Random Walk With Continuously Smoothed Variable Weights

Steven Prestwich

Cork Constraint Computation Centre
Department of Computer Science
University College Cork, Ireland
s.prestwich@cs.ucc.ie

Abstract. Many current local search algorithms for SAT fall into one of two classes. Random walk algorithms such as Walksat/SKC, Novelty+ and HWSAT are very successful but can be trapped for long periods in deep local minima. Clause weighting algorithms such as DLM, GLS, ESG and SAPS are good at escaping local minima but require expensive smoothing phases in which all weights are updated. We show that Walksat performance can be greatly enhanced by weighting variables instead of clauses, giving the best known results on some benchmarks. The new algorithm uses an efficient weight smoothing technique with no smoothing phase.

1 Introduction

Local search algorithms have a long history in combinatorial optimization. In recent years they have been applied to SAT problems, usually by treating them as MAX-SAT and trying to minimise the objective function (the number of clause violations) to zero. Local search for SAT has steadily improved and is an active area of research.

Some local search algorithms fall into the category of *random walks*, which were a significant advance over earlier successful algorithms such as GSAT [26] and those of Gu [10]. The best-known such algorithm is Walksat [17, 25] which has a number of variants. Walksat/G randomly selects a violated clause then *flips* the variable (reassigns it from true to false or vice-versa) that minimizes the total number of violations. Walksat/B selects flips that incur the fewest *breaks* (non-violated clauses that would be violated by the flip). Both select a random variable in the clause (a *random walk move*) with probability p (the *noise parameter*). Walksat/SKC (Selman-Kautz-Cohen) is a version of B that allows *freebies* to override the random walk heuristic. Freebies are flips that incur no breaks, and if at least one freebie is possible from a violated clause then a random one is always selected. HWSAT [8] is a version of G that breaks ties by preferring the least recently flipped variable, based on a similarly modified GSAT called HSAT [7]. Novelty and R-Novelty use similar but more complex criteria. These were later elaborated to the Novelty+ and R-Novelty+ variants [12] that use occasional random walk steps to avoid stagnation, and are among the most

competitive local search algorithms. In TABU search [13, 15, 17] variables that were flipped less recently than a threshold number of flips ago (the *tenure*) cannot be flipped. The tenure replaces the noise parameter of random walk algorithms. In the Iterated Robust Tabu Search (IRoTS) algorithm [27] variables that have not been flipped for a given period are automatically flipped to avoid stagnation.

Some problems defeat random walk algorithms, but are solved quite easily by an alternative form of local search based on *clause weighting*. These algorithms modify the objective function during search. They attach a weight to each clause and minimize the sum of the weights of the violations. The weights are varied dynamically, making it unlikely for the search to be trapped in local minima. Clauses that are frequently violated tend to be assigned higher weights. An early SAT algorithm of this form was Breakout [19] which increments violated clause weights at local minima. A similar approach was used in the later WEIGHT algorithm [2], and variants of GSAT increment weights at local minima [24] or at each flip [4], and may allow weights to slowly decay [5]. The Discrete Lagrangian Method (DLM) [34] periodically smooths the weights to reduce the effects of out-of-date local minima, and is based on the Operations Research technique of Lagrangian relaxation. MAX-AGE [28] is a simplified DLM with fewer runtime parameters and new heuristics. Guided Local Search (GLSSAT) [18] is related to DLM but differs in detail, and derives instead from work on Neural Networks. Smooth Descent and Flood (SDF) [23] introduced the Machine Learning technique of multiplicative weights, and was later elaborated to the Exponentiated SubGradient (ESG) method [22]. The Scaling And Probabilistic Smoothing (SAPS) algorithm [14] is related to ESG but uses a more efficient smoothing mechanism. The Pure Additive Weighting Scheme (PAWS) [29] is a version of SAPS that increases weights additively instead of multiplicatively.

Clause weighting algorithms are excellent at guiding local search out of local minima by consulting the *clause violation* history. An interesting question is: can we achieve a similar effect by consulting the *variable flip* history? This might lead to useful new heuristics for random walk algorithms, and would have technical advantages discussed in Section 5. We shall describe new flip heuristics for random walks that emulate clause weighting performance by maintaining variable weights. Section 2 examines a simple additive heuristic for variable weights. Section 3 describes a new smoothing technique for both clause and variable weights that requires no smoothing phase, reducing runtime overheads. Section 4 evaluates an algorithm with smoothed variable weights. Section 5 discusses the relative merits of variable and clause weighting, and future work.

