Propositional logic lecture 3

Hadjila Fethallah

Associate professor at the Computer science department

fethallah.hadjila@univ-tlemcen.dz

Motivations for SAT-Solvers

- The resolution is no longer effective when the K-CNF formula is satisfiable.
- Even if the formula is unsatisfiable, the search may take a long time due to lack of guidance
- Need for new algorithms, especially for k>=3!

SAT Problems

- 02 large families to check satisfiability propositional:
- Complete algorithms (with back tracking)
  - DPLL algorithm (Davis, Putnam, Logemann, Loveland, 62)
- Incomplete algorithms (local search)
  - WalkSAT algorithm
- Problems are easy if the size of each clause <=2
- It is generally difficult for a size >=3

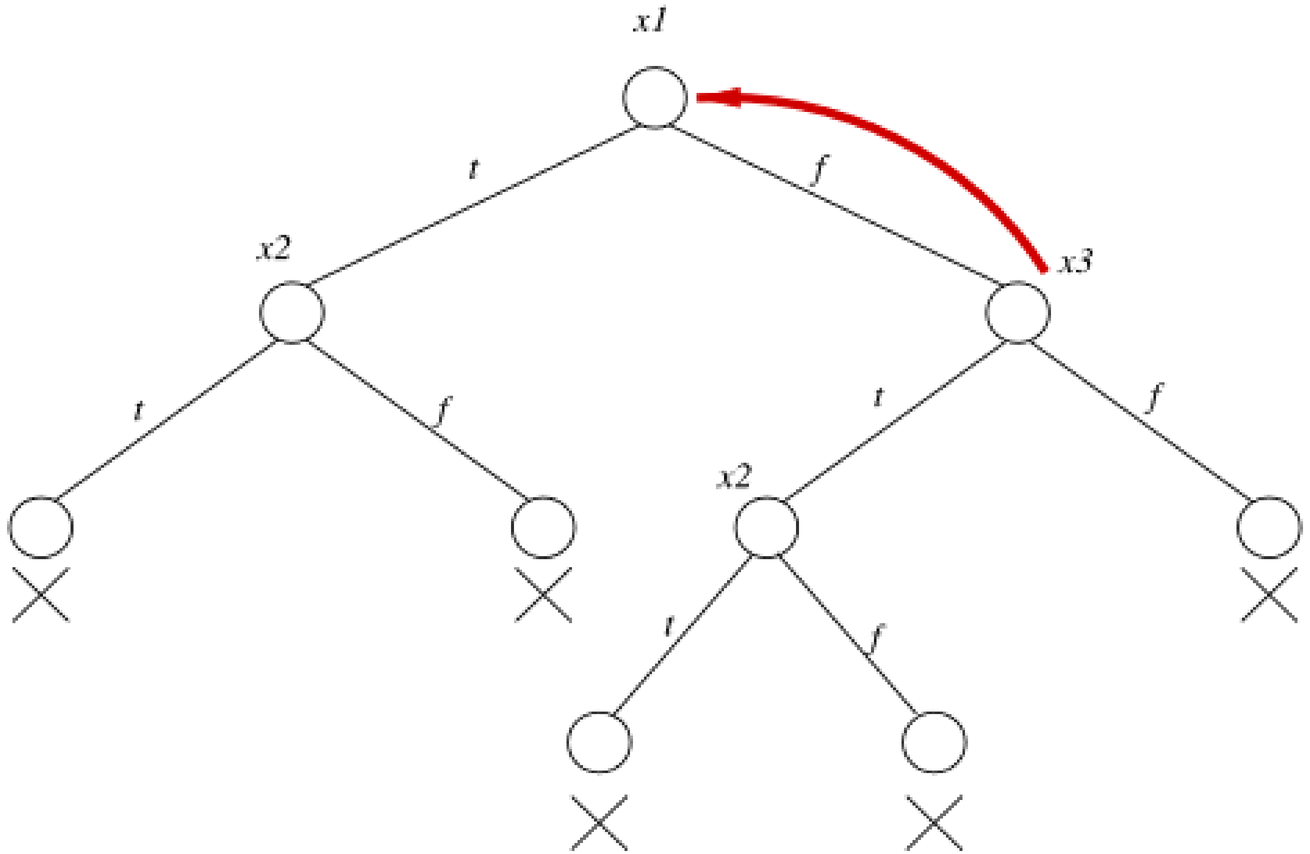
DPLL Algorithm

- IT determines whether a formula in CNF format is satisfiable or not
- Huge improvement with respect to the truth table:
- **1. Early termination**: a clause is true, if at least one literal is true. The formula is false if at least one clause is false.
- 2. **Pure symbol heuristic (monotonic)** A symbol is pure, if it always appears with the same "sign" in all clauses. Ex: $(A \vee \neg B)$, $(\neg B \vee \neg C)$, $(C \vee A)$, A, B are pure, C is not. We assign the pure symbol to true
- 3. **Unitary clause heuristic:** if there is only one literal in the clause, we assign it to true

