Tuning Local Search for Satisfiability Testing*

Andrew J. Parkes
CIS Dept. and CIRL
1269 University of Oregon
Eugene, OR 97403-1269
U.S.A.
parkes@cirl.uoregon.edu

Joachim P. Walser
Programming Systems Lab
Universität des Saarlandes
Postfach 151150, 66041 Saarbrücken
Germany
walser@cs.uni-sb.de

Abstract

Local search algorithms, particularly GSAT and WSAT, have attracted considerable recent attention, primarily because they are the best known approaches to several hard classes of satisfiability problems. However, replicating reported results has been difficult because the setting of certain key parameters is something of an art, and because details of the algorithms, not discussed in the published papers, can have a large impact on performance. In this paper we present an efficient probabilistic method for finding the optimal setting for a critical local search parameter, Maxflips, and discuss important details of two differing versions of WSAT. We then apply the optimization method to study performance of WSAT on satisfiable instances of Random 3SAT at the crossover point and present extensive experimental results over a wide range of problem sizes. We find that the results are well described by having the optimal value of Maxflips scale as a simple power of the number of variables, n, and the average run time scale sub-exponentially (basically as $n^{\log(n)}$) over the range $n = 25, \ldots, 400$.

INTRODUCTION

In recent years, a variety of local search routines have been proposed for (Boolean) satisfiability testing. It has been shown (Selman, Levesque, & Mitchell 1992; Gu 1992; Selman, Kautz, & Cohen 1994) that local search can solve a variety of realistic and randomly generated satisfiability problems much larger than conventional procedures such as Davis-Putnam.

The characteristic feature of local search is that it starts on a total variable assignment and works by repeatedly changing variables that appear in violated constraints (Minton *et al.* 1990). The changes are typically made according to some hill-climbing heuristic strategy with the aim of maximizing the number of satisfied constraints. However, as is usual with hill-climbing, we are liable to get stuck on local maxima. There are two standard ways to overcome this problem: "noise" can be introduced (Selman, Kautz, & Cohen 1994); or, with a frequency controlled by a *cutoff* parameter Maxflips we can just give up on local moves and restart with some new assignment. Typically both of these techniques are used together but their use raises several interesting issues:

1. What value should we assign to Maxflips? There are no known fundamental rules for how to set it, yet it can have a significant impact on performance and deserves optimization. Also, empirical comparison of different procedures should be done fairly, which involves first optimizing parameters.
2. If we cannot make an optimal choice then how much will performance suffer?
3. How does the performance scale with the problem size? This is especially important when comparing local search to other algorithm classes.
4. Local search routines can fail to find a solution even when the problem instance is actually satisfiable. We might like an idea of how often this happens, i.e. the false failure rate under the relevant time and problem size restrictions.

At present, resolving these issues requires extensive empirical analysis because the random noise implies that different runs can require very different runtimes even on the same problem instance. Meaningful results will require an average over many runs. In this paper we give a probabilistic method to reduce the computational cost involved in Maxflips optimization and also present scaling results obtained with its help.

The paper is organized as follows: Firstly, we discuss a generic form of a local search procedure, and present details of two specific cases of the WSAT algorithm (Selman, Kautz, & Cohen 1994). Then we describe the optimization method, which we call "*retrospective parameter variation*" (RPV), and show how it allows data collected at one value of Maxflips to be reused to produce runtime results for a range of values. We note that the same concept of RPV can also be used to study the effects of introducing varying amounts of

*This work has been supported by ARPA/Rome Labs under contracts F30602-93-C-0031 and F30602-95-1-0023 and by a doctoral fellowship of the DFG to the second author (Graduiertenkolleg Kognitionswissenschaft).

parallelization into local search by simulating multiple threads (Walser 1995).

Finally, we present the results of experiments to study the performance of the two versions of WSAT on Random 3SAT at the crossover point (Cheeseman, Kanefsky, & Taylor 1991; Mitchell, Selman, & Levesque 1992; Crawford & Auton 1996). By making extensive use of RPV, and fast multi-processor machines, we are able to give results up to 400 variables. We find that the optimal Maxflips setting scales as a simple monomial, and the mean runtime scales subexponentially, but faster than a simple power-law.

