# Refutation by Randomised General Resolution

**Steven Prestwich**
Cork Constraint Computation Centre
University College, Cork, Ireland
s.prestwich@cs.ucc.ie

**Inês Lynce**
IST/INESC-ID
Technical University of Lisbon, Portugal
ines@sat.inesc-id.pt

### Abstract

Local search is widely applied to satisfiable SAT problems, and on some problem classes outperforms backtrack search. An intriguing challenge posed by Selman, Kautz and McAllester in 1997 is to use it instead to prove unsatisfiability. We design a greedy randomised resolution algorithm called RANGER that will eventually refute any unsatisfiable instance while using only bounded memory. RANGER can refute some problems more quickly than systematic resolution or backtracking with clause learning. We believe that non-systematic but greedy inference is an interesting research direction for powerful proof systems such as general resolution.

## Introduction

Most satisfiability (SAT) solvers can be classed either as *complete* or *incomplete*, and the complete algorithms may be based on resolution or backtracking. Resolution provides a complete proof system by refutation (Robinson 1965). The first resolution algorithm was the Davis-Putnam (DP) procedure (Davis & Putnam 1960) which was then modified to the Davis-Putnam-Logemann-Loveland (DPLL) backtracking algorithm (Davis, Logemann, & Loveland 1962). Because of its high space complexity, resolution alone is often seen as impractical for real-world problems, but there are problems on which general resolution proofs are exponentially smaller than DPLL proofs (Ben-Sasson, Impagliazzo, & Wigderson 2004). DPLL has recently been greatly enhanced by the use of clause learning, and by improved implementation techniques.

Incomplete SAT algorithms are usually based on *local search* following early work by (Gu 1992; Selman, Levesque, & Mitchell 1992). SAT local search has subsequently been an active area of research, and modern algorithms are considerably improved. SAT local search usually explores a space of total variable assignments, and tries to minimise the number of violated clauses by flipping variable truth assignments, guided by various heuristics. The best algorithms currently use some form of dynamic adjustment to the objective function, usually by minimising the sum of weights of violated clauses: clause weights are dynamically adjusted so that frequently-violated clauses have greater weights, enabling faster escape from local minima.

Local and backtrack search have complementary strengths and weaknesses. Local search has superior scalability on many large problems, but it cannot (in the form described above) prove unsatisfiability. Backtrack search is complete, and its use of unit propagation, clause learning, dedicated data structures and other methods enables it to outperform local search on some highly-structured problems. This complementarity has inspired research on hybrid approaches such as the use of unit propagation in local search, and more flexible backtracking strategies.

An interesting question is: can local search be applied to *unsatisfiable* problems? Such a method might be able to refute (prove unsatisfiable) SAT problems that defy complete algorithms. This was number five of the ten SAT challenges posed by (Selman, Kautz, & McAllester 1997): *design a practical stochastic local search procedure for proving unsatisfiability*. While substantial progress has been made on several challenges, this one remains wide open (Kautz & Selman 2003). We describe a new form of local search that explores a space of multisets of resolvents using general resolution, and aims to derive the empty clause non-systematically but greedily. We show that it will eventually refute any unsatisfiable instance while using only bounded memory, and demonstrate the existence of problems that it can refute more quickly than current DPLL and systematic resolution algorithms. This paper summarizes the key contributions of our earlier paper (Prestwich & Lynce 2006).

## Local search on multisets of resolvents

We begin by reviewing a theoretical result from (Esteban & Torán 2001). Given an unsatisfiable SAT formula $\phi$ with $n$ variables and $m$ clauses, a general resolution refutation can be represented by a series of formulae $\phi_1, \ldots, \phi_s$ where $\phi_1$ consists of some or all of the clauses in $\phi$, and $\phi_s$ contains the empty clause. Each $\phi_i$ is obtained from $\phi_{i-1}$ by (optionally) deleting some clauses in $\phi_{i-1}$, adding the resolvent of two remaining clauses in $\phi_{i-1}$, and (optionally) adding clauses from $\phi$. The *space* of a proof is defined as the minimum $k$ such that each $\phi_i$ contains no more than $k$ clauses.

