Old Resolution Meets Modern SLS

Anbulagan[1], **Duc Nghia Pham**[2], **John Slaney**[1,3], **Abdul Sattar**[2]
[1]Logic and Computation Program, National ICT Australia Ltd., Canberra, Australia
[2]IIIS, Griffith University, Brisbane, Australia
[3]Computer Sciences Laboratory, Australian National University, Canberra, Australia
{anbulagan, john.slaney}@nicta.com.au, {d.n.pham, a.sattar}@griffith.edu.au

Abstract

Recent work on Stochastic Local Search (SLS) for the SAT and CSP domains has shown the importance of a dynamic (non-markovian) strategy for weighting clauses in order to escape from local minima. In this paper, we improve the performance of two best contemprorary clause weighting solvers, PAWS and SAPS, by integrating a propositional resolution procedure. We also extend the work to AdaptNovelty$^+$, the best non-weighting SLS solver in the GSAT/WalkSAT series. One outcome is that our systems can solve some highly structured problems such as quasigroup existence and parity learning problems which were previously thought unsuitable for local search and which are completely out of reach of traditional solvers such as GSAT. Here we present empirical results showing that for a range of random and real-world benchmark problems, resolution-enhanced SLS solvers clearly outperform the alternatives.

Introduction

Stochastic Local Search (SLS) is the method of choice in many SAT and CSP domains, provided proofs of unsatisfiability or strict optimality are not required. Here we consider SAT problems restricted to conjunctive normal form or clause form. This is a standard form in which to represent problems of propositional logic, in which class we may include finite domain CSPs. Most of the performance benchmarks in the propositional reasoning literature are for this class of problems.[1] SLS for SAT has been researched intensively since the introduction of GSAT (Selman, Levesque, & Mitchell 1992). GSAT, based on greedy random hill-climbing, was the first SLS solver to deal successfully with large size random and structured SAT problems that are too hard to be solved by systematic search methods. During the last decade, there have been many more systems based on new ideas such as noise strategies (Selman & Kautz 1993; Selman, Kautz, & Cohen 1994), Novelty and R-Novelty schemes (McAllester, Selman, & Kautz 1997), Novelty$^+$ (Hoos 1999) and different clause weighting strategies (Frank 1997; Hutter, Tompkins, & Hoos 2002; Morris 1993; Schuurmans & Southey 2000; Schuurmans, Southey, & Holte 2001; Thornton *et al.* 2004; Wu & Wah 2000). See (Hoos & Stulze 2005) for a history and a lucid account of these and many more local search methods.

The basic concept of clause weighting is simple: whenever a local minimum is encountered, the weight of unsatisfied clauses is increased, helping the search to avoid getting trapped in local minima. Several different clause weighting algorithms have been found to enhance the performance of SLS solvers. The current state of the art is represented by the systems SAPS (scaling and probabilistic smoothing) and RSAPS (Hutter, Tompkins, & Hoos 2002), and PAWS (pure additive weighting scheme) (Thornton *et al.* 2004). The winner of the 2004 SAT competition was a rather different solver, AdaptNovelty$^+$, also by Hoos and Tompkins, which uses a novelty scheme to order variables (rather than clauses) for selection and a history-dependent noise strategy. We include AdaptNovelty$^+$ in the experimental comparisons below.

In the present paper, we report on a technique for enhancing the performance of SLS solvers by incorporating a preprocessing phase in which resolution is used to deduce consequences of the input clauses, exposing hidden structure in the problems which the solvers are then able to exploit. We show that all of the SLS solvers AdaptNovelty$^+$, RSAPS and PAWS benefit markedly from this technique, especially on structured problems such as those resulting from real-world applications. The resolution-enhanced solvers R+ANOV$^+$, R+RSAPS and R+PAWS all outperform their parents, and are all able to solve problems which are beyond the range of earlier solvers such as GSAT and WalkSAT.

As a particularly challenging benchmark for empirical comparison of the solvers, we choose a range of problems concerning quasigroups (i.e. essentially, Latin squares). We report experiments with quasigroup existence problems, requiring solutions that satisfy particular algebraic equations. In passing, it is worth noting that SLS solvers with good clause weighting schemes are also applicable to quasigroup completion problems in which a Latin square is sought with some of the values specified in advance, though this problem class is not relevant to the present investigation because no nontrivial resolution between the input clauses is possible.

[1]This is not, of course, to deny the importance of the usual formulations of CSPs in terms of many-valued domains, nor of non-clausal encodings of propositional problems, but SAT *is* one of the central problem classes of automated reasoning, and enough is enough for one paper.

