

Towards Fewer Parameters for SAT Clause Weighting Algorithms

John Thornton, Wayne Pullan, and Justin Terry

School of Information Technology,
Griffith University Gold Coast,
Qld, 4215, Australia

Email: {j.thornton, w.pullan, j.terry}@mailbox.gu.edu.au

Keywords: Constraints, Search

Abstract. Considerable progress has recently been made in using clause weighting algorithms such as DLM and SDF to solve SAT benchmark problems. While these algorithms have outperformed earlier stochastic techniques on many larger problems, this improvement has been bought at the cost of extra parameters and the complexity of fine tuning these parameters to obtain optimal run-time performance. This paper examines the use of parameters, specifically in relation to DLM, to identify underlying features in clause weighting that can be used to eliminate or predict workable parameter settings. To this end we propose and empirically evaluate a simplified clause weighting algorithm that replaces the tabu list and flat moves parameter used in DLM. From this we show that our simplified clause weighting algorithm is competitive with DLM on the four categories of SAT problem for which DLM has already been optimised.

1 Introduction

One of the basic aims of artificial intelligence research is to replace tasks requiring human expertise with automated or algorithmic solutions. For instance, the constraint satisfaction problem (CSP) formalism and the development of general purpose constraint solving technologies is intended to replace the task of writing specific algorithms to solve specific problems. However, human input is still required to model problems, select appropriate constraint solving techniques and to fine tune parameters that in turn optimise performance in particular problem domains. Typically, this tuning process requires a significant period of trial and error (especially for stochastic search techniques). Further, the complexity of parameter setting grows exponentially rather than linearly as the number of parameters increases. For these reasons we can conclude that, given an algorithm with a set of n parameters, removal of a subset k of these parameters without significant impact on the sensitivities and range of the remaining $n - k$ parameters generates a more effective and usable algorithm.

In this paper we look at parameter elimination for clause weighting algorithms in the satisfiability (SAT) problem domain. We have chosen SAT due to the significant and ongoing improvement in the performance of SAT algorithms that has occurred in the last decade. In particular, we are interested in clause weighting, because current techniques such as the Discrete Lagrangian Method (DLM) [10], Smooth Descent

and Flood (SDF) [6] and the Exponentiated Subgradient algorithm (ESG) [7] represent state-of-the-art performance on the widely used SATLIB and DIMACS benchmark problems. DLM offers the further advantage that it is a general purpose technique applicable to the broader domain of CSPs.

Specifically, the paper examines three important DLM parameters described in [10] that (i) control the number of zero cost moves taken in the weighted cost space (ii) set the length of the tabu list and (iii) control the frequency with which weights are reduced. We propose a simplified clause weighting algorithm that removes parameters (i) and (ii) and compare the performance of this algorithm with DLM in the four problem domains for which DLM was originally optimised (namely the SATLIB and DIMACS random 3-SAT, parity function learning, graph colouring and blocks world planning benchmarks). These results show that simplified clause weighting is competitive with DLM and suggest future directions for a self-tuning, general purpose clause weighting heuristic.

2 Clause Weighting

2.1 A Brief History

The clause weighting algorithm for SAT was simultaneously proposed in [5] and [8] in 1993. Developed to improve on GSAT [8], clause weighting is an incomplete local search method that escapes traps or minima by adding weight to currently false clauses. This changes the *weighted* cost surface of a problem, allowing further cost reducing moves by partially “filling in” [5] each minimum. As weights build up during the search, flip selection is biased towards moves that satisfy more heavily weighted clauses. This is analogous to a human problem solver fixing the most difficult parts of a problem first, and then moving around the less constrained resources until a solution is found.

Various enhancements to clause weighting were proposed in the mid-90s, most notably Jeremy Frank’s work on multiplicative weighting and weight decay [1]. Frank anticipated much of the later work on clause weighting, particularly in controlling weight growth so that the relative clause weight magnitudes remain fairly constant during the search. However, it was not until the development of DLM that these insights were translated into significant performance improvements.

