Attempts to produce minimal Resolution refutations

Olivier Bailleux

Attempts to produce minimal Resolution refutations

Olivier Bailleux
LIB, Université de Bourgogne
olivier.bailleux@u-bourgogne.fr

july, 2018

Abstract

We address the challenge of searching minimal refutations proofs of inconsistent CNF formulae using the Resolution rule. We propose two algorithms which can only afford formulae of at most 5 variables with a desktop computer. A faster but incomplete algorithm is used to produce "hard" 5 variables 3CNF formulae though a stochastic greedy search. It allowed us to find formulae that can be refuted by producing clauses of at most 3 literals, but whose all *minimal* refutations contain at least one clause of 4 literals.

1 Introduction

Almost all modern complete SAT solvers indirectly use the Resolution rule [Rob65]. It is applicable to any pair of clauses with exactly one variable occurring with opposed signs, namely $x \vee \alpha$ and $\neg x \vee \beta$ such that for any other literal y of α, $\neg y$ is not in β. This very simple inference rule is complete for refutation, meaning that applied to an inconsistent set of clauses, it always produces the empty clause after a finite number of steps. But from the same set of clauses, namely, the input CNF formula, the number of steps before the empty clause is produced can drastically vary according to the clauses which are used as premises at each application of the rule.

In this report, the following question is addressed in a practical perspective : given an inconsistent CNF formula Σ, what is the smallest number of deductions needed to produce the empty clause ? Any refutation of Σ with such a number of deductions will be called a *minimal* refutation. Except in special cases, for example when all the clauses have a length of 2 [BOM07], producing a minimal refutation is a very hard problem [ABMP01].

2 Producing minimal refutations

We tried two ways for producing minimal refutations. The first one consists in producing all the possible Resolution inferences in a single graph. The second one consists in producing all the possible refutations one by one.

2.1 Parallel search

This first approach consists in producing the graph of all possible derivations, one layer after another. The layer 0 contains the initial clauses. Each layer $i > 0$ contains the clauses occurrences derived from one premise of the preceding layer and another premise belonging to any layer $j < i$. This concept of *clause occurrences* is important because some clauses can be derived many times in different ways. In the following, any clause occurrence will be called a *deduction*.

The key point of the efficiency of such an approach is to avoid producing (or to remove) as many unnecessary deductions as possible. To this end, we compute two values related to each deduction q : the *weight*, defined as the number of inferences used to produce q, and the *cover*, defined as the number of initial clauses used to produce q. The weights are given in the gray disks in figure 1. Let min denote the weight of the refutation of the lowest weight produced so far. Clearly, any deduction q of size k and weight at least $min - k$ can be rejected because any path from q to an empty clause would have length at least k. Figure 2 shows the clauses that can be removed thanks to this criterion.

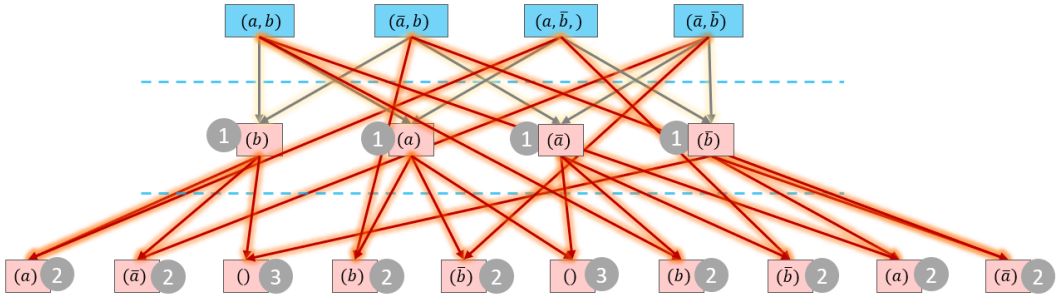

Figure 1: Producing all the deductions, one level after another.