Part of the interest of quasigroup problems as benchmarks stems from early work by Gent and Walsh (1995) who found these problems to be extremely hard for the SLS solvers at that time and who conjectured that SLS methods are inherently unsuitable for them. Latin square problems exhibit an extreme "small world" topology: in the graph whose vertices are the variables and whose edges represent co-occurrence in clauses, the neighbours of each vertex consist of two cliques, while the longest path between any two vertices is of length 2. Consequently, a flip of any variable rapidly propagates everywhere, making it very difficult to detect nearby minima in the state space. Since 1995 it has been part of the "folk wisdom" of automated reasoning that quasigroup problems require systematic search methods. We therefore find it most interesting that contemporary SLS solvers can now achieve performance on satisfiable quasigroup problems comparable with that of systematic SAT solvers. Our results suggest that the superior performance of systematic methods stems not so much from the power of unit propagation as from the ability of a little reasoning to simplify problem encodings early in the search.

In our experimental study, we compare R+ANOV$^+$, R+PAWS, R+RSAPS and the underlying SLS solvers without resolution not only on quasigroup existence problems but also on random problems and a range of real-world problems taken from SATLIB (www.satlib.org) and used in previous SAT competitions. For further comparison, we include results for WalkSAT and R+WalkSAT on the same problems. In brief, except in the case of problems such as purely random ones where resolution gets little purchase, the results show that the resolution preprocessor brings order of magnitude improvements over the current state of the art.

In the next section we describe the resolution procedure and briefly review some of its uses in SAT solvers. In the subsequent section, we briefly outline the state-of-the-art SLS solvers to which resolution is most successfully added. We then present and discuss detailed experimental results[2], before concluding and indicating some future research directions.

Resolution Procedure for SAT

Resolution (Quine 1955; Davis & Putnam 1960; Robinson 1965) is widely used as a rule of inference in first order automated deduction, where the clauses tend to be few in number and contain few literals, and where the reasoning is primarily driven by unification. As a procedure for propositional reasoning, however, resolution is rarely used on its own because in practice it has not been found to lead to efficient algorithms.

It has been used as a *component* of propositional reasoning systems. In the world of complete search, it is sometimes used to enhance the performance of complete SAT solvers. The first efficient integration of resolution into such a solver was done by Billionnet and Sutter (1992) helping them to solve randomly generated 3-SAT problems with up to 300 variables. Later, the system Satz (Li & Anbulagan 1997) used a restricted resolution procedure, adding resolvents of length ≤ 3, as a first phase process before running the complete backtrack search. This implementation gave modest performance gains (around 10%) on random 3-SAT problems, but was very important to the ability of Satz to solve many real-world benchmark problems.

[2] All experiments were conducted on an Intel Pentium 4 PC with 2.4 GHz CPU, under Linux.

Algorithm 1 ComputeResolvents()

```
1: for each clause c_1 of length ≤ 3 in F do
2:    for each literal l of c_1 do
3:       for each clause c_2 of length ≤ 3 in F s.t. l̄ ∈ c_2 do
4:          Compute resolvent r = (c_1 \ {l}) ∪ (c_2 \ {l̄});
5:          if r is empty then
6:             return "unsatisfiable";
7:          else
8:             if r is of length ≤ 3 then
9:                F := F ∪ {r};
10:            end if
11:         end if
12:      end for
13:   end for
14: end for
```

In Algorithm 1, we sketch the resolution process implemented in Satz. When two clauses of a CNF formula have the property that some variable x_i occurs positively in one and negatively in the other, the resolvent of the clauses is a disjunction of all the literals occuring in the clauses except x_i and $\overline{x_i}$. For example, the clause $(x_2 \vee x_3 \vee \overline{x_4})$ is the resolvent for the clauses $(\overline{x_1} \vee x_2 \vee x_3)$ and $(x_1 \vee x_2 \vee \overline{x_4})$ and is added to the clause set. The new clauses, provided they are of length ≤ 3, can in turn be used to produce other resolvents. The process is repeated until saturation. Duplicate and subsumed clauses are deleted, as are tautologies and any duplicate literals in a clause. Clearly, the resolution phase runs in polynomial time, because in n variables there are only $\mathcal{O}(n^3)$ possible clauses of length 3, and any two clauses with a non-tautologous resolvent have exactly one pair of complementary literals, so the total number of resolvents generated, including duplicates, is bounded above by $\mathcal{O}(n^6)$. Since in real-world problems it is common for n to be on the order of 10^4, this theoretical upper bound is not especially reassuring; fortunately, as shown in the experimental results below, in practice it is far from tight.