2.2 DLM

DLM was first proposed as a general purpose optimisation technique rather than as a special purpose SAT algorithm [9]. However, it has been widely recognised as the state-of-the-art for solving the larger SAT benchmark problems [6] and subsequent versions have introduced SAT specific heuristics [11]. In addition to achieving performance gains over other SAT techniques, DLM provides a mathematical foundation to clause weighting, extending the theory of Lagrangian multipliers from continuous to discrete space problem solving. When applied to SAT, DLM can be considered as a clause weighting algorithm, with the discrete Lagrangian multipliers representing the clause weights. The main differences between DLM and earlier clause weighting techniques

are in the use of a tabu list [3] to guide the search over plateau areas, and in the use of a weight reduction heuristic that periodically reduces clause weights. While tabu lists had previously been used in SAT [4] and similar weight reduction schemes had already been suggested [1], the success of DLM hinges on the careful *combination* of these heuristics within a clause weighting algorithm. This is illustrated in the pseudocode for *DLM-SAT* (in Figure 1) which is derived from *DLM-98-BASIC-SAT*, *DLM-99-SAT* [10] and the *DLM-2000-SAT* source code [11]. It represents the key features of DLM without the later *DISTANCE-PENALTY* and *SPECIAL-INCREASE* heuristics (designed to solve the harder parity, graph and hanoi problems). We present *DLM-SAT* in some detail as this algorithm acts as the base for our further development and also to isolate the three main parameters used to tune DLM to particular problem domains, namely the tabu list length (TABU), the maximum number of flat moves allowed (FLAT) and the number of weight increases before a weight decrease occurs (DECREASE).

```

procedure DLM-SAT
begin
  Generate a random starting point
  bestWeightedCost  $\leftarrow$  number of false clauses
  Initialise counters and clause weights to zero
  while solution not found and flips < maxFlips do
    B  $\leftarrow$  set of best weighted cost single flip moves
    if no improving  $x \in B$  then
      Remove all  $x \in B$  with  $flips - tabuAge(x) < TABU$ 
    if weighted cost < bestWeightedCost then
      bestWeightedCost  $\leftarrow$  weighted cost
      flatMoves  $\leftarrow$  0
    else if ++flatMoves > FLAT and no improving  $x \in B$  then
      B  $\leftarrow$   $\emptyset$ 
      flatMoves  $\leftarrow$  0
    end if
    if  $B \neq \emptyset$  then
      Randomly pick and flip  $x \in B$ 
      flips  $\leftarrow$  flips + 1
      if  $flips - tabuAge(x) > TABU$  then
        tabuAge(x)  $\leftarrow$  flips
      else
        Increase weight on all false clauses
        if ++increases % DECREASE = 0 then
          Decrease weight on all weighted clauses
          bestWeightedCost  $\leftarrow$  weighted cost
        end if
      end if
    end if
  end while
end

```

Fig. 1. The basic DLM algorithm

Weight Reduction in DLM and SDF *DLM-SAT* extends clause weighting by using a weight reduction scheme controlled by the DECREASE parameter shown in Figure 1. This scheme reduces the weights on all *weighted* clauses by a standard decrement (usually one) after the search has added weight DECREASE times. Weight increases are also of a simple additive nature. Other weighting schemes, most notably SDF [6], have used multiplicative weighting and a continuous normalisation of relative weights after each increase. While SDF has produced some improvement over DLM in terms of the flip count on smaller sized problems, there is a significant run-time overhead in maintaining SDF’s real valued weights. This is caused by having to recalculate the weights on *all* clauses each time weight is added. In contrast, DLM only updates false clause weights during an increase (generally less than 5% of the total clauses) and then only updates *weighted* clauses during a reduction (with reductions occurring after each DECREASE number of weight increases). As the run-times in [6] show, SDF is up to 4 times slower than DLM, a result largely explained by the different operation of the weight control schemes.

While multiplicative weighting and weight smoothing offer a less ad-hoc approach to weight control, DLM’s additive reduction scheme works well in practice, is more efficient and is controlled by a single parameter (rather than the two required for SDF). For these reasons we decided to continue with a DLM type scheme in our own simplified clause weighting scheme described in Section 2.3.