Intuitively each $\phi_i$ represents the set of *active* clauses at step $i$ of the proof. Inactive clauses are not required for fu-

```
1  RANGER(φ, p_i, p_t, p_g, w, k):
2    i ← 1 and φ_1 ← {any k clauses from φ}
3    while φ_i does not contain the empty clause
4      with probability p_i
5        replace a random φ_i clause by a
         random φ clause
6      otherwise
7        resolve random φ_i clauses c, c' giving r
8        if r is non-tautologous and |r| ≤ w
9          with probability p_g
10           if |r| ≤ max(|c|, |c'|) replace the
             longer of c, c' by r
11         otherwise
12           replace a random φ_i clause by r
13     with probability p_t
14       apply any satisfiability-preserving
         transformation to φ, φ_i
15     i ← i + 1 and φ_{i+1} ← {the new formula}
16   return UNSATISFIABLE
```

Figure 1: The RANGER architecture

ture resolution, and after they have been used as needed they can be deleted. It is proved in (Esteban & Torán 2001) that the space $k$ need be no larger than $n + 1$: possibly fewer clauses than in $\phi$ itself.

The *width* of a proof is the length (in literals) of the largest clause in the proof. Any non-tautologous clause must have length no greater than $n$, so this is a trivial upper bound for the width used for our algorithm. However, in practice we may succeed even if we restrict resolvent length to a smaller value, which may be useful for saving memory on large problems.

Thus we can in principle find a large refutation using a modest amount of working memory. But finding such a proof may not be easy. We shall use the above notions as the basis for a novel local search algorithm that performs a randomised but biased search in the space of formulae $\phi_i$. Each $\phi_i$ will be of the same constant size, and derived from $\phi_{i-1}$ by the application of resolution or the replacement of a clause by one taken from $\phi$. We call our algorithm RANGER (RANdomised GEneral Resolution).

## The algorithm

The RANGER architecture is shown in Figure 1. It has six parameters: the formula $\phi$, three probabilities $p_i, p_t, p_g$, the width $w$ and the size $k$ of the formula $\phi_i$.

RANGER begins with any sub-multiset $\phi_1 \subseteq \phi$ (we interpret $\phi, \phi_i$ as multisets of clauses). It then performs iterations $i$, each either replacing a $\phi_i$ clause by a $\phi$ clause (with probability $p_i$) or resolving two $\phi_i$ clauses and placing the result $r$ into $\phi_i$. In the latter case, if $r$ is tautologous or contains more than $w$ literals then it is discarded and $\phi_{i+1} = \phi_i$. Otherwise a $\phi_i$ clause must be removed to make room for $r$: either (with probability $p_g$) the removed clause is the longer of the two parents of $r$ (breaking ties randomly), or it is randomly chosen. In the former case, if $r$ is longer than the parent then $r$ is discarded and $\phi_{i+1} = \phi_i$. At the end of the iteration, any satisfiability-preserving transformation may (with probability $p_t$) be applied to $\phi$, $\phi_{i+1}$ or both. If the empty clause has been derived then the algorithm terminates with the message "unsatisfiable". Otherwise the algorithm might not terminate, but a time-out condition (omitted here for brevity) may be added.

Local search algorithms usually use *greedy* local moves that reduce the value of an objective function, and *plateau traversal* moves that leave it unchanged. However, they must also allow non-greedy moves in order to escape from local minima. This is often controlled by a parameter known as *noise* (or *temperature* in simulated annealing). But what is our objective function? Our goal is to derive the empty clause, and a necessary condition for this to occur is that $\phi_i$ contains at least some small clauses. We will call a local move *greedy* if it does not increase the number of literals in $\phi_i$. This is guaranteed on line 10, so increasing $p_g$ increases the greediness of the search, reducing the proliferation of large resolvents. There may be better forms of greediness but this form is straightforward, and in experiments it significantly improved performance.

RANGER has a useful convergence property: for any unsatisfiable SAT problem with n variables and m clauses, RANGER finds a refutation if $p_i > 0$, $p_i, p_t, p_g < 1$, $w = n$ and $k \geq n + 1$. For a proof see (Prestwich & Lynce 2006). The space complexity of RANGER is $O(n + m + kw)$. To guarantee convergence we require $w = n$ and $k \geq n + 1$ so the complexity becomes at least $O(m + n^2)$. In practice we may require $k$ to be several times larger, but a smaller value of $w$ is often sufficient.

Lines 13–14 provide an opportunity to apply helpful satisfiability-preserving transformations to $\phi$ or $\phi_i$ or both (if we do not aim for a pure resolution refutation). We apply the subsumption and pure literal rules in several ways. Using $\phi_i$ clauses to transform $\phi$, a feature we shall call *feedback*, preserves useful improvements for the rest of the search. (We believe that for these particular transformations we can set $p_t = 1$ without losing completeness, but we defer the proof until a later paper.) Note that if $\phi$ is reduced then this will soon be reflected in the $\phi_i$ via line 5 of the algorithm.

