Local Search Algorithms for SAT: An Empirical Evaluation

Holger H. Hoos (hoos@cs.ubc.ca)
Department of Computer Science
University of British Columbia
Vancouver, Canada

Thomas Stützle* (tstutzle@ulb.ac.be)
IRIDIA, Université Libre de Bruxelles
Brussels, Belgium

Abstract. Local search algorithms are among the standard methods for solving hard combinatorial problems from various areas of Artificial Intelligence and Operations Research. For SAT, some of the most successful and powerful algorithms are based on stochastic local search and in the past 10 years a large number of such algorithms have been proposed and investigated. In this article, we focus on two particularly well-known families of local search algorithms for SAT, the GSAT and WalkSAT architectures. We present a detailed comparative analysis of these algorithms' performance using a benchmark set which contains instances from randomised distributions as well as SAT-encoded problems from various domains. We also investigate the robustness of the observed performance characteristics as algorithm-dependent and problem-dependent parameters are changed. Our empirical analysis gives a very detailed picture of the algorithms' performance for various domains of SAT problems; it also reveals a fundamental weakness in some of the best-performing algorithms and shows how this can be overcome.

Keywords: SAT, Stochastic Local Search, Empirical Evaluation, Run-time Distributions, Robustness

1. Introduction

The satisfiability problem in propositional logic (SAT) is the task to decide for a given propositional formula whether it has a model. This problem plays a prominent role in various areas of computer science, as it is one of the conceptually simplest $\mathcal{NP}$-complete problems. Local search approaches for SAT became prominent in 1992, when independently Selman, Levesque, and Mitchell [60] as well as Gu [27] introduced algorithms based on stochastic local hill-climbing which could be shown to outperform state-of-the-art systematic SAT algorithms like ASAT [13] on a variety of hard subclasses of SAT [4, 59]. Since then, numerous other stochastic local search (SLS) schemes for SAT have been proposed. To date, state-of-the-art SLS algorithms can solve hard SAT problems with up to several thousand variables, including SAT-encoded problems from other domains. But because of their inherent randomness, the behaviour of SLS algorithms is generally difficult to analyse theoretically and even in the cases where theoretical results do

* On leave from FG Intellektik, TU Darmstadt, Germany

exist, their practical applicability is often very limited. Given this situation, the performance of SLS algorithms is usually analysed using empirical methodology.

In this paper, we empirically analyse the behaviour of algorithms from the GSAT and WalkSAT architectures of SLS algorithms for SAT. These algorithm families were chosen for two reasons: Firstly, they provided a major driving force in the development and application of SLS algorithms for SAT. Secondly, for most algorithms from these families, very efficient, thoroughly optimised implementations are available which could easily be modified and extended to cover some more variants and to yield a sufficiently coherent framework for practically conducting experiments. Although our empirical methodology allows us to abstract from machine- and implementation-specific details of algorithmic performance, some of our analyses are so computationally expensive that efficient implementations are indispensable.

Our empirical analysis addresses several important issues. Maybe the most straightforward question to ask is how the performance of different algorithms compares across a range of problem instances from different subclasses of SAT. But furthermore, as the behaviour of these algorithms (like for most other SLS algorithms) is controlled by a number of parameters which have to be chosen before running them, it is crucial to investigate the influence of these parameters on the algorithms' behaviour. In particular, optimal parameter settings are often not known exactly, but rather a range of reasonably good parameter settings is available. Therefore, one might be interested in the robustness of SLS performance *w.r.t.* these parameters. Similarly, the influence of problem dependent parameters, in particular problem size, on SLS behaviour is of considerable interest.

To adequately address these issues, a refined empirical methodology is needed. Therefore, different from many previous studies, our experimental methodology goes considerably beyond just measuring a small number of basic statistics, such as the mean run-time and its standard deviation. Instead, we use the methodology developed in [36] which is based on measuring and analysing run-time distributions (RTDs) on individual problem instances. This allows us to study the behaviour of SLS algorithms on single problem instances as well as on whole subclasses of SAT, while cleanly separating between different sources of randomness, such as the randomised generation of instances from a problem distribution and the inherent randomness of the SLS algorithm applied. Measuring RTDs does not incur any significant overhead in data acquisition, and from the RTD information, the commonly used basic descriptive statistics of the run-time, like its mean, standard deviation, and arbitrary percentiles can be easily derived. At the same time, RTDs provide important information on an algorithm's behaviour in a more precise and accessible way. For instance, the probability for finding a solution within any given time-limit or stagnation behaviour of the search process can easily be determined from a graphical representation of an RTD.

Using the RTD-based methodology, we analyse the behaviour of GSAT and WalkSAT algorithms for a carefully selected set of benchmark problems. This col-

lection has been designed to provide a basis not only for this study, but to serve as the core of a benchmark library that can be used for comprehensive future studies of SAT algorithms in general. Therefore, it covers a broad range of subclasses of SAT which are well-known from the literature and which are preferably hard for both systematic and local search algorithms. The benchmark library comprises three fundamentally different types of problems: SAT instances from randomised distributions of native SAT problems (type 1), SAT-encoded instances from randomised distributions of other combinatorial problems (type 2), and SAT-encoded, hand-crafted individual instances from other domains (type 3). For type 1, we decided to focus on sets of Random-3-SAT instances from the solubility phase transition, as these are known to be very hard for both systematic and SLS-based algorithms [10, 47, 64] and they have played a prominent role in the majority of the studies in literature [55, 60, 59, 22]. For type 2, we selected sets of graph colouring problems from randomised distributions of 3-colourable graphs [54, 35]. Finally, for type 3 we chose SAT-encoded planning instances which have been used in a number of previous studies, as well as a subset of the DIMACS Satisfiability Benchmark Library.[1] Our set of benchmark problems is described in Section 4 and Appendix A; it is, along with the WalkSAT and GSAT implementations (which have been kindly provided by Henry Kautz) used in this work, available from the SATLIB website at `http://www.informatik.tu-darmstadt.de/AI/SATLIB`.

The results of our study can be summarised as follows. For optimal parameter settings the relative performance of the SLS algorithms studied here varies with the problem domain such that there is no single best algorithm over the full benchmark set. However, we can identify a subset of the algorithms studied (this subset comprises the WalkSAT variants R-Novelty, Novelty, WalkSAT/TABU) which show superior performance compared to the remaining algorithms. At the same time, these algorithms show premature stagnation of the search when applied to some instances. As a consequence these algorithms are very sensitive with respect to the setting of the cutoff parameter (which determines after how many steps the search is aborted and restarted from a new initial assignment). However, as we show later, this deficiency can be overcome by a relatively simple and generic modification of the algorithms, the so-called *random walk extension.* The accordingly modified algorithms show improved performance and a significantly increased robustness *w.r.t.* the cutoff parameter setting.

Our investigation of the robustness of SLS performance confirms and extends earlier observations that for relatively hard problem instances, all algorithms exhibit approximately exponential RTDs and are thus very robust *w.r.t.* the cutoff parameter setting. As a consequence of this characterisation result, by using the "multiple independent tries" parallelisation scheme, which is easy to implement and extremely scalable, optimal speedup and parallelisation efficiency can be ob-

[1] Accessible via FTP at `ftp://dimacs.rutgers.edu/pub/challenge/satisfiability/benchmarks/cnf/`

tained. At the same time, the exponential RTDs suggest a novel interpretation of SLS behaviour as independent random picking from a drastically reduced search space. However, as we show, the RTDs are only exponential if the noise parameter, which controls the greediness of the search process, is sufficiently high. If the noise setting is too low, i.e., the search is very greedy, the robustness of SLS performance *w.r.t.* the cutoff parameter deteriorates rapidly. Furthermore, there are considerable differences between the individual algorithms in the robustness *w.r.t.* the setting of the so-called noise parameter.

Another interesting result is the fact that across test-sets of randomly generated problem instances (such as Random-3-SAT at the solubility phase transition) the relative performance of the different algorithms tends to be tightly correlated, such that problem instances that are hard for one algorithm tend to be hard for all the other algorithms. This is somewhat surprising when considering that the algorithms studied here are different enough to yield absolute performance differences of more than one order of magnitude. At the same time, this result suggests that the same structural features are responsible for rendering a problem instance hard or easy for all the algorithms.

Finally, we investigate the scaling of SLS performance with problem size. We found no evidence of any substantial differences between different WalkSAT algorithms *w.r.t.* their scaling behaviour for growing problem size. But, somewhat surprisingly, our results indicate that the local search cost grows exponentially only for the hardest Random-3-SAT instances, while for easier instances the scaling of SLS performance appears to be polynomial. This phenomenon could not be observed for the graph colouring test-sets, where the scaling was generally exponential.

Knowledge on which algorithms perform best under various criteria, such as maximum performance or robustness, is useful for directing future efforts in designing even better SAT algorithms. Given the recent successes of solving combinatorial problems from other domains by encoding them into SAT, especially in various domains of AI planning [45, 46], using SAT algorithms for solving encoded real-world problems seems to become a serious alternative to using specialised algorithms under certain circumstances. In the context of this development it seems to be of crucial importance to make sure that the best-performing SAT algorithms are used and that when comparing their performance to the performance of domain-specific algorithms all important aspects of their behaviour are taken into account. Moreover, since most SLS algorithms for SAT can be easily generalised to more general problem classes, such as finite Constrained Satisfaction Problems or MAX-SAT, the potential impact of this knowledge would very likely reach beyond SAT. It is unfortunate that most of the empirical studies of SLS algorithms for SAT that can be found in the literature are incompatible with each other. Because of differences in either the methodology or the reference problem instances being used their results cannot be easily assembled to yield a comprehensive picture. Additionally, most of these previous studies are rather limited in

their methodology and scope. The empirical methodology presented in this paper, the benchmark library used, and the results obtained, are intended to provide the starting point for a series of empirical studies which are compatible with each other and can thus be used to approach a comprehensive picture of the state-of-the-art in solving propositional satisfiability problems.

The remainder of the paper is structured as follows. Section 2 introduces a general framework for SLS algorithms, and in particular, the GSAT and WalkSAT families of SAT algorithms. In Section 3, we give an overview of the RTD-based methodology for empirically investigating SLS behaviour on individual problem instances as well as on problem distributions. The benchmark suite which provides the basis for our empirical study is introduced in Section 4. The problems comprising the benchmark library and the results of a comparative analysis of SLS performance across these test-sets are reported in Section 5. In Section 6 we present our characterisation of parameter robustness and scaling of SLS behaviour, including an overview of some theoretical results concerning the asymptotic behaviour of prominent GSAT and WalkSAT algorithms. Furthermore, we show how these results can be exploited to improve some of the best known SLS algorithms for SAT. The concluding Section 7 summarises the main results presented in this paper and points out directions for future research.

2. Stochastic Local Search Algorithms for SAT

Local search is a widely used, general approach for solving hard combinatorial search problems. The general idea is to examine the search space induced by the given problem instance for solutions by initialising the search at some point and from there to iteratively move from one search space position to a neighbouring position where the decision on each step is based on information about the local neighbourhood only. Thus, the following components are required to define a local search procedure for problem class Π applied to a given problem instance $\pi \in \Pi$:

- the *search space* S_π of π which is a set of *positions* $s \in S_\pi$ (also called candidate solutions or states)
- a *set of solutions* $S' \subseteq S_\pi$
- a *neighbourhood relation* $\mathbf{N} \subseteq S_\pi \times S_\pi$ on S_π which specifies for each position the set of positions which can be visited in one local search step.
- an *initialisation function* $\text{init} : \emptyset \mapsto (S_\pi \mapsto \mathbb{R})$ which specifies a probability distribution over initial search positions.
- a *step function* $\text{step} : S_\pi \mapsto (S_\pi \mapsto \mathbb{R})$ mapping each position onto a probability distribution over its neighbouring states, for specifying the local search steps.

Typically, local search algorithms also make use of an *objective function* $f : S_\pi \mapsto \mathbb{R}$, mapping each search space position onto a real (or integer) number in such a way, that the global optima of the objective function correspond to the solutions. In SLS algorithms for SAT, generally the search space is defined as the set of all possible truth value assignments for the variables occurring in the given problem instance and the solutions are the satisfying assignments (models). Of course the search space thus defined is of exponential size in the number of propositional variables. Most SLS-based SAT algorithms use a 1-flip neighbourhood relation for which two truth value assignments are neighbours if they differ in the truth value of exactly one variable. Thus, the local search steps modify the truth value assigned to one propositional variable; such a move is called a *variable flip*. Almost all SLS algorithms for SAT work on propositional formulae in conjunctive normal form (CNF), i.e., on sentences of the form $\bigwedge_{i=1}^{k} c_k$ with $c_k = \bigvee_{j=1}^{l_k} L_{kj}$ where the L_{kj} are propositional variables or their negations. Typically, they use an objective function which is defined as the number of clauses which are unsatisfied under the given variable assignment. Hence, for satisfiable problem instances the models of the given formula correspond to global minima of this function. The general idea for finding these models is to perform a random walk in the search space which is biased towards minimising the number of unsatisfied clauses (obviously, this is equivalent to maximising the number of satisfied clauses). The main difference between different SLS algorithms for SAT is in the step function, that is, in the strategy used to select the variable to be flipped next.

SLS algorithms for SAT are typically incomplete, i.e., even for satisfiable problem instances they cannot be guaranteed to find a solution. The reason for this is the non-systematic nature of the search. Furthermore, SLS algorithms can get trapped in local minima and plateau regions of the search space [22, 15], leading to premature stagnation of the search. One of the simplest mechanisms for avoiding premature stagnation of the search is random restart, which reinitialises the search if after a fixed number of steps (cutoff time) no solution has been found. Random restart is used in almost every SLS algorithm for SAT.

A general outline of an SLS algorithm for SAT is given in Figure 1. The generic procedure initialises the search at some truth assignment and then iteratively flips some variable's truth value, where the selection of the variable depends on the formula Φ and the current assignment. If after a maximum of *maxSteps* flips no solution is found, the algorithm restarts from a new random initial assignment. If after a given number *maxTries* of such tries still no solution is found, the algorithm terminates unsuccessfully.

For many concrete algorithms, and in particular, for all algorithms investigated here, *initAssign* randomly chooses the initial assignment from the set of all possible assignments according to a uniform distribution. Hence, the main difference between SLS algorithms for SAT is typically the implementation of the step function as given by the procedure *chooseVariable*. In the following, we will focus on the GSAT and the WalkSAT families of algorithms, which have provided

```
procedure StochasticLocalSearch for SAT
    input CNF formula Φ, maxTries, maxSteps
    output satisfying assignment of Φ or "no solution found"
    for i := 1 to maxTries do
        s := initAssign(Φ);
        for j := 1 to maxSteps do
            if s satisfies Φ then return s;
            else
                x := chooseVariable(Φ, s);
                s := s with truth value of x flipped;
            end if
        end for
    end for
    return "no solution found";
end StochasticLocalSearch for SAT
```

Figure 1. Outline of a general stochastic local search procedure for SAT.

a major driving force in the development of SLS algorithms for SAT, and review some of the most widely used members of these families. Later, we will briefly discuss other algorithms and further GSAT and WalkSAT variants known from the literature.

2.1. THE GSAT ARCHITECTURE

The GSAT algorithm was introduced in 1992 by Selman, Levesque, and Mitchell [60]. It is based on a rather simple idea: GSAT tries to minimise the number of unsatisfied clauses by a greedy descent in the space of variable assignments. Variable selection in GSAT and most of its variants is based on the *score* of a variable x under the current assignment s, which is defined as the difference between the number of clauses unsatisfied by s and the assignment obtained by flipping x in s.

2.1.1. *Basic GSAT*

The basic GSAT algorithm uses the following instantiation of the procedure *chooseVariable*(s, Φ). In each local search step, one of the variables with maximal score is flipped. If there are several variables with maximal score, one of them is randomly selected according to a uniform distribution. A straightforward implementation of GSAT may be rather inefficient, since in each step the score of all variables would have to be calculated from scratch. The key to efficiently implementing GSAT is to evaluate the complete set of scores only once at the beginning of each try, and then after each flip to update only the scores of those variable which were possibly affected by the flipped variable. Details on these implementation issues for GSAT and related algorithms can be found in [32]. One problem with basic GSAT is that it can easily get stuck in local minima of the objective functions. As these

might contain a large number of search space positions between which GSAT can arbitrarily wander, local minima cannot be efficiently detected in general and the built-in random restart mechanism is the only way for escaping from these. This means that for GSAT there exist search space positions from which a solution cannot be reached without reinitialising the search. We call SLS algorithms which have this property *essentially incomplete.*

2.1.2. *GSAT with Random Walk (GWSAT)*
An important extension of the basic GSAT algorithm is GSAT with Random Walk (GWSAT) [59]. Here, besides the above defined greedy local search step, a second type of local search step, the so-called *random walk step*, is introduced. In a random walk step, first a currently unsatisfied clause c' is randomly selected. Then, one of the variables appearing in c' is flipped, thus effectively forcing c' to become satisfied. The basic idea of GWSAT is to decide at each local search step with a fixed probability *wp* (called *walk probability* or noise setting) whether to do a standard GSAT step or a random walk step. The random walk mechanism has been added to basic GSAT to avoid getting stuck in local minima of the objective function. Obviously, for arbitrary $wp > 0$ this algorithm allows arbitrarily long sequences of random walk steps; as detailed in [35], this implies that from any given assignments, a model (if existent) can be reached with a positive, bounded probability. This means that applied to a satisfiable formula, even without using random restart (that is, for *maxTries*=1), the probability that GWSAT finds a solution converges to 1 as the run-time approaches infinity (the so-called *PAC property*, or *probabilistic approximate completeness*).

2.1.3. *GSAT with Tabu Search (GSAT/TABU)*
Another well-known mechanism for preventing the search from getting stuck in local optima is Tabu Search [23, 28]. The general idea is to forbid reversing the effect of a particular move for a number *tl* of iterations; the parameter *tl* is called the *tabu tenure*. This mechanism can be easily added to basic GSAT [50, 61]; in the resulting algorithm, after a variable *x* has been flipped it cannot be flipped back within the next *tl* steps. GSAT/TABU can be efficiently implemented by storing for each variable *x* the iteration number i_x when it was last flipped. When initialising the search, all the i_x are set to $-tl$ and every time a variable *x* is flipped, i_x is set to the number j of the current iteration. Obviously, a variable *x* can only be flipped if $j > i_x + tl$.

Note that, unlike for GWSAT, it is not clear whether for arbitrary satisfiable formulae there exists a *tl* value such that GSAT/TABU has the PAC property for fixed *maxTries*. This means that adding Tabu Search does not necessarily guarantee that the algorithm will never get stuck in local minima. Intuitively, the problem is that for low *tl* values the local minima region might be big enough that the algorithm still cannot escape without using restart, while for high *tl* values all the routes to a solution might be cut off because too many variables cannot be flipped.

2.1.4. *HSAT and HWSAT*

HSAT [21] is another GSAT variant where the local search steps make use of history information. When in a search step there are several variables with identical score, HSAT selects the least recently flipped variable. Only shortly after search initialisation, when there are still variables which have not been flipped, a random selection like in GSAT is done. The intuition behind HSAT is that in GSAT some variables might never get flipped although they are frequently eligible to be chosen. This might cause stagnation of the search, as such a variable might be the one which has to be flipped for the search to make further progress. Although when compared to plain GSAT, HSAT was found to show superior performance [21], it is clear that it is even more likely to get stuck in local minima from which it cannot escape, as the additional history-based tie-breaking rule effectively restricts the search trajectories when compared to GSAT. Therefore, it appears to be attractive to extend HSAT with the same random walk mechanism used in GWSAT. The resulting variant is called HWSAT [20]; like GWSAT it has the PAC property.

2.2. THE WALKSAT ARCHITECTURE

The WalkSAT architecture is based on ideas first published by Selman, Kautz, and Cohen in 1994 [59] and it was later formally defined as an algorithmic framework by McAllester, Selman, and Kautz in 1997 [51]. It is based on a 2-stage variable selection process focused on the variables occurring in currently unsatisfied clauses. For each local search step, in a first stage a currently unsatisfied clause c' is randomly selected. In a second step, one of the variables appearing in c' is then flipped to obtain the new assignment. Thus, while the GSAT architecture is characterised by a static neighbourhood relation between assignments with Hamming distance one, WalkSAT algorithms are effectively based on a dynamically determined subset of the GSAT neighbourhood relation.