DPLL Algorithm

- Function DPLL(Φ) return True or False
- If Φ doesnt contain any clauses, then return true;
- If Φ contains the empty clause, then return false;
- while there are unitary clauses L in Φ
  - Φ ← unitary-propagation(L, Φ);
- while there are pure-literal L in Φ
  - Φ ← assign-pure-literal(L, Φ);
- l ← choose-literal(Φ);
- if DPLL(Φ_L)=true then return true
- else return DPLL($\Phi_{\neg L}$)

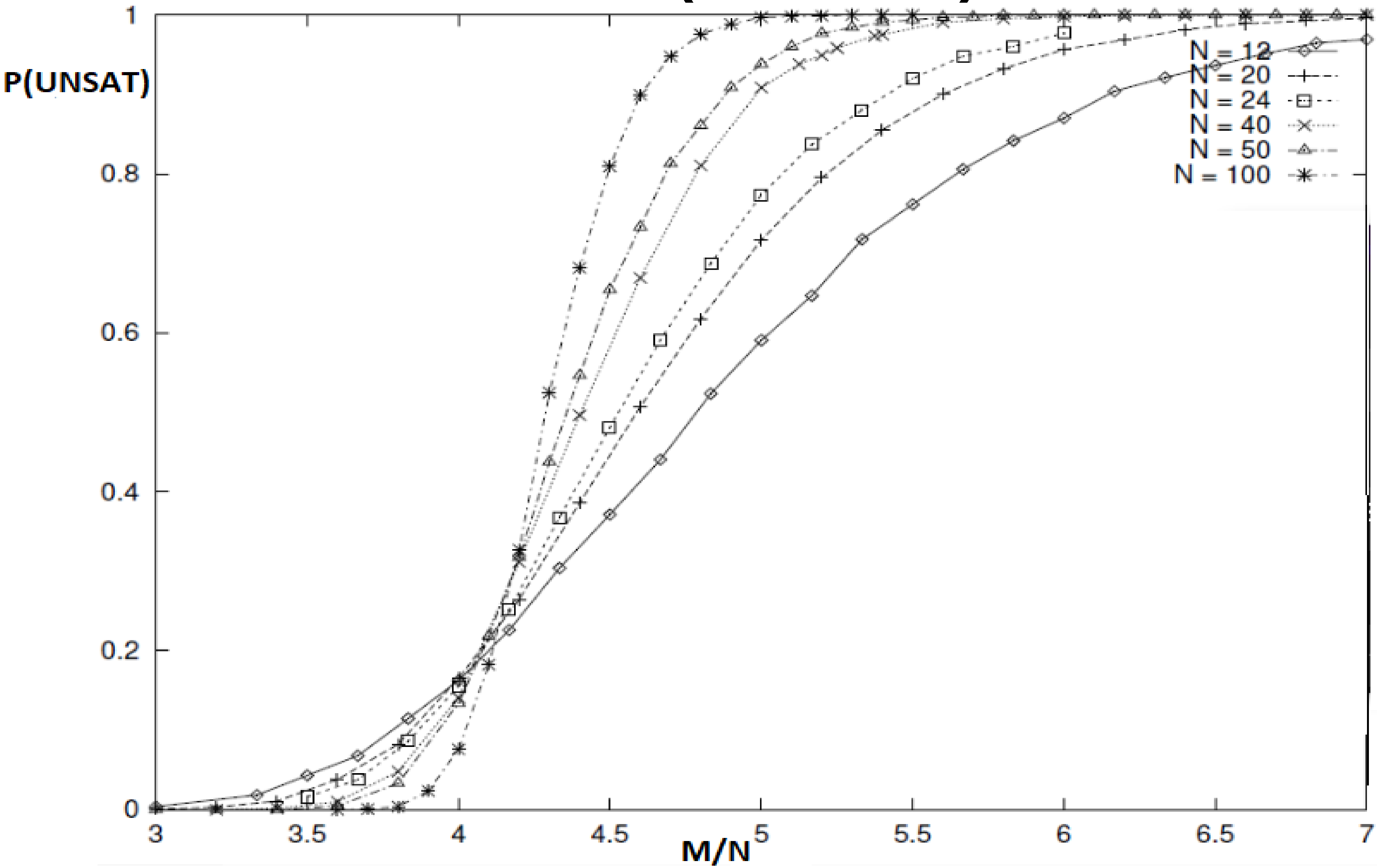
Search tree



