Characterizing the Run-time Behavior of Stochastic Local Search

Holger Hoos and Thomas Stützle

Intellektik!

Forschungsbericht AIDA–98–01

Darmstadt

Fachgebiet Intellektik, Fachbereich Informatik
Technische Hochschule Darmstadt
Alexanderstraße 10
D–64283 Darmstadt
Germany

Characterizing the Run-time Behavior of Stochastic Local Search

Holger H. Hoos and **Thomas Stützle**
Technische Universität Darmstadt
Fachgebiet Intellektik
Alexanderstr.10
D-64289 Darmstadt, Germany
{hoos,stuetzle}@informatik.tu-darmstadt.de

Abstract

Stochastic local search (SLS) algorithms have been successfully applied to hard combinatorial problems from different domains. One important feature of SLS algorithms is the fact that their run-time behavior is characterized by a random variable. Consequently, the detailed knowledge of the run-time distribution provides important information for the analysis of SLS algorithms. In this paper we investigate the empirical run-time distributions for several state-of-the-art stochastic local search algorithms for SAT and CSP. Using statistical analysis techniques, we show that on a variety of problems from both randomized distributions and encodings of the blocks world planning and graph coloring domains, the observed run-time behavior can be characterized by exponential distributions. As a first direct consequence of this result, we establish that these algorithms can be easily parallelized with optimal speedup.

Introduction

Recent successes in using stochastic local search (SLS) algorithms to practically solve hard combinatorial problems from various domains have stirred considerable interest and inspired a quickly growing body of research. Based on earlier SLS approaches (Gu 1992; Selman, Levesque, & Mitchell 1992) to solving the satisfiability problem in propositional logic (SAT), modern SLS algorithms like HSAT (Gent & Walsh 1993a), WSAT (Selman, Kautz, & Cohen 1994) or Novelty (McAllester, Selman, & Kautz 1997) do not only outperform state-of-the-art systematic search methods for a variety of problem classes, but have also shown to be competitive with the best general-purpose algorithms on domains like planning (Kautz & Selman 1996) or network routing (Jiang, Kautz, & Selman 1995). Similar SLS methods have been developed and successfully applied to the more general constraint satisfaction problem (CSP).

Because the theoretical understanding of the behavior of stochastic local search algorithms is still very limited, most of the work in this field is based on empirical methods and computational experiments. Despite the impressive successes that have been achieved, there has been some rather fundamental criticism regarding the empirical methodology that has been applied in most of the studies (Hooker 1996).

In this paper we introduce a more adequate empirical methodology for evaluating SLS algorithms. Instead of collecting simple statistics averaged over a large number of runs and large sets of instances, we are estimating and characterizing *run-length distributions* on single instances. This way a more accurate and realistic picture of the performance behavior is obtained. Concentrating on single instances is strongly motivated by the fact that the empirical hardness distributions for commonly used randomized problem distributions for both SAT and CSP have an extremely high variance. Consequently, instance sets sampled from these distributions are very inhomogeneous w.r.t. their hardness for the standard SLS algorithms.

Using this methodology we obtain some novel and surprisingly general empirical results concerning run-time behavior and parallelization of some of the most popular SLS algorithms. Our main result establishes that *on hard instances from a variety of randomized problem classes as well as encoded problems from other domains, the run-time behavior of some of the most popular SLS algorithms for both SAT and CSP can be characterized by exponential distributions.* This result has a number of significant implications, the most interesting of which might be the fact that *SLS algorithms displaying this type of behavior can be easily parallelized with optimal speedup.*

The paper is structured in the following way. After briefly reviewing basic notions concerning SLS algorithms, we introduce and motivate our empirical methodology. We then present empirical analyses of SLS algorithms on both randomized problem distributions and encoded problem instances from other domains which lead to our main result. Next, we discuss several interesting implications of this result. We conclude with a discussion of related work and indicate directions for further research.