LOCAL SEARCH IN SAT

Figure 1 gives the outline of a typical local search routine (Selman, Levesque, & Mitchell 1992) to find a satisfying assignment for a set of clauses α[1].

```
proc Local-Search-SAT
    Input clauses $\alpha$, Maxflips, and Maxtries
    for $i := 1$ to Maxtries do
        $A :=$ new total truth assignment
        for $j := 1$ to Maxflips do
            if $A$ satisfies $\alpha$ then return $A$
            $P :=$ select-variable($\alpha, A$)
            $A := A$ with $P$ flipped
        end
    end
    return "No satisfying assignment found"
end
```

Figure 1: A generic local search procedure for SAT.

Here, local moves are "flips" of variables that are chosen by *select-variable*, usually according to a randomized greedy strategy. We refer to the sequence of flips between restarts (new total truth assignments) as a "try", and a sequence of tries finishing with a successful try as a "run". The parameter Maxtries can be used to ensure termination (though in our experiments we always set it to infinity). We also assume that the *new* assignments are all chosen randomly, though other methods have been considered (Gent & Walsh 1993).

Two WSAT Procedures

In our experiments, we used two variants of the WSAT - "walk" satisfiability class of local search procedures. This class was introduced by Selman et. al. (Selman, Kautz, & Cohen 1994) as "WSAT makes flips by first randomly picking a clause that is not satisfied by the current assignment, and then picking (either at random or according to a greedy heuristic) a variable within that clause to flip." Thus WSAT is a restricted version of Figure 1 but there remains substantial freedom in the choice of heuristic. In this paper we focus on the two selection strategies, as given in Figure 2. Here, f_P denotes the number of clauses that are *fixed* (become satisfied) if variable P is flipped. Similarly, b_P is the number that *break* (become unsatisfied) if P is flipped. Hence, $f_P - b_P$ is simply the net increase in the number of satisfied clauses.

[1] A clause is a disjunction of literals. A literal is a propositional variable or its negation.

```
                    WSAT/G
proc select-variable($\alpha, A$)
    $C :=$ a random unsatisfied clause
    with probability $p$ :
            $S :=$ random variable in $C$
        probability $1 - p$ :
            $S :=$ variable in $C$ with maximal $f_P - b_P$
    end
    return $S$
end
                    WSAT/SKC
proc select-variable($\alpha, A$)
    $C :=$ a random unsatisfied clause
    $u := \min_{S \in C} b_S$
    if $u = 0$ then
        $S :=$ a variable $P \in C$ with $b_P = 0$
    else
      with probability $p$ :
            $S :=$ random variable in $C$
        probability $1 - p$ :
            $S :=$ variable $P \in C$ with minimal $b_P$
      end
    return $S$
end
```

Figure 2: Two WSAT variable selection strategies. Ties for the best variable are broken at random.

The first strategy[2] WSAT/G is simple hillclimbing on the net number of satisfied clauses, but perturbed by noise because with probability p, a variable is picked randomly from the clause. The second procedure WSAT/SKC is that of a version of WSAT by Cohen, Kautz, and Selman.[3] We give the details here because they were not present in the published paper (Selman, Kautz, & Cohen 1994), but are none-the-less rather interesting. In particular, WSAT/SKC uses a less obvious, though very effective, selection strategy. Firstly, hill-climbing is done solely on the number of clauses that *break* if a variable is flipped, and the number of clauses that get fixed is ignored. Secondly, a random move is never made if it is possible to do a move in which no previously satisfied clauses become broken. In all it exhibits a sort of "minimal greediness", in that it definitely fixes the one randomly selected clause but otherwise merely tries to minimize the damage to the already satisfied clauses. In contrast, WSAT/G is greedier and will blithely cause lots of damage if it can get paid back by other clauses.

[2] Andrew Baker, personal communication.

[3] Publically available, ftp://ftp.research.att.com/dist/ai

RETROSPECTIVE VARIATION OF MAXFLIPS

We now describe a simple probabilistic method for efficiently determining the Maxflips dependence of the mean runtime of a randomized local search procedure such as WSAT. We use the term "retrospective" because the parameter is varied after the actual experiment is over.