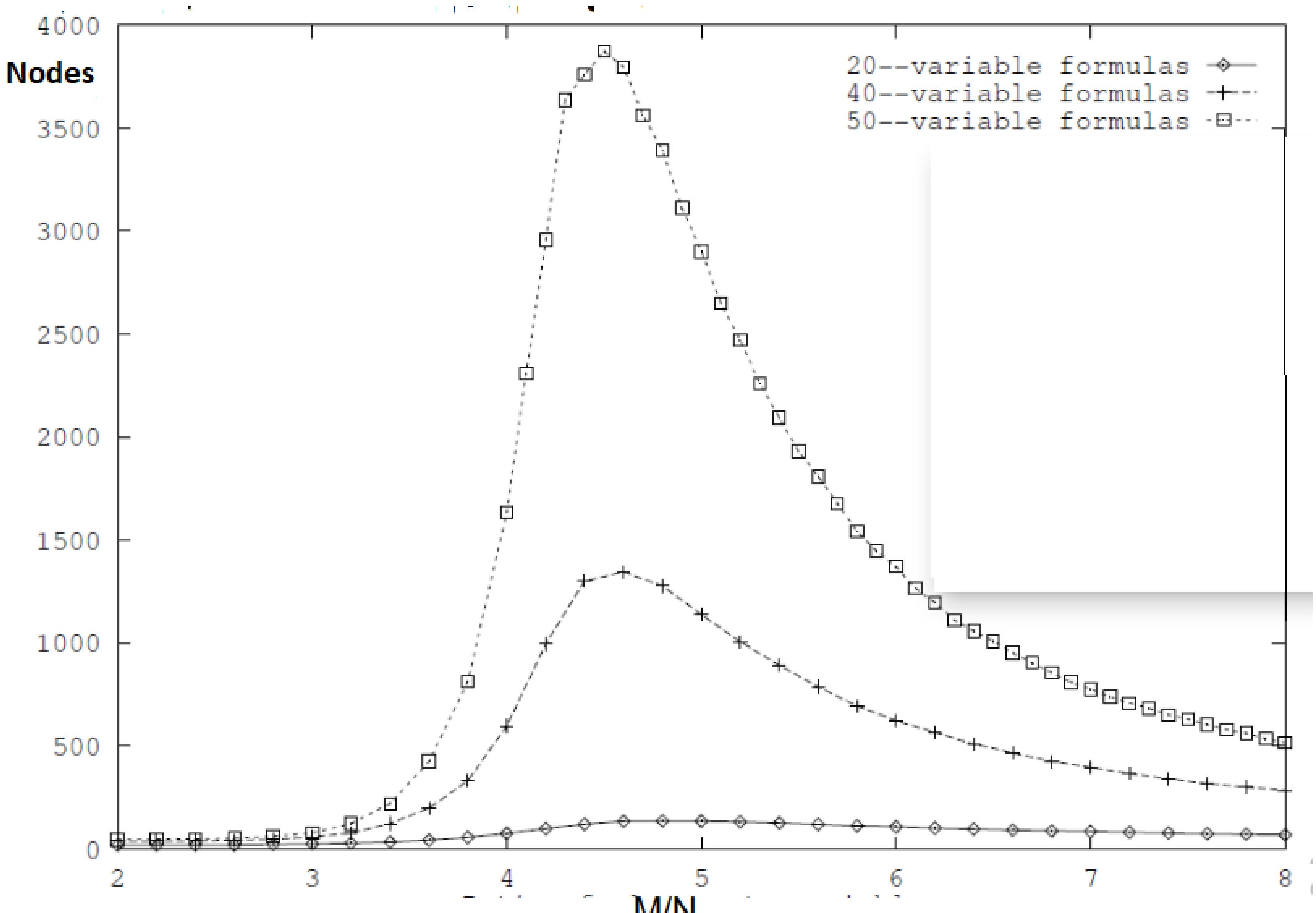
Satisfiability

- Simulate DPLL on these 3-CNF formulas:
- $(\neg D \vee \neg B \vee C) \wedge (B \vee \neg A \vee \neg C) \wedge (\neg C \vee \neg B \vee E) \wedge (E \vee \neg D \vee B) \wedge (B \vee E \vee \neg C)$
- $(\neg A \vee \neg B \vee \neg C) \wedge (\neg A \vee B) \wedge (\neg B \vee C) \wedge (\neg C \vee A) \wedge (A \vee B \vee C)$
- M= clauses' number
- N= atoms' number
- hard instances belong to the neighborhood of M/N ≈ 4.3 (critical point).