In the incomplete search world, Cha and Iwama (1996) were the first to use a restricted form of resolution procedure called neighbour resolution, which adds new resolvent clauses based on unsatisfied clauses at the local minima and their neighboring clauses. Their study showed that on hard random 3-SAT formulas, the clause weighting strategy is significantly better with neighbour resolution integrated than without. More recently, Fang and Ruml (2004) implemented the same idea along with other techniques in their complete local search solver, where it provides a slight improvement.

Adding Resolution to Modern SLS Solvers

The introduction of GSAT (Selman, Levesque, & Mitchell 1992) and many subsequent SLS variants of WalkSAT (Sel-

man, Kautz, & Cohen 1994; McAllester, Selman, & Kautz 1997) has caused considerable research interest in modelling hard combinatorial problems into SAT and applying SLS solvers to find the solutions. The best contemporary solver in the WalkSAT family is AdaptNovelty$^+$ (Hoos 2002), which is an automated version of Novelty$^+$ (McAllester, Selman, & Kautz 1997). Like other WalkSAT variants, the performance of Novelty$^+$ critically depends on the setting of its noise parameter, which controls the greediness of the search. AdaptNovelty$^+$ addresses this problem by adaptively tuning its noise level based on the detection of stagnation: it starts with a 0 noise level. If no improvement in the objective evalution value, which is the count of unsatisfied clauses, has been made after a number of flips, the noise level is increased. As soon as the value of the objective function is improved over its value at the last change of the noise level, the noise level is reduced. Experimental results have shown that this adaptive noise mechanism also works well with other WalkSAT variants (Hoos 2002).

Recently, clause weighting based SLS algorithms, most notably SAPS (Hutter, Tompkins, & Hoos 2002) and PAWS (Thornton *et al.* 2004), have been very successfully applied to hard combinatorial SAT problems. These algorithms dynamically update the clause weights (or penalties) and hence modify the search landscape to effectively avoid or escape local minima during the search. Although these clause weighting SLS algorithms differ in the way that clause weights should be updated (additive or multiplicative updating) or when these weights should be updated (deterministic or probabilistic updating after a period of time), they all share an underlying strategy, which dynamically updates clause weights based on two mechanisms, *increasing* and *reducing*,[3] to escape local minima. When the search encouters a local minimum, it increases the weights of currently unsatisfied clauses. As violating these unsatisfied clauses now costs more than violating other clauses, the search is forced to move to a new neighbour in order to satisfy those current unsatisfied clauses. As a result, it escapes the local minimum. However, this increasing mechanism has side-effects: as it locally modifies the search landscape to escape the current local minimum, it possibly gives rise to other new local minima, which may in some cases be even harder to avoid or escape (Morris 1993; Tompkins & Hoos 2004). The reducing mechanism is employed to counter these side-effects. After a number of weight increases, the weights of all weighted clauses are reduced in order to make the search forget about the high costs of violating clauses which are no longer helpful since it escaped the local minima. These two mechanisms significantly increase the mobility of the search as well as helping to focus it on any "critical" clauses.

Our main observation is that clause weighting SLS algorithms benefit markedly from resolution preprocessing. We therefore propose a two-phase clause weighting SLS algorithm that combines the benefits of effective diversification from the clause weighting mechanism and having extra information from resolution. Our most successful solvers following this scheme use PAWS or RSAPS as the clause weighting SLS solver. We call these new algorithms R+PAWS and R+RSAPS respectively. In the first stage, they call the ComputeResolvents procedure in Algorithm 1 to add resolvent clauses to $\mathcal{F}$ and also remove duplicate clauses, the tautologies, and duplicate literals in a clause. They then run the SLS solver on the resulting clause set $\mathcal{F}_r$. The two-phase model is easily extended to other clause weighting SLS algorithms and even to non-weighting SLS algorithms such as AdaptNovelty$^+$.

[3] Also known as *scaling* and *smoothing* in SAPS and other multiplicative clause weighting SLS algorithms.

Quasigroup Problems

It is now a commonplace observation that techniques optimised for random SAT problems may run into difficulties when asked to solve more realistic problems which tend to be highly structured. For that reason there has been much interest in recent years in benchmark sets taken from such structured domains. Among these benchmarks are two problem classes concerning quasigroups: quasigroup completion and quasigroup existence. While former is a valuable benchmark, exhibiting a balance between randomness and structure, our present focus is on the latter, since the quasigroup existence problems give rise to clause sets to which the resolution preprocessor applies.