2 Additive variable weighting

The usual rationale for clause weighting is that it escapes local minima by learning about features in that region of the search space. But recent evidence indicates that this picture is flawed, and that clause weighting acts instead as a diversification mechanism [32]. This suggests that random walk algorithms should be able to emulate clause weighting performance by diversifying the selection of

variables in the flip heuristic. An obvious way to do this is to prefer variables that were not flipped recently, but this is an old idea used in several algorithms. HSAT, HWSAT, Novelty and R-Novelty prefer least-recently flipped variables, the latter incorporating a tabu tenure of 1 flip. Tie-breaking flip heuristics were added to GSAT and Walksat using a first-in-first-out rule [6]. The MAX-AGE clause weighting algorithm prefers variables that were flipped longer ago than a threshold number of flips. TABU forbids recently-flipped variables from being selected, and IRoTS additionally flips variables that have not been flipped recently. Flip heuristics preferring variables that were not recently flipped have been well explored, yet they do not seem to compete with clause weighting heuristics in their ability to escape local minima.

Might variable selection be improved by importing ideas from clause weighting? To explore this idea we shall attach a dynamic weight to each variable and experiment with heuristics for updating these weights. To the best of our knowledge this is a new approach, though variable assignment (literal) weights have been used before. In [16] a literal weighting heuristic was proposed to combine local search with the DPLL backtracking procedure. When a local search algorithm fails to solve a problem, literals that occur most often in violated clauses are assigned higher scores, which can be used to guide DPLL in a proof of unsatisfiability. In [35] a variant of Walksat for SAT (and MAX-SAT) weights literals by analysing local minima during short runs. The aim is to estimate the frequency of each assignment in (optimal) solutions, called *pseudo-backbone frequencies*. These frequencies are used as weights in longer runs to guide initial variable assignments, flip selection and violated clause selection. Some versions of Guided Local Search also weight variable assignments, though not GLSSAT.

As a first experiment we add a new tie-breaking heuristic to Walksat/SKC: select flip variables as usual, but break ties (among non-freebies) by preferring the variable that has been flipped least often in the search so far; break further ties randomly. Thus the weight of a variable is the number of times it has been flipped, and we select variables with minimal weight as long as this does not conflict with the SKC flip heuristic. We shall evaluate a C implementation of the new algorithm, which we call VW1. Other algorithms used for comparison are implemented in the UBCSAT system [30]. All experiments are performed on a Dell 2.4 GHz Pentium 4 with SuSE Linux 9.1 kernel 2.6.

2.1 Experiments on ternary chains

The *ternary chain* problem T_k studied in [21, 33] contains clauses

$$v_1 \qquad v_2 \qquad v_1 \wedge v_2 \rightarrow v_3 \qquad \ldots \qquad v_{k-2} \wedge v_{k-1} \rightarrow v_k$$

and has a single solution in which all variables are true. This highly artificial problem can be solved in linear time by unit propagation, as in DPLL backtracking or hybrid local search algorithms such as Saturn [20] and UnitWalk [11], and quite quickly by clause weighting (see below). Its interest lies in the fact that random walks solve T_k in a number of flips that is exponential in k.

Chain problems are intended to simulate the chains of variable dependency that occur in some highly-structured, real-world problems, and to guide the design of new local search techniques. One such technique is a preprocessing method for adding a small number of redundant clauses [33], which makes chain and other problems much easier to solve.

Why is this problem so hard for random walk? A random walk algorithm selects a variable from a violated clause $\bar{v}_i \vee \bar{v}_{i+1} \vee v_{i+2}$. In a violated clause all literals are false so $v_i = v_{i+1} = T$ while $v_{i+2} = F$. An unbiased random walk selects a variable randomly from these three, so it is twice as likely to flip a variable from true to false than vice-versa. The problem has a single solution in which all variables take the value true, so it is twice as likely to move away from the solution as toward it. (See [33] for a detailed analysis of this and related problems.)