2.2.1. *WalkSAT*

WalkSAT, originally introduced in [59], differs in one important aspect from the other local search variants discussed here: The scoring function $score_b(x)$ used by WalkSAT counts only the number of clauses which are broken, i.e., currently satisfied, but will become unsatisfied by flipping variable x. Using this scoring function, the following variable selection scheme is applied: If there is a variable with $score_b(x) = 0$ in the clause c' selected in stage 1, i.e., if c' can be satisfied without breaking another clause, this variable is flipped (so-called *zero-damage flip*). If no such variable exists, then with a certain probability p (noise setting) the variable with minimal $score_b$ value is selected; in the remaining cases, one of the variables from c' is randomly selected (random walk flip).

Conceptually as well as historically, WalkSAT is closely related to GWSAT. However, there are a number of significant differences between both algorithms, which in combination account for the generally superior performance of WalkSAT.

While both algorithms use the same kind of random walk steps, WalkSAT applies them only under the condition that there is no variable with $score_b(x) = 0$. In GWSAT, however, random walk steps are done in an unconditional probabilistic way. From this point of view, WalkSAT is greedier, since random walk steps, which usually increase the number of unsatisfied clauses, are only done when every variable occurring in the selected clause would break some clauses when flipped. Yet, in a greedy step, due to the two-stage variable selection scheme, WalkSAT chooses from a significantly reduced set of neighbours and can therefore be considered to be less greedy than GWSAT. Finally, because of the different scoring function, in some sense, GWSAT shows a greedier behaviour than WalkSAT: For a GSAT step, it would prefer a variable which breaks some clauses but compensates for this by fixing some other clauses, while in the same situation, WalkSAT would select a variable with a smaller total score, but breaking also a smaller number of clauses.

To our best knowledge it is unknown whether WalkSAT has the PAC property, i.e., whether it can theoretically escape from any local minima without using random restart. Unlike for GWSAT, it is not clear whether arbitrarily long sequences of random walk steps are always possible, because random walk steps can only be executed when no zero-damage flip is available. However, there is plenty of empirical evidence that in practice WalkSAT can effectively escape from any local minima [35, 37].

2.2.2. *WalkSAT/TABU*

Analogously to GSAT/TABU, there is also a WalkSAT variant which uses a Tabu Search mechanism; this algorithm is called WalkSAT/TABU [51]. It uses the same two stage selection mechanism and the same scoring function $score_b$ as WalkSAT and additionally enforces a tabu tenure of tl steps for each flipped variable. Here, if no zero-damage flip can be made, from all variables which are not tabu, the one with the highest $score_b$ value is picked; when there are several variables with the same maximal score, one of them is randomly selected according to a uniform distribution. As a result of the two-level variable selection scheme, it may happen that all variables appearing in the selected clause cannot be flipped because they are tabu. In this case, no variable is flipped (a so-called *null-flip*).

As has been shown in [35], WalkSAT/TABU is essentially incomplete for all tl settings, as it can get stuck in local minima regions of the search space. Although this is mainly caused by null-flips, it is not clear whether replacing null-flips by, e.g., random walk steps, would be sufficient for obtaining the PAC property.

2.2.3. *Novelty*

Novelty, introduced in [51], is one of the most recent SLS algorithms for SAT. Conceptually, it combines the algorithms based on the WalkSAT architecture with a history-based variable selection mechanism in the spirit of HSAT. Novelty, too, is based on the intuition, that repeatedly flipping back and forth the same variable should be avoided. Additionally, like for tabu search variants, the number of local

search steps which have been performed since it was last flipped (also called the variable's *age*) is taken into consideration. Also, differently from WalkSAT or WalkSAT/TABU, Novelty uses the same scoring function as GSAT.

In Novelty, after an unsatisfied clause has been chosen, the variable to be flipped is selected as follows. If the variable with the highest score does not have minimal age among the variables within the same clause, it is always selected. Otherwise, it is only selected with a probability of *1-p*; in the remaining cases, the variable with the next lower score is selected. In Kautz's and Selman's implementation, if there are several variables with identical score, the one appearing first in the clause is always chosen.

Note that for $p > 0$, the age-based variable selection of Novelty probabilistically prevents flipping the same variable over and over again; at the same time, flips can be immediately reversed with a certain probability if a better choice is not available. Generally, the Novelty algorithm is significantly greedier than WalkSAT, since always one of the two most improving variables from a clause is selected, where WalkSAT may select any variable if no improvement without breaking other clauses can be achieved. Also, Novelty is more deterministic than WalkSAT and GWSAT, since its probabilistic decisions are more limited in their scope and take place under more restrictive conditions. For example, different from WalkSAT, the Novelty strategy for variable selection within a clause is deterministic for both $p = 0$ and $p = 1$.

On one hand side, this often leads to a significantly improved performance of Novelty when compared to WalkSAT. On the other hand, because of this property, it can be shown that Novelty is essentially incomplete [35], as selecting only among the best two variables in a given clause can lead to situations where the algorithm gets stuck in local minima of the objective function. As we will see later, this situation can be observed for some of our benchmark problems where it severely compromises Novelty's performance.

2.2.4. *R-Novelty*

R-Novelty, also introduced in [51], is a variant of Novelty which is based on the intuition that when deciding between the best and second best variable (using the same scoring function as for Novelty), the actual difference of the respective scores should be taken into account. The exact mechanism for choosing a variable from the selected clause can be seen from the decision tree representation given in Figure 2. Note that the R-Novelty heuristic is quite complex – as reported in [51], it was discovered by systematically testing a large number of WalkSAT variants.

R-Novelty's variable selection strategy is even more deterministic than Novelty's; in particular, it is completely deterministic for $p \in \{0, 0.5, 1\}$. Since the pure R-Novelty algorithm gets too easily stuck in local minima, a simple loop breaking strategy is used: every 100 steps, a variable is randomly chosen from the selected clause and flipped. However, it can be shown that this mechanism is insufficient for

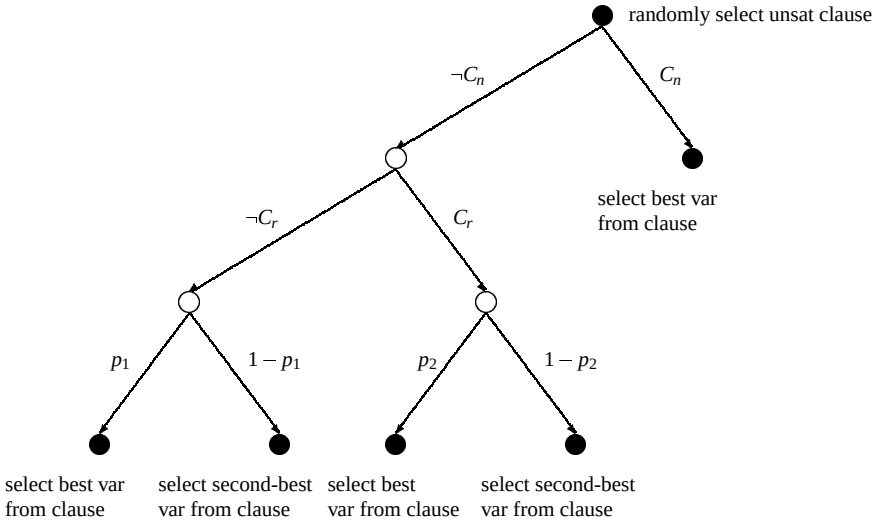

Figure 2. Decision tree representation for R-Novelty variable selection mechanism; condition $C_n \equiv$ "best var does not have minimal age"; condition $C_r \equiv$ "score diff between best and second best var > 1"; "best" and "second-best" relate to the GSAT-score of variables; $p_1 = \max\{1-2p, 0\}$, $p_2 = \min\{2-2p, 1\}$.

effectively escaping from local minima and consequently R-Novelty is essentially incomplete.

2.3. Other SLS algorithms for SAT

Apart from the presented GSAT and WalkSAT algorithms and their variants, numerous other SLS algorithms for SAT have been proposed. One particularly interesting technique which provides the basis for a number of SLS algorithms studied in literature is *clause weighting*. The underlying idea is to associate the clauses of the given CNF formula with weights which are modified during the local search process [7, 16, 17, 53, 58]. The objective function, on which the choice of the variable to be flipped is based, then reflects the weight of the satisfied clauses such that clauses with a high weight are more likely to become satisfied. Typically, the weights of the clauses which are unsatisfied at a local minimum are increased by some constant. One interpretation of clause weighting is that local minima are "filled in" by adapting the weights.

In [58], weights are only modified after each GSAT try by increasing the weight of unsatisfied clauses by one. A different approach, the breakout method [53], adjusts the clause weights during a single try. In particular, each time the algorithm encounters a local minimum, the weights of the currently violated clauses are increased. In [7, 16, 17] different strategies of how to change the weights during a single try are investigated. When evaluating the performance of a par-

ticular weighting scheme on a class of randomly generated instances which have only one single solution [2], it was observed that for the hardest of these instances the weighing scheme performed considerably better than GSAT and GWSAT [7]. Yet, the instances used in that comparison are not intrinsically hard because they can be solved by polynomial simplifications (unit propagation and unary / binary failed clauses). Experimental results presented in [16, 17] show that the weighting strategies significantly improve over GSAT and give somewhat better results than HSAT on Random-3-Sat instances (see Section 4.1 for a description of this problem class). More concretely, using the weighting schemes proposed, the average number of flips needed to find a solution is reduced by ca. 50% when compared to HSAT. Yet it is not clear how efficiently the proposed schemes can be implemented since the weights have to be modified rather frequently and therefore the single search steps are more expensive — in terms of CPU time — than for GSAT. Considering our results reported in Section 5, it will become clear that even when arguing based on the most efficient implementations of GSAT, these variants will generally not be competitive with the best-performing WalkSAT algorithms. In this study we do not investigate local search algorithms with clause weighting schemes. There are two main reasons for this: On the one side, using clause weighting is a general mechanism which could be combined with almost any SLS algorithm; in particular it could be added to each of the GSAT and WalkSAT algorithms studied here. On the other side, except for GSAT with clause weighting, as proposed in [58], there are no efficient implementations of the weighting schemes available, which would render the type of empirical analysis performed in this study extremely expensive in terms of computation time. However, we feel that a systematic investigation of weighting schemes and their effect on the performance of various SLS algorithms, especially the best-performing WalkSAT variants, would be an important extension of the work presented here.

Another interesting SLS algorithm is the GSAT variant using cycle-cutsets (GSAT+CC) [43]. This algorithm additionally incorporates a number of other extensions to GSAT, including random walk and clause weighting. To our best knowledge, so far it has only been applied to binary constraint satisfaction problems (CSPs); the results reported in [43] suggest that GSAT+CC might show a performance roughly similar to WalkSAT for a very limited subclass of CSPs (those with small cutsets) while on other subclasses of CSPs it seems to be about a factor of 2 inferior to the GWSAT variant used for comparison.

Many other local search variants which are very popular in the Operations Research community have been applied to SAT. These include methods based on Simulated Annealing [3, 63, 59], Evolutionary Algorithms [26, 14], and Greedy Randomized Adaptive Search Procedures (GRASP) [56]. Some of these algorithms have been directly compared to the algorithms used in this study, but from the data published on the performance of these algorithms, there is no evidence that they might generally perform significantly better than the algorithms considered here.

3. Empirical Methodology

SLS algorithms like those presented in Section 2 strongly involve random decisions such as the choice of the initial assignment, random tie-breaking, or biased random moves. Due to this inherent randomness, given a specific soluble problem instance, the time needed by an SLS algorithm to find a solution is a random variable, as it varies from run to run. Consequently, the most detailed characterisation of such an algorithm's behaviour is given by its run-time distribution (RTD), which for a given instance maps the run-time t to the probability of finding a solution within time t [36, 37, 35].

3.1. RTD-BASED ANALYSIS OF SLS ALGORITHMS

To measure RTDs, one has to take into account that most SLS algorithms have some cutoff parameter bounding their run-time, like the *maxSteps* parameter in the generic algorithm schema of Figure 1. Practically, we measure empirical RTDs by running the respective Las Vegas algorithm for a fixed number of times (without using restart, i.e., setting *maxTries* to one) on a given problem instance up to some (very high) cutoff value (optimal cutoff settings may then be determined a posteriori using the empirical run-time distribution) and by recording the time required to find a solution for each successful run (*cf.* Figure 3, *left*). The empirical run-time distribution is the cumulative distribution associated with these observations. More formally, let $rt(j)$ denote the run-time for the jth successful run, and n be the number of runs performed, then the cumulative empirical RTD is defined by $\widehat{P}(rt \leq t) = |\{j|rt(j) \leq t\}|/n$. Instead of actually measuring run-*time* distributions in terms of CPU-time, it is often preferable to use representative operation counts as a more machine (and implementation) independent measure of an algorithm's performance. An appropriate operation count for local search algorithms for SAT is the number of local search steps; the run-time distributions obtained by measuring the run-time in terms of local search steps (here: variable flips) instead of CPU time are also called run-*length* distributions (RLDs) (*cf.* Figure 3, *right*).

From the RTD information, the commonly used basic descriptive statistics of the run-time, like its mean, standard deviation, and arbitrary percentiles can be easily derived. At the same time, measuring RTDs does not incur any significant overhead in data acquisition, because the amount of experimentation required for obtaining stable estimates for basic statistics (like the mean run-time) is also sufficient for measuring RTDs with a reasonable accuracy. But compared to measuring standard descriptive statistics, an RTD-based empirical analysis has a number of advantages.

Firstly, it does not make any assumptions regarding the application scenario, i.e., the time constraints in finding a solution. To see why this is important, consider a SAT-based planner working in a real-time environment, where the run-time of the SAT solver is severely limited. When using an SLS algorithm in such an

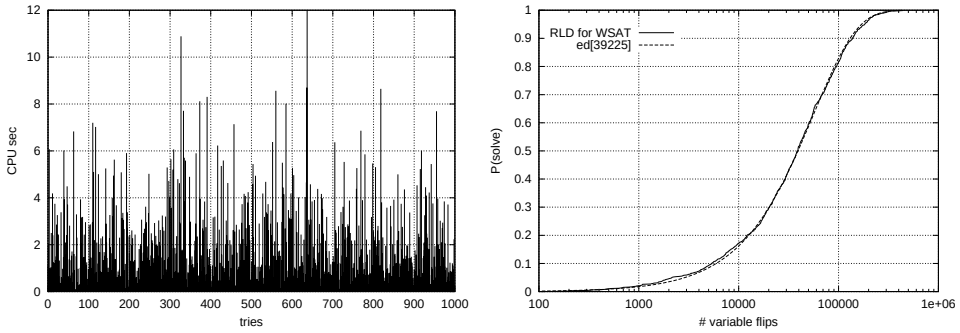

Figure 3. Run-time data of WalkSAT, applied to a hard Random-3-SAT instance for approx. optimal setting of *wp*, 1000 tries. *Left:* Bar diagramm of $rt(i)$ (vertical axis: run-time in CPU seconds). *Right:* Corresponding RLD with best-fit approximation by an exponential distribution (horizontal axis: run-time in local search steps; vertical axis: probability of solving the instance).

application context, the expected time for solving a particular problem instance is typically not relevant; the correct performance criterion is the probability of solving the problem within the given time-limit — which can be easily derived from an estimate of the RTD.

Secondly, RTDs give a graphic representation of the algorithm's behaviour which is well suited for analysing and describing the qualitative behavior of an algorithm. Especially when comparing the performance of different algorithms, a graphic representation of the RTDs is very useful. Let us consider two algorithms which, when applied to the same problem instance, show a cross-over in the corresponding RTDs. This immediately implies that for short runs one algorithm gives a higher solution probability than the other, while for longer run-times the situation is reversed. The location of the cross-over indicates the critical time at which both algorithms give the same solution probability. If on the other hand no such cross-over occurs, this provides evidence that for the given problem instance, one algorithm dominates the other in the sense that it gives consistently higher success probabilities, regardless of the run-time. Note that when only comparing the mean or median run-time, or even a number of percentiles, domination can typically not be tested but is often implicitely and misleadingly suggested by this data. The graphic representation of an RTD often also gives an indication whether an algorithm shows stagnation behaviour and thus gives hints on the possible incompleteness of an algorithm. Stagnation in an RTD is indicated by the fact that the RTD shows a horizontal section with a solution probability smaller than one over one or more orders of magnitude on the run-time axis. Moreover, as argued in [36], given the RTD data, optimal cutoff times (i.e., *maxSteps* settings) can be determined in a rather straight-forward way.

3.2. Analysing SLS performance for problem distributions

As specified above, RTDs are measured for individual problem instances rather than for sets of instances. While at the first glance this might seem like a disadvantage of the RTD-based approach, in fact it is one of its biggest advantages. To see why this is so, consider the prominent case of evaluating an SLS algorithm on a set of problem instances sampled from a random instance distribution, such as uniform Random-3-SAT (*cf.* Section 4.1) for a given number of variables and clauses. There is two sources of randomness which have to be taken into consideration: the variability in the run-time when applying the algorithm to an individual problem instance and the stochastic generation of the instances. For various reasons we feel that it is very desirable to investigate these two types of random influences seperately; one of these is the fact that the type of randomness captured in the RTD might strongly depend on the algorithm being used while the instance-related variation might be caused by properties of the instance which affect a large number of algorithms. Later, in Section 6, we will show that for Random-3-SAT this is actually the case.

Another important argument for analysing RTDs based on individual instances is the following (see [36] for details): Consider a situation where for a set of problem instances the RTDs on all individual instances are exponential distributions. As we will show in Section 6, this implies that this algorithm is very robust *w.r.t.* the *maxSteps* setting and cannot be improved by using restart. If, however, one looks at an "average RTD" obtained by summing up the RTDs over the instances of the test-set and dividing by the number of instances, this RTD is generally not an exponential distribution, as the family of exponential distributions is not closed under the operation of averaging [57]. As argued in [36], this RTD will generally suggest the existence of an optimal setting of the *maxSteps* parameter. Using this setting will indeed maximise the algorithm's performance — but only in the sense that within a given time period, the number of problem instances randomly drawn from the instance set (or distribution) which are solved within this time period will be maximal. However, as an RTD-based analysis shows, in this case the "optimal parameter setting" will not affect the performance on any individual instance from the set — it will only make sure that not too much is wasted trying to solve hard instances. Thus, using the optimal setting will effectively introduce a bias for solving easier problems — an effect which, except for very special application situations, will most likely be undesirable and can potentially give rise to erroneous interpretations of the observed behaviour. This problem becomes very relevant when the inter-instance difficulty within the test-set has a high variance, as is the case for Random-3-SAT and Graph Colouring test-sets (see also Section 5).

Using the RTD-based approach requires estimating the RTDs for each individual problem instance from a given test-set such as the Random-3-SAT and Graph Colouring instance sets used for this study. In all cases, when evaluating an algorithm's behaviour for a test-set of instances, we follow this approach. How-

ever, it is impractical to actually report the RTD data for 100 or 1,000 instances (the typical size of our test-sets). Instead, we usually report RTD data for a small number of instances (Section 5) or perform functional characterisations of the individual RTDs (Section 6). Additionally, we analyse and report distributions of the RTDs' means or percentiles (typically the median) across the given test-set. These *hardness distributions* reflect the variation in search cost arising from the random structure of the instances. Interestingly, using this approach one can easily see (cf. Section 5) that the RTDs are usually very different from the hardness distributions; in particular, the hardness distributions are typically heavy-tailed (cf. [24]), while we never observed a heavy-tailed RTD for any of the SLS algorithms studied here.

3.3. Functional characterisations of RTDs

In addition to the advantages discussed above, an RTD-based methodology provides the basis for more advanced types of empirical evaluations, such as the functional characterisation of the RTDs for a given algorithm. Here, one idea is that if a given algorithm displays the same type of RTDs for a range of problem instances and this type can be determined, then the RTD data can be represented in a much more compact and informative way. Practically, this can be done by approximating RTDs using parameterised families of distribution functions, where the parameters are estimated from the measured RTD data (*cf.* Figure 3, *right*). Once such characterisations are obtained, they provide a basis for studying the dependence of the RTD parameters on various factors such as problem size or constrainedness.

