Efficiency of Local Search

Andrei A. Bulatov[1] and Evgeny S. Skvortsov[1]

School of Computing Science, Simon Fraser University, Burnaby, Canada
{abulatov,evgenys}@cs.sfu.ca

Abstract. The Local Search algorithm is one of the simplest heuristic algothms for solving the MAX-SAT problem. The goal of this paper is to estimate the relative error produced by this algorithm being applied to random 3-CNFs with fixed density r. We prove that, for any r, there is a constant c such that a weakened version of Local Search that we call One-Pass Local Search almost surely outputs an assignment containing $cn + o(n)$ unsatisfied clauses. Then using certain assumtions we also show this for Local Search. Although the assumptions remain unproved the results well matches experiments.

1 Introduction

In the Local Search (LS) algorithm we start with a random assignment to a CNf, and then on each step we choose at random a variable such that flipping this variable increases the number of satisfied clauses, or stop if such a variable does not exist. LS is one of the oldest algorithms for solving the SAT and MAX-SAT problems. Numerous variations of this method have been proposed starting from the late eighties, see for example [4, 8].

It is well known that the worst-case performance of 'pure' LS is not very good: the only known lower bound for local optima of a k-CNF is $\frac{k}{k+1}m$, m the number of clauses [5]. Many other approximation algorithms, for example Goemans and Williamson's [3], Karloff and Zwick's [6] algorithms and their improvements (see for instance [2]), guarantee much better results than LS. However, it is also an empirical fact that the *expected* performance of LS is much better than $\frac{k}{k+1}m$ bound. This is also supported by the result by Koutsoupias and Papadimitriou [7] that for almost all satisfiable 3-CNFs LS almost surely finds a satisfying assignment.

The goal of this paper is to study the expected performance of LS on random 3-CNFs of the form $\Phi(n, \varrho n)$, ϱ fixed, where n is the number of variables and ϱn is the number of clauses. We consider also a weakened version of LS, the One-Pass LS (OLS) algorithm, in which every variable is visited only once in a certain order, and if flipping the variable increases the number of satisfied clauses then it flipped, othewise it is never considered again. For OLS we build a model similar to the 'card-game' from [1] and then use Wormald's theorem [9] to show that for any ϱ there is a constant c such that almost surely OLS finds an assignment satisfying $cn + o(1)$ clauses. Then we extend this framework to LS and obtain the same result under a certain assumption that remains unproved.

INPUT: a random 3-CNF Φ

OUTPUT: the number of unsatisfied assignments in the resulting formula Φ'

Step 1 **compute** the set W of variables X such that $A(X) > B(X)$
Step 2 **while** $W \neq \emptyset$ **do**
Step 2.1 **choose** a variable X from W uniformly at random
Step 2.2 **for every** clause $C \in \Phi$ containing X or $\neg X$ **replace** X with $\neg X$ and $\neg X$ with X
Step 2.3 **compute** the set W of variables X such that $A(X) > B(X)$
endwhile
Step 3 **output** the number of clauses in Φ with 3 negative literals

Fig. 1. The Local Search algorithm

2 Preliminaries

2.1 MAX-SAT and Local Search

First we give formal definitions of the main objects involved.

In the MAX-SAT problem we are given a CNF Φ. The goal is to find an assignment of the variables of Φ that satisfies maximum number of clauses. The MAX-SAT problem restricted to k-CNFs is called the MAX-k-SAT problem.

The model of random CNF we use is the random 3-CNFs with fixed density. Thus $\Phi(n, m)$ denotes a random CNF with m 3-clauses over n Boolean variables, where clauses are chosen uniformly and independently among $8\binom{n}{3}$ clauses without repetitions of variables. Repetitions of clauses are allowed.

We consider the following version the of LS. Given a CNF, choose a random assignment π and repeat the following steps while this is possible: Find the set of variables W such that flipping a variable X from W, that is setting $\pi'(X) = \neg\pi(X)$ keeping all other values, decreases the number of unsatisfied constraints; choose a variable X from W uniformly at random; flip X.

The standard setting of the LS algorithm involves on the first step two random objects — a formula and an assignment. It is easy to see that for the purpose of studying performance one of them can be eliminated. We are going to fix a assignment, it will be $\pi(X) = 1$ always and change the formula. By $A_\Phi(X), B_\Phi(X)$ we denote the number of unsatisfied clauses in Φ containing $\neg X$ (this means that all literals in such clases are negative), and the number of clauses in Φ containing X, and therefore satisfied, such that the two other literals are negative, respectively. As Φ is always clear from the context, we shall omit it. A formal description of LS is given in Fig. 1.

Along with the standard LS algorithm we will study its weakened version that we call the One-Pass Local Search algorithm (OLS), see Fig. 2.

Obviously, instead of choosing variables at random on every step of the OLS algorithm, one can choose a predefined order on the variable set in the very beginning.

INPUT: a random 3-CNF Φ with n variables

OUTPUT: the number of unsatisfied assignments in the resulting formula Φ'

Step 1 **for** $i = 1$ **to** n **do**
Step 1.1 **choose** a variable X from the set of not visited variables uniformly at random
Step 1.2 **if** $A(X) > B(X)$ **then do**
Step 1.2.1 **for every** clause $C \in \Phi$ containing X or $\neg X$ **replace** X with $\neg X$ and $\neg X$ with X
endif
endfor
Step 2 **output** the number of clauses in Φ with 3 negative literals

Fig. 2. The One-Pass Local Search algorithm

2.2 Wormald's Theorem

The key stage in our analysis is the theorem by Wormald [9] that allows one to replace probabilistic analysis of combinatorial algorithm with analysis of a deterministic system of differential equations.