Another rejection criterion can be used if the input formula is critically inconsistent, i.e., each initial clause is used in any refutation. In such a case, any deduction q with weight $w(q)$ and cover $c(q)$ can be rejected if $w(q) \geq min - m + c(q)$, where m is the number of initial clauses.

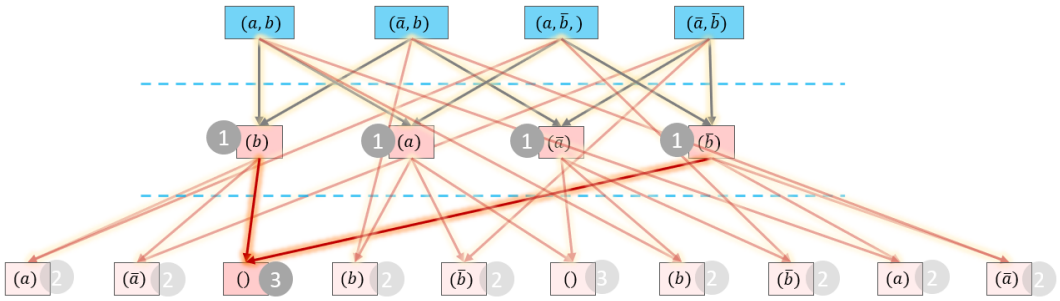

Figure 2: Removing unnecessary deductions.

Subsumption and clause duplication can be used to reject some deductions in a restricted way, namely, if a deduction q is an ancestor of a deduction p and q subsumes p or is equals to p, then p can be removed. The reason is given in [JI97] (chapter 7, corollary 7.16).

Property 2.1. *[JI97] Let S be a set of clauses, and S' be a subset of S. Assume that every clause C in S is in S' or is subsumed by a clause in S'. If the empty clause is derivable from S in k resolution steps, then it is derivable from S' in at most k resolution steps.*

Unfortunately, we found no other obvious way to use subsumption or clause duplication as a rejection criterion, because the deductions belonging to a minimal refutation are not necessarily themselves minimal. Consider the following example, where $\Sigma_i(x,y)$ is the formula $(s_i \vee t_i \vee x \vee y) \wedge (\neg s_i \vee t_i \vee x \vee y) \wedge (s_i \vee \neg t_i \vee x \vee y) \wedge (\neg s_i \vee \neg t_i \vee x \vee y)$, which produces the clause $x \vee y$ with 3 resolution steps, and $\sigma(z)$ is a formula with variables not in Σ_i, from which the clause (z) can be minimally produced with 14 deductions. Figure 3 shows a minimal refutation producing the deduction $()_1$ with 24 resolution steps, including the deduction $(a)_1$ of weight 15. The clause (a) can also be produced with only 14 inferences thanks to the deduction $(a)_2$, but using this deduction instead of $(a)_1$ would not allow to produce a minimal refutation. The reason is that some deductions used to produce $(a)_1$ are also used to produce $(\neg a)$.

2.2 Incomplete parallel search

As we will see in the experimental part, beyond 4 variable, the time required to produce a minimal refutation becomes prohibitive. In order to produce shorts refutation, but without any guarantee of minimality, we propose to modify the previous algorithm in the following way: if any deduction q with weight $w(q)$ is subsumed by or is equal to an existing deduction p with $w(p) \leq w(q)$, then q is rejected. The example of figure 3 shows that this new algorithm can fail to produce a minimal refutation. We failed to find such examples with critically inconsistent formulae, so we leave open the question of the minimality of the refutations produced in such a case.