The quasigroup existence problems studied by Fujita *et al* (1993) consist in determining the spectra (that is, the sizes for which solutions exist) of certain algebraic equations over quasigroups. Problem qg7, for instance, calls for quasigroups satisfying the condition that $a.ba = ba.b$ for all elements a and b. It is known that solutions exist of all sizes n congruent to 1 (mod 4) with the possible exception of size $n = 33$. For our experiment we used sizes $n = 9$ and $n = 13$, the latter of which is moderately hard even for systematic search methods. The nomenclature is that followed in SATLIB and the SAT competition from which these particular encodings are taken: problem qg$i$-x is the i-th existence problem as numbered in (Fujita, Slaney, & Bennett 1993) for quasigroups of size x.

Table 1 shows the runtimes and flip counts of the various solvers on ten quasigroup existence problems, together with the number of runs out of 100 on which a solution was found within the limit of 10 million flips. As observed by Gent and Walsh (1995) GSAT cannot solve any of these problems within a reasonable time, and WalkSAT without preprocessing reliably solves only the smaller cases of problems qg1 and qg2. The null results for GSAT are omitted from the table. For the other solvers, the results are fairly consistent across the range of problems. In every case there is a significant improvement in runtime, and in every case the number of flips required to reach a solution decreases dramatically when the resolution step is performed. Problem qg7-09 in particular is reduced to triviality for two of the solvers.

The runtimes in Table 1 do not include the time taken by the resolution preprocessor. It is therefore important to record the preprocessing times (which are the same for all three solvers, of course) together with data about the effect of preprocessing on the problem sizes. It will be seen from Table 2 that resolution with the length 3 restriction serves to

		Success	Time (secs)		Flips	
Problem	Algorithm	%	Mean	Median	Mean	Median
qg1-07	WalkSAT	100	105.63	79.45	4,292,083	3,228,312
	ANOV$^+$	100	10.35	7.53	438,368	319,176
	RSAPS	100	1.58	1.13	27,744	19,840
	PAWS	100	3.63	2.94	88,497	71,276
	R+WalkSAT	100	0.08	0.06	59,015	41,208
	R+ANOV$^+$	100	0.03	0.02	15,465	10,928
	R+RSAPS	100	0.02	0.01	5,190	3,811
	R+PAWS	100	0.02	0.01	4,453	2,743
qg1-08	WalkSAT	0	n/a	n/a	10M	10M
	ANOV$^+$	57	588.88	718.99	6,997,215	8,543,181
	RSAPS	21	891.75	985.13	9,052,119	10M
	PAWS	78	205.74	166.16	2,847,046	2,997,505
	R+WalkSAT	25	21.24	24.64	8,616,412	10M