Tabu Lists and Short-Term Memory The second of DLM’s extensions to clause weighting is the use of a tabu list to control the selection of non-cost improving moves. Earlier SAT heuristics, such as HSAT [2], used a similar approach to break ties between equal cost moves based on when a variable was last flipped. DLM’s tabu list differs from these strategies by storing the most recently flipped variables in a list, the length of which is set by the TABU parameter (see Figure 1). Any variable on the list is then *tabu* and cannot be flipped unless it produces a cost improvement. The rationale behind a tabu list is to avoid cycles of repeated moves and assist the search to escape from a local minimum [3]. In DLM’s case, clause weighting already acts as a minima escaping mechanism, so the tabu list is used primarily to navigate over plateaus (i.e. areas in the search space where there are no cost improving moves). Simple clause weighting algorithms generally deal with plateaus by immediately adding weight [5] whereas the tabu list in DLM delays the weight increase in order to explore a plateau more thoroughly. Given that clause weighting when combined with a weight reduction scheme is itself a form of short-term memory [1], the question arises why DLM requires a *another* short-term memory heuristic (namely a tabu list) to search plateaus.

2.3 Simplified Clause Weighting

Eliminating the Tabu List Our aim in this study is to produce a simplified clause weighting algorithm, and particularly to reduce the number of parameters required to tune an algorithm to different problem domains. From our analysis of clause weighting, and DLM in particular, we identified two basic choices that define the effectiveness of a search:

- When and by how much to *increase* clause weights
- When and by how much to *reduce* clause weights

DLM uses both the TABU and FLAT parameters to decide on weight increases, i.e. either weights are increased because the maximum number of flat moves has been exceeded or because all plateau moves have become tabu. Similarly, the DECREASE parameter is used to decide when to reduce weights. On the basis of our discussion in Section 2.2 we decided to eliminate the TABU and FLAT parameters from *DLM-SAT* by removing the tabu list, and so to look for a simpler approach to increase weights and control plateau searches. Early clause weighting algorithms avoided plateau search by adding weight as soon as a plateau is encountered [5]. However, we found such techniques do not scale well to larger problems (even if a weight reduction scheme is included). We therefore decided to randomise the choice between adding weight or taking a plateau move (see Figure 2). While this creates another potential parameter, we found the best plateau move selection probability P remains fairly constant across different problem domains (see Section 3). In addition P replaces replaces *both* the TABU and FLAT parameters from *DLM-SAT*.

The Maximum Age Heuristic An analysis of the TABU parameter settings for DLM shows an unusually long list length is required to solve the larger DIMACS random 3-SAT problems [9]. Our randomised plateau move selection heuristic assumes sufficient information is stored in the clause weights alone to guide the search trajectory. However, the longer tabu list for 3-SAT suggests a longer term-memory is sometimes useful (as pointed out in [1], clause weight reduction schemes provide only *short-term* memory). We therefore developed a maximum age (*MAX-AGE*) heuristic to exploit longer-term information about when a variable was last flipped. As with a tabu list, *MAX-AGE* stores the number of flips since a variable was last flipped (this is a variable’s age). However, instead of using a list, the age of each plateau flip is compared to a *maximum age* value, where $maximum\ age = total\ flips - maximum\ age\ counter$ and *maximum age counter* is incremented each time *MAX-AGE* causes a plateau move to be accepted. If the age of a plateau move equals or exceeds *maximum age* then it is accepted, otherwise we use our randomised selection heuristic described above. In this way the search is biased towards flipping rarely used variables and is encouraged to occasionally take steps into previously unexplored regions (the complete *MAX-AGE* algorithm is shown in Figure 2).

