A k-flip local search algorithm for SAT and MAX SAT

Chris Patuzzo

August 27, 2020

Abstract

Local search can be applied to SAT by determining whether it is possible to increase the number of satisfied clauses for a given truth assignment by flipping at most k variables. However, for a problem instance with v variables, the search space is of order v^k. A naive approach that enumerates every combination is impractical for all but the smallest of problems. This paper outlines a hybrid approach that plays to the strength of modern SAT solvers to search this space more efficiently. We describe an encoding of SAT to a related problem – k-Flip MAX SAT – and show how through repeated application, it can be used to solve SAT and MAX SAT problems. Finaly, we test the algorithm on a benchmark set with different values of k to see how it performs.

1 Introduction

- sat problems have hundreds or thousands of variables, doesn't scale
  - explain k-flip max sat
  - explain ipasir and justify it for this problem

2 The encoding

At a high-level, the encoding works by introducing a set of variables A that represents a hypothetical SAT solver's current truth assignment of variables within some formula F. A corresponding set of variables A' is introduced that is allowed to differ by at most k truth assignments from A. We use a counter circuit and a less-than comparator to enforce this constraint.

For each clause in F, we introduce a variable whose intended meaning is that the related clause has not been satisfied by A'. Collectively, we call this set U. We enforce that the number of true literals in U is less than the SAT solver's current number of unsatisfied clauses for F. We once again use a counter circuit and less-than comparator to enforce this constraint.

2.1 Flipped variables

2.2 Unsatisfied clauses

2.3 Parallel counter

2.4 Less-than comparator

3 Repeated application

4 Empirical results

	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
430	0	0	0	1	4	4	8	10	17	17	17	22	21	28	24	32	42	31	41	37
429	0	0	1	10	16	15	20	22	17	25	26	21	27	26	30	22	14	21	18	22
428	0	3	11	13	14	14	16	17	14	12	13	11	10	6	6	6	4	7	1	1
427	0	7	11	18	14	23	8	9	10	5	4	6	2	0	0	0	0	1	0	0
426	1	6	15	9	9	3	7	2	1	1	0	0	0	0	0	0	0	0	0	0
425	3	9	10	7	2	1	0	0	1	0	0	0	0	0	0	0	0	0	0	0
424	4	13	5	2	1	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0
423	5	10	5	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
422	7	7	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
421	3	4	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
420	9	4	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
419	6	1	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
418	8	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
417	11	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
416	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
415	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
414	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
413	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1: caption