A Two Level Local Search for MAX-SAT Problems with Hard and Soft Constraints [⋆]

John Thornton, Stuart Bain, Abdul Sattar, and Duc Nghia Pham

School of Information Technology,
Griffith University Gold Coast, Southport, Qld, Australia, 4215
{j.thornton,s.bain,a.sattar,duc.pham}@mailbox.gu.edu.au

Abstract. Local search techniques have attracted considerable interest in the AI community since the development of GSAT for solving large propositional SAT problems. Newer SAT techniques, such as the Discrete Lagrangian Method (DLM), have further improved on GSAT and can also be applied to general constraint satisfaction and optimisation. However, little work has applied local search to MAX-SAT problems with hard and soft constraints. As many real-world problems are best represented by hard (mandatory) and soft (desirable) constraints, the development of effective local search heuristics for this domain is of significant practical importance.

This paper extends previous work on dynamic constraint weighting by introducing a two-level heuristic that switches search strategy according to whether a current solution contains unsatisfied hard constraints. Using constraint weighting techniques derived from DLM to satisfy hard constraints, we apply a Tabu search to optimise the soft constraint violations. These two heuristics are further combined with a dynamic hard constraint multiplier that changes the relative importance of the hard constraints during the search. We empirically evaluate this new algorithm using a set of randomly generated 3-SAT problems of various sizes and difficulty, and in comparison with various state-of-the-art SAT techniques. The results indicate that our dynamic, two-level heuristic offers significant performance benefits over the standard SAT approaches.

1 Introduction

Problems containing hard and soft constraints are very common in real world situations. For example, a typical university timetabling problem contains hard constraints specifying that only one class can be scheduled in a particular room at a particular time, with additional soft constraints expressing preferences for class times (e.g. a lecturer may prefer not to teach on Fridays). The addition of soft constraints changes the standard formulation of a Constraint Satisfaction Problem (CSP) into an over-constrained optimisation problem, where the objective is to satisfy all hard constraints and maximise the level of satisfaction of the soft constraints (according to some predefined metric).

[⋆] The authors gratefully acknowledge the financial support of the Australian Research Council, grant A00000118, in the conduct of this research

1.1 Representing and Solving Over-Constrained Problems

Various approaches have been proposed to represent over-constrained problems within the constraint satisfaction literature, including Freuder and Wallace's seminal paper on partial constraint satisfaction [5]. Here the objective is to maximise the total number of satisfied constraints (rather than to satisfy the hard constraints and then maximise the number of satisfied soft constraints). In [2] the idea of a two-level distinction between hard and soft constraints is extended to a multiple level constraint hierarchy, where constraints are classified into separate levels, such that the constraints of each succeeding level are strictly less important than any one constraint of the previous level. More recently, [1] proposed the more general semiring framework for representing over-constrained problems, which is capable of modelling traditional CSPs, constraint hierarchies, fuzzy CSPs and probabilistic CSPs.

Although these formalisms for representing over-constrained problems are independent of the algorithms used to solve these problems, most of the work in the area has concentrated on the use of complete techniques. For example, in [5] the standard backtracking approach for CSPs is extended to solve over-constrained problems using branch-and-bound and in both [1] and [2] arc-consistency techniques are developed, with [1] introducing extensions to the Davis-Putnam algorithm for use with semirings, and [2] detailing linear programming methods for use with constraint hierarchies. However, more recent work [7] has successfully applied local search to hierarchical constraint satisfaction.

