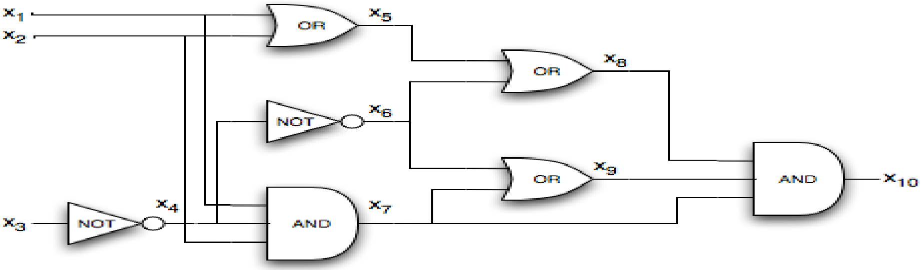
Karlsruhe Institute of Technology

Practical SAT Solving

Lecture 3

Carsten Sinz, Tomáš Balyo | May 17, 2018

INSTITUTE FOR THEORETICAL COMPUTER SCIENCE



SAT Tools – Resolution

The Resolution Rule

$$\frac{(l \vee x_1 \vee x_2 \vee \dots \vee x_n) \wedge (\bar{l} \vee y_1 \vee y_2 \vee \dots \vee y_m)}{(x_1 \vee x_2 \vee \dots \vee x_n \vee y_1 \vee y_2 \vee \dots \vee y_m)}$$

The upper two clauses are called *Input Clauses* the bottom clause is called the *Resolvent*

SAT Tools – Resolution

The Resolution Rule

$$\frac{(l \vee x_1 \vee x_2 \vee \cdots \vee x_n) \wedge (\bar{l} \vee y_1 \vee y_2 \vee \cdots \vee y_m)}{(x_1 \vee x_2 \vee \cdots \vee x_n \vee y_1 \vee y_2 \vee \cdots \vee y_m)}$$

The upper two clauses are called *Input Clauses* the bottom clause is called the *Resolvent*

Examples

- $(x_1 \vee x_3 \vee \overline{x_7}) \wedge (\overline{x_1} \vee x_2) \vdash (x_3 \vee \overline{x_7} \vee x_2)$

SAT Tools – Resolution

The Resolution Rule

$$\frac{(l \vee x_1 \vee x_2 \vee \cdots \vee x_n) \wedge (\bar{l} \vee y_1 \vee y_2 \vee \cdots \vee y_m)}{(x_1 \vee x_2 \vee \cdots \vee x_n \vee y_1 \vee y_2 \vee \cdots \vee y_m)}$$

The upper two clauses are called *Input Clauses* the bottom clause is called the *Resolvent*

Examples

- $(x_1 \vee x_3 \vee \overline{x_7}) \wedge (\overline{x_1} \vee x_2) \vdash (x_3 \vee \overline{x_7} \vee x_2)$
- $(x_4 \vee x_5) \wedge (\overline{x_5}) \vdash (x_4)$

SAT Tools – Resolution

The Resolution Rule

$$\frac{(l \vee x_1 \vee x_2 \vee \cdots \vee x_n) \wedge (\bar{l} \vee y_1 \vee y_2 \vee \cdots \vee y_m)}{(x_1 \vee x_2 \vee \cdots \vee x_n \vee y_1 \vee y_2 \vee \cdots \vee y_m)}$$

The upper two clauses are called *Input Clauses* the bottom clause is called the *Resolvent*

Examples

- $(x_1 \vee x_3 \vee \overline{x_7}) \wedge (\overline{x_1} \vee x_2) \vdash (x_3 \vee \overline{x_7} \vee x_2)$
- $(x_4 \vee x_5) \wedge (\overline{x_5}) \vdash (x_4)$
- $(x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2}) \vdash$

SAT Tools – Resolution

The Resolution Rule

$$\frac{(l \vee x_1 \vee x_2 \vee \cdots \vee x_n) \wedge (\bar{l} \vee y_1 \vee y_2 \vee \cdots \vee y_m)}{(x_1 \vee x_2 \vee \cdots \vee x_n \vee y_1 \vee y_2 \vee \cdots \vee y_m)}$$

The upper two clauses are called *Input Clauses* the bottom clause is called the *Resolvent*