All random processes 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 dependance of 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 $\max\{x|\mathbf{P}(X = x) \neq 0\} = o(f(n))$. An event occurs *almost surely* (a.s.) if its probability in Ω_n is $1 - o(1)$. We denote by S^+ the set of all $h_t = (q_0, \ldots, q_t)$, each $q_t \in S$ for $t = 0, 1 \ldots$. By H_t we denote the *history* of the processes, that is the $n \times (t+1)$-matrix with entries $Q_i(j)$, $0 \leq i \leq t, 1 \leq j \leq n$.

A function $f(u_1, \ldots, u_j)$ satisfies a *Lipschitz condition* on $D \subseteq \mathbb{R}^j$ if a constant $L > 0$ exists with the property that

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

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

Theorem 1 (Wormald, [9]). *Let k be fixed. For $1 \leq \ell \leq k$, let $y^{(\ell)}: S^+ \to \mathbb{R}$ and $f_\ell: \mathbb{R}^{k+1} \to \mathbb{R}$, 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 i and uniformly over all $t < m$, $\mathbf{P}(|Y_{t+1}^{(\ell)} - Y_t^{(\ell)}| > n^{1/5} \mid H_t) = o(n^{-3})$ always;

(ii) for all ℓ and uniformly over all $t < m$,
$\mathbf{E}(Y_{t+1}^{(\ell)} - Y_t^{(\ell)} \mid H_t) = f_\ell(t/n, Y_t^{(1)}/n, \ldots, y_t^{(k)}/n) + o(1)$ *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 $\{(t, z^{(1)}, \ldots, z^{(k)}) \mid t \geq 0\}$ with some neighbrhood of $\{(0, z^{(1)}, \ldots, z^{(k)}) \mid \mathbf{P}(Y_0^{(\ell)} = z^{(\ell)}n, 1 \leq \ell \leq k) \neq 0$ for some $n\}$.

Then:

(a) For $(0, \hat{z}^{(1)}, \ldots, \hat{z}^{(k)}) \in D$ *the system of differential equations* $\dfrac{dz_\ell}{ds} = f_\ell(s, z_1, \ldots, z_k)$, $\ell = 1, \ldots, k$, *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) Almost surely $Y_t^{(\ell)} = n z_\ell(t/n) + o(n)$ *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.*

3 One Pass Local Search

3.1 Model

To analyze the OLS algorithm we use a model, which is somewhat a fusion of the 'card-game' framework from [1] and the greedy algorithm for finding an independent set from [9]. Note that the OLS algorithm uses much more information about the formula than any of the 'card-game' algorithms. This is why the resulting system of differential equations is much more sophisticated.

We assume that a given random 3-CNF Φ has a fixed density ϱ, that is has the form $\Phi = \Phi(n, \varrho n)$, and that it comes with a predefined order of variables, that is the variables of Φ are enumerated $X_1, X_2, \ldots$. Every variable and its negation are considered as baskets that contain points corresponding to occurences of the literal in Φ. The number of points in each basket will be defined on the first step of the algorithm.

Let $D_{\varrho,n}$ be the probability distribution on the set of $2n$-tuples $(a_1, b_1, \ldots, a_n, b_n)$ such that the probability of tuple $(a_1, \ldots, b_n)$ that sums up to $3\varrho n$ equals $\frac{G(a_1,\ldots,b_n)}{G_n}$ where G_n is the number of all 3-CNFs of density ϱ with n variables and $G(a_1, \ldots, b_n)$ is the number of 3-CNFs such that X_i $[\neg X_i]$ occurs in a_i $[b_i]$ clauses. These numbers can easily be computed. The algorithm in Fig. 3 generates a random 3-CNF and at the same time simulates OLS.

Note that this algorithm sometimes generates a formula containing clauses with repeated literals or a literal and its negation. However, as the following lemma shows, the probability of this event tends to a constant as n grows. Therefore, since we prove that some event happens with probability $1 - o(1)$ on the set of all formulas, it is also true on the set of formulas with the restrictions specified.

Lemma 1. *The probability that the algorithm in Fig.3 generates a formula with repeated clauses, or with a clause with repeated literals, or with a clause with a variable and its negation tends to* $1 - e^{-3\varrho}$.

3.2 Analisys

We need some notation. Let Φ be a 3-CNF and X a variable in Φ. By $A_{i\alpha}(X)$, where $i \in \{0, 1, 2\}$ and $\alpha \in \{-, +\}$, we denote the set of clauses C such that

INPUT: A number n

OUTPUT: The number of unsatisfied clauses in a random 3-CNF Φ after applying the OLS algorithm

Step 1 **choose** a sequence of non-negative integers
$K(X_1), K(\neg X_1), K(X_2), \ldots$ according to distribution $D_{\varrho,n}$
Step 2 **for** $i = 1$ **to** n **set** $K(X_i, 0) = K(\neg X_i, 0) = 0$
Step 3 **for** $t = 1$ **to** n **do**
Step 3.1 **for each** point left in the basket X_t **do**
Step 3.1.1 **choose** uniformly at random a pair l_1, l_2 of unused points
Step 3.1.2 **if** $l_1 \in L_i, l_2 \in L_j$ where $L_i \in \{X_i, \neg X_i\}$ and $L_j \in \{X_j, \neg X_j\}$, **then create** the clause (X_t, L_i, L_j) in Φ
endfor
Step 3.2 **for each** point left in the basket $\neg X_t$ **do**
Step 3.2.1 **choose** uniformly at random a pair l_1, l_2 of unused points
Step 3.2.2 **if** $l_1 \in L_i, l_2 \in L_j$ where $L_i \in \{X_i, \neg X_i\}$ and $L_j \in \{X_j, \neg X_j\}$, **then create** the clause $(\neg X_t, L_i, L_j)$ in Φ
endfor
Step 3.3 **if** $A(X_t) > B(X_t)$ **then for every** clause $C \in \Phi$ containing X_t or $\neg X_t$ **replace** X_t with $\neg X_t$ and $\neg X_t$ with X_t
endfor
Step 4 **output** the number of unsatisfied clauses in Φ