1.2 Local Search with Hard and Soft Constraints

In this paper we are interested in applying local search techniques to over-constrained problems with hard and soft constraints. While local search can be trivially applied to the problem of maximising the total number of satisfied constraints, little work has been done in extending local search to over-constrained problems with hard and soft constraints. A start in this direction was made in [9] where the well known WalkSAT local search technique was applied to the weighted MAX-SAT problem. In this paper, the hard constraints were represented by making the weighted cost of each hard constraint exceed the sum of the weighted cost of all soft constraints. In this way the search always prefers a solution that satisfies all hard constraints regardless of the level of soft constraint violation. In [3] this work was taken further by recognising that the larger the weight differential between hard and soft constraints, the slower the search. This insight was factored into a constraint weighting algorithm by setting the hard constraint weight differential to a hand-tuned optimal level (generally slightly exceeding the average count of soft constraint violations during the search). Parallel work on Tabu search [14] looked at dynamically adjusting the weight on constraint subclasses according to whether all constraints in the subclass are satisfied (so reducing weight) or unsatisfied (so increasing weight) during a fixed period of iterations. This work was combined in [18], which introduced a dynamic constraint weighting technique capable of both adding weights to frequently violated constraints and maintaining a dynamic weight differential between the hard and soft constraints.

1.3 Recent Applications of Local Search to MAX-SAT

More recently, state-of-the-art local search SAT techniques, such as the Discrete Lagrangian Method (DLM) [16] and Guided Local Search (GLS) [12] have been successfully applied to the DIMACS benchmark over-constrained *jnh* MAX-SAT problems. The *jnh* problems are (mostly) over-constrained, with each constraint having a weight between 1 and 1000, and the search objective being to find a minimum weighted cost solution. Both DLM and GLS are *constraint weighting* techniques that add weight to unsatisfied constraints while simultaneously keeping track of original *jnh* problem weights. In the reported studies, GLS marginally outperformed DLM on the *jnh* problems but was unable to match DLM on the larger DIMACS challenge problems [11].

The motivation of this study is to advance the state-of-the-art in solving large hard and soft constraint problems. Given that constraint weighting heuristics (such as DLM and GLS) represent the state-of-the-art for SAT and weighted MAX-SAT problems [19], we decided to use a weighting heuristic as the basis for a new approach. As existing work in the area indicates that the simple fixed hard constraint multiplier used by DLM and GLS (on the weighted MAX-SAT problems) is not the best way to deal with hard constraints, we further decided to incorporate the dynamic hard constraint multiplier proposed in [18] into our new approach.

In the remainder of the paper we explain in more detail the principles underlying constraint weighting and introduce a modified constraint weighting algorithm based on DLM. We then explain and combine the dynamic constraint weighting heuristic from [18] into the modified algorithm. As a result of empirical testing of this technique on a range of randomly generated MAX-SAT problems with hard and soft constraints, we further introduce a second level Tabu search heuristic [6] that controls the search in areas where only soft constraints are violated. Finally, this two level dynamic constraint weighting heuristic is evaluated against the original one-level heuristic, a pure Tabu search and the well known NOVELTY+ SAT algorithm.

2 Constraint Weighting Local Search

The basic idea of constraint weighting is to change the cost of the solution space by adding weights to frequently violated constraints. In the SAT domain, a constraint weighting algorithm (such as [13]) will start with a complete random instantiation of variables and proceed to take local cost improving moves (i.e. flipping or changing the variable that causes the greatest decrease in the number of false clauses) until a *local minimum* is reached where no more cost improving moves are available. The algorithm will then add weight to all currently false constraints (i.e. clauses) and continue the search. In this way the local minimum is "filled in" [13] and the search can progress to new areas. Constraint weighting continues visiting local minima and adding weights until either a global minimum is reached (with cost zero) or the algorithm is timed out.

2.1 Weight Reduction Heuristics

Although simple constraint weighting techniques met with success on smaller SAT problems, other non-weighting local search techniques (such as RNOVELTY [10]) proved superior on larger and more difficult problems. It was reasoned that the deterioration of weighting techniques on these problems was due an excessive build up of weights, making the addition of weight in the later stages of the search less and less effective [4]. Therefore various weight reduction or normalisation schemes were proposed, most notably DLM [16] and later GLSSAT [11] and Smoothed Descent and Flood (SDF) [15]. Of these, SDF opted for a continuous normalisation of weights after each weight increase and adopted a multiplicative weight increment scheme, whereas both DLM and GLSSAT used an additive scheme with periodic weight reduction. On the basis of the published empirical results for these techniques, SDF proved promising on smaller problems but due to the large overhead of continuous weight adjustments was unable to match the overall performance of DLM. Similarly GLSSAT proved competitive on small problems, but not on the larger DIMACS instances (for instance compare the *par* problem results in [11] with [19]). On this basis we decided to use DLM as the starting point for our own work.

