An Empirical Study of Optimal Noise and Runtime Distributions in Local Search*

Lukas Kroc[1,2], Ashish Sabharwal[1], and Bart Selman[1]

[1] Dept. of Computer Science, Cornell University, Ithaca NY 14853-7501, USA
[2] CCS-3, Los Alamos National Lab, Los Alamos, NM 87545, USA
{kroc,sabhar,selman}@cs.cornell.edu

Abstract. This paper presents a detailed empirical study of local search for Boolean satisfiability (SAT), highlighting several interesting properties, some of which were previously unknown or had only anecdotal evidence. Specifically, we study hard random 3-CNF formulas and provide surprisingly simple analytical fits for the optimal (static) noise level and the runtime at optimal noise, as a function of the clause-to-variable ratio. We also demonstrate, for the first time for local search, a power-law decay in the tail of the runtime distribution in the low noise regime. Finally, we discuss a Markov Chain model capturing this intriguing feature.

Designing, understanding, and improving, local search methods for constraint reasoning, and in particular for Boolean satisfiability (SAT), has been the focus of hundreds of research papers since the 1990's and even earlier. For SAT, techniques such as greedy local search, tabu search, solution guided search, focused random walk, and reactive or adaptive search have led to much success. Specifically, `Walksat` [7] stands out as one of the initial solvers that introduced many of the key ideas in use today and, is still competitive with the state of the art.

While many attempts have been made to understand the behavior of local search methods in terms of local minima, exploring "plateaus", the exploration vs. exploitation tradeoff, etc., our formal understanding is limited mostly to relatively simple variants of local search, such as a pure greedy search, a pure random walk, or a combination of the two. This is not surprising as the techniques employed by `Walksat` and other state-of-the-art local search solvers are too complex to allow a formal analysis in terms of, for example, a traditional Markov Chain. At the same time, there is a wealth of information available from observations of the behavior of local search methods on a variety of domains, most notably for random 3-SAT. There is either formal or anecdotal evidence of various features, such as `Walksat` scaling linearly at optimal noise but exponentially at sub-optimal noise, and suggestions that the runtime distribution of local search on a single random instance has an exponentially decaying tail. This work provides convincing empirical evidence in favor of, or even against, such

* Supported by NSF (Expeditions in Computing award for Comp. Sustainability, 0832782; IIS 0514429; EMT 0829861) & AFOSR (IISI, FA9550-04-1-0151). The authors thank Yahoo! for generously providing access to their M45 compute cloud.

anecdotal insights and observations. We study the behavior of `Walksat` on hard, large, random 3-CNF formulas and investigate its time complexity in relation to the *clause-to-variable ratio* α and the (static) *noise level*—both of which `Walksat` is highly sensitive to. Unlike previous studies, our conclusions are based on very large formulas and are thus free of "small N effects". This might explain the difference between our conclusions and those of, e.g., Hoos and Stützle [4].

While many new local search SAT solvers are based on "adaptive" or "dynamic" noise, these solvers are apparently unable to settle on the optimal noise setting for hard random 3-CNF formulas, doing much worse than optimal static noise. E.g., we found that the SAT Competition 2009 winners in the satisfiable Random category, `TNM` and `gnovelty+2`, were slower than `Walksat` at optimal noise by a factor of roughly 4x for N=10,000 variable formulas with $\alpha = 4.2$, 13x for N=20,000, 31x for N=30,000, 54x for N=40,000, and 785x for N=50,000. This also shows that, unlike `Walksat`, these adaptive noise solvers scale super-linearly in this domain, justifying the interest in our study of static noise.

Our first result is a surprisingly simple step-linear analytical fit for the value of the *optimal noise* as a function of α, and an equally simple analytical expression for the mean running time of `Walksat` (measured as the number of flips) at this optimal noise. This fit as well as our data exhibit linear scaling with N for α close to the phase transition region for 3-SAT. Second, we study the runtime distribution of `Walksat` on single instances and find first clear evidence of *power-law decay* in the probability of failure in T flips in the tail of the distribution. Power-law decays and heavy-tailed runtime distributions [3] have been one of the key observations for DPLL-style systematic search solvers and have led to methodologies such as rapid restarts and algorithm portfolios. This phenomenon, however, is usually not associated with local search. We show that after a (relatively long) "flat" region, the probability of failure decays exponentially in the high noise regime but as a power-law in the low noise regime. Third, we show that as `Walksat` proceeds, the number of unsatisfied clauses exhibits an interesting *gradual decay* that happens only at near-optimal noise.

