DM811

Heuristics for Combinatorial Optimization

# Lecture 6
# SAT

Marco Chiarandini

Department of Mathematics & Computer Science
University of Southern Denmark

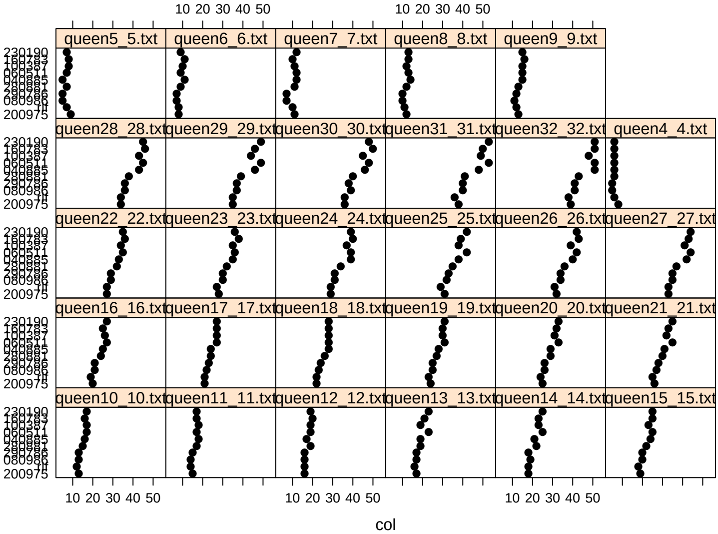
# Outline

# Recap

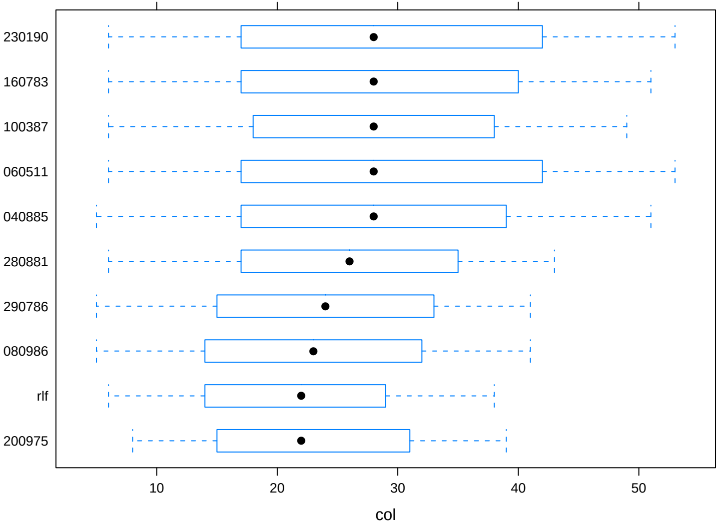
- Combinatorial Optimization and Terminology
- Basic Concepts in Algorithmics
  Graphs • Notation and runtime • Machine model • Pseudo-code • Computational Complexity • Analysis of Algorithms
- Construction Heuristics + Local Search + Metaheuristics
- Software systems and Working Environment
  [Comet, EasyLocal++, unix]
- Assignment 1 + Analysis of Results in R
  [RStudio, Cheat Sheet, My Notes, script]
- Construction Heuristics (search tree, variable + value model)
  [from complete (DFS, Best first, A*) to incomplete search (greedy)]
- Metaheuristics on CH