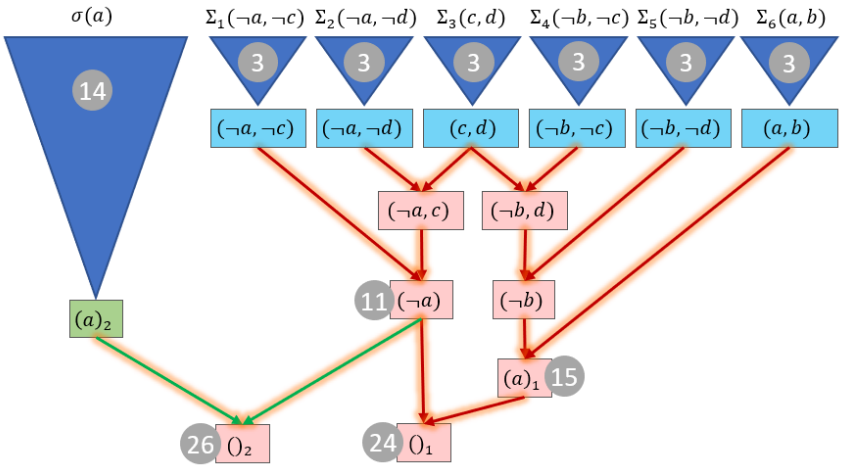

Figure 3: Some duplicated clauses cannot be rejected.

2.3 Sequential search

As with the parallel search algorithm, any refutation graph is organized in layers, were at least one premises of each deduction of the layer i is in the the layer $i-1$.

The algorithm we used enumerates all the possible refutation graphs with a given number of resolution steps. Let the *depth* of an inference graph denote its number of layers excepts the one containing the initial clauses. For any possible depth of the derivation graph, we try to populate the first layer with any admissible sequence of deductions at this level. Theses sequences of deductions will be called *clusters*. For each cluster of the first layer, we try to populate the second layer, and so on.

Some simple properties allow us to restrict the possible numbers of deductions in any layer i, given the sizes of the preceding ones and the $depth$ of the derivation graph. Clearly, the layer $depth$ must contain exactly one deduction corresponding to the empty clause. The the layer $depth-1$ cannot contain more that 2 deductions, and any layer $depth-k$ cannot contain more than 2^k deductions. In addition, each layer must contain at least one deduction. It follows that if r denotes the number of deductions that remains to be done after the level $i-1$, and k denotes the number of layers after the layer i, then the size of the layer i is at least $\max(1, r-2^k+1)$ and at most $\min(r-k, 2^k)$.

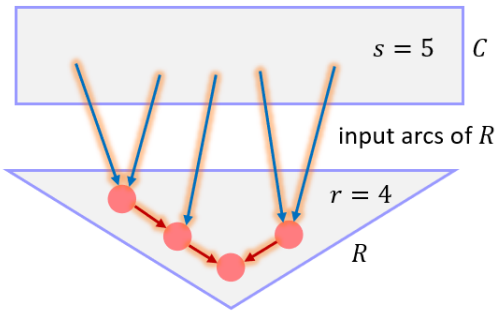

Figure 4: Dealing with the current number of sinks and the number of remaining nodes.

The search space can also be reduced by tacking into account the number s of sinks at each stage of the construction of the current derivation graph C, as well as the number r of deductions that remains to be done. This current graph C must be connected to the graph R consisting of the r deductions to be produced. Then R must have one sink, s input arcs, and r nodes, including the sink. On the example of figure 4, we can see that if $s=5$, the graph can be completed only if $r \geq 4$. More generally, each new node in R remove one input arc and add two ones, then we must have $r \geq s-1$.

Of course, in addition to the preceding criteria, any new occurrence of an already produced (or initial) clause can be rejected, as well as any deduction of a clause that is subsumed by an already produced one in the current, partially built

inference graph.

The algorithm used for enumerating all the inference graphs is summarized figure 5.

- `cur` represents the current level, i.e., the level of the layer in construction.
- `layer[cur]` represents the layer of depth `cur`.
- `init layer[cur].n` and `init layer[cur].max` set the current (minimal) and the maximal size of the current layer, according to the previously seen criteria.
- `init layer[cur].cluster` puts the first admissible cluster into the current cluster, and `layer[cur].cluster++` produces the next admissible cluster for the current layer, according to the enumeration order of the clusters.