SLS Algorithms for SAT and CSP

A discrete CSP is defined by a finite set of variables $\mathcal{X} = \{X_1, X_2, \ldots, X_n\}$, every variable X_i having an associated finite and discrete domain D_i, and a set of constraints $\mathcal{C}$. A *k-ary constraint* between variables $\{X_{i1}, \ldots, X_{ik}\}$ is a relation $C \subseteq D_{i1} \times \ldots \times D_{ik}$ specifying the admissible value combinations for the variables involved. SAT may be interpreted as a special case of CSP in which the variables only take two values 1 (*true*) and 0 (*false*). An assignment s is an instantiation of every variable with some value of its domain. We say a constraint (or, for SAT, a clause) is *violated* by a given assignment s if the k-tuple of values of its variables are not contained in the relation. We assume that the SAT problem is given in conjunctive normal form (CNF), i.e. as a conjunction of disjunctions of literals, a literal being a variable or its negation. We restrict ourselves to binary CSPs, while in the SAT instances we use always clauses with more than two variables.

Stochastic local search algorithms for SAT or CSP can be seen as operating on an objective function which maps assignments onto the number of unsatisfied clauses or violated constraints. These algorithms typically start with an initial complete assignment and perform a sequence of moves that lead from one assignment to another. The possible moves that are applicable to a given assignment are defined by a neighborhood function. SLS algorithms rely heavily on random decisions made during the search process. During their run several kinds of random decisions are made, such as the choice of the initial solution, randomly broken ties, or randomly biased moves being made with a certain probability. SLS methods can be characterized as *Las Vegas algorithms* (Alt *et al.* 1996; Luby, Sinclair, & Zuckerman 1993): An algorithm A is a Las Vegas algorithm if the output of A is always correct but its running time, T_A, is a random variable. Hence, to obtain knowledge on the runtime behavior of a Las Vegas algorithm, we need information on its run-time distribution, an issue which has not been addressed so far for SLS algorithms on SAT or CSP.

A generic stochastic local search procedure for SAT and CSPs is outlined in Figure 1. Two well known local search algorithms for SAT and CSPs are GSAT (Selman, Levesque, & Mitchell 1992) and the *min conflicts* heuristic (MCH) (Minton *et al.* 1992), respectively. Both heuristics start from an initial random assignment and perform hillclimbing on the number of violated constraints. If after *maxSteps* local hill-climbing steps no solution is found, the algorithm restarts the local search by choosing a new random initial assignment. If a given number *maxTries* of restarts fail to find a solution, the algorithm terminates unsuccessfully. GSAT and the *min conflicts* heuristic differ in the hill-climbing step and the neighborhood definition. In GSAT all variables are considered for a hill-climb step and a variable is flipped that maximally decreases the objective function value. The MCH selects in each move one variable involved in some violated constraint and chooses a value for this variable among those that minimize the number of constraint violations.

```
procedure Local Search
  for i := 1 to maxTries do
    s := random assignment for all X ∈ 𝒳;
    for j := 1 to maxSteps do
      if s is solution of 𝒞 then return s
      else
        P := possibleSteps(s, 𝒞);
        Step := random element of P;
        perform Step on s;
      end if;
    end for;
  end for;
  return "no solution found";
end Local Search;
```

Figure 1: General local search procedure applicable to SAT and CSPs.

A difficulty with local search methods is that they can get "stuck" in local minima of the objective function which do not represent solutions. To cope with that problem, several strategies that allow the algorithm to escape from local optima have been proposed. The most basic of these is to restart the local search from a new randomly chosen solution, also realized in the algorithm as stated above. An important extension of the basic algorithm is the "random walk" strategy (Selman, Kautz, & Cohen 1994) in GWSAT: Here, with a fixed probability wp instead of a standard hill-climb step a variable appearing in one of the currently unsatisfied clauses is flipped. Basic GSAT is a special case of GWSAT with $wp = 0$. Other extensions of GSAT are WSAT (Selman, Kautz, & Cohen 1994), Novelty and R-Novelty (McAllester, Selman, & Kautz 1997). For MCH, also random walk strategies (WMCH) similar to those for GSAT (Hao & Dorne 1996) and Tabu Search (TMCH) (Steinmann, Strohmaier, & Stützle 1997) can be applied. Today, Novelty and WSAT belong to the most effective algorithms for SAT. With respect to CSPs, the same holds for the WMCH and TMCH extensions of the *min conflicts* heuristic. For a detailed description of the different strategies we refer to the literature.

Empirical Methodology

Development and investigation of stochastic local search algorithms, particularly for SAT and CSPs, have given rise to a large number of empirical studies in the past few years. At the same time there has been some severe criticism of the methodology that has been applied in empirical computer science in general, and in empirical AI in particular (Hooker 1996; 1994). Since the theoretical understanding of the behavior of stochastic local search algorithms is still very limited, empirical methods are the most important means of studying these algorithms. Due to their significance in designing, evaluating, and understanding stochastic local search algorithms, empirical studies should be based on a thoroughly designed, adequate methodology.