# Outline

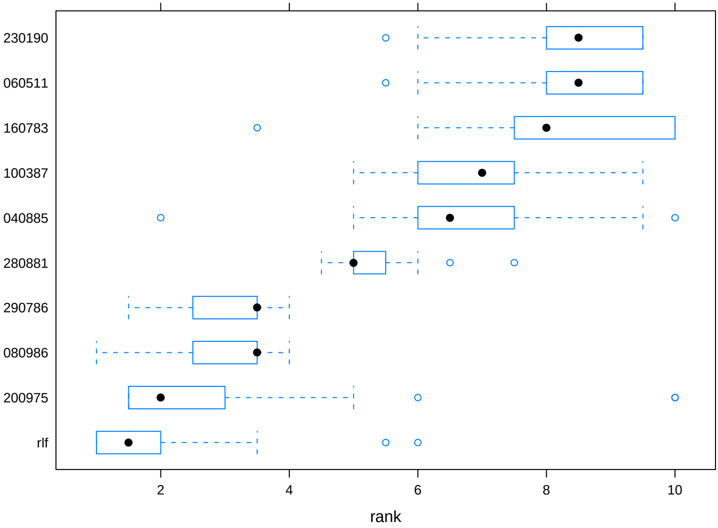
# Outline

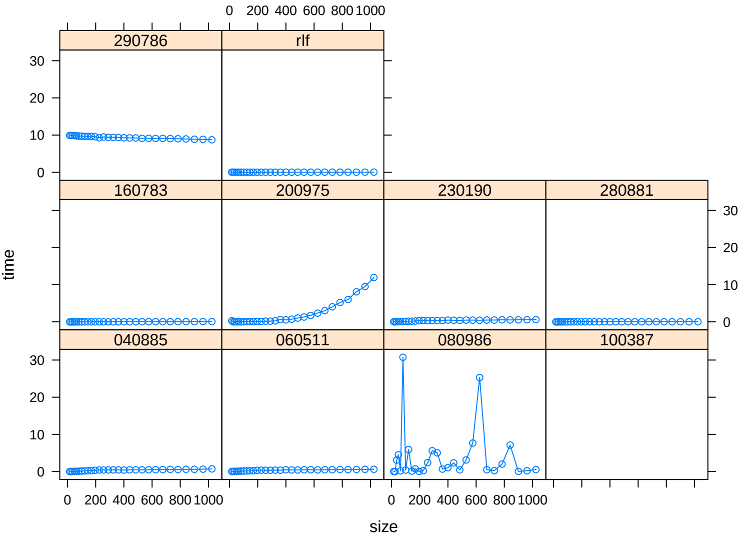


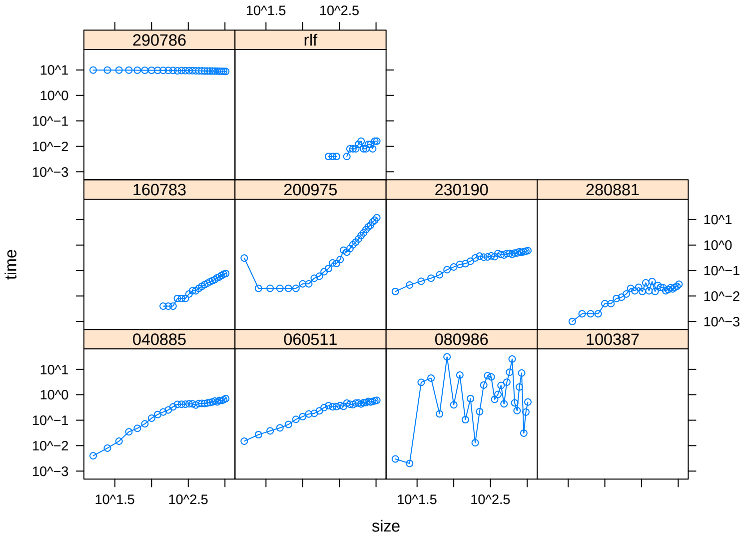

Note the floor/ceiling effect on the small instances



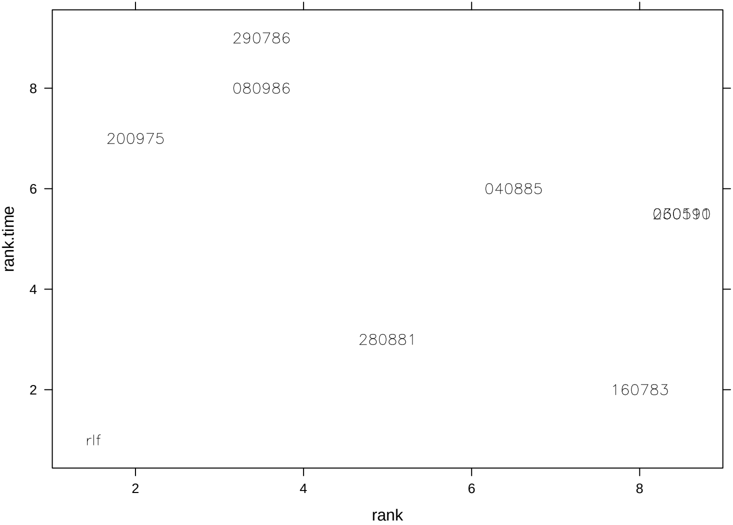

Different scales among instances hide differences


230190
060511
160783
100387
040885
280881
290786
080986
200975
rlf
2
4
6
8
10
rank


290786
rlf
160783
200975
230190
280881
040885
060511
080986
100387
time
size
0
10
20
30
0
200
400
600
800
1000




log-log transformation ⇝ polynomial is a straigth line


290786
080986
200975
040885
280881
160783
rlf
rank.time
rank
2
4
6
8

# Outline

# Examples

```
import cotls;
include "loadDIMACS";
// int nv;
// int me;
// float alpha;
// bool adj[nv,nv];
range Vertices = 1..nv;
range Colors = 1..nv;
int nbc = Colors.getUp();

Solver<LS> m();

var{int} col[Vertices](m,Colors) := 1;
ConstraintSystem<LS> S(m);

forall (i in Vertices, j in Vertices: j>i && adj[i,j])
S.post(col[i] != col[j]);
S.close();

m.close();

// CONSTRUCTION HEURISTIC
set{int} dom[v in Vertices] = setof(c in Colors) true;
RandomPermutation perm(Vertices);
forall (i in 1..nv) {
  int v = perm.get();
  selectMin(c in dom[v])(c) {
    col[v] := c;
    forall(w in Vertices: adj[v,w])
      dom[w].delete(c);
  }
}
nbc = max(v in Vertices) col[v];
Colors = 1..nbc;
cout<<"Construction␣heuristic,␣done:␣"<<nbc<<"␣colors"<< endl;
```

code1.java/png code3.cpp