Fig. 3. Combined generation + OLS algorithm

$X \in C$ if $\alpha = +$, $\neg X \in C$ if $\alpha = -$, and among the other two literals there are exactly i positive. If $C \in A_{i\alpha}$ we also say that C has *type* $i\alpha$ for X. Sometimes we need to indicate that $A_{i\alpha}$ is a set of clauses existing before tth iteration of the algorithm. In this case we use $A_{i\alpha}(X,t)$. Also $a_{i\alpha}(X), a_{i\alpha}(X,t)$ denote the size of $A_{i\alpha}(X), A_{i\alpha}(X,t)$. By $E_{\overline{a}}$ $[E_{\overline{a}}(t)]$, $\overline{a} = (a_{0-}, a_{0+}, a_{1-}, a_{1+}, a_{2-}, a_{2+})$ we denote the set of all variables X of Φ such that $a_{i\alpha}(X) = a_{i\alpha}$ $[a_{i\alpha}(X,t) = a_{i\alpha}]$ for all i and α. If $X \in E_{\overline{a}}$ then variable X is said to have type $\overline{a}$. Moreover, we shall distinguish between those variables, which already visited by the algorithm and those which are not. Let us set $E'_{\overline{a}}(t) = E_{\overline{a}}(t) \cap \{X_1, \ldots, X_{t-1}\}$ and $E''_{\overline{a}}(t) = E_{\overline{a}}(t) \setminus E'_{\overline{a}}(t)$. Finally, $e'_{\overline{a}}(t), e''_{\overline{a}}(t)$ denote the size of sets $E'_{\overline{a}}(t), E''_{\overline{a}}(t)$ respectively. We would also like to be able to derive from the history if variable X_t is flipped on tth iteration. To this end we also use variable $d(t)$ that equals the number of variables flipped by iteration t. By $\mathsf{typ}(X_t)$ we shall denote the tuple $(a_{0-}(X_t), a_{0+}(X_t), a_{1-}(X_t), a_{1+}(X_t), a_{2-}(X_t), a_{2+}(X_t))$ on iteration t *before* possible flipping.

The numbers $e'_{\overline{a}}(t), e''_{\overline{a}}(t), d(t)$ will be our random variables from Wormald's theorem. A problem with these random variables is that as n grows there are more and more of them, since the entries of $\overline{a}$ are not bounded. We overcome this problem by considering variables $e'_{\overline{a}}, e''_{\overline{a}}$ for only those $\overline{a}$ the sum of the components of which is bounded with a constant M. Thus we consider only those variables that have bounded degree. Clearly, this does affect the analysis, but in a certain controllable way, as we shall see.

Now we check conditions (i)–(iii) of Theorem 1.

(i) As is easily seen $|e'_{\overline{a}}(t+1) - e'_{\overline{a}}(t)| \le 2(K(X_t) + K(\neg X_t))$, $|e''_{\overline{a}}(t+1) - e''_{\overline{a}}(t)| \le 2(K(X_t) + K(\neg X_t))$ and $|d(t+1) - d(t)| \le 1$. Therefore, it is sufficient to show that $\mathbf{P}(K(X_t) + K(\neg X_t) > n^{1/5}) = o(n^{-3})$. To this end we note that

$$\mathbf{P}(K(X) = k) = \mathbf{P}(K(\neg X) = k) = \binom{\varrho n}{k}\left(\frac{1}{2n}\right)^k \left(1 - \frac{1}{2n}\right)^{\varrho n - k}, \qquad (1)$$

and

$$\mathbf{P}(K(X) > n^{1/5}) = \mathbf{P}(K(\neg X) > n^{1/5}) \le \binom{\varrho n}{n^{1/5}}\left(\frac{1}{2n}\right)^{n^{1/5}}.$$

Using Stirling's formula the number in the right side can be asymptotically estimated as

$$\sqrt{\frac{\varrho n}{2\pi(\varrho n - n^{1/5})n^{1/5}}} \cdot \left(\frac{\varrho}{e}\right)^{n^{1/5}} \cdot (2n^{1/5})^{-n^{1/5}} = o(n^{-3}).$$

(ii) We compute the expectation of differences of the random values on iteration t. First of all, we need some auxiliary probabilities and expectations. All of them are conditional to the history H_t. Notice that H_t does not tell us the sequence $\{K(X_i), K(\neg X_i)\}$. Instead it suggests that (a) the number S of undefined clauses is known (it is not hard to see that $S = \varrho n - 1/3\sum_{\overline{a}}(e'_{\overline{a}}(t) + e''_{\overline{a}}(t)))$; and (b) these clauses eventually constitute a random 3-CNF on $n-t$ variables with density $\frac{S}{n-t}$.

- $\mathbf{P}(X_t \in E''_{\overline{a}}(t))$. Since on the previous iterations clauses were chosen independently, we have

$$\mathbf{P}(X_t \in E''_{\overline{a}}(t)) = \frac{e''_{\overline{a}}(t)}{n-t}, \qquad (2)$$

for any tuple $\overline{a}$.