2.2 Simplifying DLM

There have been several versions of DLM, mainly developed to solve the larger and more challenging DIMACS *par* and *hanoi* problems. In this study we decided to remove the specialised heuristics from DLM and concentrate on the core effectiveness of the technique. We therefore retained the DLM weight reduction scheme but removed the Tabu list and Flat Moves parameters that are used to help DLM explore plateau areas[1]. This resulted in the development of the *MAX-AGE* heuristic [17], which uses a simple random probability of taking a zero cost move on a plateau (otherwise weight is added) and also has a bias to accept infrequently used zero cost moves. In this way, the difficult to tune Tabu list length and Flat Moves parameters from DLM are replaced by a fairly robust zero cost move probability (set at 15% in the current study). Our empirical studies have already shown *MAX-AGE* to be comparable to DLM on a range of the larger DIMACS problems, trading slightly fewer moves for the slightly increased overhead of adding and reducing weight more often [17] (an adapted *MAX-AGE* algorithm for solving hard and soft constraint problems is shown in Figure 1).

[1] A plateau is an area where only equal or deteriorating cost moves are available.

```
procedure MAX-AGE
begin
    Generate a random starting point
    $bestSolutionCost \leftarrow$ unweighted solution cost
    $hardMultiplier \leftarrow 1$
    Initialise counters and clause weights to zero
    while $bestSolutionCost <$ objective and $flips <$ maxFlips do
        $B \leftarrow$ 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
                $B \leftarrow x$
                $maxAge \leftarrow maxAge + 1$
            else if $random(p) \leq P$ then
                $B \leftarrow \emptyset$
            end if
        end if
        if $B \neq \emptyset$ then
            Randomly pick and flip $x \in B$
            $age(x) \leftarrow ++flips$
            if unweighted solution cost $< bestSolutionCost$ then
                $bestSolutionCost \leftarrow$ unweighted solution cost
        else
            Increase weight on all false clauses
            if false hard clauses exist then $++hardMultiplier$
            else if $hardMultiplier > 1$ then $--\ hardMultiplier$
            if $++increases$ % DECREASE = 0 then
                Decrease weight on all weighted clauses
        end if
    end while
end
```

Fig. 1. The *MAX-AGE* Algorithm for Hard and Soft Constraints

2.3 Dynamic Hard and Soft Constraint Weighting

The *MAX-AGE* algorithm was initially developed for solving *satisfiable* problems. However, it is a trivial exercise to adapt local search for over-constrained problems, as it already searches in the space of inconsistent assignments. Hence *MAX-AGE* also keeps track of the unweighted cost of each solution point visited in the search, in order to recognise when a new best cost solution has been found[2] (this is done using *bestSolutionCost* in Figure 1). As previously discussed (in Section 1.2), hard and soft constraints can be represented by giving a weight penalty to the hard constraints. This can be incorporated into a constraint weighting algorithm by considering the hard constraint weight to be a *hard constraint multiplier*, such that the weighted cost of a particular hard constraint is the product of the hard constraint multiplier and the actual weight currently added to the constraint. In effect, using a hard constraint multiplier with a value n is equivalent to solving a problem where each hard constraint is repeated n times [3].

In [18] it was further demonstrated that performance gains could be obtained by dynamically adjusting the value of the hard constraint multiplier during the search. This idea is incorporated into Figure 1 using *hardMultiplier*. Here *MAX-AGE* continuously increases *hardMultiplier* at each local minimum (from the

[2] In practice a copy of the current best solution found would also be kept.

initial value of one) until a solution is found that satisfies all hard constraints. In this way, the relative importance of the hard constraints is raised until they are all satisfied. Then, in hard constraint satisfying local minima, the value of *hardMultiplier* is decremented to bring the search back to the point where the summed cost of the currently false soft constraints just balances the cost of making a hard constraint false.