# Where do speedups come from?

Where can maximum speedup be achieved?
How much speedup should you expect?

# Code Tuning

- Caution: proceed carefully! Let the optimizing compiler do its work!
- Expression Rules: Recode for smaller instruction counts.
- Loop and procedure rules: Recode to avoid loop or procedure call overhead.
- Hidden costs of high-level languages
- String comparisons in C: proportional to length of the string, not constant
- Object construction / de-allocation: very expensive
- Matrix access: row-major order $\neq$ column-major order
- Exploit algebraic identities
- Avoid unnecessary computations inside the loops

# Where Speedups Come From?

McGeoch reports conventional wisdom, based on studies in the literature.

- Concurrency is tricky: bad -7x to good 500x
- Classic algorithms: to 1trillion and beyond
- Data-aware: up to 100x
- Memory-aware: up to 20x
- Algorithm tricks: up to 200x
- Code tuning: up to 10x
- Change platforms: up to 10x

# Relevant Literature

Bentley, **Writing Efficient Programs; Programming Pearls** (Chapter 8 Code Tuning)

Kernighan and Pike, **The Practice of Programming** (Chapter 7 Performance).
Shirazi, **Java Performance Tuning**, O'Reilly

McCluskey, **Thirty ways to improve the performance of your Java program.** Manuscript and website: `www.glenmcci.com/jperf`

Randal E. Bryant e David R. O'Hallaron: **Computer Systems: A Programmer's Perspective**, Prentice Hall, 2003, (Chapter 5)