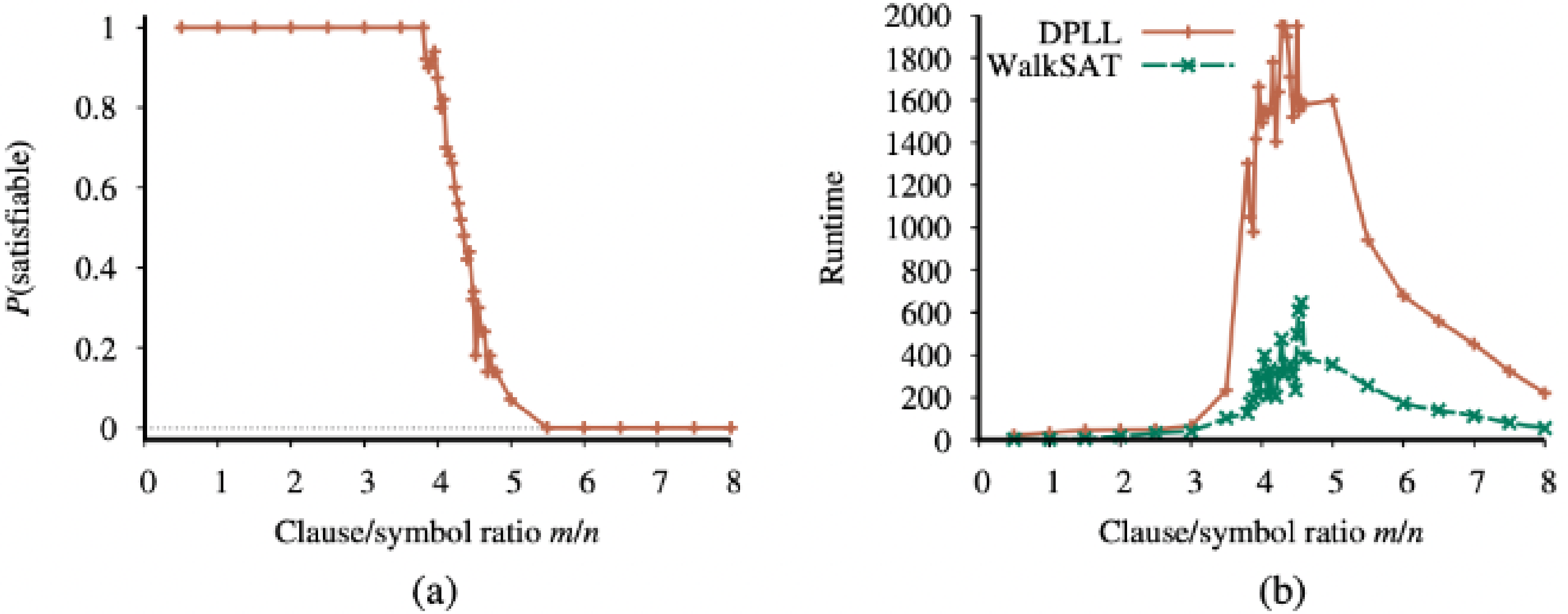
Hard instances (3-SAT)



Hard instances (3-SAT)



Hard instances (3-SAT)



(a) (b)

number of atoms n=50

Industrial instances (2-SAT)

The instance bmc-ibm-6.cnf, IBM LSU 1997:

```
p cnf 51639 368352
-1 7 0
-1 6 0
-1 5 0
-1 -4 0
-1 3 0
-1 2 0
-1 -8 0
-9 15 0
-9 14 0
-9 13 0
-9 -12 0
-9 11 0
-9 10 0
-9 -16 0
-17 23 0
-17 22 0
```

i.e. $((\neg x_1) \text{ or } x_7)$
and $((\neg x_1) \text{ or } x_6)$
and … etc.

- ✓ 15000 pages de clauses
- ✓ 50000 variables booléennes

The problem is solved in 2 Sec by the MiniSat Solver (Een&Sorensson)

Industrial instances (3-SAT)