As discussed earlier, a side-effect of the randomness introduced into local search procedures is that the runtime now varies between runs. It is often the case that this "*inter-run*" variation is large and to give meaningful runtime results we need an average over many runs. Furthermore, we will need to determine the mean runtime over a range of values of Maxflips. The naive way to proceed is to do totally independent sets of runs at many different Maxflips values. However, this is rather wasteful of data because the successful try on a run often uses many fewer flips than the current Maxflips, and so we should be able to re-use it (together with the number of failed tries) to produce results for smaller values of Maxflips.

Suppose we take a *single* problem instance ν and make many runs of a local search procedure with Maxflips=m_D, resulting in a sample with a total of N tries. The goal is to make predictions for Maxflips = $m < m_D$. Label the *successful* tries by i, and let x_i be the number of flips it took i to succeed. Write the bag of all successful tries as $S_0 = \{x_1, \ldots, x_l\}$ and define a reduced bag S_0^m by removing tries that took longer than m

$$S_0^m := \{x_i \in S_0 \mid x_i \leq m\}. \qquad (1)$$

We are asssuming there is randomness in each try and no learning between tries so we can consider the tries to be independent. From this it follows that the new bag provides us with information on the distribution of flips for successful tries with Maxflips=m. Also, an estimate for the probability that a try will succeed within m flips is simply

$$p_m \approx \frac{|S_0^m|}{N} \qquad (2)$$

Together S_0^m and p_m allow us to make predictions for the behaviour at Maxflips=m.

In this paper we are concerned with the expected (mean) number of flips $E_{\nu,m}$ for the instance ν under consideration. Let $\overline{S_0^m}$ be the mean of the elements in the bag S_0^m. With probability p_m, the solution will be found on the first try, in which case we expect $\overline{S_0^m}$ flips. With probability $(1 - p_m)p_m$, the first try will fail, but the second will succeed, in which case we expect $m + \overline{S_0^m}$ flips, and so on. Hence,

$$E_{\nu,m} = \sum_{k=0}^{\infty}(1 - p_m)^k \, p_m \, (k\, m + \overline{S_0^m}) \qquad (3)$$

which simplifies to give the main result of this section

$$E_{\nu,m} = (1/p_m - 1)\, m + \overline{S_0^m} \qquad (4)$$

with p_m and $\overline{S_0^m}$ estimated from the reduced bag as defined above. This is as to be expected since $1/p_m$ is the expected number of tries. It is clearly easy to implement a system to take single data-set obtained at Maxflips=m_D, and estimate the expected number of flips for many different smaller values of m.

Note that it might seem that a more direct and obvious method would be to take the bag of all *runs* rather than *tries*, and then simply discard runs in which the final try took longer than m. However, such a method would discard the information that the associated tries all took longer than m. In contrast, our method captures this information: the entire population of tries is used.

Instance Collections To deal with a collection, C, of instances we apply RPV to each instance individually and then proceed exactly as if this retrospectively simulated data had been obtained directly. For example, the expected mean number of flips for C is

$$E_m = \frac{1}{|C|} \sum_{\nu \in C} E_{\nu,m}. \qquad (5)$$

Note that this RPV approach is not restricted to means, but also allows the investigation of other statistical measures such as standard deviation or percentiles.

Practical Application of RPV The primary limit on the use of RPV arises from the need to ensure that the bag of successful tries does not get too small and invalidate the estimate for p_m. Since the bag size decreases as we decrease m it follows that there will be an effective lower bound on the range over which we can safely apply RPV from a given m_D. This can be offset by collecting more runs per instance. However, there is a tradeoff to be made: If trying to make predictions at too small a value of m it becomes more efficient to give up on trying to use the data from Maxflips=m_D and instead make a new data collection at smaller Maxflips. This problem with the bag size is exacerbated by the fact that different instances can have very different behaviours and hence different ranges over which RPV is valid. It would certainly be possible to do some analysis of the errors arising from the RPV. The data collection system could even do such an analysis to monitor current progress and then concentrate new runs on the instances and values of Maxflips for which the results are most needed. In practice, we followed a simpler route: we made a fixed number of runs per instance and then accepted the RPV results only down to values of m for some fixed large fraction of instances still had a large enough bagsize.