Naturally, the methods required for evaluating SLS algorithms depend on the application scenario one has in mind. While, if the run-time is unlimited, the expected time to find a solution is a good performance criterion, for limited run-time the probability of finding a solution within the given time is the more relevant concept. In a more general application scenario where the utility (or usefulness) of a solution depends on the time that was actually needed to find it, the performance should be evaluated based on the functional dependence of the success probability on run-time which is equivalent to the run-time distribution (rtd). The rtd can be estimated by running the algorithm for a number of times and recording the time needed to find a solution for each successful run. The empirical distribution thus obtained approximates the rtd which in turn uniquely and completely characterizes the run-time behavior of the algorithm. Consequently, other statistics like expected run-time needed to find a solution, the standard deviation of this measure, or the expected success probability for a given time-limit can be easily obtained from the rtd.

Vice versa, given only an estimate for the mean and standard deviation of the run-time, there is no way of predicting success probabilities because the actual rtds might still be wildly different. As a simple example, take an exponential distribution $Exp(\lambda)$ with parameter $\lambda = 0.01$ and a normal distribution $N(\mu, \sigma)$ with $\mu = 100, \sigma = 100$. Both distributions have the same mean and standard deviation, but the 99% percentile of $Exp(\lambda)$ is ≈ 460 while for $N(\sigma, \mu)$ it is ≈ 333 meaning that the run-time to achieve 99% solution probability for algorithms characterized by the respective rtds are different by a factor of ≈ 1.38.

One method that has been commonly used for evaluating stochastic local search algorithms is to test them on a set of randomly generated problem instances and reporting statistics averaged over the results on the individual problem instances. In particular, this approach has been applied to problem classes for which phase transition phenomena have been observed, like Random-3-SAT at a clauses/variable ratio of approximately 4.26 or Random-CSP at a certain critical constrainedness. But actually, even when taken from the phase transition region the hardness of these problems for stochastic local search varies dramatically.

This can be seen in Fig. 2 which shows hardness distributions for samples from three random problem distributions. These distributions were obtained by running a stochastic local search algorithm to find 100 solutions on each problem instance and determining the average number of steps per solution as a rough estimate for the hardness of the instance. The algorithms we used are GWSAT with a walk probability of 0.6 (which turned out to be optimal for these problems) and MCH with random walk (WMCH) with an optimal walk probability of 0.1. The test sets were a set of Random-3-SAT instances with 100 variables and 430 clauses (obtained by unforced problem generation and filtering using a complete SAT solver), a set of Random-CSP instances with 20 variables with 10 possible values each, a constraint graph density of 0.5 and a constraint tightness of 0.38 (also obtained by unforced generation and filtering), and a set of SAT encoded graph coloring problems taken from the DIMACS Challenge database.[1] Each test set contained 100 instances.

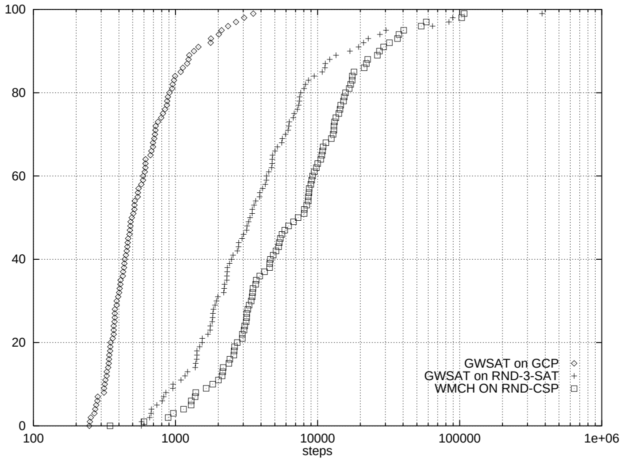

Figure 2: Hardness distributions for different problem classes using stochastic local search algorithms, see text for details.

As can be seen from Fig. 2, for all three problem classes there are differences in hardness covering several orders of magnitude. Note, that for both Random-3-SAT and Random-CSP the variability in hardness is significantly higher than for the more structured graph coloring instances.