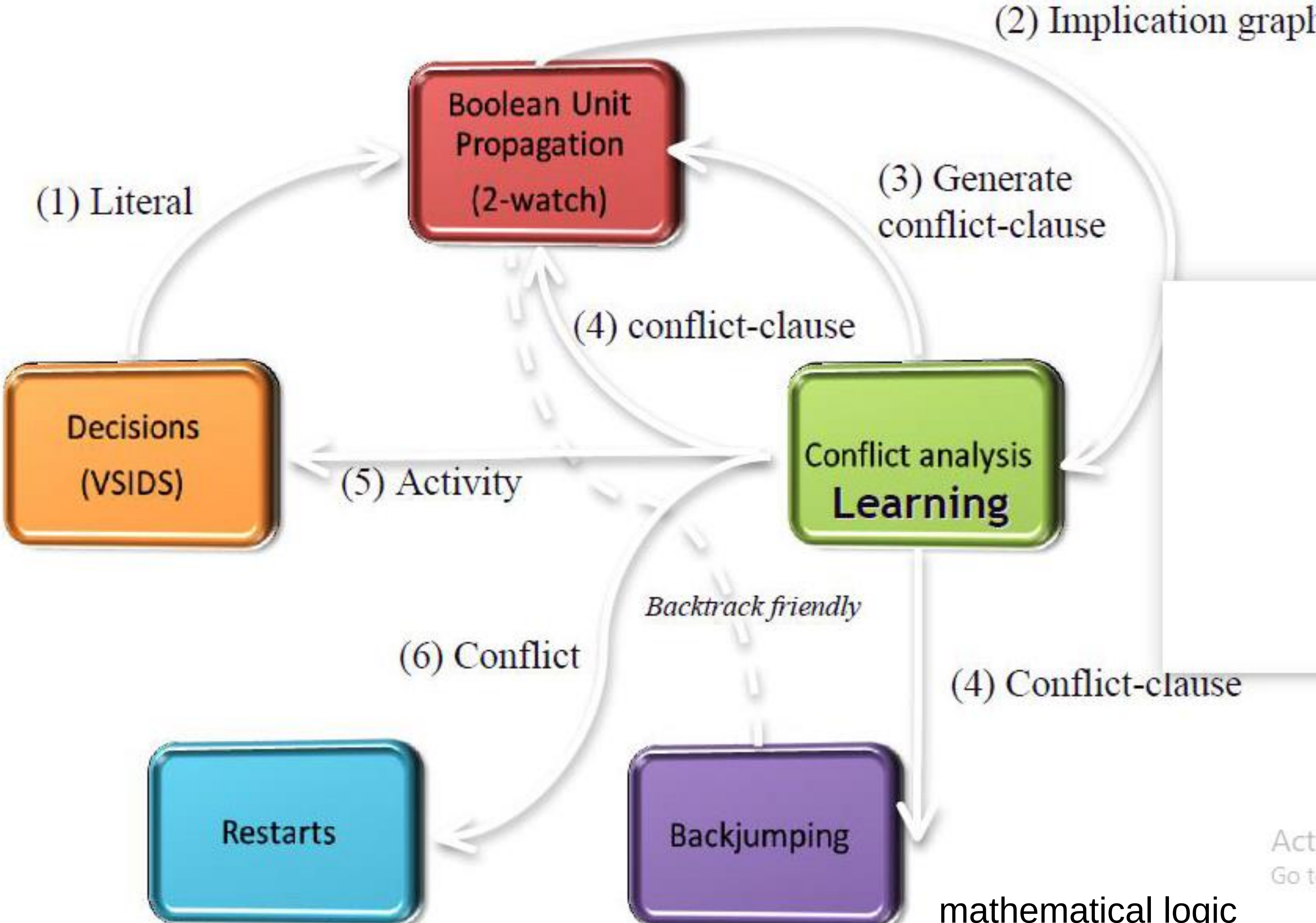
exqmple : post-cbmc-zfcp-2.8-u2.cnf

p cnf **11 483 525** (vars) **32 697 150** (clauses)
1 -3 0
2 -3 0 $\leftarrow x_1 = \wedge(x_2, x_3)$
-1 -2 3 0
...
...
-11482897 -11483041 -11483523 0
11482897 11483041 -11483523 0
11482897 -11483041 11483523 0 $\leftarrow (x_3 \Leftrightarrow x_2 \Leftrightarrow x_3)$
-11482897 11483041 11483523 0
-11483518 -11483524 0
-11483519 -11483524 0
-11483520 -11483524 0
-11483521 -11483524 0 $\leftarrow x_6 = \wedge(x_7, x_8, x_9, x_{10}, x_{11}, x_{12})$
-11483522 -11483524 0
-11483523 -11483524 0
11483518 11483519 11483520 11483521 11483522 11483523 11483524 0
-8590303 -11483524 -11483525 0
8590303 11483524 -11483525 0
8590303 -11483524 11483525 0 $\leftarrow (x_{13} \Leftrightarrow x_{14} \Leftrightarrow x_{15})$
-8590303 11483524 11483525 0
-11483525 0