Examples

- $(x_1 \vee x_3 \vee \overline{x_7}) \wedge (\overline{x_1} \vee x_2) \vdash (x_3 \vee \overline{x_7} \vee x_2)$
- $(x_4 \vee x_5) \wedge (\overline{x_5}) \vdash (x_4)$
- $(x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2}) \vdash$
- $(x_1) \wedge (\overline{x_1}) \vdash$

SAT Tools – Resolution

The Resolution Rule

$$\frac{(l \vee x_1 \vee x_2 \vee \cdots \vee x_n) \wedge (\bar{l} \vee y_1 \vee y_2 \vee \cdots \vee y_m)}{(x_1 \vee x_2 \vee \cdots \vee x_n \vee y_1 \vee y_2 \vee \cdots \vee y_m)}$$

Special Cases

- Tautological Resolvent $(x_1 \vee x_2) \wedge (\overline{x_1} \vee \overline{x_2}) \vdash (x_1 \vee \overline{x_1})$
  - Usually forbidden, does no harm, will be useful later
- Empty Clause $(x_1) \wedge (\overline{x_1}) \vdash \perp$
  - The empty clause a.k.a conflict clause a.k.a "$\perp$" is unsatisfiable

Notation

- $\mathrm{R}((x_1 \vee x_2), (\overline{x_1} \vee x_3)) = (x_2 \vee x_3)$

SAT Tools – Resolution

Theorem: Resolution maintains satisfiability

Let F be a CNF formula and C_1 and C_2 two of it's clauses with a pair complementary literals. Then F is satisfiable if and only if $F \wedge \mathrm{R}(C_1, C_2)$ is satisfiable.

SAT Tools – Resolution

Karlsruhe Institute of Technology

Theorem: Resolution maintains satisfiability

Let F be a CNF formula and C_1 and C_2 two of it's clauses with a pair complementary literals. Then F is satisfiable if and only if $F \wedge \mathrm{R}(C_1, C_2)$ is satisfiable.

Proof:

- If F is not satisfiable then $F \wedge C$ for any C is also not satisfiable.
- If F is satisfiable and ϕ is a satisfying assignment of F then we show that ϕ also satisfies $\mathrm{R}(C_1, C_2)$.
  - If $C_1 = (l \vee P_1)$ and $C_2 = (\bar{l} \vee P_2)$ then $\mathrm{R}(C_1, C_2) = (P_1 \vee P_2)$
  - Since ϕ satisfies both C_1 and C_2 it must satisfy at least one of the literals in P_1 or P_2.
    - if ϕ satisfies l then it satisfies some literal in P_2
    - if ϕ satisfies $\bar{l}$ then it satisfies some literal in P_1

SAT Tools – Resolution

Theorem: Resolution maintains satisfiability

Let F be a CNF formula and C_1 and C_2 two of it's clauses with a pair complementary literals. Then F is satisfiable if and only if $F \wedge \mathrm{R}(C_1, C_2)$ is satisfiable.

Consequences

- If we manage to resolve the empty clause ($\perp$) the original formula is unsatisfiable

SAT Tools – Resolution

Theorem: Resolution maintains satisfiability

Let F be a CNF formula and C_1 and C_2 two of it's clauses with a pair complementary literals. Then F is satisfiable if and only if $F \wedge \mathrm{R}(C_1, C_2)$ is satisfiable.

Consequences

- If we manage to resolve the empty clause ($\perp$) the original formula is unsatisfiable

Usage

- Proof of unsatisfiability – Resolution Proof
  - A resolution proof is a sequence of clauses such that each clause is either a clause of the original formula or a resolvent of two previous clauses ending with $\perp$.

SAT Tools – Resolution

Theorem: Resolution maintains satisfiability

Let F be a CNF formula and C_1 and C_2 two of it's clauses with a pair complementary literals. Then F is satisfiable if and only if $F \wedge \mathrm{R}(C_1, C_2)$ is satisfiable.

Consequences

- If we manage to resolve the empty clause ($\perp$) the original formula is unsatisfiable

Usage

- Proof of unsatisfiability – Resolution Proof
  - A resolution proof is a sequence of clauses such that each clause is either a clause of the original formula or a resolvent of two previous clauses ending with $\perp$.