Figure 1 compares VW1 with five other local search algorithms on ternary chains. Each point is the median number of flips over 1000 runs. Optimal values of the SKC, HWSAT and VW1 random walk parameter p are used (found by trying values 0.1–1.0 in steps of 0.1), similarly for the Novelty+ probability parameter used to choose between variables in a clause, and the SAPS smoothing parameter ρ. Novelty+ sets the random walk parameter p to a default value of 0.01. SAPS performance is reported to be robust under different values of its other parameters, and a reactive version called RSAPS varies only ρ [14]. TABU was used with a fixed tabu tenure of 10.

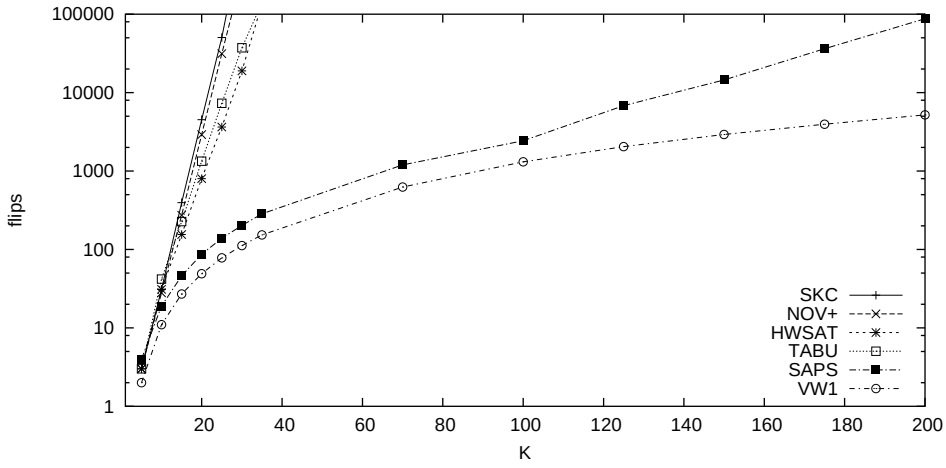

Fig. 1. Local search algorithms on ternary chain problems

The graphs show that most algorithms scale exponentially. However, VW1 scales polynomially: on a log-log plot (not shown) its graph is a straight line. SAPS also scales polynomially for k up to approximately 100, after which it scales

exponentially. It is unclear why this occurs, but we conjecture that multiplicative weights cause small floating-point errors that erase long-term information. HWSAT and TABU use simpler forms of memory and scale only slightly better than Novelty+, which scales slightly better than SKC. Surprisingly, increasing the tabu tenure did not improve TABU. These problems seem to require a particular form of long-term memory currently provided only by clause and variable weighting.

We propose ternary chains as a benchmark for clause and variable weighting algorithms, and other techniques designed to escape local minima. However, it is a very artificial problem so we will also evaluate variable weighting on common SAT benchmarks.

2.2 Experiments on SAT benchmarks

Next we compare SKC, Novelty+ and SAPS with VW1 on other problems. Figure 2 shows results on selected SAT benchmark problems from the SATLib repository.[1] The figures shown are numbers of flips, taking medians over 100 runs. Again optimal parameter values are used, and in these experiments we tune both the ρ and α SAPS parameters using α values $\{1.1, 1.3, 2.0, 3.0\}$. Figures greater than 10^7 flips are denoted by —.

- The AIM benchmarks [1] are random 3-SAT problems modified to be very hard for random walk. They can be solved with polynomial preprocessing so they are not intrinsically hard. Each instance i is denoted by aim$n$-r-i, meaning that it has n variables and clause/variable ratio r. We use satisfiable instances with clause-variable ratios of 2.0, which are known to be particularly hard for Walksat-style algorithms. Variable weighting greatly boosts SKC performance so that VW1 outperforms SKC and Novelty+, though not SAPS.
- On logistics planning problems VW1 lies between SKC and Novelty+ in performance, and SAPS is again the best algorithm.
- On Blocks World (bw) planning problems VW1 is the best algorithm.
- On All-Interval Series (ais) problems VW1 is similar to SKC, beating Novelty+ but beaten by SAPS.
- On large random 3-SAT problems (f) VW1 is the worst algorithm.

These results show that guiding random walks by variable weighting can greatly boost random walk performance, though not on all problems. Our next step is to import another important technique from clause weighting: smoothing.