All these procedures can either succeed (blue arrows) or fail (red arrows).

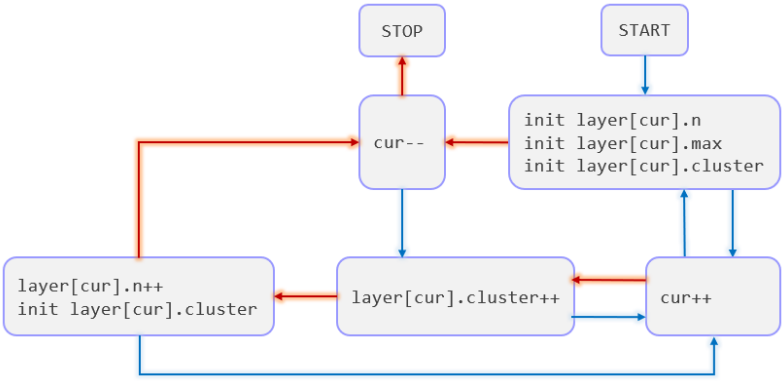

Figure 5: Enumerating all the derivation graphs.

3 Experimental investigations

3.1 Comparing sequential and parallel search

Our first experiment compares the three algorithms applied to a minimalist pebbling formula [BSIW04] with 4 variables, namely

$$\Sigma_1 = \{(\bar{1}\bar{3}), (\bar{1}\bar{4}), (34), (\bar{2}\bar{3}), (\bar{2}\bar{4}), (12)\}$$

.

Algorithm	Resolution attempts	Time in s	Refutation size
Parallel search	140k	< 0.1	6
Parallel search without subsumption	249k	< 0.1	6
Incomplete parallel search	0.5k	< 0.1	6
Sequential search	2.7k	< 0.1	6
Sequential search without subsumption	2.9k	< 0.1	6

The second experiment is about the following 3-CNF formula presented in [EK08] as "The shortest interesting formula in 3CNF", namely (with renaming of literals)

$$\Sigma_2 = \{(123), (\bar{1}\bar{2}\bar{3}), (\bar{1}3\bar{4}), (1\bar{3}4), (1\bar{2}\bar{4}), (\bar{1}24), (\bar{2}34), (2\bar{3}\bar{4})\}$$

Algorithm	Resolution attempts	Time in s	Refutation size
Parallel search	> 20B	> 4800	?
Parallel search without subsumption	> 20B	> 4800	?
Incomplete parallel search	2k	< 0.1	11
Sequential search	0.6M	25	11
Sequential search without subsumption	0.7M	23	11

The third experiment is about the following small 3CNF formula with 5 variables. This formula is intractable using the parallel search algorithm with our desktop computer, and requires Billions of Resolution tests using the sequential algorithms. The incomplete parallel search algorithm quickly produces an optimal refutation.

$$\Sigma_3 = \{(123), (\bar{1}\bar{2}3), (1\bar{2}\bar{3}), (\bar{1}2\bar{3}), (\bar{3}\bar{4}\bar{5}), (\bar{3}45), (3\bar{4}5), (34\bar{5}), (\bar{1}\bar{4}), (1\bar{4}), (\bar{2}\bar{5}), (2\bar{5})\}$$

Algorithm	Resolution attempts	Time in s	Refutation size
Incomplete parallel search	13k	< 0.1	13
Sequential search	13B	830	13
Sequential search without subsumption	20B	750	13

3.2 Producing difficult formulae

As proved in [BSW01], the largest Refutation proofs also are the widest ones, in the sense that if the initial clauses have a fixed width (i.e. number of literals), the size of any minimal refutation is exponentially related to the width of the widest clauses produced in such a refutation, namely,

$$S(F) = \exp \Omega \left(\frac{w(F \vdash 0) - w(F)}{n} \right)$$

were $S(F)$ is the size of any minimal refutation of a formula F, $w(F)$ is the width of the widest clause of of F, $w(F \vdash 0)$ is the width of the widest clause of any minimal refutation of F, and n is the number of variables in F. Then, asymptotically, the kCNF formulae whose minimal refutations produce "wide" clauses are "hard" for Resolution. On the other hand, because there are only a polynomial amount of clauses of width k on n variables, "large" refutations of kCNF formulae *do* contain "wide" clauses.

