Compressing UNSAT Search Trees with Caching: an update

Anthony BLOMME, Daniel LE BERRE, Anne PARRAIN, Olivier ROUSSEL

CRIL, Université d'Artois & CNRS

July, 2023

Context

Machine Learning is everywhere

The rationale of the outcome of those “black boxes” is hard to explain making XAI a very trendy topic

Many people no longer trust computer programs

Even if it is a deterministic constraint programming solver

What is an explanation for a SAT solver?

- several target users (solver expert, modeling expert or user)
- several levels of explanation (clauses, high level constraints, ...)
- foundation of the explanation (logical reasoning, statistical reasoning, ...)
- ...

What is an explanation for a SAT solver?

- several target users (solver expert, modeling expert or user)
- several levels of explanation (clauses, high level constraints, ...)
- foundation of the explanation (logical reasoning, statistical reasoning, ...)
- ...

This work: focus on the information provided by the solver, i.e. it's search tree

Case of satisfiable formulas

Why is it a solution ?

- All the clauses are satisfied
- Compact representation (prime implicant)

Why this particular solution ?

- Logical justification: logical implication (which reduces to UNSAT proof), backbone
- Statistical justification: probability($x =$ true)
- Solver's decisions have no logical explanation!

```
$ java -Dcolor -jar org.sat4j.core.jar file.cnf
```

```
s SATISFIABLE
v 1 2 -3 4 5 -6 -7 -8 -9 -10 11 -12 -13 14 15 16 -17 -18 19 -20 -21 -22 23 24 25 26 -
27 28 -29 -30 -31 32 -33 34 35 36 37 38 39 -40 -41 42 -43 -44 -45 -46 -47 -48 49 50 -
51 -52 53 -54 55 -56 57 -58 59 60 -61 62 -63 -64 65 -66 67 68 -69 -70 -71 -72 73 -74
-75 -76 -77 -78 79 -80 -81 -82 83 -84 -85 86 -87 88 89 -90 -91 -92 -93 94 95 -96 97 -
98 99 -100 0
c UNASSIGNED: 0 DECIDED: 0 PROPAGATED_ORIGINAL: 69 PROPAGATED_LEARNED: 29 DECIDED_PRO
PAGATED: 1 DECIDED_PROPAGATED_LEARNED: 0 DECIDED_CYCLE: 1
c Total wall clock time (in seconds) : 0.01
```

Why is there no solution?

Prove the impossibility of a solution

- UNSAT certificate or MUS (too large in general)
- Reduce a posteriori the size of the search tree:
  - Delete useless decisions and propagations
  - Reorder the nodes
  - Recognize recurring patterns

Case of unsatisfiable formulas

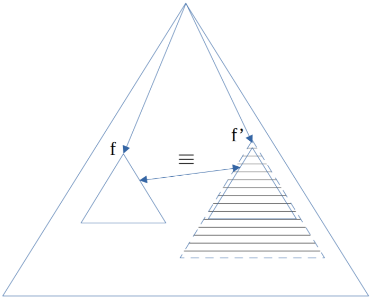
Why is there no solution?

Prove the impossibility of a solution

- UNSAT certificate or MUS (too large in general)
- Reduce a posteriori the size of the search tree:
  - Delete useless decisions and propagations
  - Reorder the nodes
  - Recognize recurring patterns

Recognize recurring patterns

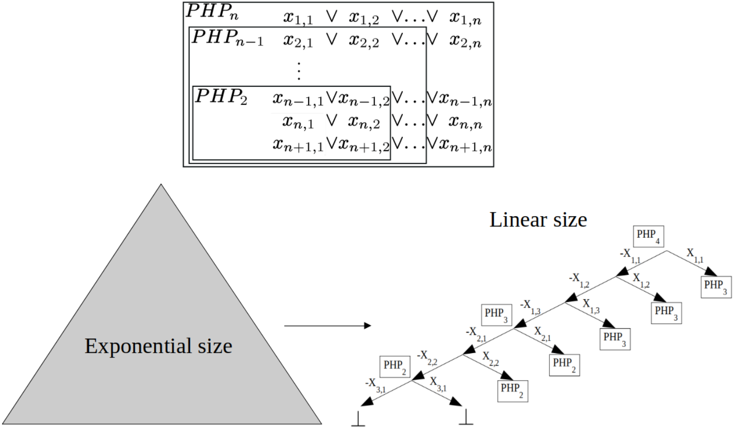
Recognize equivalent subformulas (renamings, inclusions).



Do not explain the unsatisfiability of a formula twice.

Link similar proofs together (single explanation).