	R+ANOV$^+$	100	2.73	1.89	839,387	580,205
	R+RSAPS	100	13.91	9.69	2,062,302	1,436,738
	R+PAWS	100	1.94	1.38	293,365	208,404
qg2-07	WalkSAT	93	65.84	46.66	2,910,442	2,062,669
	ANOV$^+$	100	5.86	4.49	257,675	197,236
	RSAPS	100	0.64	0.48	10,759	8,060
	PAWS	100	1.44	1.11	35,719	27,257
	R+WalkSAT	100	0.05	0.03	36,053	21,960
	R+ANOV$^+$	100	0.01	0.01	7,610	5,196
	R+RSAPS	100	0.01	0.01	2,603	1,795
	R+PAWS	100	0.01	0.01	2,538	1,892
qg2-08	WalkSAT	0	n/a	n/a	10M	10M
	ANOV$^+$	24	888.31	1,003.18	8,854,866	10M
	RSAPS	16	1,005.66	1,080.50	9,307,338	10M
	PAWS	61	512.34	569.72	3,795,823	6,910,973
	R+WalkSAT	8	25.06	26.18	9,575,403	10M
	R+ANOV$^+$	98	9.79	8.47	2,852,501	2,467,205
	R+RSAPS	83	35.23	27.85	4,705,153	3,719,797
	R+PAWS	100	19.27	15.02	2,413,993	1,880,763
qg3-08	WalkSAT	28	11.77	14.00	8,409,379	10M
	ANOV$^+$	89	8.08	6.38	4,249,504	3,355,453
	RSAPS	100	0.23	0.16	46,904	32,580
	PAWS	100	0.66	0.50	140,685	105,531
	R+WalkSAT	100	0.16	0.12	167,288	127,427
	R+ANOV$^+$	100	0.10	0.07	84,833	58,417
	R+RSAPS	100	0.04	0.03	13,418	8,947
	R+PAWS	100	0.03	0.02	10,236	7,450
qg4-09	WalkSAT	10	16.66	17.77	9,374,716	10M
	ANOV$^+$	48	15.03	20.67	7,272,489	10M
	RSAPS	100	1.98	1.27	285,221	182,326
	PAWS	99	9.03	7.44	1,394,090	1,223,705
	R+WalkSAT	100	0.74	0.46	661,461	406,315
	R+ANOV$^+$	100	0.27	0.20	196,105	139,914
	R+RSAPS	100	0.18	0.12	55,096	36,995
	R+PAWS	100	0.10	0.07	29,713	20,321
qg5-11	WalkSAT	0	n/a	n/a	10M	10M
	ANOV$^+$	0	n/a	n/a	10M	10M
	RSAPS	94	106.01	70.69	2,853,395	1,902,617
	PAWS	97	94.96	67.47	2,372,620	1,851,463
	R+WalkSAT	1	29.77	29.94	9,943,134	10M
	R+ANOV$^+$	56	18.34	23.93	6,042,269	7,883,243
	R+RSAPS	100	3.03	1.97	143,001	93,143
	R+PAWS	100	1.80	1.38	82,516	62,950
qg6-09	WalkSAT	1	25.00	25.10	9,958,346	10M
	ANOV$^+$	0	n/a	n/a	10M	10M
	RSAPS	100	0.88	0.71	55,598	44,726
	PAWS	100	9.81	8.40	634,952	540,475
	R+WalkSAT	100	0.32	0.14	241,304	106,581
	R+ANOV$^+$	100	1.54	0.91	943,542	554,951
	R+RSAPS	100	0.02	0.02	3,617	3,078
	R+PAWS	100	0.02	0.02	4,022	3,004
qg7-09	WalkSAT	1	27.14	27.38	9,910,996	10M
	ANOV$^+$	34	21.08	25.21	8,363,211	10M
	RSAPS	100	0.54	0.48	30,937	27,312
	PAWS	100	3.39	2.56	212,259	159,752
	R+WalkSAT	100	0.02	0.01	12,856	9,740
	R+ANOV$^+$	100	0.02	0.00	15,573	2,849
	R+RSAPS	100	0.00	0.00	697	584
	R+PAWS	100	0.01	0.01	808	595
qg7-13	WalkSAT	0	n/a	n/a	10M	10M
	ANOV$^+$	0	n/a	n/a	10M	10M
	RSAPS	1	525.17	526.87	9,968,002	10M
	PAWS	2	513.75	518.76	5,148,754	10M
	R+WalkSAT	0	51.16	51.16	10M	10M
	R+ANOV$^+$	26	34.70	42.52	8,160,433	10M
	R+RSAPS	83	172.10	173.99	5,112,670	5,168,583
	R+PAWS	100	58.89	43.93	1,738,133	1,296,636