Hence, RPV does not always remove the need to consider data collection at various values of Maxflips, however, it does allow us to collect data at more widely separated Maxflips values and then interpolate between the resulting "direct" data points: This saves a time-consuming fine-grained data-collection, or binary search through Maxflips values.

EXPERIMENTAL RESULTS

To evaluate performance of satisfiability procedures, a class of randomized benchmark problems, Random 3SAT, has been studied extensively (Mitchell, Selman, & Levesque 1992; Mitchell 1993; Crawford & Auton 1996). Random 3SAT provides a ready source of hard scalable problems. Problems in random k-SAT with n variables and l clauses are generated as follows: a random subset of size k of the n variables is selected for each clause, and each variable negated with probability 1/2. If instances are taken from near the crossover point (where 50% of the randomly generated problems are satisfiable) then the fastest systematic algorithms, such as TABLEAU (Crawford & Auton 1996), show a well-behaved increase in hardness: time required scales as a simple exponential in n.

In the following we present results for the performance of both variants of WSAT on satisfiable Random 3SAT problems at the crossover point. We put particular emphasis on finding the Maxflips value m^* at which the mean runtime averaged over all instances is a minimum. Note that the clause/variable ratio is not quite constant at the crossover point but tends to be slightly higher for small n. Hence, to avoid "falling off the hardness peak", we used the experimental results (Crawford & Auton 1996) for the number of clauses at crossover, rather than using a constant value such as $4.3n$. To guarantee a fair sample of satisfiable instances we used TABLEAU to filter out the unsatisfiable instances. At n=400 this took about 2-4 hours per instance, and so even this part was computationally non-trivial, and in fact turned out to be the limiting factor for the maximum problem size.

For WSAT/SKC, the setting of the noise parameter p has been reported to be optimal between 0.5 and 0.6 (Selman, Kautz, & Cohen 1994). We found evidence that such values are also close to optimal for WSAT/G, hence we have produced all results here with $p = 0.5$ for both WSAT variants, but will discuss this further in the next section.

We present the experiments in three parts. Firstly, we compare the two variants of WSAT using a small number of problem instances but over a wide range of Maxflips values to show the usage of RPV to determine their Maxflips dependencies. We then concentrate on the more efficient WSAT/SKC, using a medium number of instances, and investigate the scaling properties over the range $n = 25, \ldots, 400$ variables. This part represents the bulk of the data collection, and heavily relied on RPV. Finally, we look at a large data sample at $n = 200$ to check for outliers.

Overall Maxflips Dependence

Our aim here is to show the usage of RPV and also give a broad picture of how the mean runtime E_m varies with m for the two different WSAT variants. We took a fixed, but randomly selected, sample of 10^3 instances at $(n, l) = (200, 854)$, for which we made 200 runs on each instance at m=5k, 20k, 80k, 800k, 8000k, ∞ using both variants of WSAT. At each m we directly calculated E_m. We then applied RPV to the samples to extend the results down towards the next data-point.

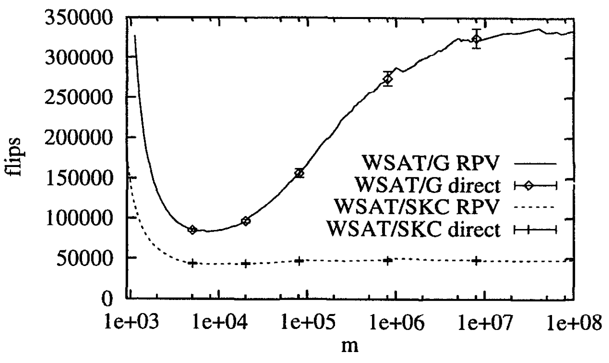

Figure 3: Variation of E_m against Maxflips=m. Based on 1k instances of (200,854) with 200 runs per instance.