The use of a dynamic hard constraint multiplier also changes the problem of measuring *bestSolutionCost*. In a system where each hard constraint is more important than the sum of all soft constraints, a solution that satisfies all hard constraints will always be preferred over a solution that does not. However allowing the hard constraint multiplier to vary destroys this property. Hence, when calculating the unweighted solution cost in Figure 1, we must reintroduce the property that each hard constraint violation costs $ns + 1$ where n is the number of soft constraints and s is the cost of violating a soft constraint.

3 A Two-Level Heuristic for Hard and Soft Constraints

In order to evaluate the hard and soft constraint version of *MAX-AGE* we developed two non-weighting local search algorithms using *NOVELTY+* [8] and an augmented Tabu search (*TABU*) based on the SAT source code developed in [15]. Starting with this code, we incorporated the hard constraint multiplier (described in the previous section) into both algorithms and additionally incorporated a random move feature and aspiration condition [6] into *TABU*. Our original SAT Tabu heuristic disallowed the undoing of a move during the search until t subsequent moves have been made (i.e. simulating a Tabu list of length t). To this we added an aspiration condition that always allows cost improving moves and moves that improve on the best cost yet achieved for a given variable value (otherwise tabu moves are disallowed). Then, in a situation where all moves are tabu, we randomly select a move from those variables that are involved in constraint violations.

Initial tests on *TABU* showed that it is relatively poor at finding hard constraint satisfying instantiations on problems where this task was *in itself* difficult. However, on problems where the hard constraint problem is relatively easy, *TABU* proved very effective at optimising the level of soft constraint violations. These observations lead us to develop a two-level heuristic using constraint weighting to satisfy the hard constraints and a Tabu search to satisfy the soft constraints (shown in Figure 2). This *TWO-LEVEL* move selection heuristic replaces the *MAX-AGE* heuristic that populates B in Figure 1. In *MAX-AGE*, moves are selected according to the minimum cost given by the function:

$$solutionCost = \alpha \sum_{i=1}^{n} w(h_i) + \sum_{j=1}^{m} w(s_j)$$

where α = the current value of *hardMultiplier*, n = the total number of hard constraints, m = the total number of soft constraints, h_i is hard constraint i, s_j

is soft constraint j and $w(x_i)$ returns the weighted cost of constraint x_i if x_i is false, zero otherwise. In the *TWO-LEVEL* cost function, the $\sum_{j=1}^{m} w(s_j)$ term is replaced by $\sum_{j=1}^{m} c(s_j)$ where $c(s_j)$ returns one (rather than the weighted cost) if soft constraint s_j is false, zero otherwise.

```
function TWO-LEVEL MOVE SELECTION
begin
    $bestChange \leftarrow 0$
    $hCount \leftarrow$ total false hard constraints in current solution
    for each false constraint $f_i$ do
        for each variable $v_j \in f_i$ do
            $hChange \leftarrow$ change in weighted hard constraint cost from flipping $v_j$
            $sChange \leftarrow$ change in unweighted soft constraint cost from flipping $v_j$
            $costChange \leftarrow (hardMultiplier \times hChange) + sChange$
            if $costChange \leq bestChange$ then
                $age \leftarrow$ number of flips since $v_j$ was last flipped
                $sCount \leftarrow$ total false soft constraints resulting from flipping $v_j$
                if $age \leq$ hardTabu then $tabu \leftarrow$ true
                else if $hCount \leq hChange$ and $age \leq sCount$ then $tabu \leftarrow$ true
                else $tabu \leftarrow$ false
                $hBest \leftarrow$ least count of false hard constraints yet achieved by $v_j$
                $sBest \leftarrow$ least count of false soft constraints yet achieved by $v_j$
                if $hCount - hChange < hBest$ or
                    ($hCount = 0$ and $sCount < sBest$) then $aspired \leftarrow$ true
                else $aspired \leftarrow$ false
                if not $tabu$ or $aspired$ then
                    if $costChange < bestChange$ then
                        $B \leftarrow \emptyset$
                        $bestChange \leftarrow costChange$
                    end if
                    $B \leftarrow v_j$
                end if
            end if
        end for
    end for
    if $B = \emptyset$ and $hCount = 0$ then
        $B \leftarrow$ randomly selected $v_j$ from randomly selected $f_i$
    return $B$
end
```