3 Empirical Analysis

As our work is based on the original DLM algorithms, we decided to evaluate *MAX-AGE* in comparison with the most recent publicly available version of DLM, namely *DLM-2000-SAT* (or *DLM2K*). *DLM2K* contains the *SPECIAL-INCREASE* heuristic described in [10] and developed to solve the harder DIMACS parity learning, Towers of Hanoi and graph colouring problems. Secondly, for a more direct comparison, we generated results for the *DLM-SAT* algorithm shown in Figure 1. *DLM-SAT* is our own cut-down version of DLM that uses only the TABU, FLAT and DECREASE parameters, but is otherwise derived from *DLM2K*. Finally, we generated results for the

```

procedure MAX-AGE
begin
  Generate a random starting point
  Initialise counters and clause weights to zero
  while solution not found and flips < maxFlips do
    B ← set of best weighted cost single flip moves
    if no improving  $x \in B$  then
      if oldest  $x \in B$  has  $age(x) \geq maxAge$  then
        B ←  $x$ 
         $maxAge \leftarrow maxAge + 1$ 
      else if  $random(p) \leq P$  then
        B ←  $\emptyset$ 
      end if
    end if
    if  $B \neq \emptyset$  then
      Randomly pick and flip  $x \in B$ 
       $age(x) \leftarrow ++flips$ 
    else
      Increase weight on all false clauses
      if  $++increases$  % DECREASE = 0 then
        Decrease weight on all weighted clauses
      end if
    end if
  end while
end

```

Fig. 2. The MAX-AGE Algorithm

MAX-AGE algorithm from Figure 2, which alters *DLM-SAT* by replacing the tabu list and the TABU and FLAT parameters with the heuristics described previously in Section 2.3 (for further comparison of DLM with other leading SAT algorithms see [9], [10] and [6]).

3.1 Problem Domains

For our problem set we chose the four problem domains for which existing DLM parameters have already been developed (namely random 3-SAT, parity learning, graph colouring and blocks world). Using the DIMACS benchmarks we selected *f400* to *f3200* for random 3-SAT, *par16-1-c* to *par16-5-c* for parity learning, all the *g* graph colouring problems and the SATLIB *bw-large-a* to *bw-large-d* for blocks world. Due to the length of run-times, we did not produce a full set of results for the more difficult unsimplified *par16*, *par32* and *hanoi* problems. However we did confirm that *DLM2K* has the superior performance for these problems (due to the operation of the *SPECIAL-INCREASE* heuristic). We expect the addition of an equivalent heuristic to *MAX-AGE* would produce a similar performance improvement, but did not explore this option as it adds a further parameter to the problem.

3.2 Parameter Setting

One of the main advantages of *MAX-AGE* is that it eliminates the task of setting the TABU and FLAT parameters. We were therefore able to quickly tune *MAX-AGE* by selecting a single problem from each domain and varying the value of DECREASE until an optimum point was found. The probability P of taking a plateau move in *MAX-AGE* was treated as a constant and set at 0.85, although we did examine the effects of varying P on several example problems (see Section 3.3). For *DLM-SAT* we took the published values of FLAT and TABU for each domain, but again experimented with varying DECREASE to see if the optimum DECREASE for *DLM-SAT* was equivalent to *MAX-AGE* (the final DECREASE values for each method are shown in the D column of Table 1). Finally the *DLM2K* parameters were read directly from the *dmlparam* files supplied for each problem domain with the *DLM2K* source code.

3.3 Results

The average flips over 100 runs on the complete problem set, allowing 100 million flips per run, are shown in Table 1. In Table 2 we present the median flips (to provide an idea of the shorter-term behaviour of *DLM-SAT* and *MAX-AGE*) and the CPU time usage for each method and problem.

The average flip data in Table 1 shows *MAX-AGE* performs competitively with *DLM-SAT* and that both *MAX-AGE* and *DLM-SAT* can equal or exceed *DLM2K* on all four problem domains. In particular, *MAX-AGE* achieves above average performance on the *par16* problems and equals *DLM2K* on the harder graph colouring *g* problems. The graph colouring results are interesting because *DLM-SAT* performs considerably worse on these problems, implying that the *SPECIAL-INCREASE* heuristic in *DLM2K* is playing an important role which *MAX-AGE* is able to replace. However, on the larger *par* and *hanoi* problems (not reported here) *SPECIAL-INCREASE* still provides a decisive advantage.