Modern Sat Solvers

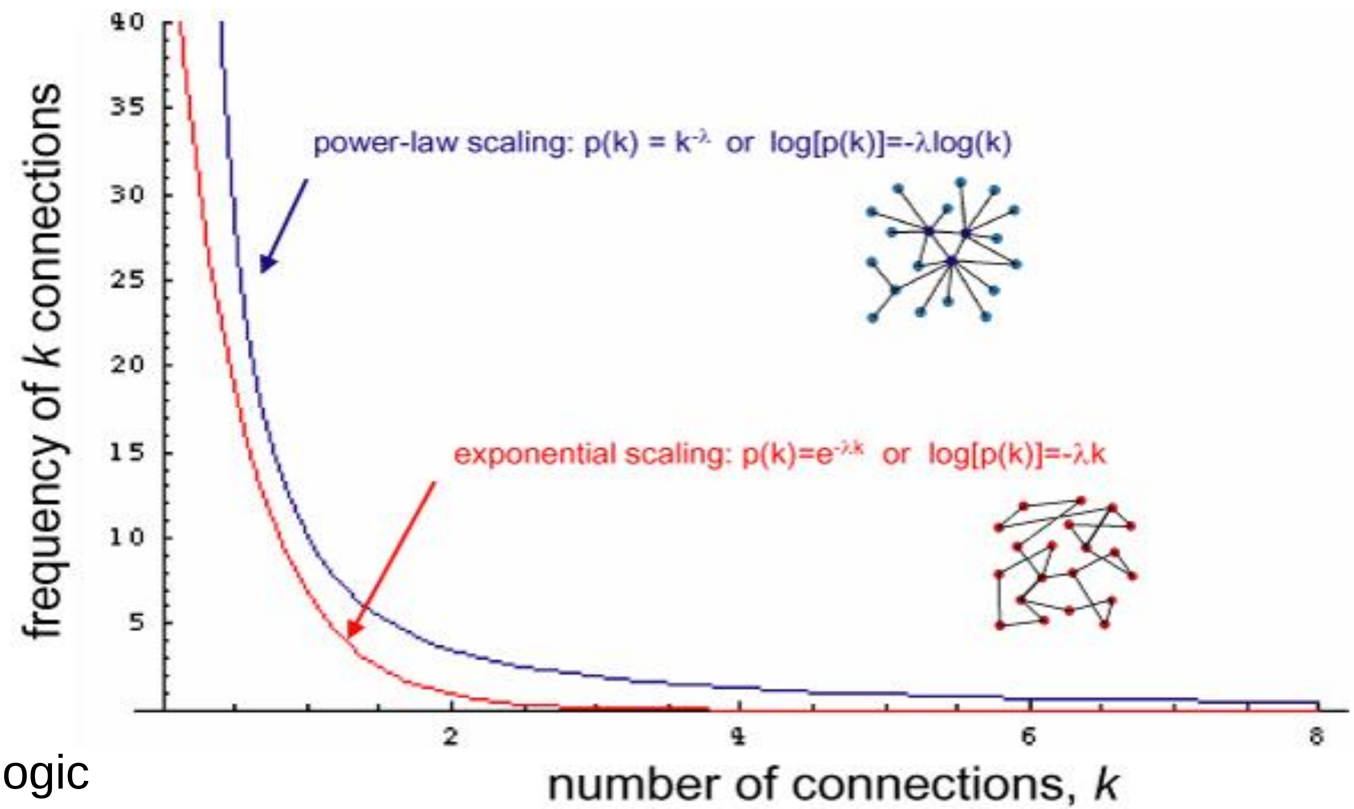
- 1.Heavy tailed phenomena: Execution time of 3-SAT problems can be large due to bad choices of Boolean variables [Gomes et al, 97]
  - Solution: random restart
- 2. Learning conflict clauses: adding some specific clauses can stop searching in unnecessary areas [Marques Silva et al. 96]
- 3. Sorting of variables according to their activity: [Brisoux et al, 99], [Moskewicz et al, 01]: Effective heuristics for the simplification of the problem
- 4. Watched literals: [H. Zhang el al,97], [Moskewicz et al,01] acceleration of unit propagation
- 5.Symmetries
- 6.Parallelism