Fig. 2. The *TWO-LEVEL* move selection heuristic

The main principle of the *TWO-LEVEL* heuristic is that weight is only considered on violated *hard* constraints. For this reason a random move is required in the Tabu heuristic to escape local minima in regions where all hard constraints are satisfied (as adding weight in these areas will have no effect). Additionally, a hardTabu constant (= 3) is used to avoid immediately undoing the random move. As a major aim in developing *MAX-AGE* was to reduce the complexity of the parameters used in DLM (see Section 2.2) the length of the soft Tabu list in *TWO-LEVEL* was set equal to the current number of unsatisfied soft constraints in the search (rather than introduce another Tabu list length parameter). This decision was based on empirical observations rather than a strong theoretical justification, and may therefore have to be revised in the light of further evidence.

Finally the *DECREASE* parameter from Figure 1 was augmented in *TWO-LEVEL* so that the frequency of constraint weight decreases depends on the proportion of time the search spends on satisfying soft constraints, according to the following scheme:

$$AugmentedDecrease = DECREASE + \frac{softMoves \times 100}{softMoves + hardMoves}$$

where $DECREASE$ has a constant value of 10 (taken from [19]), $softMoves =$ the number of moves taken with all hard constraints satisfied and $hardMoves$ $=$ the number of moves taken with at least one hard constraint violated. The *AugmentedDecrease* feature therefore slows the rate of decrease of the hard constraint weights when the search remains in areas where all hard constraints are satisfied.

4 Experimental Study

4.1 Problem Generation

In order to evaluate the *TWO-LEVEL* algorithm we decided to look at four problem dimensions: problem size, overall problem difficulty, the hard constraint problem difficulty and the ratio of hard to soft constraints. As we were unable to find benchmark problems that vary in all these dimensions, we decided to sample the space of randomly generated 3-SAT problems. While random problems do not measure the effect of structure on algorithm performance they enable us to control average problem difficulty by varying the clause to variable ratio. In addition, using 3-SAT problems we can arbitrarily divide clauses into hard and soft constraints and still have a measure of the relative difficulty of the hard constraint sub-problem.

We based our empirical study on three sets of randomly generated 3-SAT problems all sampled from the phase transition region (i.e. with a clause to variable ratio of 4.3). To make the problems relatively challenging (in terms of size) we generated 10 problems with 400 variables, 10 with 800 variables and 10 with 1600 variables, plus we used one f-series problem at each size from the DIMACS SAT benchmark library. Then, for each problem, we randomly generated and added a extra clause for each clause in the original problem (e.g. for a 400 variable 1720 clause problem we added another 1720 clauses), resulting in a second problem set with a clause to variable ratio of 8.6. This problem set was further multiplied by dividing up the hard and soft constraints into three ratios: 50%, 75% and 100%, where an $n\%$ ratio means the first $n\%$ of the clauses in the original problem are defined as hard constraints (e.g. in a 400 variable 3440 clause problem, if $n = 50$, the first 50% of the 1720 clauses which made up original problem are defined as hard). Using these problem generation procedures we constructed 4 problem classes at each of the 3 problem sizes, making a total of 12 data sets, each containing 11 individual problems. The data sets are identified using the format hxhns, where x specifies the number