Example: $(x_1 \vee x_2), (\overline{x_1} \vee x_2), (x_1 \vee \overline{x_2}), (\overline{x_1} \vee \overline{x_2})$

SAT Tools – Resolution

Theorem: Resolution maintains satisfiability

Let F be a CNF formula and C_1 and C_2 two of it's clauses with a pair complementary literals. Then F is satisfiable if and only if $F \wedge \mathrm{R}(C_1, C_2)$ is satisfiable.

Consequences

- If we manage to resolve the empty clause ($\perp$) the original formula is unsatisfiable

Usage

- Proof of unsatisfiability – Resolution Proof
  - A resolution proof is a sequence of clauses such that each clause is either a clause of the original formula or a resolvent of two previous clauses ending with $\perp$.

Example: $(x_1 \vee x_2), (\overline{x_1} \vee x_2), (x_1 \vee \overline{x_2}), (\overline{x_1} \vee \overline{x_2}), (x_2), (\overline{x_2}), \perp$

SAT Tools – Saturation Algorithm

Saturation Algorithm

- INPUT: CNF formula F
- OUTPUT: $\{SAT, UNSAT\}$

while (true) **do**
    $R = \text{resolveAll}(F)$
    if $(R \cap F \neq R)$ **then** $F = F \cup R$
    else break
if $(\bot \in F)$ **then return** $UNSAT$ **else return** SAT

SAT Tools – Saturation Algorithm

Saturation Algorithm

- INPUT: CNF formula F
- OUTPUT: $\{SAT, UNSAT\}$

while (true) **do**
  $R = \text{resolveAll}(F)$
  if $(R \cap F \neq R)$ **then** $F = F \cup R$
  else break
if $(\bot \in F)$ **then return** $UNSAT$ **else return** SAT

Properties of the saturation algorithm:

- it is sound and complete – always terminates and answers correctly
- has exponential time and space complexity (always for Pigeons)

SAT Tools – Unit Propagation

Unit Resolution

= at least one of the resolved clauses is unit (has one literal).

Example:

- $\mathrm{R}((x_1 \vee x_7 \vee \overline{x_2} \vee x_4), (x_2)) = (x_1 \vee x_7 \vee x_4)$

SAT Tools – Unit Propagation

Unit Resolution

= at least one of the resolved clauses is unit (has one literal).

Example:

- $\mathrm{R}((x_1 \vee x_7 \vee \overline{x_2} \vee x_4), (x_2)) = (x_1 \vee x_7 \vee x_4)$

Unit Propagation

= a process of applying unit resolution as long as we get new clauses.

Example:

- $(x_1) \wedge (x_7 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_3)$
- $(x_1) \wedge (x_7 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_3) \wedge (x_3)$
- $(x_1) \wedge (x_7 \vee x_2 \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_3) \wedge (x_3) \wedge (x_7 \vee x_2)$

Easy Cases

SAT is not always hard, in the following cases it is polynomially solvable

- 2-SAT
- Horn-SAT
- Hidden Horn-SAT
- SLUR

2-SAT

2-SAT Formula

= each clause has exactly 2 literals.

Example:

- $(x_1 \vee x_3) \wedge (x_7 \vee \overline{x_3}) \wedge (\overline{x_1} \vee x_3)$
- $(x_1 \vee x_2) \wedge (\overline{x_1} \vee x_2) \wedge (x_1 \vee \overline{x_2}) \wedge (\overline{x_1} \vee \overline{x_2})$

Also called Binary SAT or Quadratic SAT

How to solve 2-SAT?

How to solve 2-SAT?

Saturation Algorithm

The resolution saturation algorithm is polynomial for 2-SAT

Proof:

- Only 2-literal resolvents are possible
- There are only $\mathcal{O}(n^2)$ 2-literal clauses on n variables

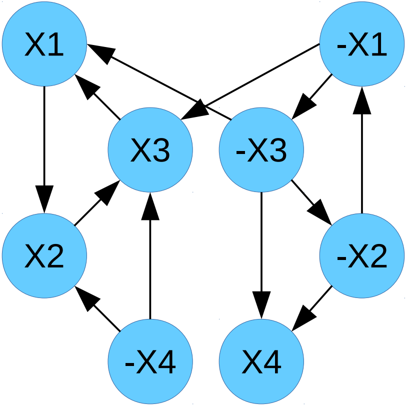
Complexity:

- Both time and space $\mathcal{O}(n^2)$
- There exists a linear algorithm! [1]

Implication Graph

Implication graph of a formula F is an oriented graph that has:

- a vertex for each literal of F
- 2 edges for each clause $(l_1 \vee l_2)$
  - $\bar{l}_1 \rightarrow l_2$
  - $\bar{l}_2 \rightarrow l_1$

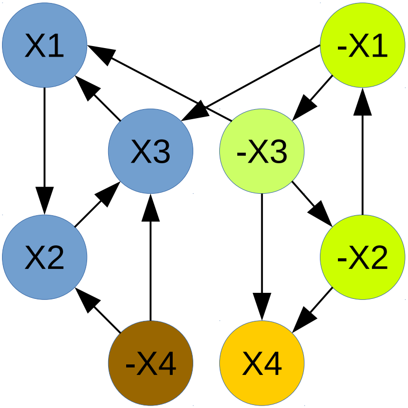


Example:
$(\overline{x}_1 \vee x_2) \wedge (\overline{x}_2 \vee x_3) \wedge (\overline{x}_3 \vee x_1) \wedge (x_2 \vee x_4) \wedge (x_3 \vee x_4) \wedge (x_1 \vee x_3)$

Implication Graph

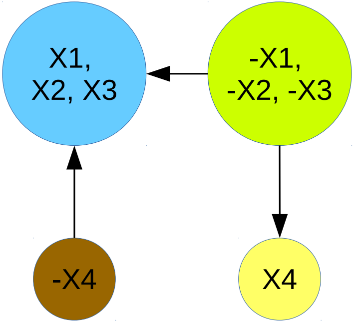
The next step is to analyze the *Strongly Connected Components* of the implication graphs

- SCC = there is a path in each direction between each pair
- Tarjan's algorithm finds SCCs in $\mathcal{O}(|V| + |E|)$
- If any x and $\overline{x}$ literal pair is in the same SCC then the formula is UNSAT
  - All the literals in an SCC must be all True or all False



How to find the solution?

- Construct the *Condensation* of the implication graph
  - contract each SCC into one vertex
- Topologically order the vertices of the condensation
- In reverse topological order, if the variables do not already have truth assignments, set all the terms to true.



Example: $x_1 = x_2 = x_3 = True$, $x_4 = True$, the rest is already assigned.

How to solve 2-SAT?

Linear Algorithm

- Construct the Implication Graph
- Find all the SCCs
- Check if any SCC contains a complementary pair
- Construct a condensation of the implication graph
- Run topological sort on the condensation
- Construct the solution

Complexity:

- All the steps can be done in linear time

HornSAT

Horn Formula

A CNF formula is a *Horn formula* is each of its clauses contains at most one positive literal.

Example: $(\overline{x}_1 \vee \overline{x}_7 \vee x_3) \wedge (\overline{x}_2 \vee \overline{x}_4) \wedge (x_1)$

HornSAT

Horn Formula

A CNF formula is a *Horn formula* is each of its clauses contains at most one positive literal.

Example: $(\overline{x}_1 \vee \overline{x}_7 \vee x_3) \wedge (\overline{x}_2 \vee \overline{x}_4) \wedge (x_1)$

Solving Horn Formulas

- Perform unit propagation on the input formula
- If you resolve $\perp$ then the formula is UNSAT otherwise it is SAT
- Get the solution:
  - Assign the variables in unit clauses to satisfy them
  - Set the rest of the variables to False

Mixed Horn

Mixed Horn Formula

A CNF formula is Mixed Horn if it contains only quadratic and Horn clauses.

Example: $(\overline{x}_1 \vee \overline{x}_7 \vee x_3) \wedge (\overline{x}_2 \vee \overline{x}_4) \wedge (x_1 \vee x_5) \wedge (x_3)$

Questions:

- How to solve a Mixed Horn formula?
- How to hard is it to solve a Mixed Horn formula?