Note that this type of empirical analysis can be used to produce hypotheses, such as "when applying WalkSAT and using an optimal setting of *wp* to hard Random-3-SAT problem instances, the RTDs are well approximated by exponential distributions", which can be tested for statistical significance. Therefore, this type of RTD-based analysis follows suggestions made by Hooker [30, 31], that the empirical analysis of algorithms should not remain at the stage of simply collecting data. Rather, analogous to empirical methodologies used in other sciences, one should attempt to formulate hypotheses based on this data which, in turn, can be experimentally refuted or validated. As we will see in Section 6, this approach is extremely useful for investigating the robustness of SLS performance. Furthermore, functional characterisations of RTDs may also help in relating the empirical behaviour of algorithms to more general statistical phenomena and thus provide a basis for new interpretions of the observed behaviour [24, 18, 37, 35].

Finally, it should be noted that the same RTD-based methodology we are using here for analysing the run-time behaviour of SLS algorithms for SAT can generally be applied to empirical studies of a much broader class of algorithms, the so-called *Las Vegas algorithms.* These are algorithms which, if they find a solution, guarantee its correctness but generally have a run-time which is characterised by a random

variable [49, 1, 35]. Also, this methodology can be generalised to optimisation problems, which is of interest as many SLS algorithms for decision problems can be analogously used for solving the corresponding optimisation problems (e.g., the SAT algorithms covered here can be equally well used for solving MAX-SAT problems) [35, 62].

4. The Benchmark Set

The benchmark suite we are using as a basis for the empirical evaluation of SLS algorithms for SAT comprises three different types of problems: test-sets sampled from Random-3-SAT, a well-known random problem distribution; test-sets obtained by encoding instances from a random distribution of hard Graph Colouring instances into SAT; and SAT-encoded instances from AI planning domains, in particular, from the Blocks World and the Logistics domain [45]. All these benchmark instances are hard in general and difficult to solve for SLS algorithms. For the SAT-encoded problems, the hardness of the instances is inherent rather than just induced by the encoding scheme that was used for transforming them into SAT. The SAT-encodings used here are mostly very simple and well-known from the literature. In addition, we used some of the satisfiable benchmark instances from the second DIMACS challenge [41].

The selection of the benchmark problems was influenced by a number of criteria. Firstly, we want to facilitate the combination of the results reported in this study with future evaluations of other SAT algorithms (not necessarily based on SLS) using the same benchmark library, in order to obtain an overall picture of the performance of different SAT algorithms. Thus, the benchmark set should not be biased by the selection of algorithms studied here. This approach also allows to investigate whether there are performance differences between these algorithms which are so fundamental that they can be consistently observed across the full benchmark set. Our choice of particular problem classes focuses on fairly well-known instances and instance distributions, which have been used widely in the literature. Secondly, as the goal here is to evaluate SLS algorithms which are typically incomplete, we have to restrict the benchmark suite to satisfiable instances. Consequently, when sampling from instance distributions like Random-3-SAT, the cost for filtering out unsatisfiable instances can be very high and restrict the size and number of instances included in our test sets. Finally, because of the detailed RTD-based analysis to be applied, each SLS algorithm has to solve every problem instance a large number of times (typically, at least 250) which again limits the size and number of problem instances in our benchmark collection. Nevertheless, the benchmark library used in the context of this work contains ca. 4,500 problem instances.

In the following, we give a brief overview of the benchmark problems. More detailed information can be found in Appendix A of this paper. All benchmark

instances are available from the SATLIB website (http://www.informatik.tu-darmstadt.de/AI/SATLIB).

4.1. UNIFORM RANDOM-3-SAT

We use a number of test-sets sampled from Uniform Random-3-SAT distributions from the solubility phase transition region at ca. 4.26 clauses per variable [52, 47], where the average instance hardness for both, systematic and stochastic local search algorithms is maximal [9, 10, 64]. Since in the context of (incomplete) SLS algorithms for SAT it makes no sense to include insoluble instances in a benchmark suite, we filtered the randomly generated test-sets for soluble instances using a systematic algorithm which was always run to completion. Thus, for each generated instance we definitely determined whether it is satisfiable or not, and included all satisfiable instances in our test-sets. This way, we ensured that there was no bias towards generating satisfiable instances. As this kind of sampling is computationally expensive, both, the size and the number of instances in the test-sets are effectively limited. Using this generation method, which is well-known from the literature (see [21, 50, 51] for some examples), we generated test-sets for $n = 25$ to $n = 250$ instances with typically 100 instances per test-set. The test-sets for $n = 50$ and $n = 100$ contain 1,000 instances each.

4.2. GRAPH COLOURING

The Graph Colouring problem (GCP) is a well-known combinatorial problem from graph theory: Given a graph $G = (V,E)$, where $V = \{v_1, v_2, ..., v_n\}$ is the set of vertices and $E \subseteq V \times V$ the set of edges connecting the vertices, find a colouring $C : V \mapsto \mathbb{N}$, such that neighbouring vertices always have different colours. When transforming GCP in SAT, we encode a decision variant in which the objective is to find a colouring for a given number of colours. The optimisation variant, which searches for a colouring with a minimal number of colours, can then be solved by solving a series of such decision problems.

We generated seven test-sets of 3-colourable flat random graphs with 50 to 200 vertices. The connectivity (edges per vertex) of these graphs was adjusted in such a way that the instances have maximal hardness (on average) for graph colouring algorithms like the Brelaz heuristic [29]. Each test-set contains 100 instances except for the 50 vertex test-set, which comprises 1,000 instances. The GCP instances were transformed into SAT by using a straightforward, yet efficient encoding known from the literature [12] (see also Appendix A).

4.3. PLANNING INSTANCES

Recently it has been shown that some AI Planning problems can be efficiently solved by encoding them into SAT and then finding models of the SAT formulae using standard SAT algorithms like the SLS algorithms applied in our study.

This approach has shown to be competitive or even to outperform state-of-the-art general-purpose planning algorithms [45]. For our benchmark set we use SAT encodings from two planning domains, the Blocks World domain and the Logistics domain. In Blocks World Planning, starting from some initial configuration, a number of blocks has to be moved to reach some given goal situation; in Logistics Planning, packages have to be moved between locations in different cities using trucks and airplanes with limited capacity.

Our benchmark set contains the four largest Blocks World Planning instances and four Logistics Planning instances from Henry Kautz's and Bart Selman's SATPLAN distribution. These problems range from 459 to 6,325 variables and contain up to 131,937 clauses. At the time of this writing, the two largest Blocks World instances, `bw_large.c` and `bw_large.d`, are among the hardest problem instances which can be solved by state-of-the-art SAT algorithms in reasonable time.

4.4. Instances from the DIMACS Benchmark Set

Other SAT instances we used in our experiments are taken from the benchmark set established during the Second DIMACS Challenge [41]. This benchmark set contains satisfiable as well as unsatisfiable instances; but since the SLS algorithms used in our study cannot be used to establish unsatisfiability, we selected only satisfiable instances for our empirical study. As we were mainly interested in challenging SAT-encoded problems from other domains, we chose the three large SAT-encoded Graph Colouring instances and the smaller SAT-encoded instances from the problem of learning a parity function from given data.

Other instances, like the larger parity learning instances can be solved by some of the best-performing SLS algorithms identified in this study, but the search cost is too high for a systematic empirical analysis of SLS behaviour. Other problems were not considered because they were extremely easy for the algorithms studied here or they could be solved by polynomial preprocessing techniques (for details regarding the selection of problems from the DIMACS collection, the interested reader is referred to Appendix A.)

5. Peak Performance Comparison

In this section, we present the results of our comprehensive performance analysis for the algorithms introduced in Section 2 applied to the suite of benchmark problems described in the previous section. As motivated and introduced in Section 3, we use an RLD-based methodology. In particular, two basic methods are applied: For individual instances, we compare the RLDs of the algorithms using approximately optimal settings of the noise parameters *wp*, *p*, or *tl*. As an optimality criterion we use the expected number of steps required for finding a solution. For test-sets sampled from random instance distributions, we additionally analyse the

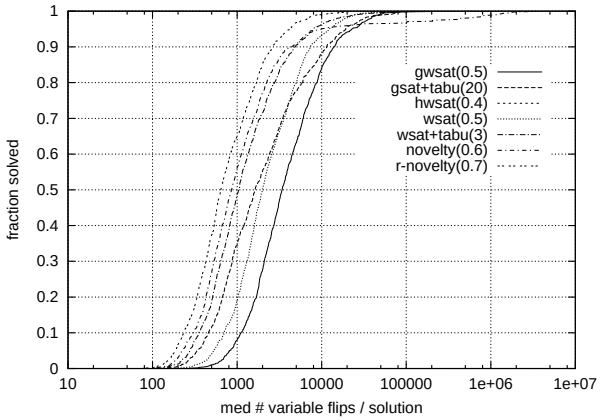

Figure 4. Hardness distributions across test-set `uf100-430` for various algorithms; *x*-axis: median number of local search steps per solution.

distribution of the mean search cost per instance for different algorithms, again using approximately optimal noise settings. More advanced aspects of our empirical investigation, in particular functional characterisations of the RLDs, as well as robustness and scaling issues, are deferred to Section 6, where we also study the influence of the noise parameters in more detail.

5.1. RANDOM-3-SAT

For Random-3-SAT, we first empirically analysed SLS performance for test-set `uf100-430` (1,000 instances) by measuring an RLD for each algorithm applied to each individual instance using approximately optimal noise settings (*wp*, *p*, or *tl*).[2] In this case, the RLDs are based on 100 tries per instance; using an extremely high cutoff parameter setting of $maxSteps = 10^7$ we made sure to obtain a maximal number of successful tries. To get an impression of the variability of the search cost within the test-set, we analysed the distribution of the mean search cost (number of local search steps) over the instances from the test-set. Figure 4 shows these cumulative hardness distributions, while Table I reports the corresponding basic statistics.

For all algorithms, we observe a huge variability in search cost between the instances of each test-set. In particular, as can be seen from the long tails of these distributions, a substantial part of the problem instances is dramatically harder than the rest of the test-set. Furthermore, looking at the graphic representation of the hardness distributions in Figure 4, one might notice that the curves are roughly parallel in a semi-log plot. This indicates that the differences in search cost for these algorithms can be characterised by a uniform factor across the whole hardness dis-

[2] The noise settings have been optimised in preliminary runs. It was found that the approximately optimal noise settings did not considerably differ between different instances from the same test-set.

Table I. Test-set `uf100-430`, basic descriptive statistics of hardness distributions for various algorithms with approx. optimal noise, based on 100 tries / instance; noise settings as for medium instance (*cf.* Table XXIV).

algorithm	mean	stddev	$\frac{\text{stddev}}{\text{mean}}$	med	Q_{75}	Q_{90}	$\frac{Q_{75}}{Q_{25}}$	$\frac{Q_{90}}{Q_{10}}$
GWSAT	6,532	10,639	1.63	3,398	7,040	13,880	3.82	12.62
GSAT/TABU	4,783	9,741	2.04	1,621	4,260	11,399	5.62	24.93
HWSAT	3,039	3,365	1.11	1,914	3,622	6,605	3.33	9.71
WalkSAT	3,672	5,698	1.55	1,995	4,142	7,296	3.62	10.40
WalkSAT/TABU	2,485	6,263	2.52	1,041	2,215	4,751	3.92	13.51
Novelty	28,257	191,668	6.78	851	1,845	4,390	3.85	14.50
R-Novelty	1,245	1,893	1.52	640	1,402	2,625	3.73	11.30

tribution. However, examining the top 10% of the cumulative distribution curves, there are three notable exceptions from that rule: those for GSAT/TABU, WalkSAT/TABU, and — far more dramatically — Novelty have a significantly heavier tail than the curves for the other algorithms. That GSAT/TABU, WalkSAT/TABU, and Novelty cause considerably more inter-instance hardness variation can also be seen when comparing the normalised standard deviations and percentile ratios in Table I. This oberservation indicates that the variants using tabu search, as well as Novelty, have greater difficulties escaping from highly attractive areas in the search spaces of some instances than the other variants. For the tabu search variants this is most probably caused by the fixed length of the tabu list which limits the ability to escape from local minima regions. For Novelty, 33 of the 1,000 instances had a solution rate between 75% and 99%, whereas for identical *maxSteps* settings all other algorithms solved all instances in each try. This explains the huge standard deviation observed for the Novelty hardness distribution. It also indicates that due to its greedy bias, Novelty can get permanently stuck in local minima regions.

Next, we analysed the RLDs obtained by applying the algorithms to individual problem instances. For practical reasons, we restrict our presentation here to the instances corresponding to the minimum, median, and maximum of the hardness distribution for WalkSAT; these are referred to as the "easy", "medium" (med), and "hard" hardness instances from the underlying test-set. The RLDs shown in Figures 5 and 6 offer an easily accessible, yet comprehensive and precise picture of our results. Nevertheless, we additionally include the corresponding descriptive statistics in numerical form. This data and some additional discussion on minor details of our results can be found in Appendix B. As can be seen from the figures, the RLDs for the different algorithms have all roughly the same shape, and, with one exception (R-Novelty and WalkSAT/TABU on the medium instance), no cross-overs occur. Thus, generally there is a dominance relation between the algorithms when applied to the same instance, i.e., when comparing two algorithms' performance, one is uniformly better or worse than the other. Note that based on our

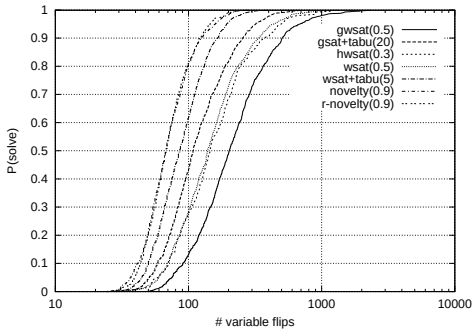

Figure 5. Problem instance `uf100-430/easy`, RLDs for various algorithms, approx. optimal noise, based on 1,000 tries.

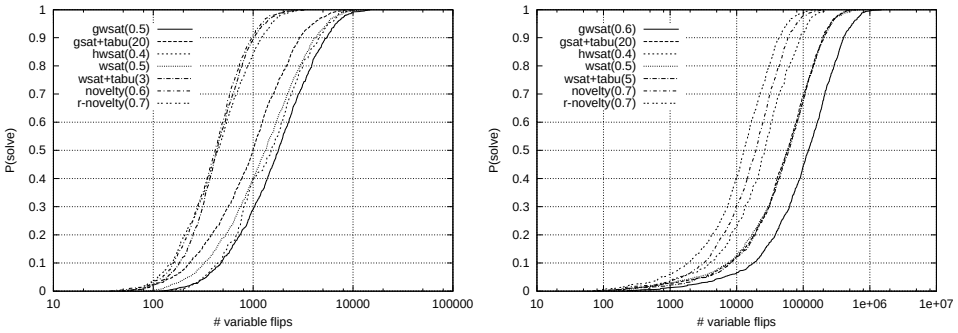

Figure 6. Problem instance `uf100-430/med` *(left)* and `uf100-430/hard` *(right)*, RLDs for various algorithms, approx. optimal noise, based on 1,000 tries / instance.

earlier observation of stagnation behaviour for some of the algorithms, there are instances, for which this is not true, but it holds for the majority of the instances. This indicates that for Random-3-SAT problems, the relative effectivity of different search mechanisms, when compared to each other, does typically not change with increasing length of the search trajectory. Thus, search space features which affect the different search mechanisms non-uniformly are the exception rather than the norm.

Note also that for all algorithms, there is a huge variance in the length of different runs of the same algorithm. Interestingly, for harder problem instances, the variability in the length of the algorithms' runs is even higher, as can be seen by the fact that the distribution curves in the semi-log plots of the RLDs (Figures 5 and 6) get less steep with increasing problem hardness.

From the RLD data, one can easily see that GWSAT is generally the worst performing algorithm, while R-Novelty performs best. Apart from this, however, the relative performance of the algorithms on Random-3-SAT generally depends on the instance hardness. Only for the hard instance, a significant difference between Novelty and R-Novelty can be observed, which suggests that the more complicated

variable selection mechanism used in R-Novelty only pays off for extremely hard instances. At the same time, the performance differences observed between WalkSAT, WalkSAT/TABU, and GSAT/TABU for the easier instances are no longer observed for the hard instance. This suggests that the tabu mechanism might be considerably more effective for easier instances. HWSAT ranks between GWSAT and GSAT/TABU for the easy and medium instance; for the hard instance however, this algorithm shows very competitive performance and ranks third after R-Novelty and Novelty. Thus, compared to rigid tabu lists the more flexible memory-based mechanism used in HWSAT, Novelty, and R-Novelty seems to be more effective for hard instances. It may be further noted that the differences in performance between the best and the worst performing algorithm (GWSAT and R-Novelty) is far more dramatic for the hard instance (a factor of approx. 10 in the median) than for the medium (factor ≈ 4 in the median) and the easy instance (factor ≈ 3 in the median). This suggests that especially for the features which make instances hard for the SLS algorithms considered here, R-Novelty's sophisticated search mechanism is particularly effective. As we will show in Section 6.3, by using advanced analysis techniques, additional evidence for this interpretation can be obtained.

5.2. Graph Colouring

Applying the same RLD analysis as used for Random-3-SAT to the `flat100-329` Graph Colouring test-set (100 instances), we observe some surprising differences (*cf.* Figures 7 and 8; for numerical data on these RLDs, see Appendix B, Tables XXVIII–XXX). These concern mainly the performance of GSAT/TABU and R-Novelty. The latter algorithm, which dominated the scene for the Random-3-SAT test-set, shows a relatively weak performance on the Graph Colouring domain tested here, where it ranks third (on the easy and hard instance) or fourth (on the medium problem). Novelty is significantly better than R-Novelty on all three test instances; the performance difference is uniformly characterised by a factor of ca. 2 in the number of steps required to find a solution with a given probability. This suggests that R-Novelty's sophisticated variable selection mechanism might be too much tuned towards search space features which tend to occur only in Random-3-SAT problems. Alternately, R-Novelty might have difficulties with the more structured search spaces of SAT-encoded problems.

GSAT/TABU, on the other hand, is much stronger when applied to the Graph Colouring test-set than for Random-3-SAT. Here, for the easy and medium instance, GSAT/TABU is only second after Novelty, which is the best algorithm for these instances; for the hard instance, GSAT/TABU is slightly better than Novelty and thus leads the field. But analogously to what we observed for Random-3-SAT, WalkSAT/TABU's performance is relatively poor on the hard instance where, for `flat100-239`, it is inferior to WalkSAT. This is somewhat surprising given the huge advantage GSAT/TABU realises over GWSAT (for the hard instance, we

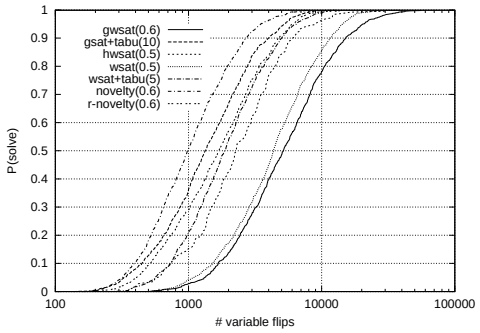

Figure 7. Problem `flat100-239/easy`, RLDs for various algorithms, approx. optimal noise, based on 1,000 tries.

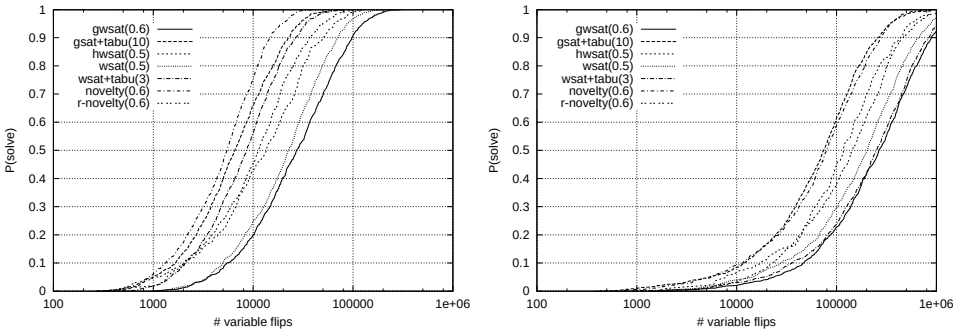

Figure 8. Problem `flat100-239/med` *(left)* and `flat100-239/hard` *(right)*, RLDs for various algorithms, approx. optimal noise, based on 1,000 tries / instance.

observe a factor of ca. 3 between the performance of these two algorithms). Obviously, the more global tabu mechanism in GSAT/TABU is particularly effective for the more structured search spaces of the Graph Colouring instances. HWSAT's performance is consistenly inferior to both, Novelty's and GSAT/TABU's.