Knowing nothing yet about the nature of the run-time distributions on individual problem instances, it already seems questionable whether it makes sense to evaluate algorithms on test-sets showing such hardness distributions by simply averaging over the results on individual instances. As we will show, the run-time behavior of a broad range of SLS algorithms on individual instances can be characterized by exponential distributions. But averaging over exponentially distributed random variables generally yields a random variable of a different distribution type (see (Rohatgi 1976) for details), so averaging statistics over test sets as described above will significantly influence the nature of the observations thus obtained. Therefore this practice seems even more questionable, especially if different algorithms are not tested on exactly identical

[1] DIMACS benchmark instances can be downloaded via ftp from `dimacs.rutgers.edu/pub/challenge/`

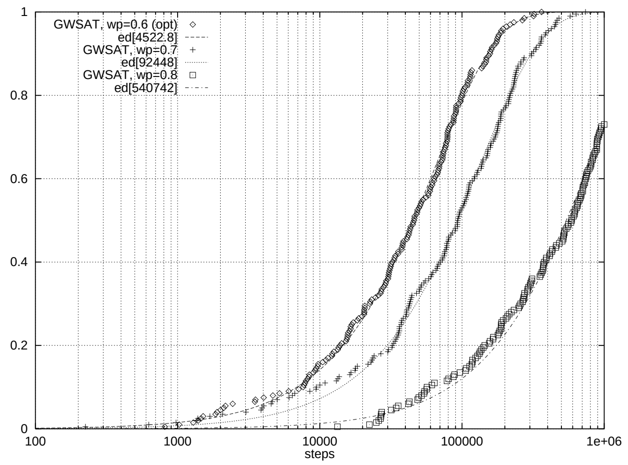

Figure 3: Run length distributions for GWSAT with various $wp \geq wp_{opt}$ on a hard Random-3-SAT instance (log scale representation on x-axis).

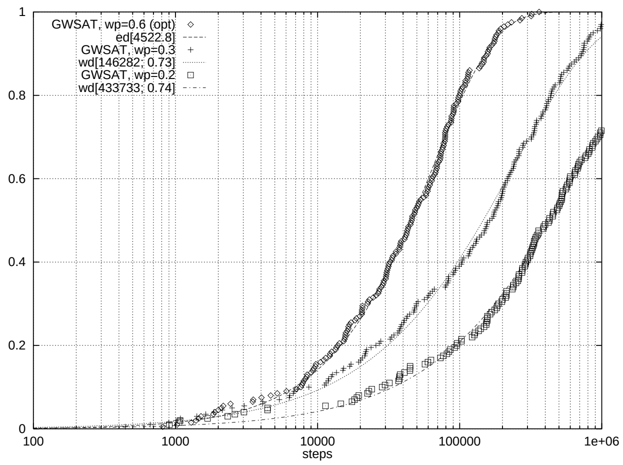

Figure 4: Run length distributions for GWSAT with various $wp \leq wp_{opt}$ on a hard Random-3-SAT instance.

test sets. Consequently, to obtain a realistic impression of the run-time characteristics in this paper we are only reporting rlds for single problem instances.

Results on Random Problems

As motivated in the previous section, we did all our empirical analyses on single problem instances. In order to abstract from machine and implementation dependent factors, we measured run-length distributions (rlds) instead of run-time distributions. Note, that for the algorithms considered here, on each problem instance, the CPU time per local search step is roughly constant. Practically, the empirical rlds are determined by running the respective SLS algorithm for k times and recording for each successful run the number of steps required to find a solution (run length). To get reasonable estimates, we chose k between 200 and 1000. We are not using the restart feature for these measurements (by setting $maxTries = 1$) because we are interested in the behavior of the (pure) SLS strategies without restart, which can be easily added to any SLS algorithm. Note, that due to the nature of the rlds, by using sufficiently high values of $maxSteps$ (10^6 or more), we can guarantee that almost every run is successful. If $rl(j)$ denotes the run-length for the jth successful run, the cumulative empirical rld is defined by $P(rl \leq i) = \#\{j|rl(j) \leq i\}/k$.

