sci.*, 62(2):367–375, 2001.

[34] Daniel Johannsen, Igor Razgon, and Magnus Wahlström. Solving SAT for CNF formulas with a one-sided restriction on variable occurrences. In Kullmann [43], pages 80–85.

[35] David S. Johnson. Approximation algorithms for combinatorial problems. *J. Comput. Syst. Sci.*, 9(3):256–278, 1974.

[36] Anil Kamath, Rajeev Motwani, Paul Spirakis, and Krishna Palem. Tail bounds for occupancy and the satisfiability threshold conjecture. *Random Struct. Algorithms*, 7(1):59–80, 1995.

[37] Anil P. Kamath, Narendra Karmarkar, K. G. Ramakrishnan, and Mauricio G. C. Resende. Computational experience with an interior point algorithm on the satisfiability problem. In Ravi Kannan and William R. Pulleyblank, editors, *IPCO*, pages 333–349. University of Waterloo Press, 1990.

[38] 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.

[39] Lefteris M. Kirousis, Evangelos Kranakis, Danny Krizanc, and Yannis C. Stamatiou. Approximating the unsatisfiability threshold of random formulas. *Random Struct. Algorithms*, 12(3):253–269, 1998.

[40] Elias Koutsoupias and Christos H. Papadimitriou. On the greedy algorithm for satisfiability. *Inf. Process. Lett.*, 43(1):53–55, 1992.

[41] Michael Krivelevich and Dan Vilenchik. Solving random satisfiable 3CNF formulas in expected polynomial time. In *SODA*, pages 454–463. ACM Press, 2006.

[42] Alexander S. Kulikov and Konstantin Kutzkov. New bounds for MAX-SAT by clause learning. In *CSR*, pages 194–204, 2007.

[43] Oliver Kullmann, editor. *Theory and Applications of Satisfiability Testing - SAT 2009, 12th International Conference, SAT 2009, Swansea, UK, June 30 - July 3, 2009. Proceedings*, volume 5584 of *Lecture Notes in Computer Science*. Springer, 2009.

[44] Oliver Kullmann and Horst Luckhardt. Algorithms for SAT/TAUT decision based on various measures. Technical report, Information and Computation, 1999.

[45] Tracy Larrabee and Yumi Tsuji. Evidence for a satisfiability threshold for random 3CNF formulas. *Proceedings AAAI Symposium on AI and NP-hard problems*, 1993.

[46] Monaldo Mastrolilli and Luca Maria Gambardella. Maximum satisfiability: How good are tabu search and plateau moves in the worst-case? *European Journal of Operational Research*, 166(1):63–76, October 2005.

[47] David G. Mitchell, Faraz Hach, and Raheleh Mohebali. Faster phylogenetic inference with mxg. In Nachum Dershowitz and Andrei Voronkov, editors, *LPAR*, volume 4790 of *Lecture Notes in Computer Science*, pages 423–437. Springer, 2007.

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

[49] Raghavan P. Motwani R. *Randomized Algorithms*. Cambridge University Press, 1995.

[50] Christos H. Papadimitriou. On selecting a satisfying truth assignment (extended abstract). In *FOCS*, pages 163–169. IEEE, 1991.

[51] Mukul R. Prasad, Armin Biere, and Aarti Gupta. A survey of recent advances in SAT-based formal verification. *STTT*, 7(2):156–173, 2005.

[52] Jussi Rintanen and Jörg Hoffmann. An overview of recent algorithms for AI planning. *KI*, 15(2):5–11, 2001.

[53] Bart Selman, Henry Kautz, and Bram Cohen. Local search strategies for satisfiability testing. In *DIMACS Series in Discrete Mathematics and Theoretical Computer Science*, pages 521–532, 1995.

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

[55] Bart Selman, David Mitchell, and Hector Levesque. Generating hard satisfiability problems. *Artificial Intelligence*, 81:17–29, 1996.

[56] Min te Chao and John Franco. Probabilistic analysis of a generalization of the unit-clause literal selection heuristics for the k-satisfiability problem. *Information Science*, 51:289–314, 1990.

[57] Ming te Chao and John Franco. Probabilistic analysis of two heuristics for the 3-satisfiability problem. *SIAM Journal on Computing*, 15:1106–1118, 1986.

[58] Dan Vilenchik Uriel Feige. A local search algorithm for 3SAT. *Technical report MCS04-07 of the Weizmann Institute*, 2004.

[59] Van H. Vu. A general upper bound on the list chromatic number of locally sparse graphs. *Comb. Probab. Comput.*, 11(1):103–111, 2002.

[60] Toby Walsh. The constrainedness knife-edge. In *AAAI/IAAI*, pages 406–411, 1998.

[61] Nick Wormald. Differential equations for random processes and random graphs. *The Annals of Applied Probability*, 5(4):1217–1235, 1995.

[62] Mihalis Yannakakis. On the approximation of maximum satisfiability. In *SODA*, pages 1–9, Philadelphia, PA, USA, 1992. Society for Industrial and Applied Mathematics.