Additionally to these RLD analyses, like in the case of Random-3-SAT, we performed an analysis of the instance hardness distribitions for the different algorithms. For practical reasons (computing time limitations and higher accuracy because of a larger number of instances), we used the `flat50-115` test-set for this analysis; however, our experimental experience indicates that these results translate directly to `flat100-239`. The hardness distributions shown in Figure 9 confirm and refine the observations from studying the three sample instances. Novelty, R-Novelty, and GSAT/TABU clearly dominate the other algorithms on the major part of the test-set; the hardness distributions of these algorithms appear to be almost identical. However, for Novelty and R-Novelty, as well as for Walk-SAT/TABU, the distributions have heavy tails which indicate that the algorithms get stuck in local minima for a relatively small number of instances. Consequently, their worst-case performance on the given test-set is drastically worse than the oth-

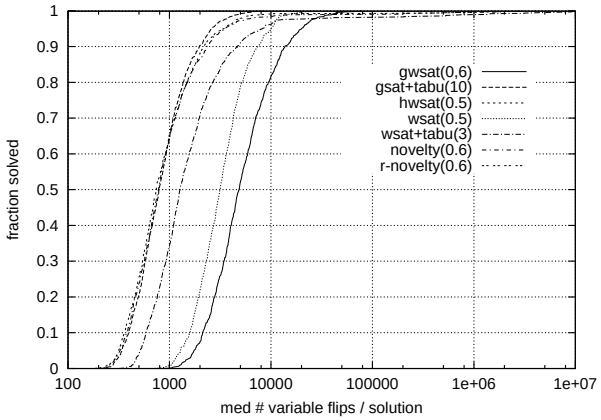

Figure 9. Hardness distributions across test-set `flat50-239` for various algorithms; *x*-axis: median number of local search steps / solution.

Table II. Test-set `flat50-115`, basic descriptive statistics of hardness distributions for various algorithms with approx. optimal noise, based on 100 tries / instance; noise settings as for medium instance (*cf.* Table XXVII).

algorithm	mean	stddev	$\frac{\text{stddev}}{\text{mean}}$	median	Q_{75}	Q_{90}	$\frac{Q_{75}}{Q_{25}}$	$\frac{Q_{90}}{Q_{10}}$
GWSAT	7,023	7,060	1.01	4,846	8,122	13,640	2.65	6.44
GSAT/TABU	1,040	855	0.82	787	1,220	1,963	2.26	4.99
HWSAT	2,641	2,056	0.78	2,042	3,139	4,926	2.30	4.95
WalkSAT	3,913	2,779	0.71	3,132	4,648	7,287	2.22	4.63
WalkSAT/TABU	61,393	720,406	11.73	1,265	2,252	4,675	2.70	8.00
Novelty	20,065	330,262	16.46	776	1,282	2,335	2.50	6.19
R-Novelty	7,109	97,543	13.72	739	1,282	2,142	2.62	6.01

er algorithms'. Note that for Novelty, a similar situation was observed in the case of Random-3-SAT — where, however, neither WalkSAT/TABU, nor R-Novelty showed any signs of essential incompleteness.

5.3. Blocks World and Logistics Planning

After evaluating the different algorithms on random distributions of hard problem instances, we now turn to single SAT-encoded instances from the Blocks World and Logistics Planning domains. Here, we compare the RLDs for individual instances, like it was done for the easy, medium, and hard instances from the distributions studied before. After preliminary experiments indicated that the GSAT variants are substantially outperformed by the more efficient WalkSAT variants on these problems (see also Section 5.5), we focused mainly on algorithms of the WalkSAT family.

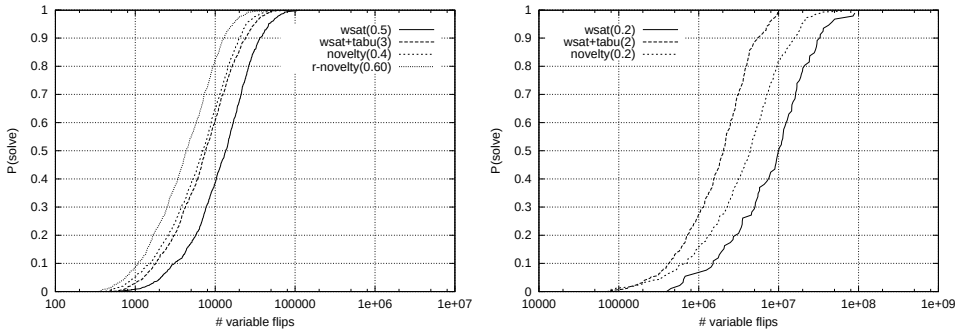

Figure 10. Problem instance `bw_large.a` *(left)* and instance `bw_large.c` *(right)* for various algorithms, approx. optimal noise, based on 1,000 and 250 tries, respectively.

Table III. Problem instance `bw_large.b`, basic descriptive statistics of RLDs for various algorithms, approx. optimal noise, based on 250 tries.

algorithm	mean	stddev	$\frac{\text{stddev}}{\text{mean}}$	median	Q_{25}	Q_{75}	$\frac{Q_{75}}{Q_{25}}$
WalkSAT(0.35)	582,384	570,626	0.98	414,744	166,326	782,842	4.71
WalkSAT/TABU(2)	152,104.03	137,108.76	0.90	107,586	54,611	215,564	3.95
Novelty(0.30)	191,715	195,045	1.02	131,494	53,629	254,779	4.75
R-Novelty(0.55)	234,304.18	215,100.55	0.92	166,922	73,673	341,827	4.64

For the smallest Blocks World instance (`bw_large.a`), as well as for similar small instances from the SATPLAN distribution, we observe a uniform performance pattern: R-Novelty is uniformly better than Novelty, which shows slightly improved performance over WalkSAT/TABU, which in turn is significantly better than WalkSAT (see Figure 10 on the left side; numerical data is given in Appendix B, Table XXVI). For the larger instances, however, the situation is considerably different: Applied to `bw_large.b`, R-Novelty ranks only third behind WalkSAT/TABU and Novelty, which show a very similar performance (see Table III). For `bw_large.c`, WalkSAT/TABU is significantly better than Novelty (factor ≈ 2), while Novelty's performance is between WalkSAT's and WalkSAT/TABU's (see Figure 10 on the right side; numerical data is given in Appendix B, Table XXVII). The big surprise, however, is that R-Novelty performs extremely poorly: for noise parameter settings up to 0.5, we observed stagnation behaviour with extremely low asymptotic solution probabilities ($< 10\%$). However, if a solution is found, this requires only a relatively small number of steps on average – thus by using restart and a small cutoff value, competitive performance can be achieved. Nevertheless, even using an optimal cutoff value, R-Novelty's performance is still inferior to WalkSAT/TABU's (the best estimated factor between the corresponding average local search steps / solution values is 1.75).

Table IV. Problem instance `logistics.a`, basic descriptive statistics of RLDs for various algorithms, approx. optimal noise, based on 1,000 tries for algorithms of WalkSAT-architecture and 250 tries for algorithms of GSAT architecture.

algorithm	mean	stddev	$\frac{\text{stddev}}{\text{mean}}$	median	Q_{75}	Q_{90}	$\frac{Q_{75}}{Q_{25}}$	$\frac{Q_{90}}{Q_{10}}$
GWSAT(0.30)	711,033	682,080	0.96	500,970	904,361	1,464,615	3.79	9.95
HWSAT(0.15)	107,143	76,633	0.72	86.609	148,109	213,285	2.96	6.75
WalkSAT(0.3)	95,205	91070	0.96	65,566	120,207	200,325	3.33	9.92
WalkSAT/TABU(1)	177,169	139,661	0.79	139,647	246,933	365,434	3.49	10.55
Novelty(0.45)	55,748	41,323	0.74	46,366	72,153	108,518	2.81	6.53
R-Novelty(0.65)	45,220	36,219	0.80	33,713	62,321	93,278	3.18	8.32

Table V. Problem instance `logistics.d`, basic descriptive statistics of RLDs for various algorithms, approx. optimal noise, based on 1,000 tries.

algorithm	mean	stddev	$\frac{\text{stddev}}{\text{mean}}$	median	Q_{75}	Q_{90}	$\frac{Q_{75}}{Q_{25}}$	$\frac{Q_{90}}{Q_{10}}$
WalkSAT(0.4)	499,956	373,354	0.75	398,277	641,438	972,765	2.56	6.29
WalkSAT/TABU(2)	400,005	276,011	0.69	332,494	526,125	750,234	2.61	5.77
Novelty(0.35)	134,695	96,288	0.71	109,297	174,384	254,862	2.58	5.91
R-Novelty(0.6)	936,537	885,835	0.95	671,914	1,251,464	2,081,376	3.88	12.57

On the SAT-encoded instances from the Logistics Planning domain, similar to the blocks world instances, none of our algorithms dominates all the others. For example, on the smallest instance (`logistics.a`, see Table IV), Novelty and R-Novelty are the best performing algorithms, followed by WalkSAT and HWSAT which take roughly twice as many steps. WalkSAT/TABU is the worst algorithm of the WalkSAT-architecture and GWSAT is quite significantly outperformed by all the others, requiring – on average – roughly 15 times as many flips as the best performing algorithm. We were not able to solve this instance with GSAT/TABU trying various values for the tabu tenure. For instances `logistics.b` and `logistics.c` (not shown here), the performance ranking of the algorithms is the same as for `logistics.a`. Yet, for instance `logistics.d`, R-Novelty performs substantially worse than all other WalkSAT variants, while Novelty shows the best performance followed by WalkSAT/TABU and WalkSAT (see Table V).

5.4. Instances from the DIMACS Benchmark Set

Finally, we present the comparison of the SLS algorithms on instances of the DIMACS Benchmark set. We started with comparing the algorithms' performance on the large SAT-encoded graph coloring instances. In Table VI we give the computational results on instance `g125.18`. All the algorithms we applied solve this

Table VI. Problem instance `g125.18`, basic descriptive statistics of RLDs for various algorithms, approx. optimal noise, based on 1,000 tries.

algorithm	mean	stddev	$\frac{\text{stddev}}{\text{mean}}$	median	Q_{75}	Q_{90}	$\frac{Q_{75}}{Q_{25}}$	$\frac{Q_{90}}{Q_{10}}$
GSAT/TABU(20)	7,799	5,168	0.66	6,466	10,087	13,874	2.39	4.52
HWSAT(0.15)	12,442	9,591	0.77	9,566	16,281	24,510	2.89	6.76
WalkSAT(0.15)	44,923	34,731	0.77	34,712	59,574	87,705	2.89	6.75
WalkSAT/TABU(2)	21,215	71,263	3.36	13,856	20,653	29,786	2.30	4.78
Novelty(0.45)	9,304	31,989	3.43	7,125	10,124	14,873	2.17	4.36
R-Novelty(0.2)	27,501	138,038	5.02	6,933	10,220	15,353	2.18	4.32

Table VII. Problem instance `g125.17`, basic descriptive statistics of RLDs for various algorithms, approx. optimal noise, based on $\geq$ 100 tries per instance.

algorithm	mean	stddev	$\frac{\text{stddev}}{\text{mean}}$	median	Q_{75}	Q_{90}	$\frac{Q_{75}}{Q_{25}}$	$\frac{Q_{90}}{Q_{10}}$
GSAT/TABU(20)	$1.21 \cdot 10^6$	$1.26 \cdot 10^6$	1.04	$0.76 \cdot 10^6$	$1.53 \cdot 10^6$	$3.06 \cdot 10^6$	5.03	29.27
WalkSAT(0.15)	$5.74 \cdot 10^6$	$5.52 \cdot 10^6$	0.77	$4.11 \cdot 10^6$	$8.29 \cdot 10^6$	$12.53 \cdot 10^6$	4.19	17.79
WalkSAT/TABU(1)	$2.94 \cdot 10^6$	$14.20 \cdot 10^6$	3.36	$0.64 \cdot 10^6$	$1.20 \cdot 10^6$	$2.01 \cdot 10^6$	2.73	7.87
Novelty(0.25)	$1.87 \cdot 10^6$	$10.10 \cdot 10^6$	5.38	$0.64 \cdot 10^6$	$1.14 \cdot 10^6$	$1.95 \cdot 10^6$	3.89	15.43

instance in relatively few steps; consistently with the results for the smaller, randomly generated graph colouring instances, GSAT/TABU and Novelty show the best performance. Novelty, however, as well as WalkSAT/TABU and R-Novelty show stagnation behaviour and hence do not find solutions in every run, even when using extremely high *maxSteps* settings (here: 10^6 steps). Of these, R-Novelty is affected worst which is reflected by the high standard deviation of the run length (see Table VI). A similar situation can be found for instance `g125.17` (corresponding to the same graph but using only 17 instead of 18 colours), which is considerably harder to solve. On this instance, R-Novelty (not shown in the table) suffered from extreme stagnation behaviour.

Next, we did an analogous performance analysis for the DIMACS parity instances mentioned in Section 4.4. In Table VIII we give the basic descriptive statistics for instance `par8-5-c`, which we found to be the hardest for the SLS algorithms considered here. Here, R-Novelty and Novelty are the top performing algorithms, followed by the two other WalkSAT variants and the algorithms based on the GSAT-architecture. Of these latter, HWSAT shows the the best performance and GSAT/TABU the worst. On the other parity instances, the relative ranking of the algorithms has been found to be identical to the one on this instance.

Table VIII. Problem instance par8-5-c, basic descriptive statistics of RLDs for various algorithms, approx. optimal noise, based on 1,000 tries.

algorithm	mean	stddev	$\frac{\text{stddev}}{\text{mean}}$	median	Q_{75}	Q_{90}	$\frac{Q_{75}}{Q_{25}}$	$\frac{Q_{90}}{Q_{10}}$
GWSAT(0.6)	27,143	44,044	1.62	18,634	37,812	63,169	4.65	21.45
GSAT/TABU(25)	62,687	80,268	1.28	45,027	88,950	146,843	5.42	25.58
HWSAT(0.45)	18,631	19,066	1.02	12,538	26,263	43,190	5.08	24.33
WalkSAT(0.2)	19,182	17,995	0.77	13,345	25,737	43,693	3.78	14.18
WalkSAT/TABU(7)	12,338	12,654	1.03	8,388	17,017	28,233	4.78	25.01
Novelty(0.80)	4,273	4,287	1.00	2,879	5,916	9,659	4.92	18.65
R-Novelty(0.85)	4,052	4,149	1.01	2,711	5,643	9,546	5.28	20.22

5.5. ABSOLUTE CPU-TIMES

In the comparative analysis above, we measured and reported the run-time in terms of variable flips, which implicitely suggests a uniform cost model for all algorithms. In practice, however, this assumption does not hold. For our experiments, we were using the highly optimised GSAT and WalkSAT implementations by Henry Kautz and Bart Selman, which we extended to implement HWSAT. Comparing the absolute CPU-time per local search step (i.e., variable flip) on a PC System with a single 400 MHz Pentium II CPU and 256 MB RAM running Redhat Linux 5.2, we found that for all benchmark instances the GSAT variants were consistently slower (in terms of variable flips per CPU second) than the WalkSAT variants by a factor of at least 3.

Table IX shows the typical numbers of flips for a selection of instances from our benchmark suite. Note that for structured problems, the observed differences are usually greater; also, the difference between GSAT and WalkSAT variants typically increases with problem size. Under the assumption that both implementations are optimised to approximately the same degree, this implies that for our benchmark suite the best WalkSAT variants would consistently outperform any GSAT variant considered here. In particular, this applies to the cases where the best GSAT variant (typically GSAT/TABU) was found to be competitive with the best WalkSAT variant (as for the Graph Colouring instances) when counting local search steps. This can be explained by the fact that different from GSAT, WalkSAT uses a two-stage variable selection scheme which means that in each search step, only the consequences of flipping a typically small number of variables occuring in one particular unsatisfied have to be considered.

It is also worth noting that for the Random-3-SAT and Graph Colouring instances, the number of flips per CPU second is almost unaffected by increasing the problem size. The reason for this is the fact that for these formulae, the connectivity of the constraint graph (in which variables correspond to nodes and edges connect variables which occur in a same clause) as well as the length (number of

Table IX. Absolute CPU timing for different algorithms and benchmark instances; all timings were measured on a 400 MHz Pentium II CPU.

instance	algorithm	flips per sec	instance	algorithm	flips per sec
uf100-430	GWSAT	38,400	uf250-1065	GWSAT	31,300
	GSAT/TABU	56,100		GSAT/TABU	60,100
	WalkSAT	248,000		WalkSAT	223,000
	R-Novelty	214,000		R-Novelty	191,000
flat100-239	GWSAT	48,300	flat200-479	GWSAT	42,900
	GSAT/TABU	44,100		GSAT/TABU	33,000
	WalkSAT	389,000		WalkSAT	378,000
	R-Novelty	320,000		R-Novelty	318,000
bw_large.a	GWSAT	21,500	bw_large.c	GWSAT	7,810
	GSAT/TABU	26,100		GSAT/TABU	16,100
	WalkSAT	170,000		WalkSAT	76,500
	R-Novelty	145,000		R-Novelty	74,000

literals) of the clauses are independent of the problem size. In the most efficient implementations, these two factors dominate the CPU time per variable flip.

In summary, our comparative analysis of different SLS algorithms' performance across various problem domains shows that there is no single best algorithm for all domains. This is an interesting and new result because previously, a comprehensive study of the most recent and best performing GSAT and WalkSAT variants was not available. In particular, while the more limited results in [51] suggested that R-Novelty might be the overall best performing SLS algorithm for SAT, our results show clearly that this is not the case. We observed a tendency indicating that Novelty and R-Novelty are performing best for random, unstructured problems, but WalkSAT/TABU and GSAT/TABU are competive or better (when run-time is measured in terms of variable flips) on some of the relatively hard, large, structured problem instances like the large Blocks World Planning and Graph Colouring instances. However, all of these algorithms potentially suffer from stagnation behaviour which, in some cases, severely compromises their performance. Additionally, given the currently best known implementations for these algorithms, GSAT/TABU is never competitive with the best-performing WalkSAT variants in terms of absolute CPU time.

6. Robustness and Scaling

Up to this point, our analysis was mainly focused on peak performance; like most other studies in this field we analysed and reported results for optimal parameter

settings (such as the cutoff time *maxSteps* or the noise parameters *wp*, *p*, or *tl*). But in practice, optimal parameter settings are often not known a priori, and as we have seen in the last section, for the same algorithm they may differ considerably between problem instances or domains. Therefore, robustness of an SLS algorithm *w.r.t.* suboptimal parameter settings is an important issue. At least of equal importance are the scaling aspects of SLS behaviour. These comprise scaling *w.r.t.* problem size, scaling *w.r.t.* constrainedness, and scaling *w.r.t.* the mean hardness of instances within test-sets sampled from problem distributions.

Of these, scaling with problem size is at least implicitly covered to some degree in many experimental studies in the sense that problem instances or test-sets for different problem sizes are used; but only a few studies, such as [19], investigate and model the scaling behaviour of SLS algorithms for SAT in more detail. The influence of constrainedness on SLS performance has been investigated in great detail in a number of studies, especially in the context of phase-transition phenomena (e.g., [19, 52]). Finally, as we will show in this section, the performance of the different SLS algorithms studied here is typically tightly correlated within test-sets, i.e., the same instances tend to be relatively hard for all algorithms. This suggests that the relative hardness of problem instances within the same test-set of, e.g., Random-3-SAT instances, is an intrinsic feature of the problem instance. Consequently, the distribution of the mean performance of an SLS algorithm over such a test-set can be viewed as a kind of scaling *w.r.t.* this intrinsic hardness.

Scaling dependencies and robustness are related in that the former can be interpreted as robustness of an algorithm's performance *w.r.t.* problem instance or domain specific properties. In this section we study the robustness and the scaling behaviour of SLS algorithms *w.r.t.* problem size and instance hardness, building on the methodology used in the previous section.