Fig. 3 shows the rlds for GWSAT with walk probabilities ≥ 0.6 on an instance from the hard end of the complexity distribution for Random-3-SAT with 100 variables and 430 clauses shown in Fig. 2. For the whole problem distribution the optimal walk probability has been shown to be $wp_{opt} = 0.6$. After determining the rlds by sampling run-lengths over 200 runs (in the graphic, each data point represents one run) we successfully approximated[2] the empirical data using the cumulative form of an exponential distribution $ed[d](x) = 1 - 2^{-x/d}$ where d is the half-time of the distribution and x the number of steps required to find a solution.[3] To verify the statistical significance of the approximation we further did standard χ^2-tests (Rohatgi 1976) for all functional approximations of rlds. All approximations were accepted by the respective χ^2-test at the $\alpha = 0.05$ level. Similar experiments were done both for many other Random-3-SAT instances from the 100 variables / 430 clauses distribution as well as for other instance sizes at or near the phase transition.

Similar experiments for sub-optimal walk probabilities $0 < wp \leq wp_{opt}$ show that the respective rlds can be approximated by Weibull distributions $wd[d, \alpha](x) = 1 - 2^{-(x/d)^{\alpha}}$ with $\alpha < 1$ (see Fig. 4). However, this does not hold for $wp = 0$, i.e., for plain GSAT, where the asymptotic behavior is entirely different from the cases $wp > 0$.

In Fig. 5 we give rlds for WMCH (for 1000 runs) obtained on randomly generated CSP instances of different hardness, taken from the lower part, the medium part and the upper part of the distribution in Figure 2. We refer to these instances as *easy*, *med*, and *hard*. For all three instances the rld is well approximated by an exponential distribution. Only for the *easy* and *med* instance, the empirical distributions show a small deviation for short runs. These deviations in the lower part of the curve are also the reason that for the easiest instance the χ^2-test rejected the approximation at the $\alpha = 0.05$ level, whereas for the harder instances the approximations by an exponential distribution passed

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

[3] 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 $ed[d]$ using $d = \ln 2/\lambda$. A similar argument applies to the Weibull distribution which is mentioned later.

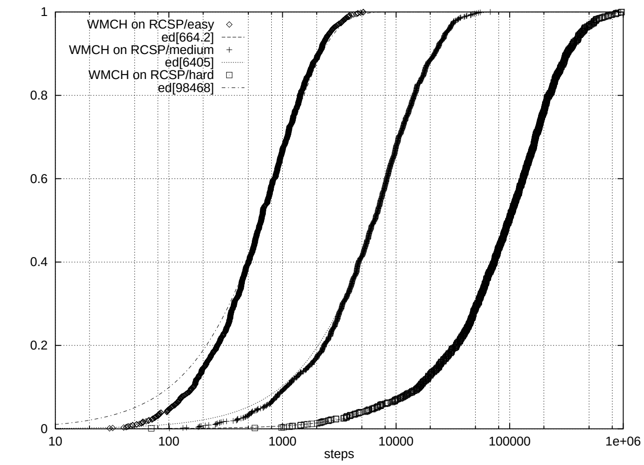

Figure 5: Run length distributions for WMCH with wp=0.1 (optimal setting) on Random CSP instances.

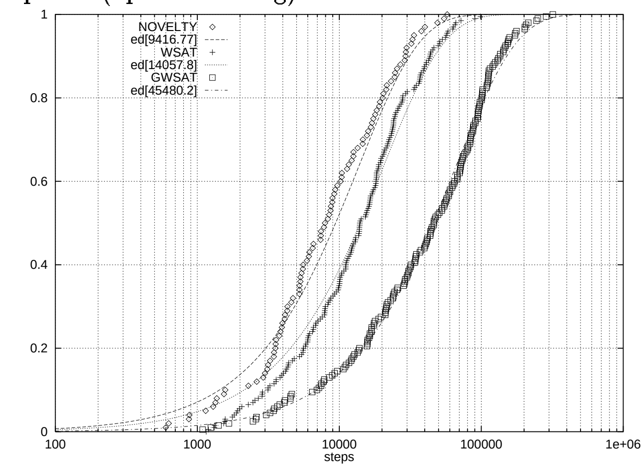

Figure 6: Run length distributions for Novelty, WSAT, and GWSAT on blocks world planning instance, see text for details.

the test. We conjecture that the deviations are caused by the initial hill-climb phase of the local search as shown in (Gent & Walsh 1993b) for SAT.

Summarizing, for instances from the hard random problem distributions analyzed above, we get the following result: *rlds for GWSAT and WMCH with optimal or super-optimal walk probability can be approximated by exponential distributions.* All experiments on other hard random CSP and SAT instances confirmed this observation. The same result was established for GSAT with tabu-lists and TMCH (given optimal tabu-list lengths), and furtheron for WSAT and Novelty.