The results are plotted in Figure 3. Here the error bars are the 95% confidence intervals for the particular set of instances, i.e. they reflect only the uncertainty from the limited number of runs, and *not* from the limited number of instances. Note that the RPV lines from one data point do indeed match the lower, directly obtained, points. This shows that the RPV is not introducing significant errors when used to extrapolate over these ranges.

Clearly, at $p = 0.5$, the two WSAT algorithms exhibit very different dependencies on Maxflips: selecting too large a value slows WSAT/G down by a factor of about 4, in contrast to a slowdown of about 20% for WSAT/SKC. At Maxflips=∞ the slowest try for WSAT/G took 90 minutes against 5 minutes for the worst effort from WSAT/SKC. As has been observed before (Gent & Walsh 1995), "random walk" can significantly reduce the Maxflips sensitivity of a local search procedure: Restarts and noise fulfill a similar purpose by allowing for downhill moves and driving the algorithm around in the search space. Experimentally we found that while the peak performance E_{m^*} varies only very little with small variation of p (± 0.05), the Maxflips sensitivity can vary quite remarkably. This topic warrants further study and again RPV is useful since it effectively reduces the two-dimensional parameter space (p, m) to just p.

While the average difference for E_{m^*} between the two WSATs on Random 3SAT is typically about a factor of two, we found certain instances of circuit synthesis problems (Selman, Kautz, & Cohen 1994) where WSAT/SKC is between 15 and 50 times faster. Having identified the better WSAT, we will next be concerned with its optimized scaling.

Extensive Experiments on WSAT/SKC

We would now like to obtain accurate results for the performance of WSAT/ SKC on the Random 3SAT do-

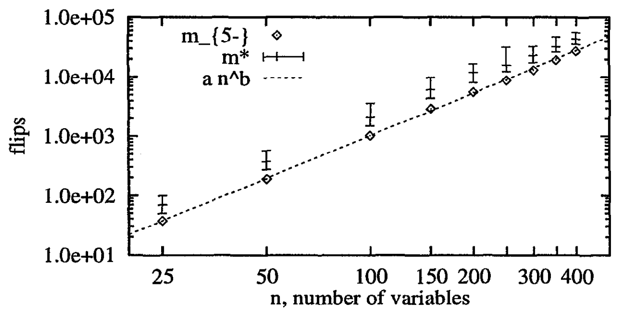

Figure 4: The variation of m_{5-} and m^* with n. We use m_{1+} and m_{1-} as the upper and lower error bounds on m^*.

main. So far, we have only considered confidence intervals resulting from the inter-run variation. Unfortunately, for the previous sample of 1000 instances, the error in E_m arising from the inter-instance variation is much larger (about 25%). This means that to obtain reasonable total errors we needed to look at larger samples of instances.

We were mostly interested in the behaviour at and near m^*. Preliminary experiments indicated that making the main data collection at just $m = 0.5n^2$ would allow us to safely use RPV to study the minimum. Hence, using WSAT/SKC at $p = 0.5$, we did 200 runs per instance at $m = 0.5n^2$, and then used RPV to find m^*. The results are summarized in Table 1, and we now explain the other entries in this table. We found it useful to characterize the E_m curves by values of Maxflips which we denote by m_{k-}, and define as the largest value of m such that $m < m^*$ and E_m is k% larger than E_{m^*}. Similarly we define m_{k+} as the smallest value of $m > m^*$ with the same property. We can easily read these off from the curve produced by RPV. The RPV actually produces the set E_{ν,m^*} and so we also sorted these and calculated various percentiles of the distribution (the 99th percentile means that we expect that 99% of the instances will, on average, be solved in this number of flips). Finally, the error on E_{m^*} is the 95% confidence level as obtained from the standard deviation of the $E_{\nu,m}$ sample (inter-instance). The error from the limited number of runs was negligible in comparison. In the next section we interpret this data with respect to how m^* and E_{m^*} vary with n. We did not convert flips to times because the actual flips rate varies remarkably little (from about 70k-flips/sec down to about 60k-flips/sec).