		Solved	Soft Cost			Number of Flips			Time
Problem	Method	% of 110	Mean	Max	Min	Mean	Median	Std Dev	Mean
h400h50s	*TABU*	100.00	105.66	118	98	385499	358115	278057	9.98
	NOVELTY+	100.00	167.55	181	156	483528	455726	289826	7.04
	MAX-AGE	100.00	135.08	146	123	412152	395993	344137	18.16
	TWO-LEVEL	100.00	104.15	114	98	218261	111305	236683	8.53
h400h75s	*TABU*	100.00	123.17	144	108	487509	463778	281986	13.48
	NOVELTY+	100.00	168.71	182	148	519810	520912	283110	7.01
	MAX-AGE	100.00	148.19	158	131	355046	289414	291971	19.14
	TWO-LEVEL	100.00	119.10	129	108	314484	264277	285989	10.07
h400h100s	*TABU*	0.00	n/a	n/a	n/a	n/a	n/a	n/a	n/a
	NOVELTY+	99.09	180.59	200	152	423190	385968	268140	3.56
	MAX-AGE	46.36	180.51	200	156	453510	394879	279855	17.87
	TWO-LEVEL	73.64	175.33	198	151	299752	162736	299894	8.36
orig400	*TABU*	98.18	0.00	0	0	95725	41601	135638	0.42
	NOVELTY+	100.00	0.00	0	0	97280	33417	144974	0.35
	MAX-AGE	100.00	0.00	0	0	29682	14164	41978	0.16
	TWO-LEVEL	100.00	0.00	0	0	41638	13217	92212	0.22
h800h50s	*TABU*	9.09	245.00	259	214	524697	512840	235755	26.78
	NOVELTY+	100.00	359.32	377	330	451359	424782	270922	10.43
	MAX-AGE	100.00	289.63	313	257	181927	1993	306001	16.75
	TWO-LEVEL	100.00	201.45	212	193	391025	334046	258612	35.61
h800h75s	*TABU*	0.00	n/a	n/a	n/a	n/a	n/a	n/a	n/a
	NOVELTY+	100.00	371.39	391	347	503110	501893	277671	11.52
	MAX-AGE	100.00	318.11	337	295	361073	352352	271479	39.10
	TWO-LEVEL	100.00	241.75	258	228	398243	399929	303827	26.83
h800h100s	*TABU*	0.00	n/a	n/a	n/a	n/a	n/a	n/a	n/a
	NOVELTY+	90.91	369.62	412	339	505453	468170	287641	6.82
	MAX-AGE	0.91	329.00	329	329	816356	816356	0.00	70.35
	TWO-LEVEL	46.36	343.78	381	313	490876	514804	268261	30.98
orig800	*TABU*	41.82	0.00	0	0	361592	281072	247563	2.23
	NOVELTY+	90.00	0.00	0	0	225248	118765	226833	1.10
	MAX-AGE	100.00	0.00	0	0	108463	68454	136127	0.69
	TWO-LEVEL	100.00	0.00	0	0	87370	49610	108346	0.63
h1600h50s	*TABU*	0.00	n/a	n/a	n/a	n/a	n/a	n/a	n/a
	NOVELTY+	100.00	788.71	822	740	470748	402339	297055	19.52
	MAX-AGE	100.00	634.36	676	570	101848	4019	240074	28.39
	TWO-LEVEL	100.00	410.84	427	393	571933	584322	252455	161.00
h1600h75s	*TABU*	0.00	n/a	n/a	n/a	n/a	n/a	n/a	n/a
	NOVELTY+	100.00	790.4	832	740	515291	553617	288798	21.28
	MAX-AGE	100.00	681.71	715	642	396506	353132	289177	118.33
	TWO-LEVEL	100.00	484.73	503	452	384046	350721	265745	78.59
h1600h100s	*TABU*	0.00	n/a	n/a	n/a	n/a	n/a	n/a	n/a
	NOVELTY+	9.09	745.90	804	708	450650	399261	304313	10.76
	MAX-AGE	0.00	n/a	n/a	n/a	n/a	n/a	n/a	n/a
	TWO-LEVEL	2.73	674.33	693	637	523352	372545	289618	100.02
orig1600	*TABU*	0.00	n/a	n/a	n/a	n/a	n/a	n/a	n/a
	NOVELTY+	6.36	0.00	0	0	411237	264014	292473	3.21
	MAX-AGE	35.45	0.00	0	0	398464	374962	253588	3.89
	TWO-LEVEL	85.45	0.00	0	0	244216	180327	203094	2.53

Table 1. Experimental Results

of variables and n specifies the ratio of hard constraints. In addition, the orig$x$ problems represent the original 3-SAT problems with the 4.3 clause to variable ratio.