Results on Encoded Problems

Applying the methodology described in the last section to SAT- or CSP-encoded problems from other domains we obtain similar results as described for the random problem distributions.

Fig. 6 shows rlds for three different SLS algorithms, GWSAT, WSAT, and Novelty on a SAT encoded plan-

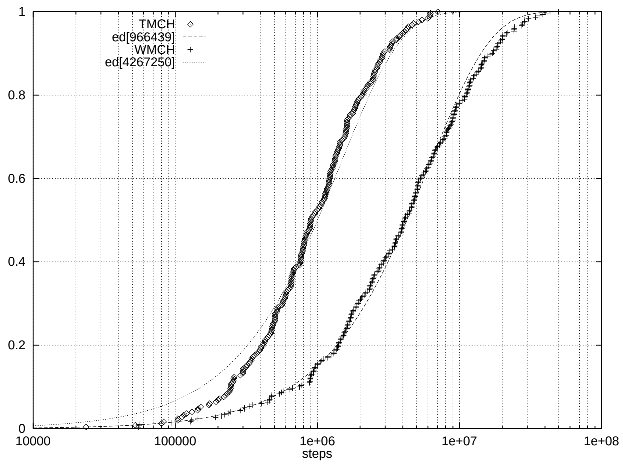

Figure 7: Run length distributions for TMCH and WMCH on graph coloring problem, see text for details.

ning problem from the blocks-world domain.[4] For each algorithm we used 200 runs to approximate the actual rlds as described before. Again, our approximations using exponential distributions passed a standard χ^2-Test. For all algorithms an approximately optimal setting of the walk or noise parameter was used. From the rld graphs it can easily be seen that while the shape of the rlds is the same for all algorithms, the difference in their run-time behavior can be characterized by a left-shift of the rld which corresponds to a speedup of a constant factor. For the SAT-encoded planning problem this factor is about 1.5 between Novelty and WSAT, and about 3 between WSAT and GWSAT.

Apart from SAT-encoded planning problems we also tested WMCH and TMCH on graph coloring problems. In Fig. 7 we present the rlds for WMCH and TMCH on the graph coloring instance `DSJC125.5.col`. This instance is taken from the DIMACS Benchmark set and comprises 125 nodes. The best known solution for this problem needs 17 colors and is conjectured to be optimal. Thus, we solved this instance for 17 colors. The rlds are obtained from 250 runs of both algorithms. Again they can be reasonably well approximated by an exponential distribution (although the approximation for TMCH passed the χ^2-test only at the 0.001 level due to the deviations for short runs). Analogous results have also been obtained for SAT-encoded graph coloring instances (whose hardness distribution is shown in 2) and various encodings of the Hamilton Circuit Problem (Hoos 1996) using GWSAT, and WSAT, and Novelty.

[4] We used the problem instance `bw_large.a` from (Kautz & Selman 1996) in the SAT formulation obtained by linear encoding and simplifying. Similar results were obtained for other problem instances from the same domain.

Interpretation

As shown in the previous sections, for optimal walk probability (or tabu list length), the rlds for various stochastic local search algorithms for SAT and CSP on a variety of both randomly generated and encoded problem instances can be approximated by exponential distributions. As an immediate consequence these algorithms, although being incomplete in general, are *approximately complete* in the sense that for arbitrarily long runs a solution will be found with a probability approaching 1. The important point here is that this result holds for each individual run and *without using random restart.*[5] Intuitively, it means that these algorithms cannot get stuck forever in local maxima.

As another consequence of the rlds being exponentially distributed, for those algorithms random restart is ineffective for optimal parameter settings. In practice this can be easily verified over a broad range of *maxSteps* settings. However, due to small deviations at the extreme left end of the rlds, for very low *maxSteps* settings the algorithms' performance usually deteriorates. For suboptimal settings of the noise parameter, as shown in Fig. 4, the rlds can be approximated by Weibull distributions $wd[d, \alpha]$ with $\alpha < 1$. In this situation, random restart is effective and an optimal *maxSteps* setting exists. On some very easy instances we also observed rlds which are best approximated by Weibull distributions with $\alpha > 1$. This was observed especially for SLS algorithms based on Tabu Search. Here, restarting the algorithm is even harmful and best performance is obtained by not using restart at all.