Motivating example: the pigeon hole problem



Use of a cache system

Idea: *Build a cache with proven unsatisfiable subformulas and try to recognize them later*

Inspired by model counters and compilers, here specialized to the UNSAT case.

- Light minimization: use only the clauses involved in the conflict (sources)
- Use a normalized representation to register subformulas
- If a subformula is equal to an entry of the cache, we can prune the branch

Use of a cache system

Idea: *Build a cache with proven unsatisfiable subformulas and try to recognize them later*

Inspired by model counters and compilers, here specialized to the UNSAT case.

- Light minimization: use only the clauses involved in the conflict (sources)
- Use a normalized representation to register subformulas
- If a subformula is equal to an entry of the cache, we can prune the branch

Does not work on PHP example: sub-PHP instances are built on different variables and clauses

If the subformula contain the cache entry it is also UNSAT

Generalizing the cache system

Generalize equality:

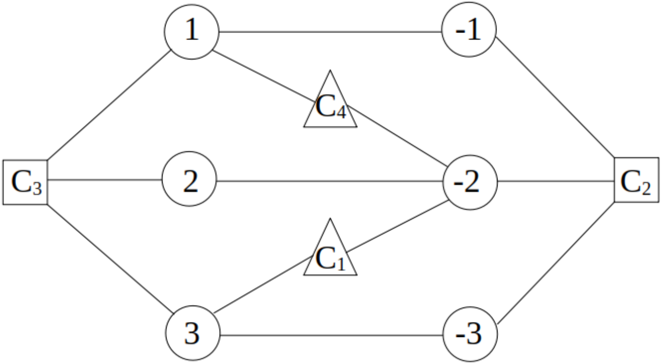
- detect if an entry of the cache is a subset of the current subformula
- allow variable renaming

Subgraph isomorphism allows to test if, after renaming the variables, an entry of the cache is included in the current subformula. If it is the case, we can prune the branch.

Glasgow Subgraph Solver is used to detect subgraph isomorphisms ($\Rightarrow$ classic encoding of subformulas).

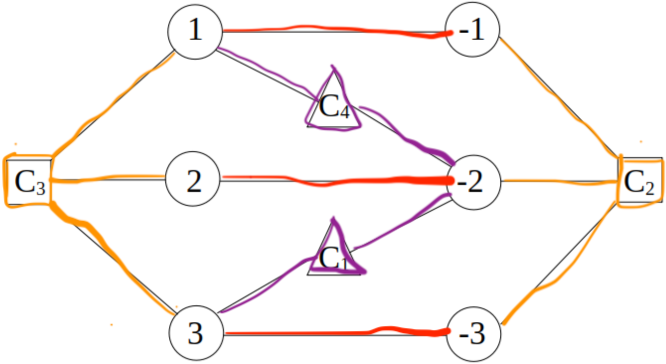
Example of colored graph representation of a CNF

Colored graph representation of the formula
$(\neg x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2)$

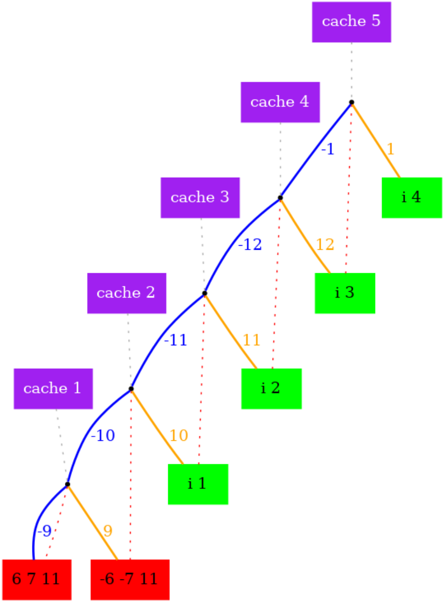


Example of colored graph representation of a CNF

Colored graph representation of the formula
$(\neg x_2 \vee x_3) \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \wedge (x_1 \vee x_2 \vee x_3) \wedge (x_1 \vee \neg x_2)$



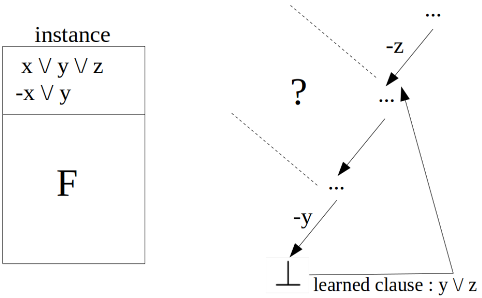
Does it work? the marg2x2.cnf instance DPLL



When to cache?