Modern Sat Solvers

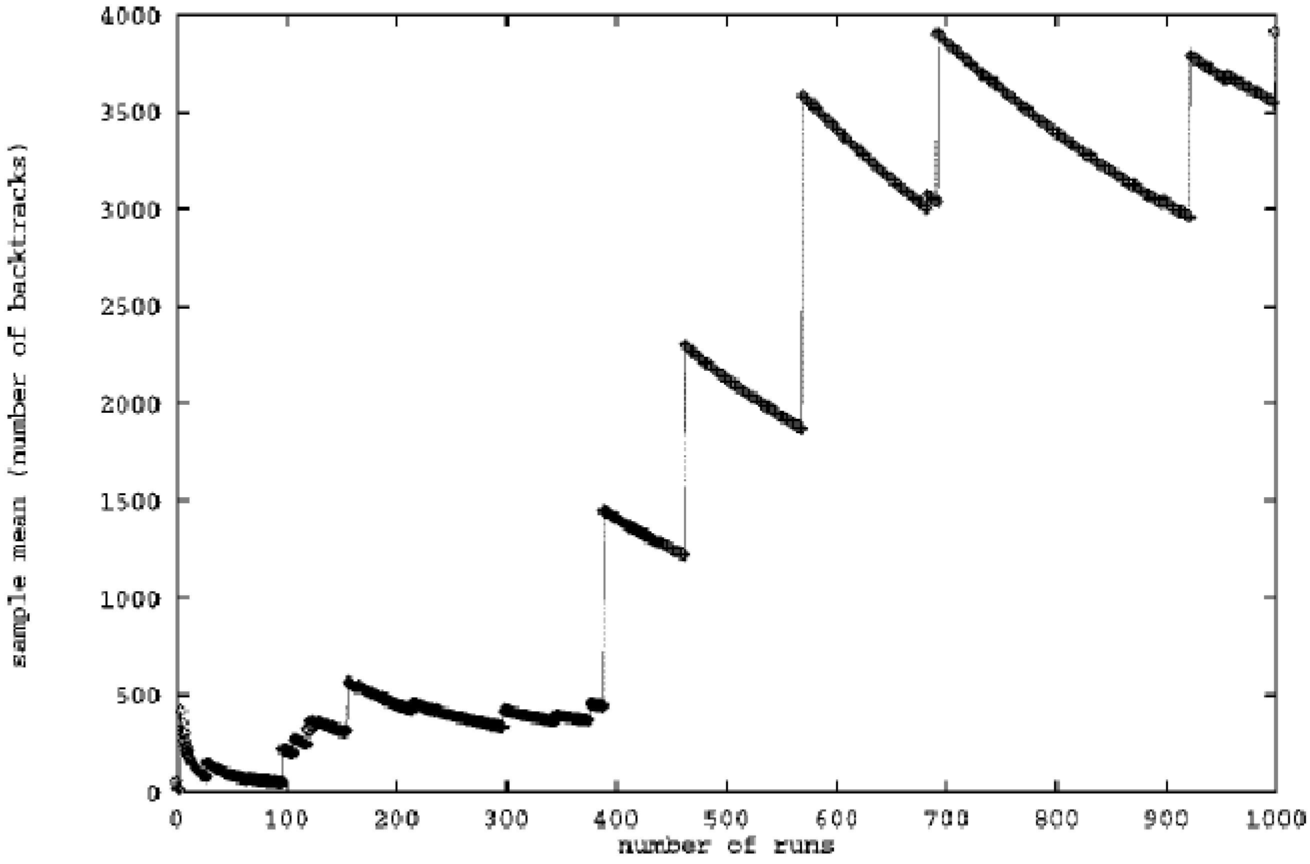


Heavy tailed phenomena

- There are random variables that don't follow exponential/normal distributions but power law distributions
- The probability density function is given as follows: pdf(x)= a.X^{-k}



Heavy tailed phenomena



mathematical logic

Random restart

- Avoid being stagnated in a branch (sub-tree) which is not satisfiable.
- With a random restart, it is possible to firstly assign the most critical variables (backdoors) of a system.
- Therefore, what is left is easy to solve.
- The execution time will be acceptable.
- The problem of heavy tails is eliminated.

Cut-off restart

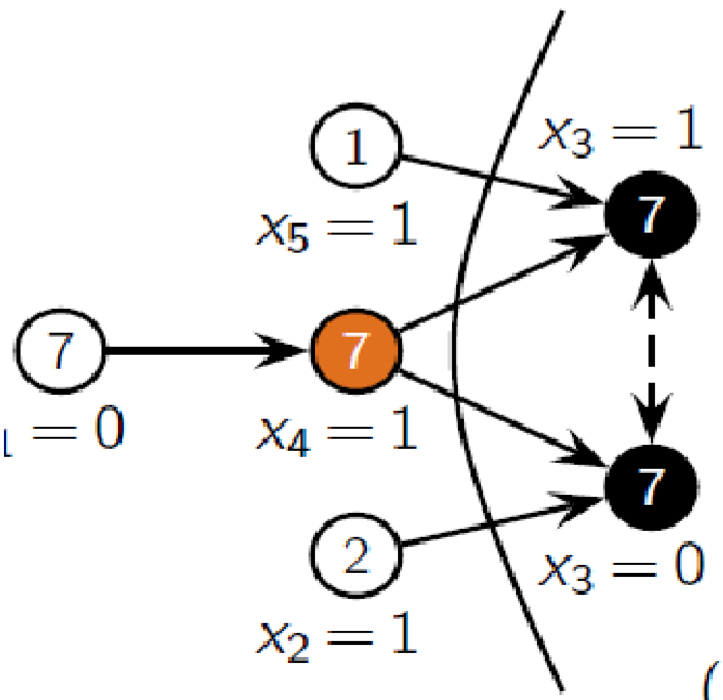
- We count the number of backtrackings (or nodes) since the last reboot or restart.
- Geometric policy:
  - $X_i = 1.5 \times X_{i-1}$.
- Arithmetic Policy:
  - $X_i = X_{i-1} + 16000$
- Luby sequence: e.g. 100, 100, 200, 100, 100, 200, 400,…