## An example

We now illustrate RANGER's behaviour with an example. Consider the following CNF formula $\phi$ with five clauses $\omega_1, ..., \omega_5$:

$$\begin{aligned} \omega_1 &= (a \vee b \vee c) \\ \omega_2 &= (\bar{a} \vee b \vee c) \\ \omega_3 &= (a \vee \bar{c}) \\ \omega_4 &= (\bar{a} \vee \bar{c}) \\ \omega_5 &= (\bar{b} \vee c) \end{aligned}$$

The formula has 3 variables. Given that we must have $k \geq n + 1$ we set $k = 4$. Suppose that $\phi_1$ contains the following four clauses:

$$\begin{aligned} \omega_1 &= (a \vee b \vee c) \\ \omega_2 &= (\bar{a} \vee b \vee c) \\ \omega_4 &= (\bar{a} \vee \bar{c}) \\ \omega_5 &= (\bar{b} \vee c) \end{aligned}$$

Now suppose that the condition in line (4) in Figure 1 is not satisfied, so that we proceed to line (7) and generate a resolvent $\omega_6 = (b \vee c)$ from $\omega_1$ and $\omega_2$. Resolvent $\omega_6$ is non-tautologous and $\omega_6$ has less than three literals, so the conditions in line (8) of the algorithm are satisfied. Suppose that the condition in line (9) is not satisfied, so that the algorithm proceeds to line (12) and replaces clause $\omega_4$ by $\omega_6$. So far $\phi_1$ has been transformed into the clauses:

$$\begin{aligned}\omega_1 &= (a \vee b \vee c)\\ \omega_2 &= (\bar{a} \vee b \vee c)\\ \omega_5 &= (\bar{b} \vee c)\\ \omega_6 &= (b \vee c)\end{aligned}$$

Assuming that the condition in line (13) is satisfied, then both clauses $\omega_1$ and $\omega_2$ are deleted from $\phi$ and $\phi_1$ because they are subsumed by $\omega_6$. Then we randomly choose $\omega_3$ and $\omega_4$ from $\phi$ to make $\phi_2$ up to four clauses:

$$\begin{aligned}\omega_3 &= (a \vee \bar{c})\\ \omega_4 &= (\bar{a} \vee \bar{c})\\ \omega_5 &= (\bar{b} \vee c)\\ \omega_6 &= (b \vee c)\end{aligned}$$

In subsequent iterations we may infer $\omega_7 = (c)$ from $\omega_6$ and $\omega_5$, and $\omega_8 = (\bar{c})$ from $\omega_3$ and $\omega_4$. From $\omega_7$ and $\omega_8$ the empty clause is derived and the algorithm terminates.

Of course this refutation depends on randomly making the correct moves. For example, before deriving the empty clause we may have replaced $\omega_3$ and $\omega_4$ by other $\phi$ clauses. But even after a bad move RANGER has a chance of making the correct moves later on, after copying the same clauses from $\phi$ to $\phi_i$.

## Experiments

To explore RANGER's behaviour we performed several experiments, which are briefly summarised here for space reasons. Firstly, we constructed an artificial benchmark that RANGER refutes in about 1 second, easily beating several SAT backtrackers (ZChaff, SATZ, Siege, POSIT and Minisat), some of which are state-of-the-art algorithms with clause learning yet take tens of minutes or more. The benchmark was a satisfiable 600-variable random 3-SAT problem augmented by a small unsatisfiable subproblem we call HIDER:

$$\begin{array}{llll}(a_1 \vee b_1 \vee c_1) & (\bar{a}_1 \vee d \vee e) & (\bar{b}_1 \vee d \vee e) & (\bar{c}_1 \vee d \vee e)\\ (a_2 \vee b_2 \vee c_2) & (\bar{a}_2 \vee \bar{d} \vee e) & (\bar{b}_2 \vee \bar{d} \vee e) & (\bar{c}_2 \vee \bar{d} \vee e)\\ (a_3 \vee b_3 \vee c_3) & (\bar{a}_3 \vee d \vee \bar{e}) & (\bar{b}_3 \vee d \vee \bar{e}) & (\bar{c}_3 \vee d \vee \bar{e})\\ (a_4 \vee b_4 \vee c_4) & (\bar{a}_4 \vee \bar{d} \vee \bar{e}) & (\bar{b}_4 \vee \bar{d} \vee \bar{e}) & (\bar{c}_4 \vee \bar{d} \vee \bar{e})\end{array}$$