- $\mathbf{P}(C \in A_{i\alpha})$. From observation (b) we get that when choosing a random clause C containing X or $\neg X$ on iteration t

$$\mathbf{P}(C \in A_{0-}) = \mathbf{P}(C \in A_{0+}) = \mathbf{P}(C \in A_{2-}) = \mathbf{P}(C \in A_{2+}) = \frac{1}{8},$$
$$\mathbf{P}(C \in A_{1-}) = \mathbf{P}(C \in A_{1+}) = \frac{1}{4}.$$

Clearly, if we are choosing several clauses like this then the probabilities for consequent clauses changes by a summand of $O\left(\frac{1}{n}\right)$.

- $\mathbf{E}((K(X_t) - K(X_t, t)) + (K(\neg X_t) - K(\neg X_t, t)))$. As is easily seen, this number is the expectation of the number of successes in a sequence of independent trials — throwing S 'empty' clauses to the set of $8(n-t)^3$ triples of variables and their negations. The probability of success — the chosen triple contain X_t or $\neg X_t$ — equals

$$\frac{4(n-t)^2 + 2(n-t)(2n-2t-1) + (2n-2t-1)^2}{8n^3} = \frac{3}{2(n-t)} + O\left(\frac{1}{n^2}\right).$$

Thus, $\mathbf{E}((K(X_t) - K(X_t, t)) + (K(\neg X_t) - K(\neg X_t, t))) = \frac{3S}{2(n-t)}$.

- $\mathbf{P}(\mathsf{typ}(X_t)=\overline{a})$. Next we compute the probability that variable X_t after completing Steps 3.1 and 3.2 has type $\overline{a}$. Clearly,

$$\mathbf{P}(\mathsf{typ}(X_t)=\overline{a})=\sum_{\overline{b}}\mathbf{P}(X_t\in E''_{\overline{b}}(t))\mathbf{P}(\mathsf{typ}'(X_t)=\overline{a}-\overline{b})=\sum_{\overline{b}}\frac{e''_{\overline{a}}(t)}{n-t}\mathbf{P}(\mathsf{typ}'(X_t)=\overline{a}-\overline{b}),$$

where $\overline{a}-\overline{b}$ denotes the tuple with components $a_{i\alpha}-b_{i\alpha}$ and $\mathbf{P}(\mathsf{typ}'(X_t)=\overline{c})$ denotes the probability that if we consider only newly generated clauses containing X_t then its type is $\overline{b}$. The latter probability can be computed for a random 3-CNF with S clauses and $n-t$ variables. Let $\overline{c}^{\Sigma}$ denotes the sum of components of $\overline{c}$.

$$\begin{aligned}\mathbf{P}(\mathsf{typ}'(X_t)=\overline{c})&=\binom{S}{c_{0-},\dots,c_{2+}}\left(\frac{3/8}{n-t}\right)^{\overline{c}^{\Sigma}+c_{1-}+c_{1+}}\left(1-\frac{3/8}{n-t}\right)^{S-\overline{c}^{\Sigma}}+O\left(\frac{1}{n^2}\right)\\&=\frac{S^{\overline{c}^{\Sigma}}}{c_{0-}!c_{0+}!\dots c_{2+}!}\left(\frac{3/8n}{n-t}\right)^{\overline{c}^{\Sigma}}e^{\frac{3(S-\overline{c}^{\Sigma})}{8(n-t)}}+O\left(\frac{1}{n^2}\right).\end{aligned}$$

- The probability that X_t is flipped on Step 3.3 equals

$$\mathbf{P}(X_t\ \text{flipped})=\sum_{a_{0-}>a_{0+}}\mathbf{P}(\mathsf{typ}(X_t)=\overline{a}).$$

- $\mathbf{E}(\hat{a}_{i\alpha}(X_t,t+1))$. Recall that $a_{i\alpha}(t+1)$ denotes the number of clauses of type $i\alpha$ after tth iteration that includes X_t (or $\neg X_t$). Here we need a similar number, but computed not after tth iteration, but just before a possible flip on tth iteration. Let us denote this number $\hat{a}_{i\alpha}(X_t,t+1)$. Then $\mathbf{E}(\hat{a}_{i\alpha}(X_t,t+1))=\sum_{\overline{c}}\mathbf{P}(\mathsf{typ}(X_t)=\overline{c})\cdot c_{i\alpha}$.

- $\mathbf{E}(e'_{\overline{a}}(t+1)-e'_{\overline{a}}(t))$. Now we are able to express the expectation of differences for $e'_{\overline{a}}$. Let $a'_{i\alpha}(t)=\sum_{\overline{a}}a_{i\alpha}\cdot e'_{\overline{a}}(t)$ be the number of all clauses of type $i\alpha$ containing $X_s,\neg X_s$, $s<t$, on tth iteration. Thus picking at random a clause of type $i\alpha$ and containing a variable from $\bigcup_{\overline{b}}E'_{\overline{b}}(t)=\{X_1,\dots,X_{t-1}\}$ the probability that this clause contains a variable from $E'_{\overline{a}}(t)$ equals $2\frac{a_{i\alpha}e'_{\overline{a}}(t)}{a'_{i\alpha}(t)}$.