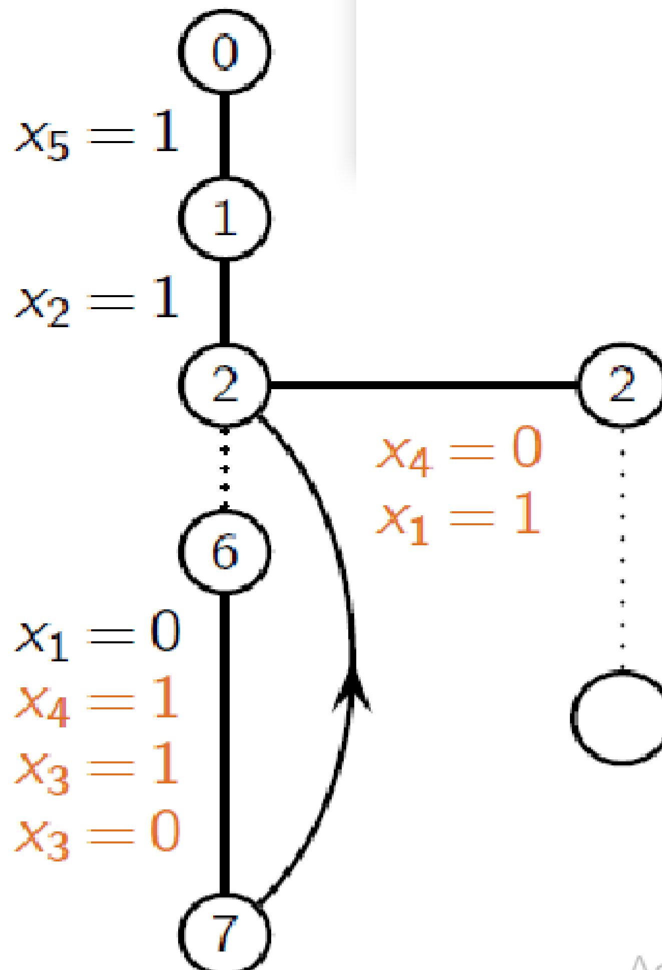
Clause learning

- Goal: prune the search tree as quickly as possible.
- Add a new clause to stop unnecessary research in the future.
- Backtracking no chronological (backjumping).

Clause learning

$(x_1 \lor x_4) \land$
$(x_3 \lor \bar{x}_4 \lor \bar{x}_5) \land$
$(\bar{x}_3 \lor \bar{x}_2 \lor \bar{x}_4) \land$
$\mathcal{F}_{\mathrm{extra}}$

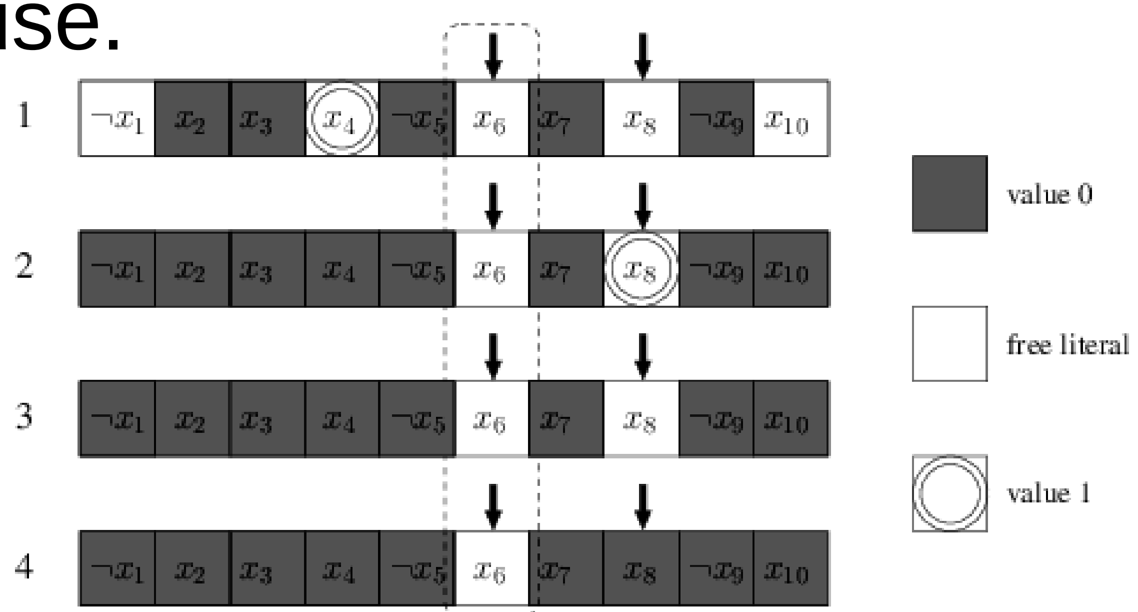




$(\bar{x}_2 \lor \bar{x}_4 \lor \bar{x}_5)$

Two watched literals

- Checking the unitary size of clauses is expensive for unitary propagation.
- Idea: maintain a data structure (02 pointers) for each clause.



Choice of division variables

- MOMS heuristic (Maximum Occurrence in clauses of Minimal Size) [JT96]
- Variable State Independent Decaying Sum (VSIDS)
  - [MMZ+01] Each variable is associated with a score which is incremented each time this variable is part of a conflict clause
  - We encourage the choice of variables involved in recent conflicts

Parallelism

- Assign the variables of division and allocate the created nodes to the processors.
- Exploitation of multi-core processors

Some references

- SAT competition:
- http://www.satcompetition.org/
- International Conference on Theory and Application of Satisfiability Testing (SAT)

Fundamental properties

- Correction(soundness)
- Every proven formula is valid

  if $\vdash$ **A** then $\models$ **A**

- Completeness
- Every valid formula is proven (A is a theorem)

  if $\models$ **A** then $\vdash$ **A**

- Decidability

  for each formula A $\in$Prop, there is a proof method (such as resolution) which says if A is valid or not within a finite amout of time.

END