Scaling of Optimal Maxflips

In Figure 4 we can see how m^* and m_{5-} vary with n. In order to interpret this data we fitted the function an^b against m_{5-} because the E_m curves are rather flat and so m^* is relatively ill-defined. However, they seem to have the same scaling and also $m^* > m_{5-}$ by definition. We obtained a best fit[4] with the values $a = 0.02$ and $b = 2.39$ (the resulting line is also plotted in Figure 4). Similar results for WSAT/G also indicate a very similar scaling of $n^{2.36}$. For comparison HSAT, a non-randomized GSAT variant that incorporates a history mechanism, has been observed to have a m^* scaling of $n^{1.65}$ (Gent & Walsh 1995).

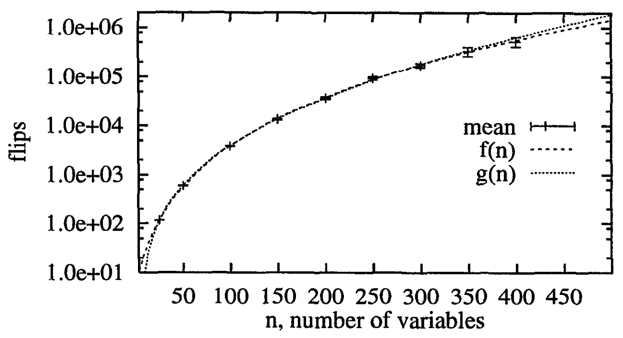

Figure 5: The scaling behaviour of WSAT/SKC at crossover. The data points are E_{m^*} together with its 95% confidence limits. The lines are best fits of the functions given in the text.

Scaling of Performance

In Figure 5 we plot the variation of E_{m^*} with n. We can see that the scaling is not as fast as a simple exponential in n, however the upward curve of the corresponding log-log plot (not presented) indicates that it is also worse than a simple monomial. Unfortunately, we know of no theoretical scaling result that could be reasonably compared with this data. For example, results are known (Koutsoupias & Papadimitriou 1992) when the number of clauses is $\Omega(n^2)$, but they do not apply because the crossover is at $O(n)$. Hence we approached the scaling from a purely empirical perspective, by trying to fit functions to the data that can reveal certain characteristics of the scaling. We also find it very interesting that m^* so often seems to fit a simple power law, but are not aware of any explanation for this. However, the fits do provide an idea of the scaling that might have practical use, or maybe even lead to some theoretical arguments.

We could find no good 2-parameter fit to the E_{m^*} curve, however the following functions give good fits, and provide the lines on Figure 5. (For $f(n)$, we found the values $f_1 = 12.5 \pm 2.02$, $f_2 = -0.6 \pm 0.07$, and $f_3 = 0.4 \pm 0.01$.)

$$\begin{aligned} f(n) &= f_1 n^{f_2 + f_3 \lg(n)} \\ g(n) &= g_1 \exp(n^{g_2}(1 + g_3/n)) \end{aligned}$$

The fact that such different functions give good fits to the data illustrates the obviously very limited dis-

[4]Using the Marquardt-Levenberg algorithm as implemented in Gnufit by C. Grammes at the Universität des Saarlandes, Germany, 1993.

Vars	Cls	m^*	m_{5-}	E_{m^*}	95%-cnf.	Median	99-perc.
25	113	70	37	116	2	94	398
50	218	375	190	591	12	414	2,876
100	430	2,100	1,025	3,817	111	2,123	27,367
150	641	6,100	2,925	13,403	486	5,891	120,597
200	854	11,900	5,600	36,973	2,139	12,915	406,966
250	1066	15,875	8,800	92,915	8,128	25,575	1,050,104
300	1279	23,200	13,100	171,991	15,455	43,314	2,121,809
350	1491	32,000	19,300	334,361	69,850	65,574	4,258,904
400	1704	43,500	27,200	528,545	114,899	96,048	11,439,288

Table 1: Experimental results for WSAT/SKC with $p = 0.5$ on Random 3SAT at crossover. The results are based on 10k instances (25–250 variables), 6k instances (300 vars), 3k instances (350 vars) and 1k instances (400 vars).

criminating power of such attempts at empirical fits. We certainly do not claim that these are asymptotic complexities! However, we include them as indicative. The fact that $f_3 > 0$ indicates a scaling which is similar to a simple power law except that the exponent is slowly growing. Alternatively, the fact that we found $g_2 \approx 0.4$, can be regarded as a further indication that scaling is slower than a simple exponential (which would have $g_2 = 1$).