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

The expectations below are computed under condition of history H_t. The first summand corresponds to the situation when X_t is not flipped. Then $e'_{\overline{a}}$ can change only if X_t has type $\overline{a}$. Otherwise this value changes in three ways. Firstly, there is a chance that X_t itself goes to $E'_{\overline{a}}$, which gives us the second summand, where $\overline{a}'$ denotes the tuple such that $a'_{i-}=a_{i+}$ and $a'_{i+}=a_{i-}$. Then X_t can appear as a member of some clauses containing variables X_j from $E'_{\overline{b}}(t)$ for some $\overline{b}$. Every such appearance can, after flipping, move X_j from $E'_{\overline{b}}$ to $E'_{\overline{a}}$ if $\overline{b}$ adjacent to $\overline{a}$, or from $E'_{\overline{a}}$ to $E'_{\overline{b}}$ if $\overline{a}$ adjacent to $\overline{b}$. Suppose that by flipping X_t some variable X is moved from $E'_{\overline{b}}$ to $E'_{\overline{a}}$, that is $\overline{a},\overline{b}$ are adjacent, and i,j,α

are the corresponding indices. We use $\hat{i}, \hat{\alpha}$ to denote the type of a clause that is changed. Clearly, for X this clause must be of type $j\alpha$, but for X_t it can have another type. Thus, $\hat{i} = j$ 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 $j > i$. The summation below is over all $\overline{b}$ adjacent to $\overline{a}$ (or all $\overline{b}$ such that $\overline{a}$ is adjacent to $\overline{b}$) and i, j, α denote the indices involved.

Finally, as we already noted, using history H_t and formula Φ' built by tth iteration one can recover the formula Φ'' that would be obtained if we omitted Step 3.3 in the algorithm. However, Φ'' is just a part of a random 3-CNF consisting of all clauses containing a variable from a certain (random) set of size t. Therefore all clauses there are chosen independently and the required expectations can be computed as follows.

$$\mathbf{E}(e'_{\overline{a}}(t+1) - e'_{\overline{a}}(t)) = (1 - \mathbf{P}(X_t \text{ flipped})) \cdot \left\{ \begin{array}{ll} \mathbf{P}(\mathsf{typ}(X_t) = \overline{a}), & \text{if } a_{0-} \le a_{0+}, \\ 0, & \text{if } a_{0-} > a_{0+} \end{array} \right\}$$
$$+ \mathbf{P}(X_t \text{ flipped}) \cdot \left(\left\{ \begin{array}{ll} \mathbf{P}(\mathsf{typ}(X_t) = \overline{a}'), & \text{if } a_{0-} \le a_{0+}, \\ 0, & \text{if } a_{0-} > a_{0+} \end{array} \right\} \right.$$
$$\left. + 2 \sum_{\substack{\overline{a}, \overline{b} \text{ are adjacent} \\ i,j,\alpha}} \left(\frac{b_{j\alpha} e'_{\overline{b}}(t) - a_{j\alpha} e'_{\overline{a}}(t)}{a'_{j\alpha}(t)} \mathbf{E}(\hat{a}_{\hat{i}\hat{\alpha}}(X_t, t+1)) \right) \right) + o(1).$$

- $\mathbf{E}(e''_{\overline{a}}(t+1) - e''_{\overline{a}}(t))$. Now we compute expectations for $e''_{\overline{a}}$. Observe that this value can change in three ways. Firstly, X_t may go from $E''_{\overline{a}}$. Secondly, newly generated clauses can increase the number of clauses in which a variable X_i, $i > t$, is involved. Thirdly, if X_t is flipped then the values $e''_{\overline{a}}$ can be rearranged similarly to $e'_{\overline{a}}$. Changes of the first types are easy to compute. The third type changes are computed in the same way as for $e'_{\overline{a}}$. For the second type changes we need some notation. Let $a''_{i\alpha}(t) = \sum_{\overline{a}} a_{i\alpha} \cdot e''_{\overline{a}}(t)$ be the number of all clauses of type $i\alpha$ containing $X_s, \neg X_s$, $s > t$, on tth iteration. If a clause $C = (L_t, L_i, L_j)$, where L_t, L_i, L_j are literals containing X_t, X_i, X_j respectively, is generated on tth iteration and $C \in A_{i\alpha}$ for X_i, then for X_t it belongs to $A_{i'\alpha'}$ or $A_{i''\alpha''}$ where $i'\alpha'$, $i''\alpha''$ depend on $i\alpha$ as the following table specifies.

$i\alpha$	$0-$	$0+$	$1-$	$1+$	$2-$	$2+$
$i'\alpha'$	$0-$	$1-$	$0+$	$1+$	$1+$	$2+$
$i''\alpha''$	$0-$	$1-$	$1-$	$2-$	$1+$	$2+$

For a tuple $\overline{a}$, let $\overline{a}^{-i\alpha}$ denote the tuple all components of which are equal to $\overline{a}$ except that the component $i\alpha$ equals $a_{i\alpha} - 1$, similarly, the $i\alpha$ component of $\overline{a}^{+i\alpha}$ equals $a_{i\alpha} + 1$. Finally $-\alpha$ denotes $+$ if $\alpha = -$ and $-$ if $\alpha = +$. The probability that for a new clause C variable X_i or X_j belong to $E''_{\overline{a}^{-i\alpha}}$ can be computed as $2\frac{e''_{\overline{a}^{-i\alpha}}(t)}{n-t}$. Therefore

$$\begin{aligned} &\mathbf{E}(e''_{\overline{a}}(t+1) - e''_{\overline{a}}(t)) = -\mathbf{P}(X_t \in E''_{\overline{a}}(t)) \\ &+ (1 - \mathbf{P}(X_t \text{ flipped}))\mathbf{E}(K(X_t) + K(\neg X_t) - K(X_t, t) - K(\neg X_t, t)) \\ &\times 2 \sum_{i,\alpha} \frac{e''_{\overline{a}^{-i\alpha}}(t)}{n-t} (\mathbf{P}(C \in A_{i'\alpha'}) + \mathbf{P}(C \in A_{i''\alpha''})) \end{aligned}$$

$$+ \mathbf{P}(X_t \text{ flipped}) \left(2 \sum_{i,\alpha} \frac{e''_{\overline{a}-i\alpha}(t)}{n-t} (\mathbf{P}(C \in A_{i'-\alpha'}) + \mathbf{P}(C \in A_{i''-\alpha''})) \right.$$

$$\left. + 2 \sum_{\substack{\overline{b} \text{ adjacent to } \overline{a} \\ i,j,\alpha}} \left(\frac{b_{j\alpha} e''_{\overline{b}}(t) - a_{j\alpha} e''_{\overline{a}}(t)}{a_{j\alpha}(t)} \mathbf{E}(\hat{a}_{\hat{i}\hat{\alpha}}(X_t, t+1)) \right) \right) + o(1).$$

- For $d_{\overline{a}(t)}$ we obviously have $\mathbf{E}(d(t+1) - d(t)) = \mathbf{P}(X_t \text{ flipped})$.