A model that captures such features and is yet simple to describe and simulate can be a very useful tool for understanding and exploiting the tradeoffs inherent in local search. We therefore propose a preliminary Markov Chain model capturing, e.g., exponential scaling with N and power-law decay at low noise.

The kind of empirical study pursued here requires a significant computational power (e.g., 100,000 runs for some low noise levels to observe a clear trend). We used Yahoo!'s Apache Hadoop based M45 cloud computing platform with the net computational effort being equivalent to around 14 years of single CPU time.

We assume basic familiarity with SAT, CNF formulas, and local search solvers such as `Walksat`. N and M will denote the number of variables and clauses, resp., of a CNF formula F, with $\alpha = M/N$. A random 3-CNF formula is created by choosing, with repetition, M clauses of size 3 each uniformly at random. Ignoring the details of the "freebie move", the noise parameter $n \in [0, 1]$ (or in $[0\%, 100\%]$) of `Walksat` is essentially the probability with which it makes a (possibly uphill) random walk move, instead of a greedily chosen downhill or plateau move.

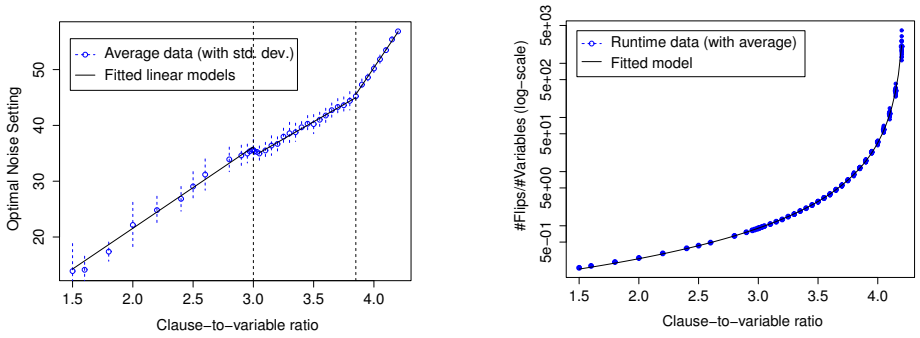

Fig. 1. Left: Linear model fitted to the optimal noise data (y-axis). Right: Model fitted to the mean runtime of `Walksat` (y-axis) at optimal noise. Both plotted as functions of clause-to-variable ratios (x-axis).

1 Local Search in SAT: Empirical Findings

1.1 Analytical Expressions for Optimal Noise and Mean Runtime

It has been observed that `Walksat` behaves quite predictably on large random formulas, in terms of the average (or median) running time or the noise levels that perform the best. For example, Seitz et al. [6] have given evidence that `Walksat` scales linearly when a noise value of 0.57 is used for formulas with $\alpha \approx 4.2$.

The left pane of Fig. 1 shows how the optimal noise $n*$ (y-axis) changes as α (x-axis) increases. For this experiment, we considered values of α in the range $[1.5, 4.2]$ and random 3-CNF formulas with N ranging from 100,000 to 400,000. For each (α, N) pair, we considered 10 formulas (the variation amongst formulas is not much at such large values of N) and did a binary search to estimate $n*$ for each formula up to a granularity of 0.1%. $n*$ values for these formulas, as one might expect, do *not* depend on N. The resulting average $n*$ and its standard deviation are plotted in the left pane of Fig. 1. We see three clear linear regimes, which can be fitted well with the following step-linear model, shown as the solid black line in the figure:

$$\text{OptNoise } (\%) \approx \begin{cases} -7.617 + 14.588\alpha & \text{for } \alpha \in [1.5, 3.0] \\ -2.558 + 12.347\alpha & \text{for } \alpha \in (3.0, 3.85) \\ -77.53 + 31.970\alpha & \text{for } \alpha \in [3.85, 4.2] \end{cases} \tag{1}$$

Interestingly, the transition points between the three regimes correspond to values of α that have been studied before. The first transition point corresponds to the threshold of $c_3 \approx 3.003$ which was proven by Frieze and Suen [2] to be the precise point up till which a simple "generalized unit clause" rule, GUC, almost surely solves random 3-SAT instances. The second transition point corresponds to the threshold of ≈ 3.9 up till which the purely greedy version of `Walksat`, namely `GSAT`, has been empirically seen to be successful (also the point at which the solution space structure is believed to change drastically [5]).