6.1. Robustness *w.r.t.* the cutoff parameter

As almost all SLS algorithms for SAT, and in particular all algorithms considered here, use a restart mechanism and therefore need the cutoff parameter *maxSteps* to be specified, robustness *w.r.t.* to this parameter is generally an important issue. Obviously, for a given algorithm and problem instance, the robustness *w.r.t.* the cutoff parameter is tightly related to the run-time distribution. Specifically, if the algorithm (without random restart) is essentially incomplete and stagnation occurs, the performance critically relies on reasonably small settings for the cutoff parameter. But good *maxSteps* settings are extremely difficult to find *a priori* and currently, to our best knowledge, there exist no theoretical results on how to effectively determine good settings. The only empirical results we are aware of, are for GSAT when applied to hard Random-3-SAT problem distributions [22]; but these results are limited to Random-3-SAT and rely on properties of the respective problem distributions rather than the individual instances. On the other hand, while using random restart with inappropriately chosen *maxSteps* settings theoretically

eliminates essential incompleteness, in practice it leads to extremely poor performance. Thus, for obtaining a reasonable robustness *w.r.t.* the *maxSteps* parameter setting, an SLS algorithm should have the PAC property.

In [35, 34], the following results regarding the essential incompleteness or the PAC property of the algorithms considered here are established:

- plain GSAT (= GWSAT with *wp=0*) is essentially incomplete.
- GWSAT is PAC for arbitrary $wp > 0$;
- WalkSAT/TABU is essentially incomplete for arbitrary *tl*;
- Novelty is essentially incomplete for arbitrary *p*;
- R-Novelty is essentially incomplete for arbitrary *p*;

Furthermore, it is easy to see the following:

- HSAT is essentially incomplete;
- HWSAT is PAC for arbitrary $wp > 0$.

Unfortunately, it has not been proven whether or not WalkSAT is PAC; the same holds for GSAT/TABU. However, our experimental analysis suggests that WalkSAT might be PAC for $p > 0$, as in thousands of experiments, we never observer any indication of stagnation behaviour. Furthermore, our results indicate that GSAT/TABU is probably essentially incomplete at least for sufficiently low tabu tenure. It should be noted that the theoretical results mentioned above do not necessarily imply that the performance of the algorithms are affected in practice. In particular, the two following questions have to be answered:

1. For PAC algorithms, is the convergence of the success probability fast enough to be observable when applying these algorithms to hard problem instances?
2. For essentially incomplete algorithms, is the characteristic stagnation behaviour observable for any of the common benchmarks for SAT algorithms?

Considering the results reported in Section 5, both questions can be answered positively: PAC behaviour as well as essential incompleteness is reflected in the presence or absence of stagnation behaviour observed in the RTDs of the algorithms under considerations when applied to our benchmark problems.

Given this situation and the fact that with Novelty and R-Novelty, two of the overall best performing algorithms are affected by the adverse effects of essential incompleteness, it seems to be very desirable to modify these algorithms in such a way that their overall superior performance is retained while their essential incompleteness is removed in a way which makes them significantly more robust *w.r.t.* the *maxSteps* parameter setting.

Fortunately, as demonstrated in [35, 34], this can be done in a rather simple and uniform way by extending them with random walk as used in GWSAT in such a way that for each local search step, with a fixed probability a random walk step is performed. Note that for any algorithm using this mechanism, arbitrarily long sequences of random walk steps can occur, which directly implies the PAC property [35, 34]. Furthermore, the amount of perturbation introduced by a random walk sequence probabilistically depends on the length of the sequence such that small perturbations are much more likely to occur than large ones which, intuitively, should make the algorithms more robust than techniques like the deterministic loop-breaker in R-Novelty (see Section 2.2.4). Based on these considerations, we extended Novelty and R-Novelty with random walk such that in each search step, with probability *wp*, the variable to be flipped is randomly picked from the selected clause, while in the remaining cases, the variable is selected according to the heuristic for Novelty or R-Novelty, respectively. Obviously, WalkSAT/TABU can be analogously extended. For R-Novelty, we furthermore omit the deterministic loop breaking strategy which randomly flips a variable from the selected clause every 100 steps. The two algorithms thus obtained, Novelty$^+$ and R-Novelty$^+$, are obviously PAC as argued above. Furthermore, testing their performance against Novelty and R-Novelty on instances from our benchmark set shows that the stagnation behaviour observed for Novelty and R-Novelty is effectively eliminated even when using a fixed setting of 0.01 for the newly introduced parameter *wp*. Consequently, the new variants show a superior performance when applied to instances in which the original algorithms suffered from the effects of their essential incompleteness as can be seen from Tables X and XI.

Note, that in cases where the original algorithm did not show stagnation behaviour, the performance is not negatively affected by adding random walk, a fact which we verified in all the experiments we have run. When evaluating the performance of Novelty$^+$ and R-Novelty$^+$, we generally used a fixed setting of $wp = 0.01$. By optimising this additional parameter, further improvements in performance and robustness can be achieved.

While, as we have seen, the PAC property is of considerable theoretical interest and also affects the robustness of SLS algorithms in practice, a more detailed analysis has to focus on the overall shape of the RTDs, which is the factor which ultimately and uniquely determines the robustness of an algorithm *w.r.t.* the cutoff parameter. As pointed out in [36], the maximal robustness *w.r.t.* the cutoff parameter is encountered for exponential run-time distributions. To see this, consider an exponential RTD characterised by the cumulative distribution function $ed[m](x) = 1 - 2^{-x/m}$. Obviously, the probability for finding a solution within the first k steps is $ed[m](k) = 1 - 2^{-k/m}$. Now, if already j search steps have been done without success, using complementary probabilities and Bayes' Rule gives the probability for finding a solution within the next k steps as:

$$1 - \frac{1 - ed[m](j+k)}{1 - ed[m](j)} = 1 - \frac{2^{j+k}}{2^j} = ed[m](k)$$

Thus, an SLS algorithm characterised by an exponential RTD is memoryless in the sense that the probability of finding a solution within a fixed number of local search steps does not depend on the number of search steps done in the past. Consequently, as also argued in [36, 35, 39], in case of an exponential RTD, the probability to find a solution within a given time would not depend on the setting of the *maxSteps* parameter and thus, the number of random restarts.

It can be shown that for approximately optimal noise parameter settings, all GSAT and WalkSAT variants considered here show exponential RTDs when applied to hard problem instances [35, 36, 37, 39]. Here, "hard instance" refers to a problem instance which is relatively hard compared to instances of comparable size and structure, such as the 25% fraction of a Random-3-SAT test-set for which the mean of the RLD for a given algorithm is highest. Note that the claim is of such a nature that it can easily be tested, especially for the test-sets sampled from random instance distributions used in this study.

We systematically tested all algorithms considered here for the problem sets `uf100-430`, `flat100-239`, as well as for the individual planning and DIMACS instances. This analysis was based on the same RLDs used in Section 5. For each individual instance and algorithm, the RLD was approximated[3] using the cumulative form of an exponential distribution $ed[m](x) = 1 - 2^{-x/m}$ where m is the median of the distribution and x the number of steps required to find a solution.[4] For testing the goodness of this approximation we use a standard χ^2-test [57]. Basically, for a given empirical RLD this is done by estimating the parameter m and comparing the deviations to the predicted distribution $ed[m]$. The result of this comparison is the χ^2 value, where χ^2 values below a critical threshold indicate a close correspondence between empirical and predicted distribution. For the results reported here, we used the acceptance thresholds corresponding to a maximal probability of 0.05 or 0.01 of false negatives for the test (these values are commonly used in the statistical literature).

Figure 11 shows a scatter plot of the correlation between the median search cost per solution and the χ^2 values for WalkSAT applied to test-set `uf100-430`. Obviously, there is a strong negative correlation, indicating that, indeed, for harder problem instances, WalkSAT's behaviour can be more and more accurately characterised by exponential distributions. As can be seen from the plot, for high median search cost, almost all instances pass the χ^2 test. Table XII shows the

[3] All approximations were done using C. Gramme's "Gnufit" implementation of the Marquart-Levenberg algorithm.

[4] In the statistical literature, the exponential distribution $Exp(\lambda)$ is usually defined by $P(X \leq x) = 1 - e^{-\lambda x}$, which is equivalent to our representation $ed[m]$ using $m = \ln 2/\lambda$. A similar argument applies to the Weibull distribution mentioned later.

Table X. Performance comparison for Novelty and R-Novelty (approx. optimal noise) *vs.* Novelty$^+$ and R-Novelty$^+$ for Random-3-SAT and Graph Colouring test-sets. The reported basic descriptive statistics refer to the hardness distributions (mean number of steps / solution) over the test-sets.

test-set	algorithm	mean	stddev	$\frac{\text{stddev}}{\text{mean}}$	median	q_{25}	q_{75}	q_{10}	q_{90}
uf100-430	Novelty(0.6)	28,257	191,668	6.78	851	479	1,845	302	4,390
	Novelty+(0.6)	1,570	2,802	1.78	801	467	1,663	288	3,049
flat50-115	R-Novelty(0.6)	7,109	97,543	13.72	739	490	1,282	356	2,142
	R-Novelty+(0.6)	1,084	1,053	0.97	747	486	1,245	358	2,167

Table XI. Performance comparison for Novelty and R-Novelty (approx. optimal noise) *vs.* Novelty$^+$ and R-Novelty$^+$ for Blocks World Planning and Graph Colouring instances from the DIMACS benchmark set; $\widehat{p_s}^*$ indicates the asymptotic maximal success probability, $\widehat{E_s}$ denotes the expected number of steps for finding a solution (using random restart); the reported basic descriptive statistics refer to the corresponding conditional RLDs.

instance	algorithm	$\widehat{p_s}^*$	*ms*	$\widehat{E}_s$	mean$_c$	stddev$_c$	median$_c$
bw_large.c	R-Novelty(0.3)	0.028	10^8	$3.47 \cdot 10^9$	169,810	157,752	32,334
	R-Novelty+(0.3)	1.0	10^8	$8.09 \cdot 10^6$	8,086,468	8,414,928	5,292,830
g125.18	R-Novelty(0.2)	0.981	10^7	27,501	8133	5107	6,866
	R-Novelty+(0.2)	1.0	10^7	10,012	10,012	27,325	7,115
g250.29	R-Novelty(0.8)	0.904	$2.5 \cdot 10^8$	1,376,845	359,146	454,448	263,995
	R-Novelty+(0.8)	1.0	$2.5 \cdot 10^8$	501,464	501,464	1,313,066	280,333

Table XII. Fraction of instances from the hardest 25% of the `uf100-430` Random-3-SAT test-set passing the χ^2 test for different algorithms with approximately optimal noise settings; the last column indicates the number of instances for which essentially incomplete behaviour was observed, these were removed from the test-set.

algorithm	acceptance level α	fraction passed	number removed
WalkSAT(0.55)	0.01	57.6%	0
WalkSAT/TABU(3)	0.01	49.6%	1
Novelty(0.7)	0.01	57.2%	35
R-Novelty(0.7)	0.01	49.6%	0

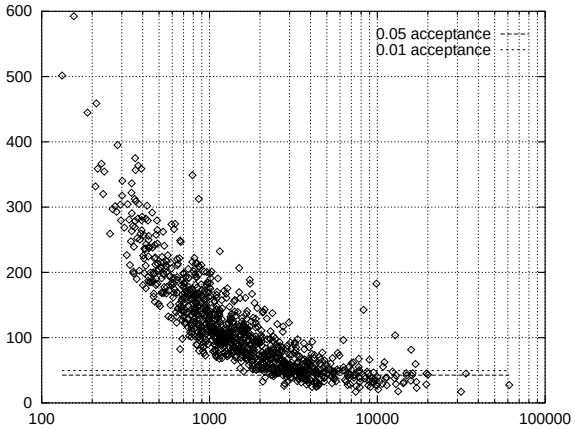

Figure 11. Correlation between hardness of problems (horizontal) and χ^2 values (vertical) from testing the RLDs of individual instances versus a best-fit exponential distribution for test-set `uf100-430`. The horizontal lines indicate the acceptance thresholds for the 0.01 and 0.05 acceptance level.

Table XIII. Fraction of instances from the hardest 25% of the `flat100-239` Graph Colouring test-set passing the χ^2 test for different algorithms with approximately optimal noise settings; the last column indicates the number of instances for which essentially incomplete behaviour was observed, these were removed from the test-set.

algorithm	acceptance level α	fraction passed	number removed
WalkSAT(0.5)	0.01	52%	0
WalkSAT/TABU(3)	0.01	64%	1
Novelty(0.6)	0.01	84%	1
R-Novelty(0.6)	0.01	100%	1

overall percentage of the instances from the hardest quarter of the test-set which passed the test for the different acceptance levels when performing this analysis for different WalkSAT variants. As indicated in Table XIII, analogous results were obtained for the `flat100-239` Graph Colouring test-set; the results for the GSAT variants (not reported here) are very similar. When applying the same analysis for different problem sizes, it can furthermore be noted that a relatively higher number of instances pass the test, i.e., the deviations of the RLDs from ideal exponential distributions apparently become less prominent for larger problems (for details see [35]).

For the single instances from various SAT-encoded problems we performed the same analysis. The results are consistent with those for the Random-3-SAT and Graph Colouring test-sets: For the harder instances, the RLDs are well approximated by exponential distributions (see Table XIV). The only exception is instance

Table XIV. RLD approximations using exponential distributions $ed[m]$ for Novelty (approximately optimal noise) applied to Blocks World Planning instances and an instance for learning the parity function; for the Graph Colouring instances we applied Novelty+ since Novelty shows stagnation behaviour on these instances. The last column indicates whether the approximation passed the χ^2 test.

instance	median #steps m	χ^2 for ed$[m]$	ν	passed (α)
bw_large.a	6,839	60.37	29	no
bw_large.b	119,680	11.40	16	yes (0.05)
bw_large.c	$4.27 \cdot 10^6$	8.71	16	yes (0.05)
bw_large.d	$8.51 \cdot 10^6$	10.14	10	yes (0.05)
par8-5-c	4,273	29.83	29	yes (0.05)
g125.18	8,558	579.24	29	no
g125.17	1,151,392	29.15	16	yes (0.01)

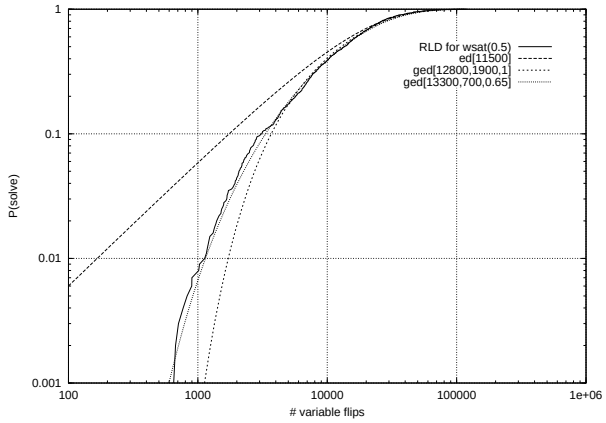

Figure 12. RLD for WalkSAT (approx. optimal noise) applied to Blocks World instance `bw_large.a` and three functional approximations.

`g125.18` which corresponds to finding a suboptimal colouring of a 125 nodes graph. For this relatively easy instance (considering its size) the test is clearly rejected. Yet, if for the same graph the conjectured optimal solution should be found (corresponding to instance `g125.17`) the χ^2 value is much smaller and, in fact, the test for an exponential distribution is passed at the 0.01 level.

Interestingly, exponential distributions are characteristic for the simplest randomised search technique: uniform random picking from a set of solution candidates. Thus, observing this type of behaviour for more sophisticated and much more efficient algorithms (such as the SLS algorithms studied here) suggests that their behaviour might be interpreted as random sampling from a much smaller

space. Intuitively, this hypothetical "virtual searchspace" should be linked to features of the actual search space, such as the number, size, and topology of local optima. Unfortunately, the existing results on the topology of search spaces for SLS algorithms cannot be exploited to establish such a link [22, 64, 15, 35, 34]. While further investigating of this issue is beyond the scope of this paper, we are confident that it can be adequately addressed by combining the methodology for characterising SLS performance applied in this study with established methodology for analysing search space topology.

Note that for relatively easy instances the exponential approximations of the RLDs are rejected by the χ^2 test. Further analysis shows that in these cases typically the tail of the RLDs is still well approximated by an exponential distribution $ed[m]$, while for shorter run-times, the success probability is systematically lower than given by $ed[m]$. Figure 12 illustrates this situation for a typical example. Intuitively, this can be explained by the fact that SLS algorithms typically go through an initial search phase (cf. [22]), during which the probability of finding a solution is rather low. For hard instances, the successful runs are long enough that the impact of this initial search phase is only minimal, whereas for easy instances, it is clearly observable from the RLD.

Based on RLD data, the effect of this initial local search phase can be modeled using a slightly generalised family of distributions. Such a distribution has been introduced in [35]; it is based on the family of Weibull distributions[5] $wd[m,\beta](x) = 1-2^{-(x/m)^{\beta}}$, a well-known generalisation of the exponential distribution. The new model is defined by the following cumulative distribution function:

$$ged[m,\gamma,\delta](x) = wd[m,1+(\gamma/x)^{\delta}](x) = 1-2^{-(x/m)^{1+(\gamma/x)^{\delta}}}$$

Although the full defining term looks quite complicated, the definition simply reflects the idea of a Weibull distribution with a dynamically changing β parameter. More precisely, the new distribution is obtained from a Weibull distribution by introducing a hyperbolically decaying β parameter. Like for the exponential and Weibull distributions, m is the median of the distribution. The two remaining parameters intuitively correspond to the length of the initial search phase (γ) and to its impact on the overall behaviour (δ). High γ values indicate a long initial search phase, while high δ values are used to model a strong influence of the initial search phase.

Note that the exponential distribution is a special case of this new class of distributions, since $ed[m] = ged[m,0,\delta]$, while generally, Weibull distributions cannot be represented within this family. As can be easily seen from the definition, $ged[m,\gamma,\delta](x)$ asymptotically approaches an exponential distribution $ed[m](x)$ for large x. For decreasing x, however, the relative difference between $ged[m,\gamma,\delta](x)$

[5] Weibull distributions are used in reliability theory to model failure rates in systems which are subject to aging phenomena [6].

and $ed[m](x)$ increases monotonically, which is qualitatively exactly what we observed for the empirical RLDs of SLS algorithms (see also Figure 12).

Using this model, the RLDs measured for the SLS algorithms considered here can be modeled in great detail. Performing an analogous analysis as above, only now using *ged* approximations instead of *ed* approximations, the fraction of instances from test-sets like `uf100-430` or `flat100-239` which pass the χ^2 test increases to ca. 80% (for details, see [35]). At the same time, the correlation between instance hardness and goodness-of-fit cannot be observed any longer — which indicates that indeed the initial search phase is correctly modeled. In the relatively few remaining cases, where even these approximations are not sufficient, we generally found multi-modal RLDs which can be well approximated by weighted linear combinations of *ged*-type distributions. Intuitively, these instances seem to contain "traps", i.e., singular, highly attractive regions of the search space which are hard to escape from. Clearly, further investigation is required for ultimately explaining this phenomenon, and the RLD-based approach, combined with advanced methods for search space analysis as proposed in [15, 35, 64] provides a good basis for such an endeavour.

6.2. Robustness *w.r.t.* the noise parameter

The noise parameter is more specific to the given algorithm than the cutoff parameter; e.g., the *wp* parameter used for the algorithms utilising random walk steps (such as GWSAT or HWSAT) is conceptually quite different from the tabu tenure parameter controlling the behaviour of SLS algorithms incorporating tabu search. Nevertheless, as emphasised in [51], the effect of high noise settings (high *wp*, *p*, *tl* parameter values) is generally similar: the algorithm's ability to escape from local minima increases. On the other hand it might be expected that this comes at a price, as high noise settings also decrease the probability for an algorithm to rapidly descend into local minima which are solutions.