- We have to be sure that the entry corresponds to an UNSAT formula.
- With a DPLL approach, it can be done for any node in the search tree
  - on the leaves, corresponding to conflicts
  - on internal nodes, once both children are found UNSAT
- With a CDCL approach, things are more complicated . . .

Problem of backjumps in CDCL solvers



Problem 1: When backjumping, the search is not complete and we do not know if the unexplored subformulas are unsatisfiable

The caching is performed at the leaves, when encountering a conflict.

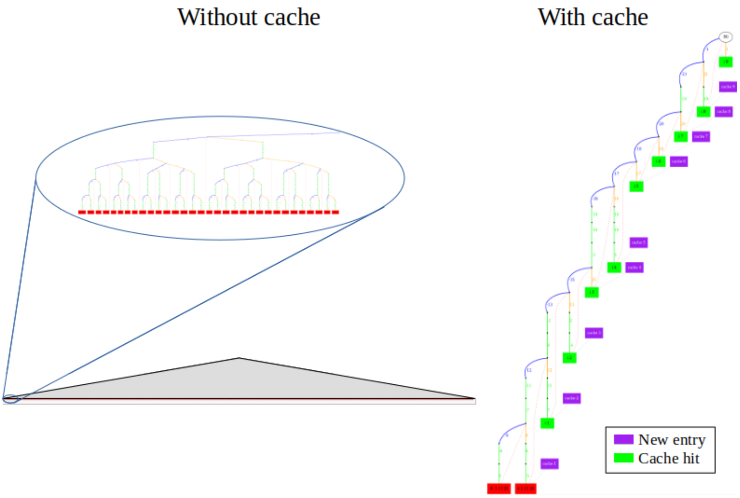
UNIVERSITÉ D'ARTOIS

Problem 2 (technical): When we hit an entry in the cache, we need a conflict clause to backtrack. How to build it?

When recognizing an entry, we create a conflict composed of the falsified literals in the matching clauses. The conflict analysis can be performed with this clause.

If those literals are not from the current decision level, backtrack to the lowest decision level before performing conflict analysis.

marg2x3.cnf instance with CDCL



Expecting $x_1 \vee x_2$ and got $x_a \vee x_b \vee x_c \vee x_d$: matches?

Expecting $x_1 \vee x_2$ and got $x_a \vee x_b \vee x_c \vee x_d$: matches?

Expecting $x_1 \vee x_2 \vee x_3$ and got $x_a \vee x_b \vee x_c \vee x_d$: matches?

Generalized isomorphisms

Expecting $x_1 \vee x_2$ and got $x_a \vee x_b \vee x_c \vee x_d$: matches?

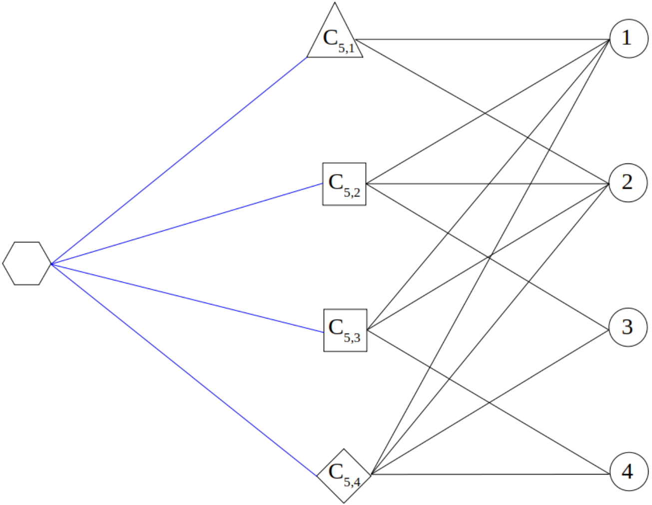
Expecting $x_1 \vee x_2 \vee x_3$ and got $x_a \vee x_b \vee x_c \vee x_d$: matches?

Detect entries of the cache even if some literals are falsified.

- Do not delete satisfied clauses and satisfied literals are considered unassigned
- Create variants of clauses with falsified literals. Create all the possibilities from the complete original clause to the clause with all falsified literals removed
- Selector nodes are used to avoid using several variants of a same clause

Encoding of exponential size but the number of added clauses is, in general, reasonable compared to the original number of clauses.

Generalized isomorphisms colored graph



Implemented on top of Minisat.

The cache lookup is performed before taking a decision.

Cache lookup is translated into a subgraph isomorphism problem and then given to Glasgow Subgraph Solver.

Time limit of 2 seconds (regular isomorphisms) or 4 seconds (generalized isomorphisms) for each call to Glasgow Subgraph Solver.