The data sets are designed to test the effects of problem size and difficulty by varying the number of variables and constraints, and the effects of the proportion of hard constraints and the relative difficulty of the hard constraint problem by varying the proportion of hard constraints. For example, an h800h100s problem represents a difficult hard constraint problem as all the clauses in the original phase transition problem are defined hard, whereas in the corresponding h800h50s problem the hard constraint problem will be easier, as 50% of the clauses in the original phase transition problem have now become soft.

4.2 Experimental Analysis

The results in Table 1 compare the performance of *TWO-LEVEL* with the three control algorithms introduced in Section 3 (*MAX-AGE*, *TABU* and *NOVELTY+*), giving averages of 10 runs on each problem (i.e. 110 runs per data set) with each run timed out after 1,000,000 flips. As *TABU* and *NOVELTY+* both perform flips in significantly less time than the clause weighting algorithms, we performed further experiments allowing *TABU* and *NOVELTY+* extra time to see if further cost reductions were possible. These CPU time results are summarised in the two graphs of Figures 3 and 4 showing anytime curves for each algorithm on the h400h75s and h1600h75s data sets respectively.

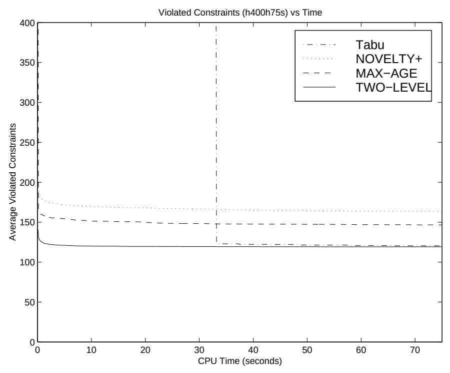

Fig. 3. h400h75s Results

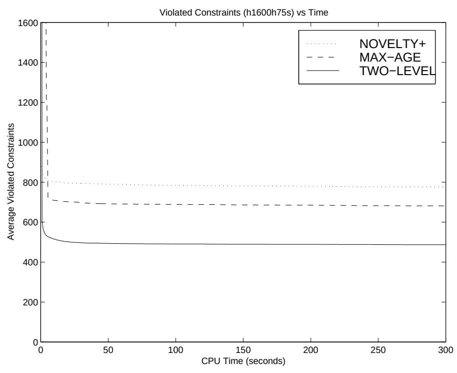

Fig. 4. h1600h75s Results

Overall the results in Table 1 confirmed our expectations that *TWO-LEVEL* would outperform both the *MAX-AGE* and *TABU* heuristics on which it is based. While *TABU* performed well on the smallest problems with the easiest hard constraint sub-problem (h400h50s), it was uncompetitive on the larger and harder problems, timing out completely on the h1600 problems. Conversely, *MAX-AGE* showed a fairly consistent ability to find solutions across all data sets (only falling back somewhat on the larger, hardest problems h800h100s and h1600h100s). Also, as expected, *MAX-AGE* performed well on the original under-constrained data sets (orig400-1600), although *TWO-LEVEL* proved better on the larger orig800 and orig1600 problems. This suggests that *TWO-LEVEL*'s use of a hard Tabu list (see Figure 2) is also beneficial on larger under-constrained problems. Overall, however, *MAX-AGE* was consistently outperformed by *TWO-LEVEL* both in terms of average soft cost and in terms of average flip count and average CPU time.

The results for *NOVELTY+* in comparison to *TWO-LEVEL* present a more interesting case. While *TWO-LEVEL* consistently finds better soft cost solutions on the over-constrained problem sets, *NOVELTY+* is considerably more reliable in finding solutions to the hardest over-constrained problems (h400h100s, h800h100s and h1600h100s). Also, as *NOVELTY+* is significantly faster than *TWO-LEVEL* in terms of flips per second, the question arises whether *NOVELTY+* could find better solutions given an equal amount of time. To this end, we re-ran *NOVELTY+* and *TABU* on the hnh75s data sets, allowing the same time cut-off that was granted to the weighting algorithms. The graphs in Figures 3 and 4 show that these increased run-times do not alter the relative performance of the algorithms, with *TWO-LEVEL* consistently achieving the lowest soft cost, independently of the time cut-off point. Therefore, we can conclude that *TWO-LEVEL* has the better average performance on all problems except those where the problem of satisfying the hard constraints is itself difficult (i.e. the hn100s problems). In these cases *NOVELTY+* is preferred, as it shows a superior ability to find a hard constraint satisfying solution in the presence of soft constraints.