But aside from these asymptotic facts, what are the smallest 3CNF formulae that cannot be refuted without producing at least a clause with more that 3 literals ? In [Urq87], such a formula with 6 variable is given. One of our aims was to try to found such a 3CNF formulae with 5 variables. To this aim, we implemented a randomized greedy algorithm with objective function provided by the incomplete parallel search of minimal refutations (IPS for short). First, a random 3CNF formula with 5 variable is produced. Then this formula is iteratively perturbed by replacing one randomly picked clause by another randomly generated one. The score of any consistent formula is the opposite of the number of deductions produced by IPS. The score of an inconsistent formula is the size of the refutation produced by IPS.

Let us denote w-refutation any refutation producing clauses of width at most w. A formula will be said w-refutable if and only if a w-refutation exists for this formula. We spent hundreds of billions Resolution attempts to try to produce 5 variables formulae with "large" minimal refutation proof, and checking them for a 3-refutation. We did not find any inconsistent but not 3-refutable formula. The most "hard" formula we found consists in 18 clauses and can be refuted with 17 deductions. The size of the minimal refutation proof for this formula was confirmed using the (complete) sequential search algorithm at nearly no cost because the formula is critically inconsistent and contains 18 clauses, then it cannot be refuted with less than 17 refutations according to the criterion illustrated figure 4.

$$\begin{aligned}\Sigma_4 = \{&(\bar{3}\bar{4}\bar{5}), (3\bar{4}5), (2\bar{3}\bar{5}), (\bar{1}2\bar{3}), (145), (134), (1\bar{2}3), (2\bar{4}\bar{5}), (1\bar{2}5),\\ &(\bar{1}\bar{3}\bar{4}), (\bar{1}\bar{4}\bar{5}), (\bar{1}\bar{3}\bar{5}), (\bar{2}34), (\bar{1}35), (2\bar{3}\bar{4}), (1\bar{2}4), (\bar{1}\bar{2}\bar{5}), (\bar{2}45)\}\end{aligned} \tag{1}$$

Interestingly, the formula Σ_4 is 3-refutable, but doe not admit any 3-refutation of size 17, as verified by the sequential search algorithm (at the cost of 15 billions resolution attempts). The smallest 3-refutation we found with the incomplete

parallel search have size 21. This means that, sometimes, producing wider clauses can allow to produce shorter refutations, which is somewhat counter-intuitive.

Below is another example of formula with minimal 3-refutations longer than minimal 4-refutations (11 and 12 deductions, respectively).

$$\Sigma_5 = \{(\bar{1}\bar{2}3), (345), (\bar{2}\bar{3}\bar{5}), (1\bar{4}\bar{5}), (24\bar{5}), (2\bar{4}\bar{5}), (\bar{1}\bar{3}4), (13\bar{5}), (1\bar{3}5), (2\bar{4}5), (\bar{3}\bar{4}\bar{5})\} \tag{2}$$