Mixed Horn Formula

Mixed Horn Complexity

Mixed Horn SAT solving is NP-complete

Proof:

- We will reduce SAT to Mixed Horn SAT
- For each non-Horn clause $C = (l_1 \vee l_2 \vee \ldots)$ do
  - for each but one positive $l_i \in C$ introduce a new variable l'_i
  - replace l_i in C by $\overline{l'_i}$
  - add $(l'_i \vee l_i) \wedge (\overline{l'_i} \vee \overline{l_i})$ to establish $l_i = \overline{l'_i}$

Example: $(x_1 \vee \overline{x}_7 \vee x_3) \wedge (\overline{x}_2 \vee \overline{x}_4) \wedge (x_1 \vee x_5) \rightsquigarrow$
$\rightsquigarrow (\overline{x'_1} \vee \overline{x}_7 \vee x_3) \wedge (\overline{x}_2 \vee \overline{x}_4) \wedge (\overline{x'_1} \vee x_5) \wedge (x'_1 \vee x_1) \wedge (\overline{x'_1} \vee \overline{x_1})$

Hidden/Renamable/Disguised Horn

Hidden Horn Formulas

A CNF formula is *Hidden Horn* if it can be made Horn by renaming some of its variables.

Example:

$$(x_1 \vee x_2 \vee x_4) \wedge (x_2 \vee \overline{x}_4) \wedge (x_1) \rightsquigarrow (\overline{x}_1 \vee \overline{x}_2 \vee x_4) \wedge (\overline{x}_2 \vee \overline{x}_4) \wedge (\overline{x}_1)$$

Questions:

- How to recognize a Hidden Horn formula?
- How to hard is it to recognize and solve a Hidden Horn formula?

Recognizing Hidden Horn Formulas

Translate into 2-SAT

Let F be original formula, R_F contains the clause $(l_1 \vee l_2)$ if and only if there is a clause $C \in F$ such that $l_1 \in C$ and $l_2 \in C$.

Example: $F = (x_1 \vee x_2 \vee x_4) \wedge (x_2 \vee \overline{x}_4) \wedge (x_1)$
$R_F = (x_1 \vee x_2) \wedge (x_1 \vee x_4) \wedge (x_2 \vee x_4) \wedge (x_2 \vee \overline{x}_4)$

Recognize Hidden Horn

If R_F is satisfiable, then F is a hidden Horn formula. Furthermore, the satisfying assignment ϕ of R_F identifies the variables to be renamed.

- if $x_i = \mathit{True}$ in ϕ then x_i needs to be renamed to $\overline{x}_i$

SLUR Formulas

Karlsruhe Institute of Technology

Single Look-ahead Unit Resolution - SLUR algorithm

01 **if** $\bot \in$ unit-prop(F) **then return** UNSAT **else return** SLUR(F)

02 **function** SLUR(F)
03 **if** all variables appear in a unit clause **then return** SAT
04 $v =$ select-variable(F)
05 $F_1 =$ unit-prop$(F \wedge (v))$
06 $F_2 =$ unit-prop$(F \wedge (\overline{v}))$
07 **if** $\bot \in F_1$ **and** $\bot \in F_2$ **then return** GIVE-UP
08 **if** $\bot \in F_1$ **and** $\bot \notin F_2$ **then** SLUR(F_2)
09 **if** $\bot \notin F_1$ **and** $\bot \in F_2$ **then** SLUR(F_1)
10 **if** $\bot \notin F_1$ **and** $\bot \notin F_2$ **then** SLUR(F_1) **or** SLUR(F_2)

A CNF formula F is SLUR if the SLUR algorithm never gives up on for F (regardless of the choices in lines 04 and 10).

SLUR Formulas

Properties of SLUR Formulas [2]:

- SLUR formulas are solvable in polynomial time (using the SLUR algorithm)
- SLUR is an umbrella class for polynomially solvable classes – All Horn and Hidden Horn formulas are SLUR formulas
  - Also true for Extended Horn, CC-balanced, and Propagation Complete formulas
- It is co-NP complete to recognize whether a given CNF is a SLUR formula or not

Stochastic Local Search (SLS)

SAT as an optimization problem: minimize the number of unsatisfied clauses

Start with a complete random assignment α:

0	0	1	0	1	1	0	0	1	1	0	1	0	1	0

Repeatedly flip (randomly/heuristically chosen) variables to decrease the number of unsatisfied clauses:

0	0	1	0	1	0	0	0	1	1	0	1	0	1	0

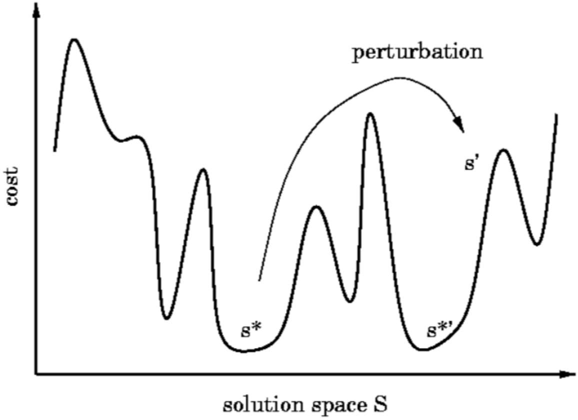
SLS Algorithms

- Local search algorithms are **incomplete**: they cannot show unsatisfiability!
- Many variants of local search algorithms
- Main question: Which variable should be flipped next?
  - select variable from an unsatisfied clause
  - select variable that increases the number of satisfied clauses most
- How to avoid local minima?