Here, we investigate the robustness of SLS algorithms *w.r.t.* the setting of the noise parameter. It should be noted that according to a result by Selman, Kautz, and McAllester [51], the optimal noise setting seems to be closely related to a simple statistical property of the search trajectory. However, it is not clear whether this result is accurate enough to provide a basis for automatically and dynamically determining the approximately optimal noise parameter setting. To our best knowledge, as of this writing there exists no successful GSAT or WalkSAT variant with self-adjusting noise. Therefore, evaluating the sensitivity of the SLS algorithms considered here *w.r.t.* non-optimal noise settings seems to be a reasonable endeavour.

Based on our results on the functional characterisation of RLDs, it is an interesting question how these are affected by non-optimal noise settings. To investigate this question, we measured and analysed RLDs for the different algorithms when applied to the instances from our benchmark library. Here, we summarise the

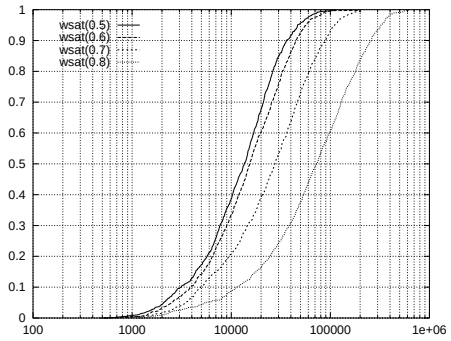

Figure 13. RLDs for WalkSAT applied to Blocks World Planning instance `bw_large.a`, using different noise parameter settings (0.5 is approximately optimal).

Table XV. $ed[m]$ approximations of RLDs for WalkSAT applied to Blocks World Planning instance `bw_large.a`, using optimal and larger-than-optimal noise parameter settings.

algorithm	median of RLD	m	χ^2 for ed$[m]$	passed (α)
WalkSAT(0.5)	13,505	13,094	98.41	no
WalkSAT(0.6)	15,339	16,605	76.17	no
WalkSAT(0.7)	27,488	27,485	42.13	yes (0.5)
WalkSAT(0.8)	72,571	74,289	20.78	yes (0.5)

corresponding results and give a few representative examples. Generally, we find that if the noise setting is increased above the optimal value, the RLDs can still be well approximated using exponential (or, when the initial search phase is modeled, *ged*) distributions. Thus, for larger-than-optimal noise settings, SLS performance is very robust *w.r.t.* the cutoff parameter. At the same time the run-times compared to optimal noise are uniformly higher for all success probabilities. This is reflected in larger values of the m parameter when fitting the exponential distributions and can also be easily seen in the semi-log plots of the corresponding RLDs, where for larger-than-optimal noise, the RLD curves have the same shape while being shifted to the right (see Figure 13 for a typical example). Furthermore, when increasing the noise, the effects of the initial search phase become less prominent — consequently, the approximations using exponential distributions are usually better for higher noise (see Table XV). The same effects can be observed when using other GSAT or WalkSAT variants or different benchmark instances.

As a second example, we report the results of analysing the effects of increasing the noise parameter for R-Novelty, applied to the Graph Colouring test-set `flat50-115`. Table XVI shows the fraction of instances from this test-set, for which a best-fit *ed* approximation passed the χ^2 test. As can be seen from this

Table XVI. Fraction of instances from the hardest 25% of the `flat50-115` Graph Colouring test-set passing the χ^2 test for Novelty with optimal and larger-than-optimal noise settings; instances, for which essentially incomplete behaviour was observed, were removed from the test-set.

algorithm	median hardness	acceptance level α	fraction passed
R-Novelty(0.6)	739	0.01	19.0%
R-Novelty(0.7)	832	0.01	23.8%
R-Novelty(0.8)	1043	0.01	40.7%

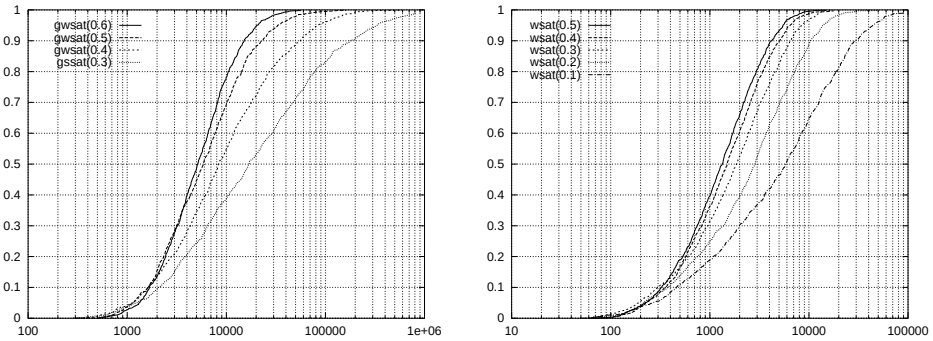

Figure 14. Left: RLDs for GWSAT applied to easy instance from Graph Colouring test-set `flat100-239`, using different noise parameter settings (0.6 is approx. optimal); *right:* same for WalkSAT applied to medium instance from Random-3-SAT test-set `uf100-430` (approx. optimal noise = 0.5).

data, the fraction of the hardest 25% of the test-set which passed the test increases for larger-than-optimal noise, as the effects of the initial search phase become less prominent. At the same time the median hardness (mean number of steps per solution) increases considerably. Analogous results were obtained for different SLS algorithms and other problem domains from our benchmark suite.

While for larger-than-optimal noise, the SLS performance decreases but the algorithms still show approximately exponential RTDs, for lower-than-optimal noise parameter settings we find a completely different situation: For all algorithms the performance decreases non-uniformly for different solution probabilities and some of them show stagnation behaviour. We demonstrate this here by giving a small number of examples, which are nevertheless typical for our overall observations. As shown in Figure 14, GSAT and WalkSAT show a similar behaviour for smaller-than-optimal noise: While for short runs, the performance even improves slightly, for longer runs and the corresponding higher success probabilities, the performance deteriorates considerably. Consequently, the RLDs are less steep than exponential distributions and suffer from an increasingly long and heavy tail. Both GSAT/TABU and R-Novelty show a similar, but additionally essential-

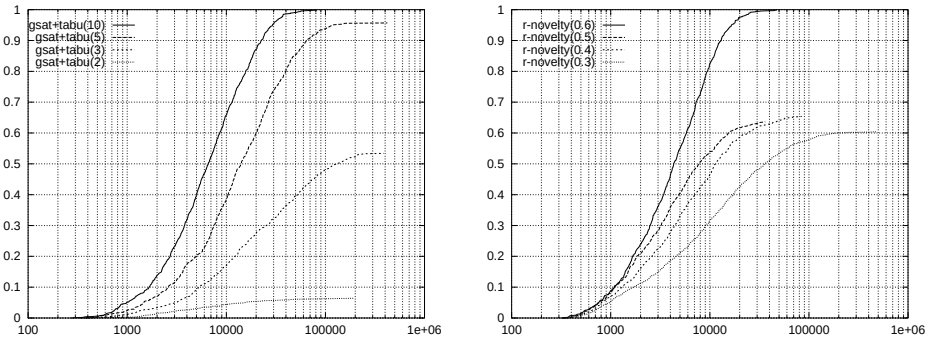

Figure 15. Left: RLDs for GSAT/TABU applied to medium instance from Graph Colouring test-set `flat100-239`, using different noise parameter settings (10 is approx. optimal); *right:* same for R-Novelty applied to Blocks World Planning instance `bw_large.a` (approx. optimal noise = 0.6).

ly incomplete behaviour; as can be seen from Figure 15, the maximal success probabilities for both algorithms are decreasing with the noise parameter. For Novelty, no increasingly essential incomplete behaviour is observed, but the performance also decays non-uniformly as for GSAT and WalkSAT. In comparison, WalkSAT/TABU seems to be relatively mildly affected; the main reason for this might be that typically, both the clause length and the optimal tabu-list-length are rather small.

Summarising these results, typically the detrimental effects of non-optimal noise parameter settings are much worse for lower-than-optimal noise settings than for larger-than-optimal noise. This is particularly the case for the higher percentiles of the corresponding RLDs; additionally, for essentially incomplete SLS algorithms, stagnation behaviour occurs more frequently with decreasing noise settings. However, the behaviour for very short runs is usually not affected by lower-than-optimal noise and sometimes, even performance improvements can be observed for the early phases of local search. This is consistent with the intuition that greedier local search behaviour should pay off during the initial search phase, where gradient descent dominates the search.

6.3. SCALING WITH INSTANCE HARDNESS

After discussing robustness *w.r.t.* the parameter settings of the algorithms, we now turn towards the influence of problem-specific properties on SLS performance. In Section 5, when comparing the distributions of mean local search cost over the instances of our Random-3-SAT and Graph Colouring test-sets we observed that these hardness distributions are mostly similarly shaped. This suggests that the hardness of instances for different algorithms might be tightly correlated. To test this hypothesis, we performed a pairwise hardness correlation analysis of the different algorithms across test-sets `uf100-430` and `flat50-235`. Like before, as a measure for hardness we chose the expected number of steps required for finding

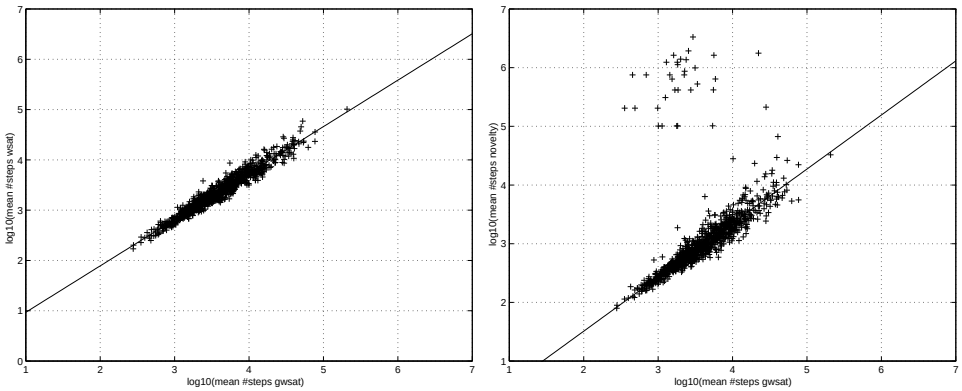

Figure 16. Left: Correlation between average local search cost for GWSAT (horizontal) and WalkSAT(vertical) for test-set `uf100-430`, using approx. optimal noise. *Right:* Same for GWSAT and Novelty. The outliers indicate instances on which Novelty showed stagnation behaviour.

Table XVII. Test-set `uf100-430`, pairwise hardness correlation for various algorithms with approx. optimal noise, based on 100 tries / instance; *r* is the correlation coefficient, *a* and *b* are the parameters of the *lms* regression analysis, and *o* the number of outliers which have been eliminated before the analysis (see text).

algorithms	*r*	*a*	*b*	*o*
GWSAT *vs.* GSAT/TABU	0.9171	1.1456	-0.8007	0
GWSAT *vs.* WalkSAT	0.9725	0.9229	0.0471	0
GWSAT *vs.* WalkSAT/TABU	0.9464	0.9962	-0.4814	0
GWSAT *vs.* Novelty	0.9431	0.9211	-0.3331	33
GWSAT *vs.* R-Novelty	0.9492	0.9044	-0.3628	0

a solution, based on 100 tries with a high cutoff value of 10^7 steps per try. In our analysis, we determined this hardness measure for each instance from the problem set and analysed the correlation between the hardness for different algorithms. To reduce the overall amount of computation, we chose GWSAT as a reference algorithm, i.e., for each other algorithm we analysed the correlation between its performance and GWSAT's on an instance per instance basis. The results of this analysis confirm the hypothesis that there is a tight correlation between the performance of any two algorithms over the test-sets, i.e., instances which are hard for one SLS algorithm tend to be hard for all the others. Figure 16 shows the correlation data for two pairs of algorithms applied to `uf100-430` as log-log scatter plots. These plots are typical for the correlation relations we observed.

To quantitatively characterise the strong correlations suggested by the scatter plots, we performed a correlation and least-mean-squares (*lms*) linear regression

Table XVIII. Test-set `flat50-115`, pairwise hardness correlation for various algorithms with approx. optimal noise, based on 100 tries / instance; *r* is the correlation coefficient, *a* and *b* are the parameters of the *lms* regression analysis, and *o* the number of outliers which have been eliminated before the analysis (see text).

algorithms	*r*	*a*	*b*	*o*
GWSAT *vs.* GSAT/TABU	0.9021	0.7673	0.0715	0
GWSAT *vs.* WalkSAT	0.9527	0.7738	0.6353	0
GWSAT *vs.* WalkSAT/TABU	0.8417	0.8885	-0.1149	27
GWSAT *vs.* Novelty	0.8398	0.8212	-0.1217	9
GWSAT *vs.* R-Novelty	0.8247	0.7848	-0.0205	6

analysis of the logarithm of the hardness (mean search cost per solution); the regression lines shown in the log-log plots (Figures 16) correspond to power functions of the type $y = x^a \cdot exp(b)$. The results of this analysis are reported in Tables XVII and XVIII. This data confirms that the correlation between the hardness for different algorithms is very strong; this holds also for algorithms which show stagnation behaviour for some instances, after these outliers have been removed from the test-set. This indicates that the hardness for any of these algorithms (neglecting the outliers) is an intrinsic property of the instances. As a possible explanation for this observation we assume that the same structural features render a particular instance hard for all SLS algorithms. From the regression data we see that the *a* parameter is relatively close to one for all algorithms; this means that there is only minor variation in the performance differences between the individual algorithms as the intrinsic hardness varies. Nevertheless, some systematic differences can be observed. In particular, for both test-sets, the tabu-variants show the lowest *b* value, indicating a superior performance on easy instances. However, except for GSAT/TABU on the Graph Colouring test-set,[6] they scale worse than the other variants as the intrinsic hardness increases. Also, we observe a consistent scaling advantage of R-Novelty over Novelty (comparing the *a* values), of Novelty over WalkSAT, and of WalkSAT over GWSAT.

Two more observations are worth noting. First, our data indicates that within a test-set, the hardness correlation gets noisier for harder problems. This holds especially for the more effective variants of WalkSAT, like WalkSAT/TABU or R-Novelty and suggests that for relatively hard instances, SLS procedures differ more in their effectiveness. Furthermore, the occurrence of stagnation behaviour, as can be seen from Figure 16, is not related to the intrinsic hardness of an instance, as

[6] Recall that for the instances considered here, GSAT/TABU is the best performing algorithm when counting variable flips; apparently it can exploit the specific structure induced by encoding Graph Colouring instances into SAT.

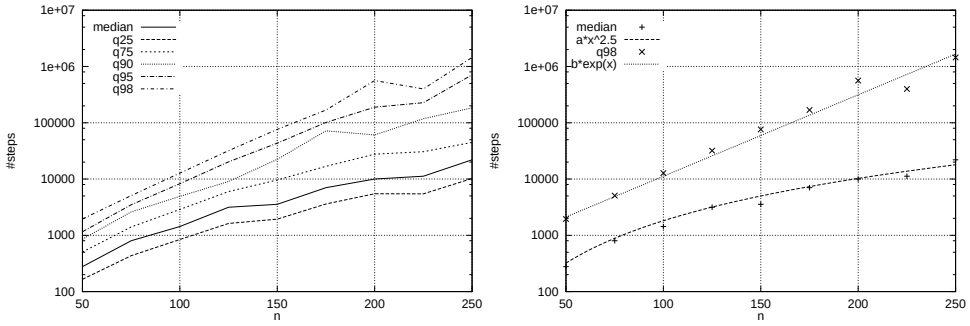

Figure 17. Left: Scaling of instance hardness with problem size for WalkSAT, approx. optimal noise, applied to Random-3-SAT test-sets. *Right:* Functional approximations of median and 0.98 percentile; the median seems to grow polynomially with n while the 0.98 percentile clearly shows exponential growth.

outliers occur for instances which are relatively easy for another algorithm as well as for intrinsically hard instances.

6.4. SCALING WITH PROBLEM SIZE

In a final analysis, we investigated the scaling of local search cost with problem size. First, we determined the hardness distributions for WalkSAT with approximately optimal noise parameter settings on the Random-3-SAT test-sets for $n = 50, 75, 100, ..., 250$ as described in Section 5. The scaling behaviour of the median and some higher percentiles thus determined is shown in Figure 17. The data clearly suggests, that the highest percentiles grow exponentially with the problem size; as shown on the left side of Figure 17, the 0.98 percentile can be well approximated with the exponential function $y = 400 \cdot exp(n/30)$. The lower percentiles, including the median, however, appear to be growing polynomially in n, the best approximation for the median was found as $y = 89.44 \cdot (n/30)^{2.5}$. Note however, that in both cases for small n, the actual value of the percentiles falls slightly off the approximations. This effect might explain why [19] observed a polynomial scaling behaviour of the 0.9 percentile in a similar analysis for GSAT when using smaller problems (up to $n = 100$); another potential explanation for the discrepancy between our result might be given by differences in the empirical methodology. However, considering the tight performance correlation established in Section 6.3, we assume that the difference in the algorithms being used is not responsible for the different results.

The qualitative difference in the scaling of the lower and higher percentiles is surprising and leaves at least two different interpretations. Either, there is a sharp transition between the polynomial and exponential behaviour as we study increasingly high percentiles of the hardness distributions. Or, all percentiles are exponential, but the lower ones show exponential behaviour only for much higher

n. Either way, the higher percentiles grow much faster than the lower ones and, as a consequence, the variation of hardness within the test-sets (as measured by the percentile ratios Q_x/Q_{1-x}), increases strictly monotonically with problem size. An intuitive explanation of this phenomenon is the assumption that with growing problem size, as more variables and clauses are available, more complex "traps" (local minima or plateau regions which are hard to escape from) for SLS-based algorithms can occur in the search spaces of the randomly generated instances.

Performing the same analysis using R-Novelty$^+$, the best-performing algorithm for Random-3-SAT according to our earlier results, yields exactly analogous results: The median curve can be well approximated by $y = 26.83 \cdot (n/30)^{2.5}$, while the 0.98 percentile's scaling is ca. $y = 200 \cdot exp(n/30)$. Comparing these results with the one for WalkSAT given above, we note that the difference in the percentiles for the two algorithms is independent of *n* (ca. 3.5 for the median, and ca. 4 for the 0.98 percentile). In other words, the performance difference between R-Novelty$^+$ and WalkSAT does not depend on the problem size, which is mildly disappointing as it means that for this problem class, R-Novelty$^+$ improves only by a small constant factor over WalkSAT.

Next, we applied an exactly analogous analysis for the Graph Colouring test-sets with $k = 50, ..., 200$ vertices in the original graph (this corresponds to 150–600 propositional variables in the SAT-encoded instances). Here, surprisingly, we obtain a different result: For WalkSAT (with approximately optimal noise for each *n*), the scaling of all the percentiles can be approximated by $y = a \cdot exp(k/b)$ with $(a,b) = (800,33)$ for the median and $(800,23)$ for the 0.98 percentile. When using Novelty$^+$ instead of WalkSAT, we get analogous results with $(a,b) = (220,30)$ for the median and $(220,20)$ for the 0.98 percentile. In general, for this problem class we have no evidence for a polynomial scaling of any percentile. At the same time, the phenomenon that for small *k* the percentile curves fall off the approximations is less obvious here. As a third difference, the variation in instance hardness is generally much smaller within our Graph Colouring test-sets than within the Random-3-SAT test-sets. As an example, consider the $Q_{0.95}/Q_{0.05}$ percentile ratio which is ca. 47 for test-set `uf150-645` and only ca. 5.1 for test-set `flat50-115` (all instances have 150 propositional variables). Hence, there seems to be a qualitative difference in the scaling behaviour for WalkSAT between Random-3-SAT and the Graph Colouring problem distribution considered here. One reason for this might be the fact that compared to Random-3-SAT, the Graph Colouring instances are structurally much more restricted because of the effects of the SAT-encoding used for transforming the original problems into CNF formulae. Also the fact that all instances are generated to be exactly 3-colourable might keep the variation of instance hardness within the test-sets somewhat smaller.