3 Phased and continuous smoothing

Smoothing techniques for clause weighting algorithms considerably improve performance. Smoothing can be adapted to variable weights, but current clause

[1] http://www.cs.ubc.ca/~hoos/SATLIB/

instance	SKC	Novelty+	SAPS	VW1
aim50-2.0-1	71933	318514	669	5510
aim50-2.0-2	10727	10338	520	4858
aim50-2.0-3	89040	361853	885	4684
aim50-2.0-4	57098	69040	603	5863
aim100-2.0-1	—	—	4423	324847
aim100-2.0-2	—	—	4898	221535
aim100-2.0-3	—	—	2878	95709
aim100-2.0-4	—	—	5044	175616
aim200-2.0-1	—	—	488547	—
aim200-2.0-2	—	—	181770	3655956
aim200-2.0-3	—	—	298473	—
aim200-2.0-4	—	—	506329	—
logistics.a	64866	46385	6288	36144
logistics.b	88186	54102	5472	25369
logistics.c	104610	81732	7578	37341
logistics.d	425746	117649	33781	183611
bw_large.a	14459	5114	2208	8085
bw_large.b	422723	100830	24649	87843
bw_large.c	8154220	3631196	1904852	582056
bw_large.d	—	5093099	3807160	630464
ais6	891	5388	331	984
ais8	19306	123949	3996	18661
ais10	106752	1523002	18228	101157
ais12	1219293	—	141211	816645
f600	130625	65476	38159	148749
f1000	491262	360709	243377	939022
f2000	2536333	4086895	2849808	—

Fig. 2. Additive variable weighting on SAT benchmarks

weighting schemes use expensive *smoothing phases* in which all weights are adjusted to reduce the differences between them. As the number of clauses is often large, this is a significant overhead. We propose a cheaper method with no smoothing phase that can be used for clause or variable weighting.

Smoothing reduces the effect of the earlier search history, placing more emphasis on recent events and adapting the search heuristics to the current search space topology. Our smoothing technique does this as follows. Associate with each variable v a weight w_v, initialised to 0 and updated each time v is flipped according to the formula:

$$w_v \leftarrow (1 - s)(w_v + 1) + s \times t$$

where t denotes time (measured as the number of flips since the start of the search) and a parameter $0 \leq s \leq 1$ controls smoothing. Setting $s = 1$ causes variables to forget their flip history: only a variable's most recent flip has an effect on its weight. Using weights for tie-breaking we obtain an HWSAT-like heuristic. Conversely $s = 0$ causes w_v to behave like a simple counter as in VW1, so that every move in the history has an equal effect. Choosing s between 0 and 1 interpolates between these two extremes, causing older events to have smaller but non-zero effects. We call this *continuous smoothing* because smoothing occurs as weights are updated, without the need for smoothing phases. It also requires only the single s parameter, though when integrating it into a search algorithm in Section 4 we find a further parameter necessary.

To compare phased and continuous smoothing we simulate the evolution of weights during search. In this simulation there are three variables v_1, v_2, v_3 that are flipped during search. The search is divided into three parts. In the first part v_1 is flipped in $\frac{2}{3}$ of the iterations and v_2 in the remaining $\frac{1}{3}$, so variables are flipped in the order $v_1, v_1, v_2, v_1, v_1, v_2 \ldots$ In the second part the same is true of v_2 and v_3 respectively, and in the third part of v_3 and v_1. The simulation shows what happens when variables change from being frequently flipped to rarely flipped, and vice-versa. We use a simple phased weighting method in which weights are increased additively by 1, and smoothed multiplicatively every 50 iterations by 0.8. We compare this with continuous smoothing using $s = 0.02$. The simulations are shown in Figures 3 and 4. The results are qualitatively similar: both methods adjust weight rankings to new situations in almost the same way, though the actual values are different.

Care must be taken when implementing continuous smoothing: if we use integer arithmetic then overflow may occur, though this only occurs on long runs because all weights are bounded above by the number of flips in the search so far. Floating-point arithmetic can be used for long runs, but as weights become very large the term $w_v + 1$ becomes indistinguishable from w_v. A solution is to periodically scale all integer weights down by some factor, similar to a smoothing phase; but whereas phased smoothing typically occurs after a few tens of flips, integer weights (and the flip counter) need not be rescaled more often than every few million flips. Alternatively weights could be implemented using infinite-precision integer arithmetic.