The right pane of Fig. 1 shows similar data for the average running time of `Walksat` at the optimal noise predicted by Eq. (1), for $\alpha \in [1.5, 4.2]$. In order to account for different formula sizes, we scaled the runtimes by dividing by N and found that this normalized runtime at optimal noise is indeed independent of N, implying linear scaling with N (as generally believed). We fit this curve with a model, depicted as a solid black line. For $\alpha < 4.2355$, our fitted model captures the linear scaling of the mean runtime with N at optimal noise. Specifically:

$$\frac{\text{numflips at OptNoise}}{N} \approx \frac{1.4358}{(4.2355 - \alpha)^{2.1894}} \tag{2}$$

1.2 Runtime Distribution: Exponential or Power-Law Decay

Fix α to 4.2 and consider the following question: *if we take a single formula F and perform several runs of `Walksat`, what is the distribution of the runtime needed to find a solution?* We measure this as the *probability of failure* as a function of the number of flips, i.e., what fraction of the runs fail to find a solution in T flips. The result for various noise levels for a 100,000 variable formula with $\alpha = 4.2$ is shown in Fig. 2, in both log-linear and log-log scales. For each low and high noise level, we performed 110,000 and 10,000 runs, resp., of `Walksat` with a cutoff of 100B (100×10^9) and 4B flips, resp. The median runtime to solve these formulas ranges between 140M to 700M flips at different noise levels.

The figure shows a clear qualitative distinction between the runtime distributions at high noise levels ($n > 0.567$, top row) and at low noise levels (bottom row). In particular, the log-linear plot in the top-left shows that the probability of failure for high noise decays exponentially with the number of flips: for large enough T, Pr[failure after T flips] $\approx \exp(-cT)$ for some constant c. On the other hand, the bottom-left plot, also in log-linear scale, shows that the decay is clearly slower than exponential for noise levels below the optimal. The bottom-right plot, showing relatively straight lines in the log-log scale, demonstrates that for low noise levels, the decay rate is very close to being a power-law (at the tail): for large enough T, Pr[failure after T flips] $\approx T^{-c'}$ for some $c' > 0$.

1.3 Evolution of the Number of Violated Clauses

Fig. 3 shows how the number of violated clauses, denoted $\hat{M}$, progresses when `Walksat` is run in a formula with 100,000 variables with $\alpha = 4.2$ at noise levels 0.50 (too low), 0.57 (optimal), and 0.58 (too high). For all noise levels considered, there is a very steep initial descent until $\hat{M}$ is under 2,500, or 2.5% of the original clauses. After this, the behavior changes drastically depending on the noise level. For optimal noise $n*$, $\hat{M}$ shows a *gradually decreasing, almost linear, trend* (with small random variations, as one would expect) until a solution is found, depicted by the middle curve in red. For high noise, $\hat{M}$ decreases rather fast in the beginning but then settles at a non-zero value that is higher the higher the noise is (the highest curve, in green). For low noise, $\hat{M}$ decreases very rapidly and settles at a relatively low but non-zero value (the lowest curve, in blue).

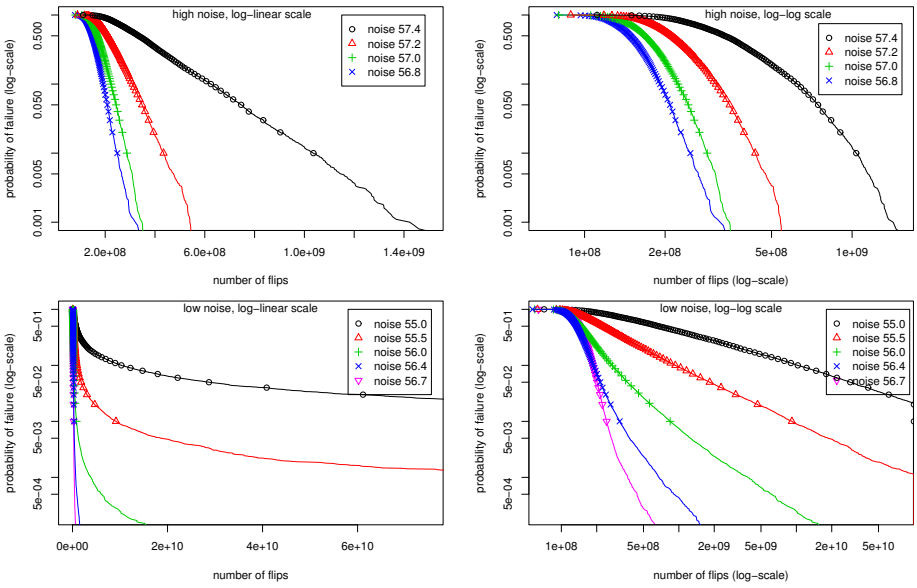

Fig. 2. Runtime distribution of `Walksat` on a large 3-CNF formula (100k variables, 420k clauses), with probability of failure on y-axis. Top: *high noise regime with exponential decay* (straight line in log-linear plot at top-left). Bottom: *low noise regime with power-law decay* (straight line in log-log plot at bottom-right).