5 Conclusion

The paper has presented a new *TWO-LEVEL* algorithm for solving problems with hard and soft constraints, based on an amalgamation of a constraint weighting heuristic for satisfying hard constraints and a Tabu list with aspiration for optimising soft constraints. Our empirical study has demonstrated the usefulness of the *TWO-LEVEL* algorithm for solving a range of randomly generated MAX-SAT problems with hard and soft constraints and has shown the overall superiority of *TWO-LEVEL* on these problems in comparison to three control algorithms.

Of the three control algorithms, *NOVELTY+* proved of greatest interest, as it was able to achieve better results than *TWO-LEVEL* on those instances where the problem of satisfying the hard constraints is itself difficult (i.e. the ratio of hard clauses to variables = 4.3). This result suggests *NOVELTY+* is the better heuristic for satisfying hard constraints when soft constraints are also present. Consequently, in our future work, we intend to investigate the combination of *NOVELTY+* and *TABU* into an alternative two-level heuristic.

References

1. U. Bistarelli, S. Montanari and F. Rossi. Semiring-based constraint solving and optimization. *Journal of ACM*, 44(2):201–236, 1997.
2. A. Borning, B. Freeman-Benson, and M. Wilson. Constraint hierarchies. *Lisp and Symbolic Computation*, 5(3):223–270, 1992.
3. B. Cha, K. Iwama, Y. Kambayashi, and S. Miyazaki. Local search algorithms for partial MAX-SAT. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, pages 332–337, 1997.
4. J. Frank. Learning short term weights for GSAT. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, pages 384–389, 1997.
5. E. Freuder and R. Wallace. Partial constraint satisfaction. *Artificial Intelligence*, 58(1):21–70, 1992.
6. F. Glover. Tabu search: Part 1. *ORSA Journal on Computing*, 1(3):190–206, 1989.
7. M. Heinz, L. Fong, L. Chong, S. Ping, J. Walser, and R. Yap. Solving hierarchical constraints over finite domains. In *Proceedings of the Sixth International Symposium on Artificial Intelligence and Mathematics*, 2000.
8. H. Hoos. On the run-time behavior of stochastic local search algorithms for SAT. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, pages 661–666, 1999.
9. H. Jiang, Y. Kautz and B. Selman. Solving problems with hard and soft constraints using a stochastic algorithm for MAX-SAT. In *First International Joint Workshop on Artificial Intelligence and Operations Research*, 1995.
10. D. McAllester, B. Selman, and H. Kautz. Evidence for invariance in local search. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI-97)*, pages 321–326, 1997.
11. P. Mills and E. Tsang. Guided local search applied to the satisfiability (SAT) problem. In *Proceedings of the 15th National Conference of the Australian Society for Operations Research (ASOR'99)*, pages 872–883, 1999.
12. P. Mills and E. Tsang. Guided local search for solving SAT and weighted MAX-SAT problems. *Journal of Automated Reasoning*, 24:205–223, 2000.
13. 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.
14. A. Schaerf. Tabu search for large high school timetabling problems. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*, pages 363–368, 1996.
15. 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.
16. Y. Shang and B. Wah. A discrete Lagrangian-based global search method for solving satisfiability problems. *J. Global Optimization*, 12:61–99, 1998.
17. J. Thornton, W. Pullan, and J. Terry. Towards fewer parameters for SAT clause weighting algorithms. In *Proceedings of the Fifteenth Australian Joint Conference on Artificial Intelligence (AI'2002)*, To appear, 2002.
18. J. Thornton and A. Sattar. Dynamic constraint weighting for over-constrained problems. In *Proceedings of the Fifth Pacific Rim Conference on Artificial Intelligence (PRICAI-98)*, pages 377–388, 1998.
19. Wu Z. *The Theory and Applications of Discrete Constrained Optimization using Lagrange Multipliers.* PhD thesis, Department of Computer Science, University of Illinois, 2000.