Table 1: Comparison results on quasigroup existence problems. ANOV$^+$ is the abbreviation of AdaptNovelty$^+$.

refine the quasigroup existence problems significantly. The addition of resolvents allows the truth values of up to half of the variables to be determined, with the result that many redundant clauses can be deleted and other clauses shortened by unit propagation. This happens in all ten cases, but the effects on runtimes are not uniform across problems. Nor are they uniform across solvers, PAWS in particular deriving greater benefit from the preprocessing than either AdaptNovelty$^+$ or RSAPS.

	Before		After		Time
Problem	#Vars	#Clauses	#Vars	#Clauses	(secs)
qg1-07	343	68,083	168	3,846	5.79
qg1-08	512	148,957	281	12,434	29.57
qg2-07	343	68,083	179	4,378	6.88
qg2-08	512	148,957	296	13,743	35.31
qg3-08	512	10,469	273	3,335	0.03
qg4-09	729	15,580	435	6,151	0.08
qg5-11	1,331	64,054	824	27,415	1.29
qg6-09	729	21,844	389	7,337	0.22
qg7-09	729	22,060	334	5,728	0.29
qg7-13	2,197	97,072	1,412	45,362	2.63

Table 2: Effect of resolution computation on problem size: quasigroup existence problems.

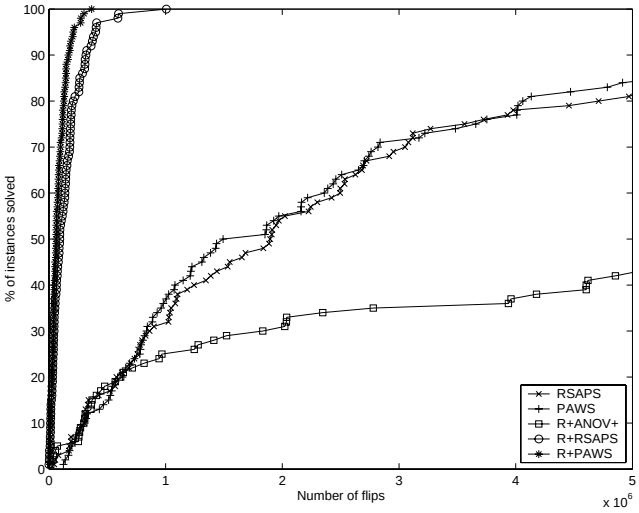

Figure 1: Detail of the *qg5-11* performances.

To further illustrate the significant improvement on the performance of SLS algorithms when the resolution step is performed, we graph the runlength distribution (RLD) for all 100 runs on the qg5-11 instance of each algorithm in Figure 1. The runlengths of WalkSAT, R+WalkSAT and AdaptNovelty$^+$ are omitted as they fail to find any solutions over 100 runs within 5 million flips. As shown in Figure 1, R+PAWS and R+RSAPS solve this instance on 100% of runs in under 1 million flips while without resolution PAWS and RSAPS find solutions within 5 million flips on around 80% of runs. In addition, within 5 million flips R+ANOV$^+$ manages solutions on more than 40% of runs

while AdaptNovelty$^+$ cannot solve the problem at all at this runlength.

Random and Real-World Problems

In order to evaluate further the performance of resolution-enhanced SLS algorithms versus their parents AdaptNovelty$^+$, RSAPS, and PAWS along with GSAT and WalkSAT, we extended the empirical study to include hard-constrained random problems used in 2004 SAT competition and some well-known benchmark problems such as bw_large.c, logistics.c, par16-*, and e*ddr2-* series. Table 3 shows the results for these problems. The results of GSAT are again not shown in this table as it failed to solve any of these problems. WalkSAT is able to find the solution for the logistics.c, e*ddr2-* and random problems, but it is always outperformed by clause weighting SLS algorithms, even without resolution enhancement.

In general, this extended study further confirms the superior performance of resolution-enhanced clause weighting SLS algorithms in comparison with their parents on structured problems. On the par16-* and e*ddr2-* series, the performance of resolution-enhanced SLS improves by an order of magnitude. On the par16-* problems, the performance of SLS solvers is significantly boosted, PAWS in particular going from 0.2% without resolution to a 56.6% success rate with it. On the e*ddr2-* series, the application of resolution results in a huge difference in the performance of AdaptNovelty$^+$ from 5.67% to nearly 100% success and ten times faster in terms of execution time and flips. R+PAWS and R+RSAPS are also two times faster than their parents. The performances of the three resolution-enhanced SLS algorithms are comparable with those of their parents on the logistics.c problem and slightly worse on the bw_large.c problem. This is expected, as the resolution preprocessor increases the problem sizes somewhat without bringing any simplification.

On the uniform random problems, resolution-enhanced SLS solvers gain no benefit from the preprocessing. In fact, resolution increases the size of the formula between 2% and 5%, making these problems harder to solve for SLS. We include results for these cases, however, to emphasize that even where it cannot bring benefits, restricted resolution preprocessing does not make matters significantly *worse*. In all such cases, the performance of resolution-enhanced SLS solvers remains comparable with that of their parent solvers.

On problems where applying resolution results in a smaller size instance than the original one, and especially where variables are removed, R+PAWS appears to benefit rather more from the preprocessing than the other solvers in our study. However, care must be taken over interpreting the results because whereas RSAPS and AdaptNovelty$^+$ are designed to run autonomously, PAWS does not appear to be provided with useful default settings, so we had to hand-tune its parameters. It is important to note that for this reason we do not intend to make absolute performance comparisons between the underlying solvers.

