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

Chris Patuzzo

August 28, 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. Finally, 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 its 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	2	7	28	75	105	163	227	303	370	417	499	571	610	645	678	797	798	838	879
429	1	16	68	171	250	345	443	478	519	555	574	609	580	588	616	595	503	520	494	465
428	0	46	156	305	376	454	429	416	389	341	316	231	217	180	121	117	91	76	65	53
427	7	122	270	369	328	305	253	197	154	110	79	50	27	20	16	9	9	6	3	3
426	14	191	307	240	213	152	86	69	35	25	18	11	4	2	2	1	0	0	0	0
425	37	251	256	158	107	32	24	14	3	4	1	0	1	0	0	0	0	0	0	0
424	78	240	182	85	43	8	6	4	1	0	0	0	0	0	0	0	0	0	0	0
423	121	209	92	35	10	2	1	0	1	0	0	0	0	0	0	0	0	0	0	0
422	173	145	34	13	2	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
421	181	104	21	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
420	165	46	7	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
419	179	22	4	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
418	161	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
417	110	5	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
416	80	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
415	38	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
414	33	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
413	17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
412	6	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
411	3	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
410	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
409	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Table 1: caption