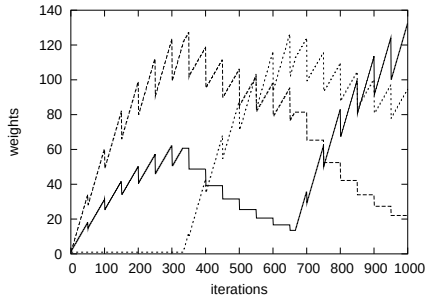

Fig. 3. Phased smoothing

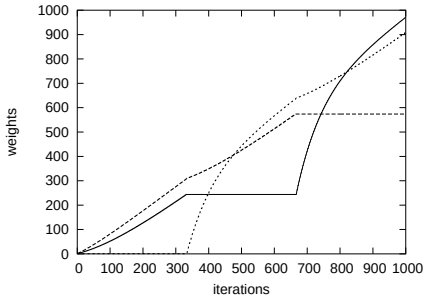

Fig. 4. Continuous smoothing

The same method may be applied to clause weighting. At each local minimum we update the weight of each violated clause using the above formula. These weights may then be used as in other clause weighting algorithms. However, in this paper we test only variable weighting.

4 A new local search algorithm

We now describe and evaluate a new local search algorithm called VW2, combining continuously smoothed variable weights with heuristics based on Walksat/SKC. The VW2 algorithm is shown in Figure 5 and is identical to SKC except for its flip heuristic. Instead of using weights for tie-breaking we use them to adjust the break counts as follows. From a random violated clause we select the variable v with minimum score $b_v + b(w_v - M)$ where b_v is the break count of v, w_v is the current weight of v, M is the current mean weight, and c is a new parameter ($c \geq 0$ and usually $c < 1$). Ties are broken randomly. VW2 has three parameters p, s, c but we shall only explore a fraction of the parameter space using p values $\{0.05, 0.1, 0.2, 0.3, 0.4\}$, s values $\{10^{-1}, 10^{-2}, 10^{-3}\}$ and c values $\{10^{-1}, 10^{-2}, 10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}\}$.

Figure 6 shows results for VW2 on the same benchmarks as in Figure 2 except for the smaller AIM problems, plus two hard graph colouring benchmarks. They are almost uniformly better than those for VW1, and we now compare them to our SKC, Novelty+ and SAPS results in Figure 2; also to published results, where available, for the clause weighting algorithms DLM and MAX-AGE from [28] and SAPS and PAWS from [14, 29] (all using medians over 100 runs).

- On the AIM problems VW2 beats SKC, Novelty+ and SAPS.
- On the logistics planning problems VW2 is beaten by SAPS but beats SKC and Novelty+.
- On the Blocks World planning problems VW2 beats SKC, Novelty+, SAPS, PAWS, DLM and MAX-AGE.
- On the AIS problems VW2 beats SKC and Novelty+, and is comparable to SAPS.

```
initialise all variables to randomly selected truth values
initialise all variable weights to 0
repeat until no clause is violated
(  randomly select a violated clause C
   if C contains freebie variables
      randomly flip one of them
   else with probability p
      flip a variable in C chosen randomly
   else with probability 1-p
      flip a variable in C chosen by the new heuristic
   update and smooth the weight of the flipped variable
)
```

Fig. 5. The VW2 algorithm

- On the random 3-SAT problems (f) VW2 beats SKC, Novelty+ and (on the largest problem) SAPS, but is beaten by DLM and MAX-AGE.
- On the graph colouring (g) VW2 beats SAPS but is beaten by PAWS, DLM and MAX-AGE.

The results are clearly competitive with those for current random walk and clause weighting algorithms, and the blocks world planning results are (as far as we know) the best reported. VW2 currently takes an order of magnitude more flips than the best algorithms on 16-bit parity learning problems, but clause weighting algorithms have undergone several generations of development and we hope to emulate their success in future work. We also hope to improve its robustness under different parameter values, following techniques already developed for the Walksat noise parameter.