		Success	Time (secs)		Flips	
Problem	Algorithm	%	Mean	Median	Mean	Median
unif04 (8 problems)	WalkSAT	65.38	3.51	3.47	5,464,756	5,404,806
	ANOV$^+$	51.38	5.25	5.96	6,610,487	3,332,986
	RSAPS	42.63	13.17	15.10	7,534,550	4,567,975
	PAWS	94.63	3.07	2.45	1,938,207	1,833,543
	R+WalkSAT	58.43	15.32	15.43	4,371,635	4,378,580
	R+ANOV$^+$	51.25	5.24	5.73	6,499,467	4,191,363
	R+RSAPS	42	14.10	15.92	7,588,603	7,164,512
	R+PAWS	93.75	3.74	2.85	2,452,301	1,877,690
bw_large.c	WalkSAT	0	n/a	n/a	10M	10M
	ANOV$^+$	71	8.91	9.17	5,579,415	5,746,796
	RSAPS	84	44.09	38.78	4,371,893	3,845,685
	PAWS	100	7.75	5.48	1,269,132	889,807
	R+WalkSAT	0	n/a	n/a	10M	10M
	R+ANOV$^+$	52	12.12	16.45	7,051,380	9,573,926
	R+RSAPS	89	50.61	43.18	4,436,449	3,785,521
	R+PAWS	100	9.41	6.62	1,093,097	791,997
logistics.c	WalkSAT	100	0.51	0.42	641,424	523,357
	ANOV$^+$	100	0.15	0.15	157,501	152,908
	RSAPS	100	0.02	0.02	9,453	8,024
	PAWS	100	0.02	0.02	10,195	8,494
	R+WalkSAT	100	1.67	1.03	1,865,203	1,149,086
	R+ANOV$^+$	100	0.08	0.07	66,874	59,761
	R+RSAPS	100	0.03	0.02	10,669	7,968
	R+PAWS	100	0.04	0.03	11,729	10,085
par16-* (5 problems)	WalkSAT	0	n/a	n/a	10M	10M
	ANOV$^+$	0	n/a	n/a	10M	10M
	RSAPS	0	n/a	n/a	10M	10M
	PAWS	0.2	10.21	10.21	9,623,108	10M
	R+WalkSAT	0	n/a	n/a	10M	10M
	R+ANOV$^+$	4	6.07	6.01	9,795,548	10M
	R+RSAPS	15.2	9.71	10.58	9,176,854	10M
	R+PAWS	56.6	6.15	7.57	6,740,259	8,312,353
e*ddr2-* (6 problems)	WalkSAT	76	9.48	9.79	6,214,625	6,436,834
	ANOV$^+$	5.67	14.12	14.25	9,907,019	10M
	RSAPS	100	1.81	1.54	64,275	53,256
	PAWS	100	0.95	0.91	53,609	48,889
	R+WalkSAT	31.69	22.71	22.54	5,412,279	5,175,889
	R+ANOV$^+$	99.17	1.68	1.35	870,454	702,870
	R+RSAPS	100	1.04	0.89	44,707	37,555
	R+PAWS	100	0.54	0.51	30,548	27,073

Table 3: Comparison results on random and realistic benchmark problems. ANOV$^+$ is the abbreviation of AdaptNovelty$^+$.

	Before		After		Time
Problem	#Vars	#Clauses	#Vars	#Clauses	(secs)
unif04-52	500	2,125	500	2,220	0.03
unif04-62	550	2,337	550	2,430	0.03
unif04-65	550	2,337	550	2,427	0.04
unif04-80	600	2,550	600	2,628	0.03
unif04-83	650	2,762	650	2,863	0.02
unif04-86	650	2,762	650	2,849	0.03
unif04-91	700	2,975	700	3,064	0.03
unif04-99	700	2,975	700	3,054	0.03
bw_large.c	3,016	50,457	3,016	54,563	0.27
logistics.c	1,141	10,719	1,141	11,628	0.07
par16-1	1,015	3,310	607	1,815	0.03
par16-2	1,015	3,374	632	1,929	0.03
par16-3	1,015	3,344	620	1,875	0.03
par16-4	1,015	3,324	619	1,853	0.03
par16-5	1,015	3,358	627	1,903	0.02
e0ddr2*1	19,500	103,887	15,549	81,864	1.19
e0ddr2*4	19,500	104,527	16,011	85,014	1.14
enddr2*1	20,700	111,567	16,019	84,182	1.39
enddr2*8	21,000	113,729	15,905	83,451	1.46
ewddr2*1	21,800	118,607	16,265	8,832	1.69
ewddr2*8	22,500	123,329	15,815	81,240	1.85

Table 4: Effect of resolution computation on problem size: random and realistic problems.

Conclusion

We have integrated restricted resolution as preprocessing in the SLS solvers AdaptNovelty$^+$, PAWS and RSAPS. The resolution-enhanced SLS solvers show their best performance on structured problems such as quasigroup existence and real-world problems. In cases where the preprocessor does not simplify clauses, there is little effect, but where some unit clauses result, order of magnitude improvements in the local search are observed. We consider it particularly interesting that this technique allows SLS methods to approach the performance of systematic ones on several problems (quasigroup existence, parity learning) which are not usually thought suitable for local search.

Current and future research includes experimenting with the effect of resolution preprocessing on other clause weighting and novelty techniques, as well as combining neighbour resolution (Cha & Iwama 1996) with resolution preprocessing.