The median flips data in Table 2 shows a similar pattern to the average flips data, indicating neither *MAX-AGE* or *DLM-SAT* would gain an advantage from a random restarts strategy (this was confirmed by further estimates of expected flips, based on the work in [6]). However the CPU time usage does highlight an additional overhead of 10-20% for *MAX-AGE* in comparison to *DLM-SAT*. This is caused by *MAX-AGE* only accepting 15% of plateau flips and otherwise increasing weight. Weight increases create overhead in terms of updating the flip cost of each affected variable. Both *DLM* techniques avoid a proportion of this cost by searching plateaus more extensively under the control of the tabu list. However *MAX-AGE*'s overhead is not as significant as that imposed by the alternative multiplicate weighting schemes of SDF and ESG, and on several problems *MAX-AGE*'s performance increase outweighs the time penalty.

Finally, we examined the effects of varying the value of P from 0.6 to 0.95 in *MAX-AGE* across the whole problem set. These experiments showed that P does affect performance, but no clear pattern emerged. For instance, on the 3-SAT *f* problems a P value of 0.85 is consistently better whereas on the *par* problems the optimum value ranges from 0.8 to 0.95 and on the larger *g* problems a value of 0.8 works better. Given this

variation, treating P as a constant at 0.85 appears the best compromise, although meeting another problem domain where a significantly different value P is required would change our conclusions. Overall the results indicate that the tabu list and its associ-

Average Flips over 100 runs						
Problem	D	DLM2K	D	DLM-SAT	D	MAX-AGE
f400	12	11,853	10	8,265	9	6,787
f600	12	66,846	10	33,947	9	30,745
f800	12	463,173	10	150,826	9	154,048
f1000	12	303,784	10	136,140	9	157,357
f1600	12	5,597,990	10	1,990,000	9	3,134,047
f2000	12	678,294	10	1,140,000	9	1,014,339
f3200	12	6,268,780	10	4,458,820	9	7,319,200
g125.17	7	813,463	6	1,693,360	4	729,628
g125.18	7	8,876	6	23,645	4	13,171
g250.15	7	2,309	6	2,272	4	2,212
g250.29	7	342,935	6	744,337	4	374,155
par16-1-c	46	3,917,350	40	2,771,220	40	1,958,802
par16-2-c	46	8,018,580	40	5,818,140	40	4,272,580
par16-3-c	46	6,920,230	40	5,672,240	40	3,941,307
par16-4-c	46	6,413,150	40	3,969,650	40	2,579,460
par16-5-c	46	6,150,910	40	5,236,300	40	4,269,350
bw-large.a	5	4,911	4	4,786	5	4,596
bw-large.b	5	51,203	4	54,480	5	76,968
bw-large.c	5	2,608,970	4	1,305,650	5	1,146,401
bw-large.d	5	6,992,680	4	2,082,610	5	1,746,582

Table 1. Average flips and DECREASE parameter (D)

ated parameters in DLM can be replaced by simpler clause weighting approach without loss of performance. Although DLM’s tabu list has some run-time advantage over using weights to escape plateaus (by adding weight less often), *MAX-AGE* balances this by showing more robust performance over the problem set, especially in outperforming *DLM-SAT* on the *par16* problems and matching the performance of *DLM2K* on the larger graph colouring problems (without using *SPECIAL-INCREASE*). Additionally, *MAX-AGE* has the advantage of not having to tune the length of the tabu list to each problem domain, or to decide on the optimum number of flat moves before adding weight.