From these clauses we can derive $(d \vee e)$, $(\bar{d} \vee e)$, $(d \vee \bar{e})$ or $(\bar{d} \vee \bar{e})$ in 3 resolution steps each. For example resolving $(a_1 \vee b_1 \vee c_1)$ with $(\bar{a}_1 \vee d \vee e)$ gives $(b_1 \vee c_1 \vee d \vee e)$; resolving this with $(\bar{b}_1 \vee d \vee e)$ gives $(c_1 \vee d \vee e)$; and resolving this with $(\bar{c}_1 \vee d \vee e)$ gives $(d \vee e)$. From these 4 resolvents we can obtain $(d)$ and $(\bar{d})$ (or $(e)$ and $(\bar{e})$ in 2 resolution steps. Finally, we can obtain the empty clause in 1 more resolution step, so this problem has a refutation of size 15. We designed HIDER to be hidden in random 3-SAT problems as an unsatisfiable sub-problem with a short refutation. All its clauses are ternary and no variable occurs more than 12 times. In a random 3-SAT problem from the phase transition each variable occurs an expected 12.78 times so these clauses blend well with the problem, and a backtracker has no obvious reason to focus on the new variables. Moreover, a resolution refutation of HIDER requires the generation of quaternary clauses, which (for example) SATZ's resolution-based preprocessor algorithm *compactor* does not generate.

Secondly, we found that, though a low-space proof may exist, performance can be greatly improved by allowing more space. Thirdly, feedback was found to be an important feature: it reduces the automotive product configuration problems of (Sinz, Kaiser, & Küchlin 2003) to about $1/20$ of their original size via feedback, and they are refuted in seconds or minutes depending on the instance; but without feedback RANGER does not refute them in a reasonable time. Fourthly, it refutes the hard unsatisfiable random 3-SAT instance aim-100-2_0-no-1 in a few seconds, whereas Rish & Dechter's DR resolution algorithm (Rish & Dechter 2000) takes tens of minutes, as does the TABLEAU backtracker. But their resolution/backtrack hybrid algorithms take under 1 second, as does SATZ's *compactor* algorithm alone. Fifthly, on unsatisfiable random 3-SAT problems RANGER performs very poorly: an interesting asymmetry, given that local search performs well on *satisfiable* random problems. This may be because such refutations are almost certainly exponentially long (Chvatal & Szemeredi 1988).

In future work we hope to find a useful class of SAT problems on which RANGER is the algorithm of choice. As with our artificial benchmark, these problems should be unsatisfiable, fairly large, not susceptible to backtrack search, and require a resolution proof of non-trivial width so that systematic resolution does not easily refute them.

## Related work and conclusion

As of 2003 no work had been done on using local search to prove unsatisfiability (Kautz & Selman 2003), but since our paper (Prestwich & Lynce 2006) another has appeared describing a related algorithm called GUNSAT (Audemard & Simon 2007). The architectures of GUNSAT and RANGER are similar but there are interesting differences in detail. For example, whereas RANGER aims for a high rate of rather unintelligent local moves (on our artificial benchmark it performs roughly 130,000 iterations per second on a 733 MHz Pentium II with Linux), GUNSAT takes longer to make more intelligent moves based on a more complex objective function. GUNSAT also uses extended resolution while RANGER uses general resolution. No convergence proof for GUNSAT is provided, nor is it shown to beat current backtrackers on any instance. The two algorithms have not been tested on a common set of benchmarks so we cannot compare their performance. Neither has yet found its "killer application" but it should be interesting to observe future developments in this emerging class of algorithm.

An alternative approach to refutation by local search was explored in (Prestwich & Lynce 2006): applying standard SAT local search to a space of (possibly incorrect) proof graphs each represented by a clause list. In order to exploit

current SAT local search technology, this was done via a new reformulation of the original SAT problem. A reformulation was presented containing $O(r^2 + mr + nr)$ variables and $O(nr^2 + rs)$ literals, where the original SAT problem contains $n$ variables, $m$ clauses and $s$ literals, and we aim for a refutation of length at most $r$. Thus for short proofs the meta-encoding may be economical, but RANGER's space complexity has the important advantage of being independent of the length of the proof. It was shown that a problem that is hard for some backtrackers (though not others) is solved easily by a standard local search algorithm applied to the reformulation. However, some quite trivial problems turn out to be hard to refute in this way. Moreover, this approach is only practical for small refutation lengths.