GSAT Algorithm [3]

```
Maybe[Assignment] GSAT(ClauseSet S)
{
  for i = 1 to MAX_TRIES do {
    α = random-assignment to variables in S
    for j = 1 to MAX_FLIPS do {
      if ( α satisfies all clauses in S ) return α
      x = variable that produces least number of
        unsatisfied clauses when flipped
      flip x
    }
  }
  return Nothing   // no solution found
}
```

SLS: Illustration



[Source: Alan Mackworth, UBC, Canada]

Walksat [4]

- Variant of GSAT
- Try to avoid local minima by introducing "random noise"
  - Select unsatisfied clause C at random
  - If by flipping a variable $x \in C$ no new unsatisfied clauses emerge, flip x
  - Otherwise:
    - With probability p select a variable $x \in C$ at random
    - With probability $1 - p$ select a variable that changes as few as possible clauses from satisfied to unsatisfied when flipped

SLS: Important Notions

- Consider a flip taking α to α'
- **breakcount:** number of clauses satisfied in α, but not in α'
- **makecount:** number of clauses unsatisfied in α, but satisfied in α'
- **diffscore:** number of unsatisfied clauses in α minus number of clauses unsatisfied in α'
- Typically, **breakcount**, **makecount** and **diffscore** are updated after each flip
- Question: How can we do this efficiently?

GSAT and Walksat Flip Heuristics

- GSAT: select variable with highest **diffscore**
- Walksat:
  - First randomly select unsatisfied clause C
  - If there is a variable with **breakcount** 0 in C, select it
  - otherwise with probability p select a random variable from C, and with probability $1 - p$ a variable with minimal **breakcount** from C

Runtime Comparison Walksat vs. GSAT

formula			DP	GSAT+w	WSAT
id	vars	clauses	time	time	time
2bitadd_12	708	1702	*	0.081	0.013
2bitadd_11	649	1562	*	0.058	0.014
3bitadd_32	8704	32316	*	94.1	1.0
3bitadd_31	8432	31310	*	456.6	0.7
2bitcomp_12	300	730	23096	0.009	0.002
2bitcomp_5	125	310	1.4	0.009	0.001

Table 4: Comparing an efficient complete method (DP) with local search strategies on circuit synthesis problems. (Timings in seconds.)

formula			DP	GSAT+w	WSAT
id	vars	clauses	time	time	time
ssa7552-038	1501	3575	7	129	2.3
ssa7552-158	1363	3034	*	90	2
ssa7552-159	1363	3032	*	14	0.8
ssa7552-160	1391	3126	*	18	1.5

Table 5: Comparing DP with local search strategies on circuit diagnosis problems by Larrabee (1989). (Timings in seconds.)

[Source: Selman, Kautz, Cohen Local Search Strategies for Satisfiability Testing, 1993]

Are you touched by SAT?

References I

- B. Aspvall, M. F. Plass, R. E. Tarjan, A linear-time algorithm for testing the truth of certain quantified boolean formulas, Information Processing Letters 8 (3) (1979) 121–123.
- O. Čepek, P. Kučera, V. Vlček, Properties of slur formulae, in: SOFSEM 2012: Theory and Practice of Computer Science, Springer, 2012, pp. 177–189.
- B. Selman, H. Levesque, D. Mitchell, A new method for solving hard satisfiability problems, in: Proceedings of the Tenth National Conference on Artificial Intelligence, AAAI'92, AAAI Press, 1992, pp. 440–446.
  URL http://dl.acm.org/citation.cfm?id=1867135.1867203

References II

D. J. Johnson, M. A. Trick (Eds.), Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, Workshop, October 11-13, 1993, American Mathematical Society, Boston, MA, USA, 1996.