Now we are in a position to write up the differential equations. Let $z'_{\overline{a}} = \frac{e'_{\overline{a}}}{n}, z''_{\overline{a}} = \frac{e''_{\overline{a}}}{n}, p = \frac{d}{n}$, and $s = \frac{t}{n}$. Recall that we only consider those tuples $\overline{a}$ for which $\overline{a}^{\Sigma} \le M$. Then

$$\frac{dz'_{\overline{a}}}{ds} = (1-u)v_{\overline{a}} + u \left(v'_{\overline{a}} + \sum_{\substack{\overline{b} \text{ adjacent to } \overline{a} \\ i,j,\alpha}} \left(\frac{b_{j\alpha} z'_{\overline{b}} - a_{j\alpha} z'_{\overline{a}}}{g'_{j\alpha}} h_{\hat{i}\hat{\alpha}} \right) \right),$$

$$\frac{dz''_{\overline{a}}}{ds} = -\frac{z''_{\overline{a}}}{1-s} + 2(1-u) \frac{3w}{2(1-s)} \sum_{i,\alpha} \frac{z''_{\overline{a}-i\alpha}}{1-s} (P_{i'\alpha'} + P_{i''\alpha''})$$

$$+ 2u \left(2 \sum_{i,\alpha} \frac{z''_{\overline{a}-i\alpha}}{1-s} (P_{i'-\alpha'} + P_{i''-\alpha''}) + \sum_{\substack{\overline{b} \text{ adjacent to } \overline{a} \\ i,j,\alpha}} \left(\frac{b_{j\alpha} z''_{\overline{b}} - a_{j\alpha} z''_{\overline{a}}}{g''_{j\alpha}} h_{\hat{i}\hat{\alpha}} \right) \right),$$

$$\frac{dp}{ds} = u$$

where

$$v_{\overline{a}}(s) = \mathbf{P}(\mathsf{typ}(X_t) = \overline{a}) = \sum_{\overline{b}} \frac{z''_{\overline{a}}}{1-s} \frac{w^{(\overline{a}-\overline{b})^{\Sigma}}}{(a_{0-} - b_{0-})! \dots (a_{2+} - b_{2+})!} \left(\frac{3/8}{1-s} \right)^{(\overline{a}-\overline{b})^{\Sigma}}$$

$$\times e^{\frac{3(w-(\overline{a}-\overline{b})^{\Sigma})}{8(1-s)}} \text{ if } a_{0-} \le a_{0+}, \text{ and 0 otherwise,}$$

$$v'_{\overline{a}}(s) = \mathbf{P}(\mathsf{typ}(X_t) = \overline{a}') \text{ if } a_{0-} \le a_{0+}, \quad w(s) = \frac{S}{n} = \varrho - \frac{1}{3} \sum_{\overline{a}} (z'_{\overline{a}} + z''_{\overline{a}}),$$

$$g'_{i\alpha}(s) = \sum_{\overline{a}} a_{i\alpha} z'_{\overline{a}}, \quad g''_{i\alpha}(s) = \sum_{\overline{a}} a_{i\alpha} z''_{\overline{a}}, \quad h_{i\alpha}(s) = \sum_{\overline{a}} v_{\overline{a}} a_{i,\alpha}.$$

(iii) The functions constructed above have one substantial deficiency: they are not defined at $(0, 0, \dots, 0)$ and cannot be defined there without destroying continuity. To avoid this problem we replace every function $z'_{\overline{a}}, z''_{\overline{a}}, p$ occuring there with their absolute values. Then the resulting function can be defined at $(0, 0, \dots, 0)$ to be 0. Another problem is that these functions are not defined when $s = 1$. However, this can be overcome using a standard trick: for each $\epsilon > 0$ and define set D such that it includes only points with $s < 1 - \epsilon$. Then we find the required value as the limit when $\epsilon \to 0$.

Theorem 2. *For any positive ϱ and there is a constant c such that for a random 3-CNF $\Phi(n, \varrho n)$ a.s. the OSL algorithm finds and assingment such that the number of satisfied clauses equals $cn + o(n)$.*

Proof. Applying Wormald's theorem we get that, for any positive ϱ and any M there is a constant c' such that for a random 3-CNF $\Phi(n, \varrho n)$ a.s. the OSL algorithm finds and assingment such that the number of satisfied clauses equals $c'n + o(n)$ 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. Using (1) we get the number of such clauses is no more than

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

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

4 Local Search

4.1 Model

In this section we try to extend the approach from Section 3 to analysis of Local Search. Although we were unable to carry out a complete rigorous analisys, it turns out that such an analysis boils down to a certain simple assumption (see below Assumption 1). This assumption looks very plausible, but we could not neither prove nor disprove it. Experiments shows 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.

As in the previous section we use sets $E_{\overline{a}}$ and values $e_{\overline{a}}$ as the random variables, but this time we do not distinguish between visited and unvisited variables, and hence no extra random variables are needed. Thus, given a formula Φ, $E_{\overline{a}}$ denotes the set of variables of type $\overline{a}$ and $e_{\overline{a}}$ denotes the number of such variables. Before checking conditions (i)–(iii) of Theorem 1 we make a simple observation.