We now compare flip rates for VW2 and SAPS. SAPS has a more efficient smoothing algorithm than most clause weighting algorithms, but continuous smoothing scales better to large problems. For example on the 600-variable random 3-SAT benchmark the ratio of the VW2 flip rate to that of SAPS is 1.62; on the 1000-variable benchmark it is 1.73; and on the 2000-variable benchmark it is 2.13. Similarly on the Blocks World planning problems the ratio is 1.75 for problem (a), 2.26 for problem (b), 2.80 for problem (c), and 2.99 for problem (d). This is despite the fact that the UBCSAT implementation is generally more efficient than ours. For example on the 600-variable random 3-SAT problem its version of Novelty+ performs roughly 1.42 more flips per second than VW2, on the 1000-variable problem the ratio is 1.78, and on the 2000-variable problem it is 1.94. Similarly on blocks world planning problem (a) the ratio is 1.37, on problem (b) 1.29, on problem (c) 2.10 and on problem (d) 2.95. Combining the implementation techniques of the UBCSAT system with continuous smoothing should yield very scalable algorithms.

instance	p	s	c	flips	sec
aim200-2.0-1	0.05	0.01	0.001	121735	0.087
aim200-2.0-2	0.05	0.01	0.001	55128	0.040
aim200-2.0-3	0.05	0.01	0.001	66884	0.046
aim200-2.0-4	0.05	0.01	0.001	83576	0.060
logistics.a	0.05	0.1	0.001	13114	0.027
logistics.b	0.05	0.1	0.001	14559	0.034
logistics.c	0.05	0.01	0.001	17734	0.046
logistics.d	0.05	0.01	0.0001	87817	0.21
bw_large.a	0.2	0.01	0.001	5854	0.012
bw_large.b	0.2	0.01	0.0001	73994	0.26
bw_large.c	0.2	0.01	0.00001	508254	3.17
bw_large.d	0.2	0.01	0.000001	570471	5.56
ais6	0.1	0.001	0.1	891	0.0015
ais8	0.1	0.001	0.1	5485	0.013
ais10	0.1	0.001	0.01	27356	0.076
ais12	0.1	0.01	0.001	135394	0.54
f600	0.4	0.1	0.000001	68040	0.10
f1000	0.4	0.1	0.000001	272392	0.53
f2000	0.4	0.1	0.000001	969650	3.05
g125.17	0.2	0.1	0.000001	1820914	29.8
g250.29	0.2	0.1	0.000001	1508571	96.3

Fig. 6. Smoothed variable weighting on SAT benchmarks

5 Discussion

It was recently conjectured that clause weighting works as a form of diversification [32]. We showed that an alternative diversification technique called *variable weighting*, based on variable flip histories, can emulate clause weighting performance. This contributes to the understanding of local search heuristics by supporting the conjecture. It also provides an alternative to clause weighting for problems with deep local minima, and is a promising source of new local search heuristics. We also introduced an efficient new *continuous smoothing* technique for variable weights, which we will test on clause weights in future work.

An advantage of variable weighting is that at each search step only one weight is adjusted, whereas an unbounded (though typically small) number of clause weights may need to be updated at a local minimum. Another advantage is that variable weighting is amenable to *lifting* [9], a technique for compressing a large set of clauses into a single formula via quantification. Lifted clauses are not represented explicitly so dynamically changing weights cannot be assigned to them, but variables can be dynamically weighted. Thus we can achieve clause weighting-like performance on extremely large problems. Variable weighting may also be better suited than clause weighting to weighted MAX-SAT, though this remains to be tested. One of the best local search algorithms on MAX-SAT is currently SAPS [31] and the authors speculate that it will also perform well on

weighted MAX-SAT, but note that there are several ways of combining the static clause weights of the problem with the dynamic clause weights of the algorithm. This may be easier with our approach because the static clause weights can be treated separately from the dynamic variable weights.

Finally, a note on completeness. Some local search algorithms have a weak form of completeness called *probabilistic asymptotic completeness* (PAC), meaning that as time tends to infinity the probability of finding a solution tends to one [12]. Does VW2 possess this property? A drawback of freebies is that they make PAC hard to prove, and it is still an open question for SKC except for the special case of 2-SAT [3]. But SKC appears to behave empirically like a PAC algorithm so a proof may eventually be found. Any such proof is likely to reason on freebies and random walk moves alone. VW2 differs from SKC only in its other moves, so we conjecture that VW2 is PAC if and only if SKC is. PAC seems to be less of an issue in clause weighting research, though an inefficient PAC version of Breakout is described in [19].