7. Conclusions

In this work, we have studied several of the most prominent and best-performing SLS algorithms for SAT. The published empirical results on the performance of these algorithms are mostly incompatible with each other because different methodologies and/or different sets of benchmark instances were used. Thus, in order to obtain a comprehensive picture of these algorithm's performance, we systematically evaluated them on newly compiled set of benchmark problems. This benchmark library contains different types of problem instances, including sets of randomly generated SAT instances and more structured SAT-encoded problems from various other domains. All problem classes are known from the literature and most of them have been used in the context of several studies of SLS performance. For our empirical study, we have used a novel empirical methodology which allowed us to characterise the behaviour of the various algorithms in great detail. This methodology has a number of advantages over traditional methods for the empirical evaluation of SLS-based SAT algorithms; in particular, it allows the functional characterisation of run-time distributions.

From our comparative analysis we obtained a number of new and interesting insights. First of all, of the algorithms considered here, no single one dominates all the others on all domains. However, we identified a subset of algorithms which show superior performance compared to the others across our benchmark set. These algorithms are: R-Novelty, Novelty, WalkSAT/TABU and GSAT/TABU; they all suffer from occasional stagnation behaviour, which for some instances severely compromises their performance. GWSAT, WalkSAT and HWSAT, on the other hand, are generally dominated by at least one of the aforementioned algorithms. Interestingly, all the best performing SLS variants for SAT make use of some aspects of the search history, such as a tabu list or the age of variables for guiding the local search process. This suggests that an adequate use of the search history may be generally crucial for obtaining superior SLS performance. This is very much in tune with recent results on local search algorithms for other combinatorial optimisation problems where, on a wide variety of problems, history-sensitive algorithms have been shown to perform better than local search methods which do not take into account the search history.

Additionally, when comparing the performance of the different SLS algorithms studied here, one has to take into account that when using the currently best-known implementations, the GSAT variants are by at least a factor of 3 slower than the WalkSAT variants, such that effectively, even GSAT/TABU is never competitive with the best-performing WalkSAT variants when absolute CPU-time is measured. Since the implementations of all GSAT and WalkSAT variants used in this study are highly optimised, we believe that this performance difference is inherent to the algorithms. There are several implementation related reasons which may be responsible for this fact. One is that WalkSAT algorithms are much simpler to implement efficiently and require less sophisticated data structures because they

evaluate only the scores of a small number of variables in each search step. Furthermore, clauses are typically rather short — in many instances stemming from SAT-encodings of other problems, most of the clauses are actually binary clauses. Therefore WalkSAT algorithms actually often select only between two variables while GSAT algorithms select the variable to be flipped from a potentially much larger set.

As we have shown, the essential incompleteness which compromises the performance especially of the high-performing WalkSAT variants can be effectively overcome by adding an unconditional random walk mechanism as used in G-WSAT. The Novelty$^+$ and R-Novelty$^+$ variants thus obtained show superior performance compared to the original algorithms and, to our best knowledge, might currently be the best-performing SLS algorithms for SAT. Compared to Novelty and R-Novelty they are particularly superior in terms of their robustness *w.r.t.* the setting of the cutoff parameter, which plays an important role in practice.

Regarding the functional characterisation of SLS behaviour, we empirically studied the run-time behaviour of WalkSAT, WalkSAT/TABU, Novelty, and R-Novelty. We showed that, using optimal noise parameter settings, the RLDs of these algorithms when applied to hard problem instances from various domains can be approximated by exponential distributions. The same phenomenon is observed for larger-than-optimal noise settings, while for smaller-than-optimal noise qualitatively different behaviour occurs. The insights gained by the functional characterisation of SLS behaviour also explain the observations made by other researchers that the use of random walk reduces the sensitivity of SLS algorithms to the particular value of the *maxSteps* parameter [20, 55]: for exponentially distributed RLDs, the solution probability does not depend on a particular *maxSteps* setting. However, due the effects of the initial hill-climbing phase, *maxSteps* should not be chosen too low.

We further introduced a refined mathematical model based on a new distribution type which is asymptotically exponential, but allows to model the effects of the initial search phase. As we have shown, this extended model allows a precise characterisation of SLS behaviour for a vast majority of the problem instances from our benchmark suite. Thus, for the first time, we can model the behaviour of some of the most powerful and prominent SLS algorithms for SAT in great detail. As a direct consequence of these results, when using optimal or larger-than-optimal noise settings, the algorithms are very robust *w.r.t.* the cutoff parameter setting.

Furthermore, we analysed the correlation between the different algorithm's performance across the Random-3-SAT and Graph Colouring test-sets. As a result, we found that there is a strong linear correlation between the logarithm of the average local search cost for each pair of algorithms. In other words, instances which are hard for one algorithm tend also to be hard for all other algorithms. Consequently it is justified to talk about the intrinsic hardness of problem instances for local search. The existence of the tight correlation in search cost we observed for various SLS algorithms is somewhat surprising, because the underlying con-

cepts are sufficiently different to cause significant differences in performance. This suggests that the local search cost for these algorithms depends on the same structural features of the search space. Finally, we investigated the scaling of SLS performance with problem size. Surprisingly, our results indicate that while for the hardest fractions of problem instances from the randomly generated test sets, the local search cost grows exponentially, for fractions of easier instances the scaling of SLS performance might be polynomial. This phenomenon could not be observed for the Graph Colouring test-sets, where the scaling was generally exponential. When comparing different SLS algorithms *w.r.t.* their scaling behaviour for growing problem size, we found no evidence of any substantial differences.

Of course, this study has a number of limitations and leaves many issues for further investigation. In particular, we did not attempt to explain the observed performance behaviour in terms of structural features of the underlying search spaces [64, 15, 35]. Clearly, any advances in obtaining a deeper understanding of the factors which affect SLS performance, such as search space topology, the effects of various SAT-encodings, and polynomial simplification techniques (which can be applied as a preprocessing step), will improve the chances of designing better performing and more robust SAT algorithms. Search space analyses may also be helpful for understanding the rather poor performance of the SLS algorithms studied here compared to algorithms using clause weighting schemes on certain types of problems, such as the single solution instances with very low clause per variable ratio generated with the AIM generator [2]. We are currently investigating these issues, using the same methodology underlying this study (for first results, see [35, 33]).

Furthermore, the algorithms considered here can be extended in many directions, e.g., by considering clause weighting schemes, multiple variable flips per local search step, and learning schemes for certain parameters or the objective function. In particular, for the algorithms considered here it would be highly desirable to develop schemes for automatically tuning the noise parameter setting to a given problem instance. Given such a method, the effort for fine-tuning the algorithms would be significantly reduced. To achieve this goal, the local search invariants from [51] or a reaction mechanism as suggested in [5] could be very useful. Another interesting direction is the design and investigation of hybrid methods which combine different SLS strategies (or even SLS methods with systematic SAT algorithms) using a simple control mechanism [35]. In fact, the improved WalkSAT variants Novelty$^+$ and R-Novelty$^+$ are examples of such hybrid algorithms as they combine the Novelty and R-Novelty strategies with the random walk mechanism used in GWSAT.

We feel, based on the empirical results presented here, that as algorithms are further improved, robustness of performance will become a major issue, and hybrid combinations of different algorithms or heuristics might become the key to obtaining maximal performance over a broad range of problem classes. Finally,

one of the big questions in SAT solving remains to be answered: how the best SLS algorithms and the best systematic methods compare in terms of peak performance and robustness. We started investigating this question [38], for which this study provides a good basis, as it provides a carefully designed benchmark set, offers an advanced empirical methodology for empirically analysing the performance of stochastic algorithms, and identifies the candidates for the best-performing SLS algorithms. We are convinced that these contributions will also prove valuable for research on many of the issues mentioned above and thus bring us another step closer to a deep understanding and profound assessment of stochastic local search algorithms for SAT and other hard combinatorial problems.

Acknowledgements

A large part of this research was done at FG Intellektik, TU Darmstadt in Germany. We would like to thank Wolfgang Bibel and the Intellectics Group for their support, as well as Bart Selman, Henry Kautz, David McAllester, David Poole, Ian Gent, Toby Walsh, and Joe Culberson for interesting discussions in earlier stages of this work. Furthermore we extend our gratitude towards Henry Kautz and Bart Selman for providing the code for GSAT and WalkSAT. Finally, we thank the anonymous reviewers and Kevin O'Neill for their comments and suggestions which considerably helped to improve this article. This work was in part supported by a Postdoctoral Fellowship awarded by the University of British Columbia to Holger H. Hoos and by a Marie Curie Fellowship awarded to Thomas Stützle (CEC-TMR Contract No. ERB4001GT973400).

References

1. H. Alt, L. Guibas, K. Mehlhorn, R. Karp, and A. Wigderson. A Method for Obtaining Randomized Algorithms with Small Tail Probabilities. *Algorithmica*, 16:543–547, 1996.
2. Y. Asahiro, K. Iwama, and E. Miyano. Random Generation of Test Instanzes with Controlled Attributes. In [41], pages 377–394.
3. A. Beringer, G. Aschemann, H.H. Hoos, M. Metzger, and A. Weiß. GSAT versus Simulated Annealing. In A.G. Cohn, editor, *Proceedings of ECAI'94*, pages 130–134. John Wiley & Sons, 1994.
4. M. Buro and H. Kleine-Büning. Report on a SAT Competition. Technical Report 110, Dept. of Mathematics and Informatics, University of Paderborn, Germany, 1992.
5. R. Battiti and M. Protasi. Reactive Search, A History-Based Heuristic for MAX-SAT. *ACM Journal of Experimental Algorithmics*, 2, 1997.
6. R.E. Barlow and F. Proschan. *Statistical Theory of Reliability and Life Testing.* Holt, Reinhart and Winston, Inc., 1981 (reprint).
7. B. Cha and K. Iwama. Performance Tests of Local Search Algorithms Using New Types of Random CNF Formula. In *Proceedings of IJCAI'95*, pages 304–309. Morgan Kaufmann Publishers, 1995.

8. B. Cha and K. Iwama. Adding New Clauses for Faster Local Search. In *Proceedings of AAAI'96*, pages 332–337. MIT Press, 1996.
9. P. Cheeseman, B Kanefsky, and W.M. Taylor. Where the really hard problems are. In *Proceedings of IJCAI'91*, pages 331–337. Morgan Kaufmann Publishers, 1991.
10. J.M. Crawford and L.D. Auton. Experimental Results on the Crossover Point in Random 3SAT. *Artificial Intelligence*, 81(1–2):31–57, 1996.
11. M. Davis, G. Logemann, and D. Loveland. A Machine Program for Theorem Proofing. *Communications of the ACM*, 5:394–397, 1962.
12. J. de Kleer. A Comparison of ATMS and CSP Techniques. In *Proceedings of IJCAI'89*, pages 290–296. Morgan Kaufmann Publishers, 1989.
13. O. Dubois, P. Andre, Y. Boufkhad, and J. Carlier. SAT versus UNSAT. In [41], pages 415–436.
14. A.E. Eiben and J.K. van der Hauw. Solving 3-SAT with Adaptive Genetic Algorithms. In *Proceedings of the 4th IEEE Conference on Evolutionary Computation*, pages 81-86. IEEE Press, 1997.
15. J. Frank, P. Cheeseman, and J. Stutz. When Gravity Fails: Local Search Topology. *(Electronic) Journal of Artificial Intelligence Research*, 7:249–281, 1997.
16. J. Frank. Weighting for Godot: Learning Heuristics for GSAT. In *Proceedings of AAAI'96*, pages 338–343. MIT Press, 1996.
17. J. Frank. Learning Short-term Clause Weights for GSAT. In *Proceedings of IJCAI'97*, pages 384–389. Morgan Kaufmann Publishers, 1997.
18. D. Frost, I. Rish, and L. Vila. Summarizing CSP Hardness with Continuous Probability Distributions. In *Proceedings of AAAI'97*, pages 327–333. MIT Press, 1997.
19. I.P. Gent, E. MacIntyre, P. Prosser, and T. Walsh. The Scaling of Search Cost. In *Proceedings of AAAI'97*, pages 315-320. MIT Press, 1997.
20. I. Gent and T. Walsh. Unsatisfied Variables in Local Search. In J. Hallam, editor, *Hybrid Problems, Hybrid Solutions*, pages 73–85. IOS Press, 1995.
21. I.P. Gent and T. Walsh. Towards an Understanding of Hill–Climbing Procedures for SAT. In *Proceedings of AAAI'93*, pages 28–33. MIT Press, 1993.
22. I.P. Gent and T. Walsh. An Empirical Analysis of Search in GSAT. *Journal of Artificial Intelligence Research*, 1:47–59, 1993.
23. F. Glover. Tabu Search – Part I. *ORSA Journal on Computing*, 1(3):190–206, 1989.
24. C.P. Gomes and B. Selman. Problem Structure in the Presence of Perturbations. In *Proceedings of AAAI'97*, pages 221–226. MIT Press, 1997.
25. C.P. Gomes, B. Selman, and H. Kautz. Boosting Combinatorial Search Trough Randomization. In *Proceedings of AAAI'98*, pages 431–437. MIT Press, 1998.
26. J. Gottlieb and N. Voss. Improving the Performance of Evolutionary Algorithms for the Satisfiability Problem by Refining Functions. In A.E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Proceedings of PPSN V*, volume 1498 of *LNCS*, pages 813–822. Springer Verlag, 1998.
27. J. Gu. Efficient Local Search for Very Large-Scale Satisfiability Problems. *SIGART Bulletin*, 3:8–12, 1992.
28. P. Hansen and B. Jaumard. Algorithms for the Maximum Satisfiability Problem. *Computing*, 44:279–303, 1990.
29. T. Hogg. Refining the Phase Transition in Combinatorial Search. *Artificial Intelligence*, 81:127–154, 1996.
30. J.N. Hooker. Needed: An Empirical Science of Algorithms. *Operations Research*, 42(2):201–212, 1994.
31. J.N. Hooker. Testing Heuristics: We Have It All Wrong. *Journal of Heuristics*, pages 33–42, 1996.

32. H.H. Hoos. Aussagenlogische SAT-Verfahren und ihre Anwendung bei der Lösung des HC-Problems in gerichteten Graphen. Master's thesis, Technische Universität Darmstadt, Darmstadt, Germany, 1996.
33. H.H. Hoos. SAT-Encodings, Search Space Structure, and Local Search Performance. In *Proceedings of IJCAI'99*, 1999.
34. H.H. Hoos. On the Run-time Behaviour of Stochastic Local Search Algorithms for SAT. In *Proceedings of AAAI'99*, pages 661–666. MIT Press, 1999.
35. H.H. Hoos. *Stochastic Local Search — Methods, Models, Applications*. PhD thesis, Department of Computer Science, Darmstadt University of Technology, 1998.
36. H.H. Hoos and T. Stützle. Evaluating Las Vegas Algorithms — Pitfalls and Remedies. *Proceedings of UAI-98*, pages 238–245. Morgan Kaufmann Publishers, 1998.
37. H.H. Hoos and T. Stützle. Characterizing the Run-time Behaviour of Stochastic Local Search. Technical Report AIDA–98–1, FG Intellektik, TU Darmstadt, January 1998.
38. H.H. Hoos and T. Stützle. Systematic vs. Local Search for SAT. To appear, *Proceedings of the 23rd National German Conference on Artificial Intelligence (KI-99)*, 1999.
39. H.H. Hoos and T. Stützle. Towards a Characterisation of the Behaviour of Stochastic Local Search Algorithms for SAT. To appear, Artificial Intelligence Journal.
40. D.S. Johnson, C.R. Aragon, L.A. McGeoch, and C. Schevon. Optimization by Simulated Annealing: An Experimental Evaluation: Part II, Graph Coloring and Number Partitioning. *Operations Research*, 39(3):378–406, 1991.
41. D.S. Johnson and M.A. Trick, editors. *Cliques, Coloring, and Satisfiability*, volume 26 of *DIMACS Series on Discrete Mathematics and Theoretical Computer Science*. American Mathematical Society, 1996.
42. A.P. Kamath, N.K. Karmarkar, K.G. Ramakrishnan, and M.G.C. Resende. A Continuous Approach to Inductive Inference. *Mathematical Programming*, 57:215–238, 1992.
43. K. Kask and R. Dechter. Graph-based Methods for Improving GSAT. In *Proceedings of AAAI'96*, pages 350–355. MIT Press, 1996.
44. H. Kautz, D. McAllester, and B. Selman. Encoding Plans in Propositional Logic. In *Proceedings of KR'96*, pages 374–384, 1996.
45. H. Kautz and B. Selman. Pushing the Envelope: Planning, Propositional Logic, and Stochastic Search. In *Proceedings of AAAI'96*, volume 2, pages 1194–1201. MIT Press, 1996.
46. H. Kautz and B. Selman. BLACKBOX: A New Approach to the Application of Theorem Proving to Problem Solving. In *Working notes of the Workshop on Planning as Combinatorial Search*, held in conjunction with AIPS-98, Pittsburgh, PA, 1998.
47. S. Kirkpatrick and B. Selman. Critical Behavior in the Satisfiability of Random Boolean Expressions. *Science*, 264:1297–1301, 1994.
48. T. Larrabee. Test Pattern Generation Using Boolean Satisfiability. *IEEE Transactions on Computer-Aided Design*, 11(1):6–22, 1992.
49. M. Luby, A. Sinclair, and D. Zuckerman. Optimal Speedup of Las Vegas Algorithms. *Information Processing Letters*, 47:173–180, 1993.
50. B. Mazure, L. Sais, and É. Grégoire. Tabu Search for SAT. In *Proceedings of AAAI'97*, pages 281–285, 1997.
51. D. McAllester, B. Selman, and H. Kautz. Evidence for Invariants in Local Search. In *Proceedings of AAAI'97*, pages 321–326. MIT Press, 1997.
52. D. Mitchell, B. Selman, and H. Levesque. Hard and Easy Distributions of SAT Problems. In *Proceedings of AAAI'92*, pages 459–465. MIT Press, 1992.
53. P. Morris. The Breakout Method for Escaping from Local Minima. In *Proceedings of AAAI'93*, pages 40–45. MIT Press, 1993.
54. S. Minton, M.D. Johnston, A.B. Philips, and P. Laird. Minimizing Conflicts: A Heuristic Repair Method for Constraint Satisfaction and Scheduling Problems. *Artificial Intelligence*, 52:161–205, 1992.

55. A.J. Parkes and J.P. Walser. Tuning Local Search for Satisfiability Testing. In *Proceedings of AAAI'96*, pages 356–362. MIT Press, 1996.
56. M.G.C. Resende, T.A. Feo. A GRASP for Satisfiability. In [41], pages 499–520.
57. V.K. Rohatgi. *An Introduction to Probability Theory and Mathematical Statistics*. John Wiley & Sons, 1976.
58. B. Selman, Henry A. Kautz. An Empirical Study of Greedy Local Search for Satisfiability Testing. In *Proceedings of AAAI'93*, pages 46–51. MIT Press, 1993.
59. B. Selman, Henry A. Kautz, and B. Cohen. Noise Strategies for Improving Local Search. In *Proceedings of AAAI'94*, pages 337–343. MIT Press, 1994.
60. B. Selman, H. Levesque, and D. Mitchell. A New Method for Solving Hard Satisfiability Problems. In *Proceedings of AAAI'92*, pages 440–446. MIT Press, 1992.
61. O. Steinmann, A. Strohmaier, and T. Stützle. Tabu Search vs. Random Walk. In *Advances in Artificial Intelligence (KI97)*, volume 1303 of *LNAI*, pages 337–348. Springer Verlag, 1997.
62. T. Stützle. *Local Search Algorithms for Combinatorial Problems — Analysis, Improvements, and New Applications*. PhD thesis, Department of Computer Science, Darmstadt University of Technology, 1998.
63. W.M. Spears. Simulated Annealing for Hard Satisfiability Problems. Technical Report, Naval Research Laboratory, Washington D.C., 1993.
64. T. Yokoo. Why Adding More Constraints Makes a Problem Easier for Hill-Climbing Algorithms: Analyzing Landscapes of CSPs. In *Proceedings of CP'97*, pages 357–370. Springer Verlag, 1997.

Appendix

A. Details on the Benchmark Set

In this appendix, we provide more detailed information on the suite of benchmark problems used for the empirical analyses reported in this study. All benchmark instances are available from the SATLIB website (http://www.informatik.tu-darmstadt.de/AI/SATLIB).