Environment for the experiments

- We consider UNSAT instances from the SAT'02 and SAT'03 competitions
  - "Easy" for DPLL and CDCL
  - Small enough for expensive algorithms
- A total of 579 UNSAT instances
  - SAT'02: 381 instances
  - SAT'03: 198 instances
- Time limit:
  - Regular isomorphisms: 15 minutes
  - Generalized isomorphisms: 30 minutes

Some results (pruning after search, regular)

Instance	Size	Conflicts (no cache)	Conflicts (cache)	Compression Ratio
PHP_7	448	$5.6\ 10^3$	853	$1.5\ 10^{-1}$
PHP_{12}	2,028	-	-	-
marg2x6.sat03-1444	528	$3.0\ 10^4$	20	$6.6\ 10^{-4}$
marg3x3add8.sat03-1449	1,056	$1.8\ 10^5$	32	$1.8\ 10^{-4}$
Urquhart-s3-b9	1,240	$1.9\ 10^4$	21	$1.1\ 10^{-3}$
Urquhart-s3-b3	2,152	$1.6\ 10^6$	29	$1.8\ 10^{-5}$
x1_16	364	$2.2\ 10^3$	20	$9.1\ 10^{-3}$
x1_24	556	$2.0\ 10^5$	78	$3.9\ 10^{-4}$
3col20_5_6	646	27	27	1
3col40_5_4	1,286	92	64	$7.0\ 10^{-1}$
homer06	1,800	-	-	-

- 117/579 instances solved
- Some traces were too large to be postprocessed

Some results (pruning during search, regular)

	CDCL (integrated cache, regular isomorphisms)					
Instance	Conflicts	Cache size	Subgraph Isomorphisms	Calls	Time (Search)	Time (GSS)
PHP_7	47	41	29 (8)	259	0.025	1.550
PHP_{12}	116	107	96 (20)	589	0.189	18.412
PHP_{16}	187	178	167 (32)	1020	0.731	166.088
marg2x6.sat03-1444	20	17	18 (17)	44	0.007	0.341
marg3x3add8.sat03-1449	32	25	20 (20)	55	0.022	0.524
marg6x6.sat03-1456	86	84	84 (84)	276	0.181	15.190
Urquhart-s3-b9	21	18	17 (17)	38	0.009	0.329
Urquhart-s3-b3	29	26	27 (25)	59	0.023	0.861
Urquhart-s5-b5	95	91	91 (90)	259	0.367	55.682
x1_16	18	15	14 (14)	60	0.010	0.693
x1_24	40	35	32 (18)	202	0.022	1.665
x1_96	2177	471	106 (76)	8513	1.759	423.444
3col20_5_6	27	5	0 (0)	15	0.005	0.099
3col40_5_4	110	22	54 (3)	786	0.107	5.535
homer06	102	95	92 (20)	462	0.485	47.701

- 185/579 instances solved

Some results (pruning during search, generalized)

	CDCL (integrated cache, generalized isomorphisms)					
Instance	Conflicts	Cache size	Subgraph Isomorphisms	Calls	Time (Search)	Time (GSS)
PHP_5	23	17	16 (11)	95	0.017	1.183
PHP_7	42	35	35 (21)	391	0.083	220.329
marg2x6.sat03-1444	20	17	18 (17)	50	0.057	3.331
marg3x3add8.sat03-1449	31	24	21 (21)	74	0.067	14.081
marg4x4.sat03-1454	41	39	39 (35)	186	0.108	130.496
Urquhart-s3-b9	21	18	17 (17)	50	0.039	2.959
Urquhart-s3-b3	29	26	27 (26)	67	0.269	17.958
x1_16	18	15	14 (14)	94	0.018	21.538
x1_24	29	24	22 (21)	117	0.044	60.011
x2_32	65	54	53 (40)	568	0.195	908.380
3col20_5_6	23	3	2 (2)	16	0.008	17.322
3col40_5_4	57	20	27 (4)	809	0.199	1474.762

- 89/579 instances solved

Conclusion

- Our goal is to reduce drastically some UNSAT search trees so that they can be shown to the user
- We propose a syntactic approach based on the detection of subgraph isomorphisms
- Interesting results obtained on some highly structured families of instances
- Future research directions:
  - Better encoding for managing assigned literals in cache entries
  - Delete entries that do not seem useful
  - Incremental use of Glasgow Subgraph Solver
  - Try other heuristics

Compressing UNSAT Search Trees with Caching: an update

Anthony BLOMME, Daniel LE BERRE, Anne PARRAIN, Olivier ROUSSEL

CRIL, Université d'Artois & CNRS

July, 2023

UNIVERSITÉ D'ARTOIS