Testing for Outliers

One immediate concern is whether the average runtimes quoted above are truly meaningful for the problem class Random 3SAT. It could easily be that the effect of outliers, instances that WSAT takes much longer to solve, eventually dominates. That is, as the sample size increases then we could get sufficiently hard instances and with sufficient frequency such that the mean would drift upwards (Mitchell 1993).

To check for this effect we decided to concentrate on $(n, l) = (200, 854)$ and took 10^5 instances. Since we only wanted to check for outliers, we did not need high accuracy estimates and it was sufficient to do just 20 runs/instance of WSAT/SKC at Maxflips=80k, not using RPV. We directly calculated the mean for each seed: in Figure 6 we plot the distribution of the $\lg(E_{\nu,m})$.

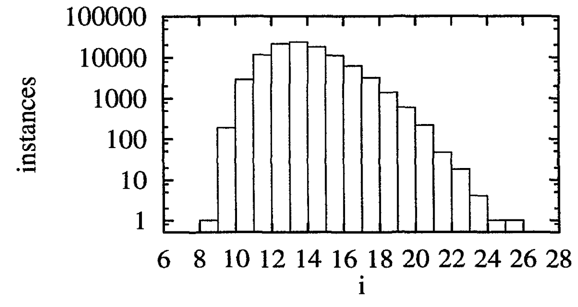

Figure 6: From a total of 10^5 instances we give the number of instances for which the expected number of flips $E_{\nu,m}$ is in the range $2^i \leq E_{\nu,m} < 2^{i+1}$. This is for WSAT/SKC at $m = 80$k, based on 20 runs per instance.

If the tail of the distribution were too long, or if there were signs of a bimodal distribution with a small secondary peak but at a very large E_{m^*} then we would be concerned about the validity of the means quoted above. However we see no signs of any such effects. On the contrary the distribution seems to be quite smooth, and its mean is consistent with the previous data. However, we do note that the distribution tail is sufficiently large that a significant fraction of total runtime is spent on relatively few instances. This means that sample sizes are effectively reduced, and is the reason for our use of 10k samples for the scaling results where computationally feasible.

RELATED WORK

Optimal parameter setting of Maxflips and scaling results have been examined before by Gent and Walsh (Gent & Walsh 1993; 1995). However, for each sample point (each n) Maxflips had to be optimized experimentally. Thus, the computational cost of a systematic Maxflips optimization for randomized algorithms was too high for problems larger than 100 variables. This experimental range was not sufficient to rule out a polynomial runtime dependence of about order 3 (in the number of variables) for GSAT (Gent & Walsh 1993).

A sophisticated approach to the study of incomplete randomized algorithms is the framework of Las Vegas algorithms which has been used by Luby, Sinclair, and Zuckermann (Luby, Sinclair, & Zuckerman 1993) to examine optimal speedup. They have shown that for a *single* instance there is no advantage to having Maxflips vary between tries and present optimal cutoff times based on the distribution of the runtimes. These results, however, are not directly applicable to average runtimes for a collection of instances.

Local Search procedures also have close relations to simulated annealing (Selman & Kautz 1993). Indeed, combinations of simulated annealing with GSAT have been tried for hard SAT problems (Spears 1995). We can even look upon the restart as being a short period of very high temperature that will drive the variable assignment to a random value. In this case we find it interesting that work in simulated annealing also has cases in which periodic reheating is useful (Boese & Kahng 1994). We intend to explore these connections further.

CONCLUSIONS

We have tried to address the four issues in the introduction empirically. In order to allow for optimizing Maxflips, we presented retrospective parameter variation (RPV), a simple resampling method that significantly reduces the amount of experimentation needed to optimize certain local search parameters.