A.1. UNIFORM RANDOM-3-SAT

Uniform Random-3-SAT is a family of SAT instance distributions obtained by randomly generating 3-CNF formulae in the following way: For an instance with n variables and k clauses, each of the k clauses is constructed from 3 literals which are randomly drawn from the $2n$ possible literals (the n variables and their negations) such that each possible literal is selected with the same probability of $1/2n$. Clauses are not accepted for the construction of the problem instance if they contain multiple copies of the same literal or if they are tautological (i.e., they contain a variable and its negation). Each choice of n and k thus induces a distribution of Random-3-SAT instances. Uniform Random-3-SAT is the union of these distributions over all n and k.

One particularly interesting property of uniform Random-3-SAT is the occurrence of a phase transition phenomenon, i.e., a rapid change in solubility which can be observed when systematically increasing (or decreasing) the number k of clauses for fixed problem size n [52, 47]. More precisely, for small k, almost all formulae are underconstrained and therefore satisfiable; when reaching some critical $k = k^*$, the probability of generating

a satisfiable instance drops sharply to almost zero. Beyond k^*, almost all instances are overconstrained and thus unsatisfiable. For Random-3-SAT, this phase transition occurs approximately at $k^* = 4.26n$ for large n; for smaller n, the critical clauses/variable ratio k^*/n is slightly higher [52, 10]. Furthermore, for growing n the transition becomes increasingly sharp.

Empirical analyses show that problem instances from the phase transition region of uniform Random-3-SAT tend to be particularly hard for both systematic SAT solvers [9, 10] and SLS algorithms [64]. Striving to test their algorithms on hard problem instances, many researchers used test-sets sampled from the phase transition region of uniform Random-3-SAT (see [21, 50, 51] for some examples). Although similar phase transition phenomena have been observed for other subclasses of SAT, including uniform Random-k-SAT with $k \geq 4$, these have never gained the popularity of uniform Random-3-SAT. Maybe one of the reasons for this is the prominent role of 3-SAT as a prototypical and syntactically particularly simple $\mathcal{NP}$-complete problem.

Table XIX. Uniform Random-3-SAT test-sets.

test-set	instances	clause-len	vars	clauses
uf50-218	1,000	3	50	218
uf75-325	100	3	75	325
uf100-430	1,000	3	100	430
uf125-538	100	3	125	538
uf150-645	100	3	150	645
uf175-753	100	3	175	753
uf200-860	100	3	200	860
uf225-960	100	3	225	960
uf250-1065	100	3	250	1,065

Our test-sets were obtained by randomly generating problem instances from the phase transition region of uniform Random-3-SAT as detailed above; insoluble instances were filtered out using a fast complete SAT algorithm. Table XIX shows the characteristics of the test-sets used in this study.

A.2. Graph Colouring

All instances were generated using Joe Culberson's random graph generator[7]. Given a graph $G = (V,E)$, where $V = \{v_1, v_2, ..., v_n\}$ is the set of vertices and $E \subseteq V \times V$ the set of edges connecting the vertices, the following, simple yet efficient encoding was used for transforming GCP instances into SAT [12]: Given k colours, each assignment of a colour g to a vertex v_i is represented by a propositional variable $x_{i,g}$; each constraint (i.e., an edge connecting two vertices) is represented by a set of clauses $(\neg x_{i,g} \vee \neg x_{j,g})$; two additional

[7] available from http://web.cs.ualberta.ca/~joe/Coloring/index.html, Joe Culbersons's Graph Coloring Page.

sets of clauses ensure that valid SAT assignments assign exactly one value to each CSP variable: $(x_{i,0} \vee x_{i,1} \vee \ldots \vee x_{i,k-1})$, $\forall v_i \in V$ (at least one colour is assigned to a node) and $(\neg x_{i,g} \vee \neg x_{i,h})$, $\forall g,h \in \{0,\ldots,k-1\}, g \neq h,\ \forall v_i \in V$ (at most one colour is assigned to a node). Table XX shows the characteristics of the test-sets used for our empirical study.

Table XX. SAT-encoded Graph Colouring test-sets (flat random graphs).

test-set	instances	vertices	edges	colours	vars	clauses
flat50-115	1,000	50	115	3	150	545
flat75-180	100	75	180	3	225	840
flat100-239	100	100	239	3	300	1,117
flat125-301	100	125	301	3	375	1,403
flat150-360	100	150	360	3	450	1,680
flat175-417	100	175	417	3	525	1,951
flat200-479	100	200	479	3	600	2,237

A.3. PLANNING INSTANCES

Our benchmark set contains the four largest Blocks World Planning instances and four Logistics Planning instances from Henry Kautz's and Bart Selman's SATPLAN distribution. These instances are described in Table XXI; despite the reductions mentioned above, they are still very large when compared to other instances of our benchmark suite. At the time of this writing, `bw_large.c` and `bw_large.d` belong to the hardest problem instances that can be solved by state-of-the-art SAT algorithms in reasonable time.

The SAT encoding used for generating the benchmark instances relies critically on techniques for reducing the size of the CNF formulae. These concern the particular way of defining the propositional variables as well as the application of well-known propositional reduction strategies, like unit propagation and subsumption, which are used to simplify the formulae before applying stochastic local search. These reductions can be computed in polynomial time and eliminate a number of propositional variables whereby the search space is efficiently reduced. Details on the SAT encoding used to generate the benchmark instances can be found in [45, 44].

A.4. INSTANCES FROM THE DIMACS BENCHMARK SET

The large SAT-encoded Graph Colouring instances originally stem from a study performed by Johnson et.al. [40]. The encoding used for these instances is the same as described in Section A.2. In our experimental analysis we used the three instances described in Table XXII; for a forth instance (`g250.15`), we found that, despite its size, it is extremely easy for all algorithms considered here and thus decided not to use it in our analysis. Instances `g125.18.cnf` and `g125.17.cnf` are based on the same graph. In the first case, the formula encodes searching for a solution with 18 colours, while in the later case one tries to find a colouring with 17 colours (which is conjectured to be an optimal colouring for this graph).

Table XXI. SAT-encoded Blocks World Planning and Logistics Planning instances.

instance	packages	plan steps	vars	clauses
logistics.a	8	11	828	6,718
logistics.b	5	13	843	7,301
logistics.c	7	13	1,141	10,719
logistics.d	9	14	4,713	21,991
bw_large.a	9	6	459	4,675
bw_large.b	11	9	1,087	13,772
bw_large.c	15	14	3,016	50,457
bw_large.d	19	18	6,325	131,973

Table XXII. SAT-encoded Graph Colouring instances from the DIMACS benchmark set.

instance	nodes	colours	vars	clauses
g125.18.cnf	125	18	2,250	70,163
g125.17.cnf	125	11	2,125	66,272
g250.29.cnf	250	29	7,250	454,622

The parity learning instances `par*` are very hard for the SLS algorithms we used. Therefore we limited our RLD-based study to the smaller instances from the DIMACS Benchmark set, which encode learning a parity function with eight variables based on a sample of (potentially noisy) input/output pairs. These formulae have been simplified using unit propagation and subsumption. The resulting formulae are rather small: each of the five instances contains approx. 70 variables and approx. 270 clauses.

Other classes of benchmark instances of the DIMACS benchmark set have been deliberately excluded from our study. Among these are the `aim*` instances obtained from the AIM instance generator [2] which are generated randomly for clauses/variable ratios of 1.6, 2.0, 3.4, and 6.0 in such a way that each formula has exactly one satisfiable assignment. However, different from all problem classes used in our benchmark collection, the resulting instances are polynomially solvable by simplification procedures (using binary failed clauses) and thus cannot be considered intrinsically hard. When practically solving SAT instances, such simplification procedures are typically applied as a preprocessing step.[8] Considering this, we regard instances which are solved by the preprocessing procedure alone to be of minor interest in the context of our study of SLS performance. Without simplification, when using reasonably optimised noise settings, the best WalkSAT variants

[8] Note that for SAT-encoded problems from other domains, this preprocessing is known to be crucial for finding solutions using SLS or systematic SAT algorithms.

typically solve the instances with high clauses per variable ratio with an average of less than 1000 flips. But for lower clause per variable ratios, such as 1.6 or 2.0, some of the instances are very difficult to solve using the algorithms considered here. For example, when running experiments on instance `aim-50-1_6-yes1-2.cnf` (50 variables, clauses per variable ratio $= 1.6$) we could, using various noise parameter settings and for each setting running 100 tries of 10^7 flips each, only find 14 solutions with either of the WalkSAT variants. In contrast, weighting schemes for GSAT solve this type of instances much more efficiently [7, 8]. It is not clear whether this is just an artifact of this problem class or a more general phenomenon which can be also observed for hard problem classes that cannot be efficiently solved by polynomial simplification methods. It would be interesting to further investigate this interesting question; however, this would require additional methodology which is beyond the scope of this study.

Similary, also the instances from a test pattern generation program for VLSI circuits [48] (`ssa*`) can be solved, despite their relatively large size (they contain from 1,363 to 1,501 variables and 3,032 to 3,575 clauses), by preprocessing techniques alone. They are also relatively easily solved by the SLS algorithms applied here. Of the other instances there are some satisfiable ones (16 out of 50) among the randomly generated instances `jnh*.cnf` with variable clause lenghts. Some initial experiments showed that these instances are easily solved and therefore we preferred to focus rather on our large-sets of hard Uniform Random-3-SAT instances. Finally, we also excluded the `ii*` instances which originally stem from an integer programming formulation of an inductive inference problem [42]. These are very easily solved by, for example, WalkSAT and therefore do not provide a challenging benchmark for the local search algorithms applied here.

B. Supplementary Data from the Comparative Study of SLS Performance

This appendix provides some additional information on the results of our comparison of SLS algorithms presented in Section 5. In particular, we report the descriptive statistics for the various algorithms' RLDs when applied to problem instances from our benchmark suite in numerical form. This information is particularly useful when reproducing our results or evaluating other algorithm on the same benchmark problems.

B.1. RANDOM-3-SAT

In Section 5.1 we compared various GSAT and WalkSAT algorithms on Random-3-SAT instances. For the three instances we studied in more detail (`easy`, `med`, and `hard`), we found that generally between different algorithm's RLDs typically no cross-overs are observed. In this situation it is admissible to base performance comparisons mainly on descriptive statistics, which are given in Tables XXIII–XXV.

In general, the numerical data confirm the discussion of the results in Section 5.1. One interesting observation which is even more obvious when looking at the numerical data rather than the graphical representation of the RLDs is the huge variability in the time required to find a solution (see standard deviation and percentile ratios in the tables) and the fact that this variability increases with instance hardness, as can be seen comparing the stddev/mean values or the percentile ratios between the easy, medium, and hard instance.

Table XXIII. Problem instance `uf100-430/easy`, basic descriptive statistics of RLDs for various algorithms, approx. optimal noise (in parentheses after algorithms' names), based on 1,000 tries.

algorithm	mean	stddev	$\frac{\text{stddev}}{\text{mean}}$	median	Q_{75}	Q_{90}	$\frac{Q_{75}}{Q_{25}}$	$\frac{Q_{90}}{Q_{10}}$
GWSAT(0.5)	270.63	240.62	0.89	202	313	504	2.35	5.54
GSAT/TABU(20)	137.78	91.31	0.66	108	170	255	2.18	4.40
HWSAT(0.3)	190.58	154.25	0.81	145	223	365	2.40	5.21
WalkSAT(0.5)	177.86	135.78	0.76	139	206	336	2.22	4.87
WalkSAT/TABU(5)	97.79	49.04	0.50	86	119	164	1.92	3.28
Novelty(0.9)	77.36	37.29	0.48	68	92	124	1.77	3.02
R-Novelty(0.9)	76.63	34.79	0.45	68	90	121	1.70	2.81

Table XXIV. Problem instance `uf100-430/med`, basic descriptive statistics of RLDs for various algorithms, approx. optimal noise, based on 1,000 tries.

algorithm	mean	stddev	$\frac{\text{stddev}}{\text{mean}}$	median	Q_{75}	Q_{90}	$\frac{Q_{75}}{Q_{25}}$	$\frac{Q_{90}}{Q_{10}}$
GWSAT(0.5)	2,432	2,159	0.89	1,785	3,345	5,434	3.78	11.51
GSAT/TABU(20)	1,368	1,297	0.95	995	1,804	2,903	3.83	12.62
HWSAT(0.4)	2,015	1,727	0.87	1,509	2,687	4,502	3.59	10.67
WalkSAT(0.5)	1,877	1,776	0.95	1,333	2,510	4,133	3.90	12.84
WalkSAT/TABU(3)	532	365	0.69	433	689	999	2.43	5.52
Novelty(0.6)	504	372	0.74	416	647	962	2.62	6.13
R-Novelty(0.7)	580	467	0.80	440	767	1,213	3.09	7.93

Table XXV. Problem instance `uf100-430/hard`, basic descriptive statistics of RLDs for various algorithms, approx. optimal noise, based on 1,000 tries.

algorithm	mean	stddev	$\frac{\text{stddev}}{\text{mean}}$	median	Q_{75}	Q_{90}	$\frac{Q_{75}}{Q_{25}}$	$\frac{Q_{90}}{Q_{10}}$
GWSAT(0.6)	177,468	181,843	1.02	119,666	247,047	400,466	5.27	22.78
GSAT/TABU(20)	84,578	84,301	1.00	57,778	120,286	196,493	5.01	24.95
HWSAT(0.4)	37,822	37,717	1.00	26,481	52,008	89,132	4.96	22.98
WalkSAT(0.5)	86,773	90,538	1.04	56,666	120,583	198,109	5.41	27.07
WalkSAT/TABU(5)	87,031	86,692	1.00	60,019	119,246	206,822	5.14	26.23
Novelty(0.7)	26,995	27,165	1.01	19,434	36,972	58,661	4.43	18.06
R-Novelty(0.7)	19,118	19,827	1.04	12,819	26,707	43,911	4.71	23.36

Table XXVI. Problem instance bw_large.a, basic descriptive statistics of RLDs for various algorithms, approx. optimal noise, based on 1,000 tries.

algorithm	mean	stddev	$\frac{\text{stddev}}{\text{mean}}$	median	Q_{75}	Q_{90}	$\frac{Q_{75}}{Q_{25}}$	$\frac{Q_{90}}{Q_{10}}$
GWSAT(0.6)	26,994	24,784	0.92	18,653	38,041	58,679	4.09	11.61
GSAT/TABU(10)	10,635	10,515	0.99	6,977	14,917	25,291	4.72	20.02
WalkSAT(0.5)	17,490	15,320	0.88	13,505	23,446	35,981	3.44	11.77
WalkSAT/TABU(3)	10,603	9,665	0.91	7,563	14,169	23,870	3.84	13.41
Novelty(0.4)	9,315	8,587	0.92	6,932	12,877	20,202	4.02	13.17
R-Novelty(0.6)	6,053	5,583	0.92	4,317	8,413	12,875	4.01	11.87

Table XXVII. Problem instance bw_large.c, basic descriptive statistics of RLDs for various algorithms, approx. optimal noise, based on 250 tries.

algorithm	mean	stddev	$\frac{\text{stddev}}{\text{mean}}$	median	Q_{75}	$\frac{Q_{75}}{Q_{25}}$
WalkSAT(0.2)	$14.06 \cdot 10^6$	$15.60 \cdot 10^6$	1.11	$9.76 \cdot 10^6$	$17.0 \cdot 10^6$	4.85
WalkSAT/TABU(2)	$2.52 \cdot 10^6$	$2.12 \cdot 10^6$	0.84	$2.00 \cdot 10^6$	$3.53 \cdot 10^6$	3.89
Novelty(0.2)	$6.4 \cdot 10^6$	$7.13 \cdot 10^6$	1.12	$4.40 \cdot 10^6$	$8.16 \cdot 10^6$	4.70

B.2. Graph Colouring

Tables XXVIII–XXX present the numerical data on the RLDs of various SLS algorithms applied to the easy, medium, and hard instances of the flat100-329 Graph Colouring test-set. Here, it is interesting to note that the (atypical) cross-over between the RLDs of R-Novelty and WalkSAT/TABU on the medium instance, which is very obvious in the graphical representation (Figure 8), is hard to see from the numerical statistics and could not be detected without the data from very low percentiles, such as Q_{10}. Nevertheless, cross-overs like this are significant, as they indicate that the performance relation between two algorithms changes depending on their run-time.

B.3. Blocks World Planning

In Table XXVI and Table XXVII we report the descriptive statistics for the RLDs of several SLS algorithms applied to the two SAT-encoded Blocks World Planning instances bw_large.a and bw_large.c. Here, it is interesting to note that the optimal noise parameter setting for WalkSAT and Novelty significantly depends on the problem size: The larger (and harder) the instance, the lower the optimal noise setting; this is confirmed by the corresponding results for the other two instances bw_large.b (see Table III in Section 5.3) and bw_large.d (not shown here). This is very different from the small deviation in the approximately optimal noise values reported for different instances of the Random-3-SAT and Graph Colouring test-sets, which rather reflect minor and non-systematic deviations between the instances without significant impact on SLS performance.

Table XXVIII. Problem instance `flat100-239/easy`, basic descriptive statistics of RLDs for various algorithms, approx. optimal noise, based on 1,000 tries.

algorithm	mean	stddev	$\frac{\text{stddev}}{\text{mean}}$	med	Q_{75}	Q_{90}	$\frac{Q_{75}}{Q_{25}}$	$\frac{Q_{90}}{Q_{10}}$
GWSAT(0.6)	7,268	6,898	0.95	5,146	9,031	15,168	3.24	9.07
GSAT/TABU(20)	1,636	1,156	0.71	1,320	2,242	3,215	2.97	6.68
HWSAT(0.5)	3,142	3,080	0.98	2,271	3,878	5,946	2.84	8.11
WalkSAT(0.5)	5,602	4,358	0.78	4,341	7,398	11,577	2.93	7.61
WalkSAT/TABU(5)	2,453	1,924	0.78	1,863	3,155	4,978	2.82	6.78
Novelty(0.6)	1,333	1,097	0.82	980	1,733	2,704	2.96	6.97
R-Novelty(0.6)	2,253	1,912	0.85	1,687	2,958	4,721	3.44	9.15

Table XXIX. Problem instance `flat100-239/med`, basic descriptive statistics of RLDs for various algorithms, approx. optimal noise, based on 1,000 tries.

algorithm	mean	stddev	$\frac{\text{stddev}}{\text{mean}}$	med	Q_{10}	Q_{75}	Q_{90}	$\frac{Q_{75}}{Q_{25}}$	$\frac{Q_{90}}{Q_{10}}$
GWSAT(0.6)	41,784	41,629	1.00	27,288	5,426	58,159	97,410	4.83	17.95
GSAT/TABU(10)	9,811	9,765	1.00	6,539	1,656	13,041	22,443	4.07	13.55
HWSAT(0.5)	20,377	21021	1.03	13,469	2,023	28,089	49,004	5.08	24.22
WalkSAT(0.5)	31,260	28,586	0.91	22,595	4,995	42,567	70,267	4.23	14.07
WalkSAT/TABU(3)	11,881	11,094	0.93	8,548	2,091	16,086	26,590	3.87	12.72
Novelty(0.6)	7,070	5,928	0.84	5,455	1,297	9,951	14,826	3.77	11.43
R-Novelty(0.6)	16,183	16,333	1.01	11,291	1,997	20,790	36,645	4.36	18.35

Table XXX. Problem instance `flat100-239/hard`, basic descriptive statistics of RLDs for various algorithms, approx. optimal noise, based on 1,000 tries.

algorithm	mean	stddev	$\frac{\text{stddev}}{\text{mean}}$	med	Q_{75}	Q_{90}	$\frac{Q_{75}}{Q_{25}}$	$\frac{Q_{90}}{Q_{10}}$
GWSAT(0.6)	306,886	249,012	0.81	240,302	451,475	700,129	4.35	15.96
GSAT/TABU(10)	107,872	111,690	1.04	73,594	142,838	251,477	4.51	21.35
HWSAT(0.5)	166,365	161,314	0.97	113,981	220,950	405,482	4.35	26.20
WalkSAT(0.5)	254,373	222,835	0.88	192,788	375,285	578,138	4.73	21.45
WalkSAT/TABU(3)	297,778	245,045	0.82	229,496	453,820	674,915	4.66	20.79
Novelty(0.6)	116,773	117,259	1.00	79,307	159,525	276,180	4.78	24.50
R-Novelty(0.6)	195,965	183,408	0.94	140,671	284,093	433,875	5.18	19.36