Another interpretation of our results concerning the rld of GSAT and MCH on individual SAT and CSP instances concerns the speedup obtainable in case of parallel processing. A straightforward (and probably the most basic) approach to parallel processing is to perform independent runs of an algorithm in parallel. The speedup one hopes to achieve is based on the following well known observation from probability theory: If for a given algorithm the probability of finding a solution in t time units is distributed exponentially $ed[d]$, then the probability of finding a solution in p independent runs of time t is distributed $ed[d/p]$.

Consequently, if we run such an algorithm once for time t we get exactly the same success probability as when running the algorithm p times for time t/p. Recall, that speedup in parallel processing is defined as $s = \frac{sequential\ time}{parallel\ time}$. Thus, for an exponential runtime distribution an optimal speedup is obtained, as $\frac{t}{t/p} = p$. As we have shown before, for an optimal setting of the noise parameters (walk probability or length of tabu-list), the rlds for a number of SAT and CSP algorithms can be characterized by an exponential distribution. Since, in all these cases the run-time distributions can be obtained form the rlds by simple rescaling, an optimal speedup can be realized by a very simple parallelization paradigm using multiple independent runs of local search algorithms. Due to the deviations for short runs (as can be seen for the easy instance in Fig. 6 or for TMCH in Fig. 7), which we conjecture to be caused by the initial hill-climb phase of local search, only for very high degrees of parallelization the speedup would be less than optimal.

[5] Using random restart produces the same phenomenon for *maxTries* $\rightarrow \infty$, only then convergence of the success probability towards 1 is in general drastically slower than what we observed in our experiments without using restart.

Conclusions

In this paper we have characterized the run length distribution for several state-of-the-art stochastic local search algorithms for SAT and CSP. We have shown that for optimal noise parameter settings the run length distribution for hard single instances can be approximated by exponential distributions. This approximation can be found for many popular algorithms for SAT and CSPs on randomly generated and structured problems of varying size. The goodness-of-fit of our approximation was shown to be statistically significant using a statistical hypothesis test, the χ^2-test. Knowing the run length distribution gives a more accurate picture of algorithm performance and allows to perform a more accurate statistical comparison of competing SLS algorithms.

Due to the enormous variability of the hardness of randomly generated instances, we concentrated on single instances. For sets of randomly generated problems we presented their hardness distributions; for SLS algorithms these show an enormous variability of problem hardness. In related work in (Frost, Rish, & Vila 1997) it is shown that for complete search algorithms the hardness of CSPs and SAT can roughly be approximated by a Weibull distribution for solvable instances and by a log-normal distribution for unsolvable instances. For SLS we could not find acceptable approximations using Weibull distributions.

Our main result also explains observations made, for example, in (Gent & Walsh 1995; Parkes & Walser 1996). There the authors find that using "random walk" in GSAT or WSAT makes the algorithm less sensitive to particular settings of *maxSteps.* They also report difficulties in finding sharply determined optimal values for *maxSteps.* This is consistent with our findings that for optimal noise parameter settings the rld for single instances can be approximated by an exponential distribution. In particular, this implies that *maxSteps* could be set to an arbitrary value, e.g. to ∞, without loss of efficiency. That in earlier studies "optimal" values for *maxSteps* could be established, might be due to the well known fact that the sum of exponentially distributed random variables is itself generally *not* exponentially distributed.

We also found that for noise parameter settings that are too high, the run length distribution can again be approximated by exponential distributions. In our figures this can be noticed by a right-shift of the rlds (note the use of a log-scale on the x-axis) that corresponds to a multiplication of the corresponding run-lengths by a constant factor. For noise parameter settings lower than optimal, we found that the run length distribution can be approximated by Weibull distributions.

For optimal (and higher than optimal) noise parameter settings our observation directly implies that independent execution of a given algorithm on several processors yields an optimal speedup. Yet, due to minor deviations in the lower part of the approximation – we conjecture these to be caused by the initial hill-climb phase of the local search – for a very high degree of parallelization the actual speedup will be suboptimal. If the noise parameter settings are too low, we would even obtain a "super-linear" speedup compared to an algorithm without restart. In this case, for a sequential version of a SLS algorithm restarting after some given fixed number of *maxSteps* pays off. Consequently, there exist optimal settings for *maxSteps* in case of, for example, plain GSAT. These optimal values of *maxSteps* can also be determined from the run length distribution.