Other research based on hybrid approaches is more loosely related to our work. Local search can be made complete by using learning techniques (Fang & Ruml 2004). But the aim of this approach is to improve performance on satisfiable problems, not to speed up proof of unsatisfiability. Hybrid approaches have also been tried for the more general class of QBF formulas. For example, WalkQSAT (Gent *et al.* 2003) has two main components: the QBF engine performs a backjumping search based on conflict and solution directed backjumping, whereas the SAT engine is a slightly adapted version of the WalkSAT used to find satisfying assignments quickly.

Proof systems such as general resolution should in principle be faster than more simple systems such as treelike resolution. However, in practice such systems are rarely used, partly because of their excessive memory consumption, but also because no good strategy is known for applying the inference rules in order to find a small proof (Alekhnovich & Razborov 2001) though they are used within other algorithms. In fact there may be no such strategy, and we suggest that a non-systematic approach is an interesting research direction for such proof systems.

**Acknowledgements** Thanks to Eli Ben-Sasson for advice on resolution proofs, and to the anonymous referees for helpful comments. This material is based in part upon works supported by the Science Foundation Ireland under Grant No. 00/PI.1/C075, and by Fundação para a Ciência e Tecnologia under research projects POSI/SRI/41926/01 and POSC/EIA/61852/2004.

## References

Alekhnovich, M., and Razborov, A. 2001. Resolution is not automizable unless w[p] is tractable. In *Forty-Second IEEE Symposium on FOCS*, 210–219.

Audemard, G., and Simon, L. 2007. Gunsat: A greedy local seach algorithm for unsatisfiability. In *Twentieth International Joint Conference on Artificial Intelligence*. Poster.

Ben-Sasson, E.; Impagliazzo, R.; and Wigderson, A. 2004. Near-optimal separation of treelike and general resolution. *Combinatorica* 24(4):585–603.

Chvatal, V., and Szemeredi, E. 1988. Many hard examples for resolution. *Journal of the Association for Computing Machinery* 35:759–768.

Davis, M., and Putnam, H. 1960. A computing procedure for quantification theory. *Journal of the Association of Computing Machinery* 7(3).

Davis, M.; Logemann, G.; and Loveland, D. 1962. A machine program for theorem proving. *Communications of the Association of Computing Machinery* 5:394–397.

Esteban, J. L., and Torán, J. 2001. Space bounds for resolution. *Information and Computation* 171(1):84–97.

Fang, H., and Ruml, W. 2004. Complete local search for propositional satisfiability. In *Nineteenth National Conference on Artificial Intelligence*, 161–166. AAAI Press.

Gent, I. P.; Hoos, H. H.; Rowley, A. G. D.; and Smyth, K. 2003. Using stochastic local search to solve quantified boolean formulae. In *Ninth International Conference on Principles and Practice of Constraint Programming*, volume 2833 of *Lecture Notes in Computer Science*, 348–362. Springer.

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

Kautz, H. A., and Selman, B. 2003. Ten challenges *redux*: Recent progress in propositional reasoning and search. In *Ninth International Conference on Principles and Practice of Constraint Programming*, volume 2833 of *Lecture Notes in Computer Science*, 1–18. Springer.

Prestwich, S. D., and Lynce, I. 2006. Local search for unsatisfiability. In *Ninth International Conference on Theory and Applications of Satisfiability Testing*, volume 4121 of *Lecture Notes in Computer Science*, 283–296. Springer.

Rish, I., and Dechter, R. 2000. Resolution versus search: Two strategies for sat. *Journal of Automated Reasoning* 24(1-2):225–275.

Robinson, J. A. 1965. A machine-oriented logic based on the resolution principle. *Journal of the Association for Computing Machinery* 12(1):23–41.

Selman, B.; Kautz, H. A.; and McAllester, D. A. 1997. Ten challenges in propositional reasoning and search. In *Fifteenth International Joint Conference on Artificial Intelligence*, 50–54. Morgan Kaufmann.

Selman, B.; Levesque, H.; and Mitchell, D. 1992. A new method for solving hard satisfiability problems. In *Tenth National Conference on Artificial Intelligence*, 440–446. AAAI Press.

Sinz, C.; Kaiser, A.; and Küchlin, W. 2003. Formal methods for the validation of automotive product configuration data. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 17(2):75–97. Special issue on configuration.