Lemma 2. *If Φ is a random 3-CNF of density ϱ with n variables, then for a variable X*

$$\mathbf{P}(a_{i\alpha}(X) = a) = \frac{\left(\frac{3\varrho}{8}\right)^a e^{3\varrho/8}}{a!} + o(1) \quad \text{if } i = 0, 2,$$

$$\mathbf{P}(a_{i\alpha}(X) = a) = \frac{\left(\frac{3\varrho}{4}\right)^a e^{3\varrho/4}}{a!} + o(1) \quad \text{if } i = 1,$$

$$\mathbf{P}(X \in E_{\overline{a}}) = \prod_{i,\alpha} \mathbf{P}(a_{i\alpha}(X) = a), \quad \mathbf{E}(e_{\overline{a}}) = n \cdot \mathbf{P}(X \in E_{\overline{a}}).$$

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

Lemma 3 provides the initial values for equations from Theorem 1. Now we are verifying conditions (i)–(iii).

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

(ii) We use the notation from the previous section. Suppose that on the current step t of LS the variable to flip is X. Since X 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 $\mathbf{P}(X \in E_{\overline{a}}(t) \mid a_{0-} > a_{0+}) = \frac{e_{\overline{a}}(t)}{b(t)}$, where $b(t) = |B(t)|$. Recall that $a_{i\alpha}(X,t)$ denotes the number of clauses of type $i\alpha$ on step t involving X. We have

$$\mathbf{E}(a_{i\alpha}(X,t) \mid a_{0-}(X,t) > a_{0+}(X,t)) = \sum_{\substack{\overline{a} \\ a_{0-} > a_{0+}}} a_{i\alpha} \cdot \mathbf{P}(X \in E_{\overline{a}}(t) \mid a_{0-}(X,t) > a_{0+}(X,t)).$$

Let $a_{i\alpha}(t) = \sum_{\overline{a}} a_{i\alpha} \cdot e_{\overline{a}}(t)$ be the number of all clauses of type $i\alpha$ containing $Y, \neg Y$ with $Y \neq X$. Thus picking at random a clause of type $i\alpha$ and containing a variable from $\bigcup_{\overline{b}} E_{\overline{b}}(t)$ the probability that this clause contains a variable from $E_{\overline{a}}(t)$ equals $2\frac{a_{i\alpha} e_{\overline{a}}(t)}{a_{i\alpha}(t)}$.

Further computation we carry out under the following assumption.

Assumption 1 *Let $\Phi(t)$ be the formula occured on tth step of the LS algorithm and X is the variable being flipped. Then under the condition of history H_t, for a randomly chosen clause C, the events 'C contains X or $\neg X$' and 'C contains a variable from $E_{\overline{a}}(t)$ other than X' are independent.*

Let $\overline{a}'$ denote the tuple such that $a'_{i-} = a_{i+}$ and $a'_{i+} = a_{i-}$. Then

$$\begin{aligned} &\mathbf{E}(e_{\overline{a}}(t+1) - e_{\overline{a}}(t) \mid H_t) \qquad (3) \\ &= \mathbf{P}(X \in E_{\overline{a}'}(t) \mid a'_{0-}(X,t) > a'_{0+}(X,t)) - \mathbf{P}(X \in E_{\overline{a}}(t) \mid a_{0-}(X,t) > a_{0+}(X,t)) \\ &+ 2 \sum_{\substack{\overline{b} \text{ adjacent to } \overline{a} \\ i,j,\alpha}} \left(\frac{b_{j\alpha} e_{\overline{b}}(t) - a_{j\alpha} e_{\overline{a}}(t)}{a_{j\alpha}(t)} \mathbf{E}(a_{\hat{i}\hat{\alpha}}(X,t) \mid a_{0-}(X,t) > a_{0+}(X,t)) \right). \end{aligned}$$

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

$$\frac{dz_{\overline{a}}}{ds} = \frac{z_{\overline{a}'} - z_{\overline{a}}}{u} + 2 \sum_{\substack{\overline{b} \text{ adjacent to } \overline{a} \\ i,j,\alpha}} \left(\frac{b_{j\alpha} z_{\overline{b}} - a_{j\alpha} z_{\overline{a}}}{g_{j\alpha}} h_{\hat{i}\hat{\alpha}} \right). \qquad (4)$$

(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 elemnts satisfying the condition $u > \epsilon$ for some $\epsilon > 0$. As before we can find the required value as limit as $\epsilon \to 0$.

Theorem 3. *If Assumption 1 is true then for any positive ϱ there is a constant c such that for a random 3-CNF $\Phi(n, \varrho n)$ a.s. the LS algorithm finds and assingment such that the number of satisfied clauses equals $cn + o(n)$.*

4.2 Experiments

In this subsection we report on experiments aiming to estimate constant c from Theorem 3 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 allows us to decrease the number of equations without loss of accuracy. We state this 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 $D(t) = \varrho n - (A(t) + B(t) + C(t))$. 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) for $\mathbf{E}(e_{\overline{a}}(t+1) - e_{\overline{a}}(t))$. The first part

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

can be converted into

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

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 $Y \in E_{ab}$ (a) a certain clause of type $0-$ for Y contains $\neg X$, (b) a certain clause of type $1-$ contains X, (c) a certain clause of type $0+$ contains $\neg X$, (d) a certain clause of type $1+$ contains X. The probabilities of these events are

$\mathbf{P}(\neg X \in C \mid C \text{ of type } 0- \text{ for } Y, a_{0-}(X) = K_1) = \frac{2K_1}{A(t)}$,

$\mathbf{P}(X \in C \mid C \text{ of type } 1- \text{ for } Y, a_{0+}(X) = K_2) = \frac{2K_2}{B(t)}$,

$\mathbf{P}(\neg X \in C \mid C \text{ of type } 0+ \text{ for } Y, a_{1-}(X) = K_3) = \frac{2K_3}{B(t)}$,

$\mathbf{P}(X \in C \mid C \text{ of type } 1+ \text{ for } Y, a_{1+}(X) = K_4) = \frac{2K_4}{C(t)}$.