Our experience shows that representing experimental results via rlds gives a better view of the data than just using averages and standard deviations. One strong motivation for investigating rlds stems from the fact that the run length of stochastic local search algorithms is actually a random variable. It is well known from statistics that the best way of characterizing a random variable is by its distribution function. One could object that obtaining rlds requires a higher computational effort. Yet, to obtain an accurate estimate of the average performance of an algorithm on given problem instances a high number of test runs is necessary. Exactly the same data can be used to estimate the rld.

There are many interesting ways of extending our work. One important extension involves the identification of run length distributions for optimization problems. To this aim one would need to consider the quality of the solutions as an additional parameter. Our results also pose questions on possible theoretical explanations of the observed run length distributions and might ultimately contribute to developing a comprehensive theory of stochastic local search.

References

Alt, H.; Guibas, L.; Mehlhorn, K.; Karp, R.; and Wigderson, A. 1996. A Method for Obtaining Randomized Algorithms with Small Tail Probabilities. *Algorithmica* 16:543–547.

Frost, D.; Rish, I.; and Vila, L. 1997. Summarizing CSP Hardness with Continuous Probability Distributions. In *Proceedings of AAAI'97*, 327–333.

Gent, I. P., and Walsh, T. 1993a. Towards an Understanding of Hill–Climbing Procedures for SAT. In *AAAI'93*, 28–33. MIT press.

Gent, I., and Walsh, T. 1993b. An Empirical Analysis of Search in GSAT. *Journal of Artificial Intelligence Research* 1:47–59.

Gent, I., and Walsh, T. 1995. Unsatisfied Variables in Local Search. In Hallam, J., ed., *Hybrid Problems, Hybrid Solutions*, 73–85. Amsterdam: IOS Press.

Gu, J. 1992. Efficient Local Search for very Large-Scale Satisfiability Problems. *SIGART Bulletin* 3:8–12.

Hao, J.-K., and Dorne, R. 1996. Empirical Studies of Local Search Procedures for Constraint Solving. In *Constraint Programming 1996*, volume 1118 of *LNCS*, 194–208. Springer Verlag.

Hooker, J. 1994. Needed: An Empirical Science of Algorithms. *Operations Research* 42(2):201–212.

Hooker, J. 1996. Testing Heuristics: We Have It All Wrong. *Journal of Heuristics* 33–42.

Hoos, H. H. 1996. Solving Hard Combinatorial Problems with GSAT — A Case Study. In *KI-96*, volume 1137 of *LNAI*, 107–119. Springer Verlag.

Jiang, Y.; Kautz, H.; and Selman, B. 1995. Solving Problems with Hard and Soft Constraints Using a Stochastic Algorithm for MAX-SAT. In *Proceedings of the 1st Workshop on Artificial Intellignce and Operations Research.*

Kautz, H., and Selman, B. 1996. Pushing the Envelope: Planning, Propositional Logic, and Stochastic Search. In *AAAI'96*, volume 2, 1194–1201. MIT Press.

Luby, M.; Sinclair, A.; and Zuckerman, D. 1993. Optimal Speedup of Las Vegas Algorithms. *Information Processing Letters* 47:173–180.

McAllester, D.; Selman, B.; and Kautz, H. 1997. Evidence for Invariants in Local Search. In *Proceedings of AAAI'97*, 321–326.

Minton, S.; Johnston, M.; Philips, A.; and Laird, P. 1992. Minimizing Conflicts: A Heuristic Repair Method for Constraint Satisfaction and Scheduling Problems. *Artificial Intelligence* 52:161–205.

Parkes, A., and Walser, J. 1996. Tuning Local Search for Satisfiability Testing. In *AAAI'96*, 356–362.

Rohatgi, V. 1976. *An Introduction to Probability Theory and Mathematical Statistics.* John Wiley & Sons.

Selman, B.; Kautz, H. A.; and Cohen, B. 1994. Noise Strategies for Improving Local Search. In *AAAI'94*, 337–343. MIT press.

Selman, B.; Levesque, H.; and Mitchell, D. 1992. A New Method for Solving Hard Satisfiability Problems. In *AAAI'92*, 440–446. MIT press.

Steinmann, O.; Strohmaier, A.; and Stützle, T. 1997. Tabu Search vs. Random Walk. In *Advances in Artificial Intelligence (KI97)*, volume 1303 of *LNAI*, 337–348. Springer Verlag.