2 Markov Chain Model Capturing Power-Law Decay

We briefly sketch a Markov Chain (MC) model that may shed light, at least qualitatively, on exponential scaling and power-law decay. The model has two parts (see Fig. 4). The first (top, horizontal) part is a linear MC with states corresponding to truth assignments that satisfy the same fraction of clauses of F, with the leftmost state encapsulating all solutions. Second, hanging from each state in the top chain is a "trap gadget", which captures the behavior of `Walksat` when it "gets lost" exploring parts of the search space without any solutions.

The probability of moving left or right in the top chain is determined by combining two effects. First, we assume that in the very high noise setting, the search will prefer to choose a neighboring state s with probability proportional to the number of truth assignments in s. Assuming a roughly binomial distribution of the number of assignments satisfying k clauses, the chain will then have a tendency to drift towards the middle, away from the solutions. Second, for relatively low noise settings, the search will be more focused and will choose among *immediately better* neighbors, i.e., either left or down into one of the trap gadgets. These two forces will balance in different ways for different noise settings. The trap gadget can itself be represented as an MC: two vertical linear chains, connected horizontally at each level (a "ladder", see Fig. 4). Entering this chain would first allow the search to either go down along the right side of the ladder

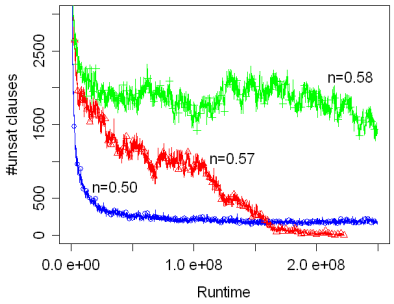

Fig. 3. Evolution of number of unsat. clauses at 3 noise levels

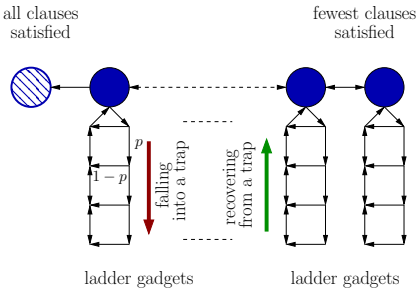

Fig. 4. A Markov Chain model for local search in SAT

(with probability p at each step), or across to the left side (probability $1-p$). This part models `Walksat`'s decision to either keep searching deeper in the trap (downward), or to try to escape (across). The left side of the ladder models the effort to escape a local minimum. If the search descends to level k of the trap before making the move across, it may become exponentially difficult for it (in k, say b^k steps) to escape, again depending on the noise; this b^k behavior is itself easily modeled by an MC biased downward while the search tries to go upward. Thus, once in a trap, the search will, with probability p^k, descend to depth k before initiating an escape move, and then spend $\approx b^k$ steps to get back to the top, yielding $\sum_k p^k \cdot b^k$ steps in expectation—and resulting in a power-law distribution, similar to previous analysis [1]. Thus, the cumulative runtime of the MC will be a sum of power-law distributions, in agreement with our data. On the other hand, for very high noise settings, the search will be successfully able to avoid traps, but will also be attracted towards the middle of the top chain, thus taking, in expectation, exponentially long to reach the left end (a solution).

References

[1] H. Chen, C. Gomes, and B. Selman. Formal models of heavy-tailed behavior in combinatorial search. In *7th CP*, 2001.

[2] A. Frieze and S. Suen. Analysis of two simple heuristics on a random instance of k-SAT. *J. Algorithms*, 20(2):312–355, 1996.

[3] C. P. Gomes, B. Selman, and H. Kautz. Boosting combinatorial search through randomization. In *15th AAAI*, pp. 431–437, Madison, WI, July 1998.

[4] H. H. Hoos and T. Stützle. Towards a characterization of the behavior of stochastic local search algorithms for SAT. *AI J.*, 112(1-2):213–232, 1999.

[5] M. Mézard, T. Mora, and R. Zecchina. Clustering of solutions in the random satisfiability problem. *Phy. Rev. Lett.*, 94:197205, May 2005.

[6] S. Seitz, M. Alava, and P. Orponen. Focused local search for random 3-satisfiability. *J. Stat. Mech.*, P06006, 2005.

[7] B. Selman, H. Kautz, and B. Cohen. Local search strategies for satisfiability testing. In D. S. Johnson and M. A. Trick, editors, *Cliques, Coloring and Satisfiability: the Second DIMACS Implementation Challenge*, vol. 26 of *DIMACS Series in DMTCS*, pp. 521–532. Amer. Math. Soc., 1996.