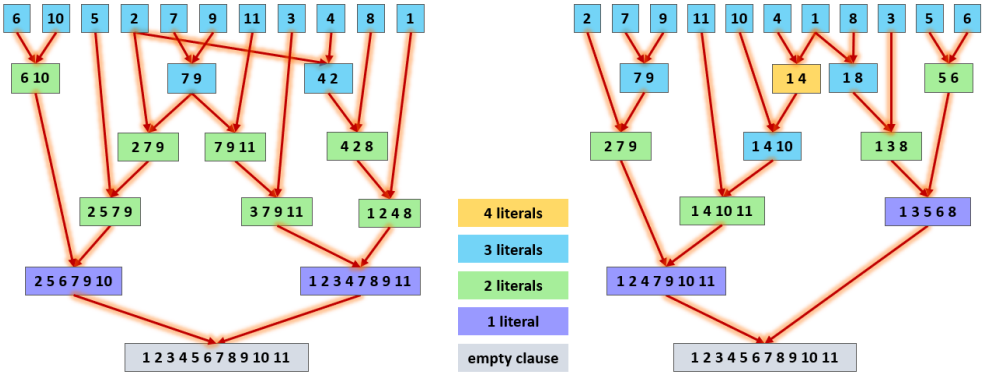

Figure 6: A minimal 3-refutation (left, size 12) and a minimal 4-refutation (right, size 11) of Σ_5.

An example of 3-refutation and an example of 4-refutation of Σ_5 are given figure 6. The numbers of the initial clauses correspond to their rank in the formula. The numbers listed in each node are the ones of the initial clauses used for the corresponding deduction.

4 Perspectives

The complexity of finding short resolution proofs has already been addressed [Iwa97] [FSW04]. Nevertheless, to the best of our known, if algorithms for finding minimal refutations have been implemented and experimentally evaluated, no such facts have been yet published. So, even if our results are somewhat disappointing, they are a first open contribution to this area. We hope this contribution will be followed by many more, and we wonder how far can be pushed the practical tractability of this challenging problem.

References

[ABMP01] Michael Alekhnovich, Sam Buss, Shlomo Moran, and Toniann Pitassi. Minimum propositional proof length is np-hard to linearly approximate. *J. Symbolic Logic*, 66(1):171–191, 03 2001. 1

[BOM07] Joshua Buresh-Oppenheim and David Mitchell. Minimum 2cnf resolution refutations in polynomial time. In João Marques-Silva and Karem A. Sakallah, editors, *Theory and Applications of Satisfiability Testing – SAT 2007*, pages 300–313, Berlin, Heidelberg, 2007. Springer Berlin Heidelberg. 1

[BSIW04] Eli Ben-Sasson, Russell Impagliazzo†, and Avi Wigderson‡. Near optimal separation of tree-like and general resolution. *Combinatorica*, 24(4):585–603, Sep 2004. 4

[BSW01] Eli Ben-Sasson and Avi Wigderson. Short proofs are narrow: Resolution made simple. *J. ACM*, 48(2):149–169, March 2001. 5

[EK08] Donald E. Knuth. *The Art of Computer Programming, Volume 4, Fascicle 0: Introduction to Combinatorial Algorithms and Boolean Functions (Art of Computer Programming)*. Addison-Wesley, 01 2008. 4

[FSW04] Michael R. Fellows, Stefan Szeider, and Graham Wrightson. On finding short resolution refutations and small unsatisfiable subsets. In Rod Downey, Michael Fellows, and Frank Dehne, editors, *Parameterized and Exact Computation*, pages 223–234, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg. 6

[Iwa97] Kazuo Iwama. Complexity of finding short resolution proofs. In Igor Prívara and Peter Ružička, editors, *Mathematical Foundations of Computer Science 1997*, pages 309–318, Berlin, Heidelberg, 1997. Springer Berlin Heidelberg. 6

[JI97] Goubault-Larrecq Jean and Mackie I. *Proof Theory and Automated Deduction*. Springer Netherlands, 1997. 2

[Rob65] J. A. Robinson. A machine-oriented logic based on the resolution principle. *J. ACM*, 12(1):23–41, January 1965. 1

[Urq87] Alasdair Urquhart. Hard examples for resolution. *J. ACM*, 34(1):209–219, January 1987. 5