Acknowledgment

Thanks to the anonymous referees for helpful comments. This material is based in part upon works supported by the Science Foundation Ireland under Grant No. 00/PI.1/C075.

References

1. Y. Asahiro, K. Iwama, E. Miyano. Random Generation of Test Instances with Controlled Attributes. In D. S. Johnson, M. A. Trick (eds), *Cliques, Coloring and Satisfiability: Second Implementation Challenge, DIMACS Series in Discrete Mathematics and Theoretical Computer Science* vol. 26, American Mathematical Society 1996, pp. 127–154.
2. B. Cha, K. Iwama. Performance Test of Local Search Algorithms Using New Types of Random CNF Formulas. *Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence*, Morgan Kaufmann 1995, pp. 304–310.
3. J. Culberson, I. P. Gent, H. H. Hoos. On the Probabilistic Approximate Completeness of WalkSAT for 2-SAT. Technical Report APES-15a-2000, APES Research Group, 2000.
4. J. Frank. Weighting for GODOT: Learning Heuristics for GSAT. *Proceedings of the Thirteenth National Conference on Artificial Intelligence*, MIT Press, 1996, pp. 338–343.
5. J. Frank. Learning Short-Term Weights for GSAT. *Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, 1997, pp. 384–389.
6. A. S. Fukunaga. Variable-Selection Heuristics in Local Search for SAT. *Proceedings of the Fourteenth National Conference on Artificial Intelligence and Ninth Innovative Applications of Artificial Intelligence Conference*, 1997, pp. 275–280.
7. I. P. Gent, T. Walsh. Towards an Understanding of Hill-Climbing Procedures for SAT. *Proceedings of the Eleventh National Conference on Artificial Intelligence*, AAAI Press, 1993, pp. 28–33.

8. I. P. Gent, T. Walsh. Unsatisfied Variables in Local Search. J. Hallam (ed.), *Hybrid Problems, Hybrid Solutions*, IOS Press, Amsterdam, The Netherlands, 1995, pp. 73–85.
9. M. L. Ginsberg, A. J. Parkes. Satisfiability Algorithms and Finite Quantification. *Seventh International Conference on Principles of Knowledge Representation and Reasoning*, Morgan Kaufmann, 2000, pp. 690–701.
10. J. Gu. Efficient Local Search for Very Large-Scale Satisfiability Problems. *Sigart Bulletin* vol. 3, no. 1, 1992, pp. 8–12.
11. E. A. Hirsch, A. Kojevnikov. Solving Boolean Satisfiability Using Local Search Guided by Unit Clause Elimination. *Seventh International Conference on Principles and Practice of Constraint Programming, Lecture Notes in Computer Science* vol. 2239, Springer, 2001, pp. 605–609.
12. H. H. Hoos. On the Run-Time Behaviour of Stochastic Local Search Algorithms. *Sixteenth National Conference on Artificial Intelligence*, AAAI Press, 1999, pp. 661–666.
13. W. Huang, D. Zhang, H. Wang. An Algorithm Based on Tabu Search for Satisfiability Problem. *Journal of Computer Science and Technology* vol. 17 no. 3, Editorial Universitaria de Buenos Aires, 2002, pp. 340–346.
14. F. Hutter, D. A. D. Tompkins, H. H. Hoos. Scaling and Probabilistic Smoothing: Efficient Dynamic Local Search for SAT. *Eighth International Conference on Principles and Practice of Constraint Programming, Lecture Notes in Computer Science* vol. 2470, Springer, 2002, pp. 233–248.
15. B. Mazure, L. Saïs, É. Grégoire. Tabu Search for SAT. *Proceedings of the Fourteenth National Conference on Artificial Intelligence*, 1997, pp. 281–285.
16. B. Mazure, L. Saïs, É. Grégoire. Boosting Complete Techniques Thanks to Local Search. *Annals of Mathematics and Artificial Intelligence* vol. 22, 1998, pp. 309–322.
17. D. A. McAllester, B. Selman, H. A. Kautz. Evidence for Invariants in Local Search. *Fourteenth National Conference on Artificial Intelligence and Ninth Innovative Applications of Artificial Intelligence Conference*, AAAI Press / MIT Press 1997, pp. 321–326.
18. P. Mills, E. P. K. Tsang. Guided Local Search for Solving SAT and Weighted MAX-SAT Problems. *Journal of Automated Reasoning, Special Issue on Satisfiability Problems*, Kluwer, Vol.24, 2000, pp. 205–223.
19. P. Morris. The Breakout Method for Escaping from Local Minima. *Proceedings of the Eleventh National Conference on Artificial Intelligence*, AAAI Press / MIT Press, 1993, pp. 40–45.
20. S. D. Prestwich. Incomplete Dynamic Backtracking for Linear Pseudo-Boolean Problems. *Annals of Operations Research* vol. 130, 2004, pp. 57–73.
21. S. D. Prestwich. SAT Problems With Chains of Dependent Variables. *Discrete Applied Mathematics* vol. 3037, Elsevier, 2002, pp. 1-22.
22. D. Schuurmans, F. Southey, R. C. Holte. The Exponentiated Subgradient Algorithm for Heuristic Boolean Programming. *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence*, Morgan Kaufmann, 2001, pp. 334–341.
23. D. Schuurmans, F. Southey. Local Search Characteristics of Incomplete SAT Procedures. *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, AAAI Press, 2000, pp. 297–302.
24. B. Selman, H. A. Kautz. Domain-Independent Extensions to GSAT: Solving Large Structured Satisfiability Problems. *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence*, 1993, pp. 290–295.