4 Conclusions

The aim of this study was to produce a simplified clause weighting algorithm with comparable performance to the state-of-the-art SAT techniques. To this end we have

Problem	DLM-SAT			MAX-AGE		
	Median Flips	CPU Time	Flips/Sec.	Median Flips	CPU Time	Flips/Sec.
f400	5,411	0.08	103,313	5,545	0.07	102,209
f600	27,875	0.35	96,991	22,533	0.34	90,507
f800	94,173	1.85	81,528	102,000	2.18	70,671
f1000	95,373	1.74	78,241	106,717	2.32	67,718
f1600	1,395,129	29.24	68,049	1,990,812	57.69	54,328
f2000	705,188	17.55	64,959	554,880	17.97	56,616
f3200	10,441,781	90.94	49,028	2,682,022	134.25	54,521
g125.17	1,148,187	100.49	16,851	607,922	53.31	13,687
g125.18	15,638	1.47	15,997	10,187	2.92	4,514
g250.15	2,257	17.34	131	2,199	18.15	122
g250.29	490,777	196.24	3,793	315,791	144.02	2,598
par16-1-c	1,901,390	21.94	126,284	1,383,090	17.72	110,549
par16-2-c	4,296,443	46.48	125,175	3,505,468	38.74	110,278
par16-3-c	3,805,261	44.45	127,617	2,849,531	34.61	113,882
par16-4-c	2,604,387	31.39	126,452	1,859,387	22.75	113,374
par16-5-c	3,474,445	40.65	128,799	3,168,043	37.56	113,657
bw-large.a	3,412	0.01	59,825	3,697	0.08	56,191
bw-large.b	36,111	1.14	47,789	62,840	1.56	49,209
bw-large.c	718,075	37.23	35,072	719,077	37.26	30,768
bw-large.d	1,364,447	118.80	17,531	1,023,887	96.91	18,023

Table 2. Median flips and CPU usage

developed the single parameter *MAX-AGE* algorithm and shown it to have comparable and, in some cases, superior performance with the latest versions of *DLM*. In terms of the future development of clause weighting algorithms *MAX-AGE* has highlighted two points:

- the use of a tabu list and its associated parameters is not a necessary feature of an efficient clause weighting technique.
- additive weighting schemes can be simply controlled by a single parameter and can be more efficiently implemented than alternative multiplicative schemes.

We consider *MAX-AGE* as a step towards developing more intelligent constraint solving technologies that do not rely on a manual fine-tuning of parameters. In future work we will look further into predicting the best value of the remaining *DECREASE* parameter via an analysis of various run-time measures and incorporate this within a self-tuning algorithm. Also for further research is the incorporation of parameter-free versions of *DLM*'s *SPECIAL-INCREASE* and/or *DISTANCE-PENALTY* into *MAX-AGE* to improve performance on the more difficult DIMACS benchmark problems.

References

1. J. Frank. Learning short term weights for GSAT. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, pages 384–389, 1997.
2. I. Gent and T. Walsh. Towards an understanding of hill-climbing procedures for SAT. In *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-93)*, pages 28–33, 1993.
3. F. Glover. Tabu search: Part 1. *ORSA Journal on Computing*, 1(3):190–206, 1989.
4. B. Mazure, S. Lakhdar, and E. Gregoire. Tabu search for SAT. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, pages 281–285, 1997.
5. P. Morris. The Breakout method for escaping local minima. In *Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-93)*, pages 40–45, 1993.
6. D. Schuurmans and F. Southey. Local search characteristics of incomplete SAT procedures. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-00)*, pages 297–302, 2000.
7. D. Schuurmans, F. Southey, and R. Holte. The exponentiated subgradient algorithm for heuristic Boolean programming. In *Proceedings of the Seventeenth International Joint Conference on Artificial Intelligence (IJCAI-01)*, pages 334–341, 2001.
8. B. Selman and H. Kautz. Domain-independent extensions to GSAT: Solving large structured satisfiability problems. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence (IJCAI-93)*, pages 290–295, 1993.
9. Y. Shang and B. Wah. A discrete Lagrangian-based global search method for solving satisfiability problems. *J. Global Optimization*, 12:61–99, 1998.
10. Z. Wu and B. Wah. Trap escaping strategies in discrete Lagrangian methods for solving hard satisfiability and maximum satisfiability problems. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, pages 673–678, 1999.
11. Z. Wu and B. Wah. An efficient global-search strategy in discrete Lagrangian methods for solving hard satisfiability problems. In *Proceedings of the Seventeenth National Conference on Artificial Intelligence (AAAI-00)*, pages 310–315, 2000.