In the wider context, this work is part of a trend towards hybrid solvers for SAT and CSP. It is clear that hybrid methods can enhance both performance and robustness, especially in real-world domains. We present our experimental results as evidence that resolution and clause weighting SLS indeed work effectively together on a wide range of such problems.

Acknowledgments

The authors would like to thank the anonymous reviewers for their valuable comments and suggestions on a previous version of this paper. We would also like to thankfully acknowledge the financial support from National ICT Australia (NICTA) and the ARC grant number A00000118 at Griffith University, Australia. National ICT Australia is funded through the Australian Government's *Backing Australia's Ability* initiative, in part through the Australian Research Council.

References

Billionnet, A., and Sutter, A. 1992. An efficient algorithm for 3-satisfiability problem. *Operations Research Letters* 12:29–36.

Cha, B., and Iwama, K. 1996. Adding new clauses for faster local search. In *Proceedings of the 13th AAAI*, 332–337.

Davis, M., and Putnam, H. 1960. A computing procedure for quantification theory. *Journal of the ACM* 7:201–215.

Fang, H., and Ruml, W. 2004. Complete local search for propositional satisfiability. In *Proceedings of the 19th AAAI*, 161–166.

Frank, J. 1997. Learning short-term clause weights for GSAT. In *Proceedings of the 15th IJCAI*, 384–389.

Fujita, M.; Slaney, J.; and Bennett, F. 1993. Automatic generation of some results in finite algebra. In *Proceedings of the 13th IJCAI*, 52–57.

Gent, I. P., and Walsh, T. 1995. Unsatisfied variables in local search. In *Proceedings of AISB'95, Hybrid Problems, Hybrid Solutions*, 73–85.

Hoos, H. H., and Stulze, T. 2005. *Stochastic Local Search*. Cambridge, Massachusetts: Morgan Kaufmann.

Hoos, H. H. 1999. On the run-time behaviour of stochastic local search algorithms for SAT. In *Proceedings of the 16th AAAI*, 661–666.

Hoos, H. H. 2002. An adaptive noise mechanism for WalkSAT. In *Proceedings of the 18th AAAI*, 655–660.

Hutter, F.; Tompkins, D. A. D.; and Hoos, H. H. 2002. Scaling and probabilistic smoothing: Efficient dynamic local search for SAT. In *Proceedings of the 8th CP*, 233–248.

Li, C. M., and Anbulagan. 1997. Look-ahead versus lookback for satisfiability problems. In *Proceedings of the 3rd CP*, 341–355.

McAllester, D. A.; Selman, B.; and Kautz, H. A. 1997. Evidence for invariants in local search. In *Proceedings of the 14th AAAI*, 321–326.

Morris, P. 1993. The breakout method for escaping from local minima. In *Proceedings of the 11th AAAI*, 40–45.

Quine, W. V. 1955. A way to simplify truth functions. *American Mathematical Monthly* 62:627–631.

Robinson, J. A. 1965. A machine-oriented logic based on the resolution principle. *Journal of the ACM* 12:23–41.

Schuurmans, D., and Southey, F. 2000. Local search characteristics of incomplete SAT procedures. In *Proceedings of the 17th AAAI*, 297–302.

Schuurmans, D.; Southey, F.; and Holte, R. C. 2001. The exponentiated subgradient algorithm for heuristic boolean programming. In *Proceedings of the 17th IJCAI*, 334–341.

Selman, B., and Kautz, H. A. 1993. Domain-independent extensions to GSAT: Solving large structured satisfiability problems. In *Proceedings of the 13th IJCAI*, 290–295.

Selman, B.; Kautz, H. A.; and Cohen, B. 1994. Noise strategies for improving local search. In *Proceedings of the 12th AAAI*, 337–343.

Selman, B.; Levesque, H.; and Mitchell, D. 1992. A new method for solving hard satisfiability problems. In *Proceedings of the 10th AAAI*, 440–446.

Thornton, J.; Pham, D. N.; Bain, S.; and Ferreira Jr., V. 2004. Additive versus multiplicative clause weighting for SAT. In *Proceedings of the 19th AAAI*, 191–196.

Tompkins, D. A., and Hoos, H. H. 2004. Warped landscapes and random acts of SAT solving. In *Proceedings of the 8th International Symposium on Artificial Intelligence and Mathematics*.

Wu, Z., and Wah, B. W. 2000. An efficient global-search strategy in discrete lagrangian methods for solving hard satisfiability problems. In *Proceedings of the 17th AAAI*, 310–315.