25. B. Selman, H. A. Kautz, B. Cohen. Noise Strategies for Improving Local Search. *Proceedings of the Twelfth National Conference on Artificial Intelligence*, AAAI Press, 1994, pp. 337–343.
26. B. Selman, H. Levesque, D. Mitchell. A New Method for Solving Hard Satisfiability Problems. *Proceedings of the Tenth National Conference on Artificial Intelligence*, MIT Press, 1992, pp. 440–446.
27. K. Smyth, H. H. Hoos, T. Stützle. Iterated Robust Tabu Search for MAX-SAT. *Proceedings of the Sixteenth Canadian Conference on Artificial Intelligence, Lecture Notes in Computer Science* vol. 2671, Springer Verlag, 2003, pp. 129–144.
28. J. R. Thornton, W. Pullan, J. Terry. Towards Fewer Parameters for Clause Weighting SAT Algorithms. *Proceedings of the Fifteenth Australian Joint Conference on Artificial Intelligence, Lecture Notes in Artificial Intelligence* vol. 2557, Springer-Verlag, 2002, pp. 569–578.
29. J. R. Thornton, D. N. Pham, S. Bain, V. Ferreira Jr. Additive versus Multiplicative Clause Weighting for SAT. *Proceedings of the Nineteenth National Conference on Artificial Intelligence*, San Jose, California, 2004, pp. 191–196.
30. D. A. D. Tompkins, H. H. Hoos. UBCSAT: An Implementation and Experimentation Environment for SLS Algorithms for SAT and MAX-SAT. *Proceedings of the Seventh International Conference on Theory and Applications of Satisfiability Testing*, 2004, pp. 37–46.
31. D. A. D. Tompkins, H. H. Hoos. Scaling and Probabilistic Smoothing: Dynamic Local Search for Unweighted MAX-SAT. *Proceedings of the Sixteenth Canadian Conference on Artificial Intelligence, Lecture Notes in Computer Science* vol. 2671, Springer, 2003, pp. 145–159.
32. D. A. D. Tompkins, H. H. Hoos. Warped Landscapes and Random Acts of SAT Solving. *Proceedings of the Eighth International Symposium on Artificial Intelligence and Mathematics*, 2004 (to appear).
33. W. Wei, B. Selman. Accelerating Random Walks. *Proceedings of the Eighth International Conference on Principles and Practice of Constraint Programming, Lecture Notes in Computer Science* vol. 2470, Springer, 2002, pp. 216–232.
34. Z. Wu, B. W. Wah. An Efficient Global-Search Strategy in Discrete Lagrangian Methods for Solving Hard Satisfiability Problems. *Proceedings of the Seventeenth National Conference on Artificial Intelligence*, AAAI Press, 2000, pp. 310–315.
35. W. Zhang, A. Rangan, M. Looks. Backbone Guided Local Search for Maximum Satisfiability. *Eighteenth International Joint Conference on Artificial Intelligence*, 2003, pp. 1179–1186.