We then studied two different variants of WSAT, which we label as WSAT/G and WSAT/SKC (the latter due to Selman et al.). The application of RPV revealed better performance of WSAT/SKC. An open question is whether the relative insensitivity of WSAT/SKC to restarts carries over to realistic problem domains. We should note that for general SAT problems the Maxflips-scaling is of course not a simple function of the number of variables and p only, but is also affected by other properties such as the problem constrainedness.

To study the scaling behaviour of local search, we experimented with WSAT/SKC on hard Random 3SAT problems over a wide range of problem sizes, applying RPV to optimize performance. Our experimental results strongly suggest subexponential scaling on Random 3SAT, and we can thus support previous claims (Selman, Levesque, & Mitchell 1992; Gent & Walsh 1993) that local search scales significantly better than Davis-Putnam related procedures.

Unfortunately RPV cannot be used to directly determine the impact of the noise parameter p, however it is still useful since instead of varying two parameters, only p has to be varied experimentally, while different Maxflips values can be simulated.

We plan to extend this work in two further directions. Firstly, we intend to move closer to real problems. For example, investigations of binary encodings of scheduling problems reveal a strong correlation between the number of bottlenecks in the problems and optimal Maxflips for WSAT/G. Secondly, we would like to understand the Maxflips-dependence in terms of behaviours of individual instances.

ACKNOWLEDGEMENTS

We would like to thank the members of CIRL, particularly Jimi Crawford, for many helpful discussions, and Andrew Baker for the use of his WSAT. We thank David McAllester and our anonymous reviewers (for both AAAI and ECAI) for very helpful comments on this paper. The experiments reported here were made possible by the use of three SGI (Silicon Graphic Inc.) Power Challenge systems purchased by the University of Oregon Computer Science Department through NSF grant STI-9413532.

References

Boese, K., and Kahng, A. 1994. Best-so-far vs. where-you-are: implications for optimal finite-time annealing. *Systems and Control Letters* 22(1):71–8.

Cheeseman, P.; Kanefsky, B.; and Taylor, W. 1991. Where the really hard problems are. In *Proceedings IJCAI-91.*

Crawford, J., and Auton, L. 1996. Experimental results on the crossover point in Random 3SAT. *Artificial Intelligence.* To appear.

Gent, I., and Walsh, T. 1993. Towards an understanding of hill-climbing procedures for SAT. In *Proceedings AAAI-93*, 28–33.

Gent, I., and Walsh, T. 1995. Unsatisfied variables in local search. In *Hybrid Problems, Hybrid Solutions (Proceedings of AISB-95).* IOS Press.

Gu, J. 1992. Efficient local search for very large-scale satisfiability problems. *SIGART Bulletin* 3(1):8–12.

Koutsoupias, E., and Papadimitriou, C. 1992. On the greedy algorithm for satisfiability. *Information Processing Letters* 43:53–55.

Luby, M.; Sinclair, A.; and Zuckerman, D. 1993. Optimal speedup of Las Vegas algorithms. Technical Report TR-93-010, International Computer Science Institute, Berkeley, CA.

Minton, S.; Johnston, M. D.; Philips, A. B.; and Laird, P. 1990. Solving large-scale constraint satisfcation and scheduling problems using a heuristic repair method. *Artificial Intelligence* 58:161–205.

Mitchell, D.; Selman, B.; and Levesque, H. 1992. Hard and easy distributions of SAT problems. In *Proceedings AAAI-92*, 459–465.

Mitchell, D. 1993. An empirical study of random SAT. Master's thesis, Simon Fraser University.

Selman, B., and Kautz, H. 1993. An empirical study of greedy local search for satisfiability testing. In *Proceedings of IJCAI-93.*

Selman, B.; Kautz, H.; and Cohen, B. 1994. Noise strategies for improving local search. In *Proceedings AAAI-94*, 337–343.

Selman, B.; Levesque, H.; and Mitchell, D. 1992. A new method for solving hard satisfiability problems. In *Proceedings AAAI-92*, 440–446.

Spears, W. 1995. Simulated annealing for hard satisfiability problems. *DIMACS Series on Discrete Mathematics and Theoretical Computer Science.* To appear.

Walser, J. 1995. Retrospective analysis: Refinements of local search for satisfiability testing. Master's thesis, University of Oregon.