By Assumption 1,

$$P_1 = \mathbf{P}(\neg X \in C \mid C \text{ of type } 0- \text{ for } Y) = \sum_{K_1} \mathbf{P}(a_{0-}(X) = K_1) \frac{2K_1}{A(t)} = 2\frac{\mathbf{E}(a_{0-}(X))}{A(t)},$$

$$P_2 = \mathbf{P}(X \in C \mid C \text{ of type } 1- \text{ for } Y) = \sum_{K_2} \mathbf{P}(a_{0+}(X) = K_2) \frac{2K_2}{B(t)} = 2\frac{\mathbf{E}(a_{0+}(X))}{B(t)},$$

$$P_3 = \mathbf{P}(\neg X \in C \mid C \text{ of type } 0+ \text{ for } Y) = \sum_{K_3} \mathbf{P}(a_{1-}(X) = K_3)\frac{2K_3}{B(t)} = 2\frac{\mathbf{E}(a_{1-}(X))}{B(t)},$$

$$P_4 = \mathbf{P}(X \in C \mid C \text{ of type } 1+ \text{ for } Y) = \sum_{K_4} \mathbf{P}(a_{1+}(X) = K_4)\frac{2K_4}{C(t)} = 2\frac{\mathbf{E}(a_{1+}(X))}{C(t)}.$$

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

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

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

Observation 1 *For a randomly chosen Y and any i, α, $i \neq 0$, and a, b*

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

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

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

The second part of the expectation equals

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

Similarly we have

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

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 $h_1(s) = \frac{1}{g}\sum_{\substack{a,b \\ a>b}} a z_{ab}$, $\quad h_2(s) = \frac{1}{g}\sum_{\substack{a,b \\ a>b}} b z_{ab}$ we get

$$\begin{aligned} \frac{dz_{ab}}{ds} &= \frac{g_{ab}}{g} + 2\left(\frac{h_2}{g}z_{(a-1)b} + \frac{(a+1)h_1}{g}z_{(a+1)b} + rz_{a(b-1)} + (b+1)z_{a(b-1)}\right) \\ &\quad - 2z_{ab}\left(\frac{ah_1}{p} + q + b + r\right), \\ \frac{dr}{ds} &= 2q + u - 3r. \end{aligned} \tag{5}$$

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

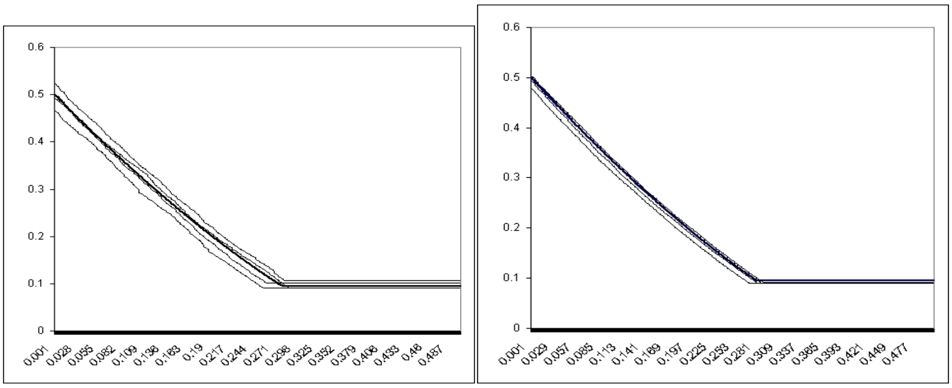

Fig. 4. Empirical performance of LS and its prediction

particular random problems, and the thick lines are computed from a numerical solution of the system above. In the examples shown $\varrho = 4, M = 30$, $n = 1000$ 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 c from Theorem 3 both empirical and predicted by the system (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 (5))	1.98	2.95	3.91	4.38	4.85	5.80	6.73	9.52	14.14	18.74	23.32

References

1. D. Achlioptas. Lower bounds for random 3-SAT via differential equations. *Theor. Comput. Sci.*, 265(1-2):159–185, 2001.
2. T. Asano and D. Williamson. Improved approximation algorithms for MAX SAT. *J. Algorithms*, 42(1):173–202, 2002.
3. M. Goemans and D. Williamson. New 3/4-approximation algorithms for the maximum satisfiability problem. *SIAM J. Discrete Math.*, 7(4):656–666, 1994.
4. J. Gu. Efficient local search for very large-scale satisfiability problem. *ACM SIGART Bulletin*, 3(1):8–12, 1992.
5. P. Hansen and B. Jaumard. Algorithms for the maximum satisfiability problem. *Computing*, 44:279–303, 1990.
6. H. Karloff and U. Zwick. A 7/8-approximation algorithm for MAX 3SAT? In *FOCS*, pages 406–415, 1997.
7. E. Koutsoupias and C. Papadimitriou. On the greedy algorithm for satisfiability. *Inf. Process. Lett.*, 43(1):53–55, 1992.
8. B. Selman, H. Levesque, and D. Mitchell. A new method for solving hard satisfiability problems. In *AAAI*, pages 440–446, 1992.
9. N. Wormald. Differential equations for random processes and random graphs. *The Annals of Applied Probability*, 5(4):1217–1235, 1995.