# Outline

# SAT Problem

## Satisfiability problem in propositional logic

$$
\begin{array}{l}
(x_5 \vee x_8 \vee \bar{x}_2) \wedge (x_2 \vee \bar{x}_1 \vee \bar{x}_3) \wedge (\bar{x}_8 \vee \bar{x}_3 \vee \bar{x}_7) \wedge (\bar{x}_5 \vee x_3 \vee x_8) \wedge \\
(\bar{x}_6 \vee \bar{x}_1 \vee \bar{x}_5) \wedge (x_8 \vee \bar{x}_9 \vee x_3) \wedge (x_2 \vee x_1 \vee x_3) \wedge (\bar{x}_1 \vee x_8 \vee x_4) \wedge \\
(\bar{x}_9 \vee \bar{x}_6 \vee x_8) \wedge (x_8 \vee x_3 \vee \bar{x}_9) \wedge (x_9 \vee \bar{x}_3 \vee x_8) \wedge (x_6 \vee \bar{x}_9 \vee x_5) \wedge \\
(x_2 \vee \bar{x}_3 \vee \bar{x}_8) \wedge (x_8 \vee \bar{x}_6 \vee \bar{x}_3) \wedge (x_8 \vee \bar{x}_3 \vee \bar{x}_1) \wedge (\bar{x}_8 \vee x_6 \vee \bar{x}_2) \wedge \\
(x_7 \vee x_9 \vee \bar{x}_2) \wedge (x_8 \vee \bar{x}_9 \vee x_2) \wedge (\bar{x}_1 \vee \bar{x}_9 \vee x_4) \wedge (x_8 \vee x_1 \vee \bar{x}_2) \wedge \\
(x_3 \vee \bar{x}_4 \vee \bar{x}_6) \wedge (\bar{x}_1 \vee \bar{x}_7 \vee x_5) \wedge (\bar{x}_7 \vee x_1 \vee x_6) \wedge (\bar{x}_5 \vee x_4 \vee \bar{x}_6) \wedge \\
(\bar{x}_4 \vee x_9 \vee \bar{x}_8) \wedge (x_2 \vee x_9 \vee x_1) \wedge (x_5 \vee \bar{x}_7 \vee x_1) \wedge (\bar{x}_7 \vee \bar{x}_9 \vee \bar{x}_6) \wedge \\
(x_2 \vee x_5 \vee x_4) \wedge (x_8 \vee \bar{x}_4 \vee x_5) \wedge (x_5 \vee x_9 \vee x_3) \wedge (\bar{x}_5 \vee \bar{x}_7 \vee x_9) \wedge \\
(x_2 \vee \bar{x}_8 \vee x_1) \wedge (\bar{x}_7 \vee x_1 \vee x_5) \wedge (x_1 \vee x_4 \vee x_3) \wedge (x_1 \vee \bar{x}_9 \vee \bar{x}_4) \wedge \\
(x_3 \vee x_5 \vee x_6) \wedge (\bar{x}_6 \vee x_3 \vee \bar{x}_9) \wedge (\bar{x}_7 \vee x_5 \vee x_9) \wedge (x_7 \vee \bar{x}_5 \vee \bar{x}_2) \wedge \\
(x_4 \vee x_7 \vee x_3) \wedge (x_4 \vee \bar{x}_9 \vee \bar{x}_7) \wedge (x_5 \vee \bar{x}_1 \vee x_7) \wedge (x_5 \vee \bar{x}_1 \vee x_7) \wedge \\
(x_6 \vee x_7 \vee \bar{x}_3) \wedge (\bar{x}_8 \vee \bar{x}_6 \vee \bar{x}_7) \wedge (x_6 \vee x_2 \vee x_3) \wedge (\bar{x}_8 \vee x_2 \vee x_5)
\end{array}
$$

Does there exists a truth assignment satisfying all clauses?

Search for a satisfying assignment (or prove none exists)

# SAT Problem

## Satisfiability problem in propositional logic



$$
\begin{array}{l}
(x_5 \lor x_8 \lor \bar{x}_2) \land (x_2 \lor \bar{x}_1 \lor \bar{x}_3) \land (\bar{x}_8 \lor \bar{x}_3 \lor \bar{x}_7) \land (\bar{x}_5 \lor x_3 \lor x_8) \land \\
(\bar{x}_6 \lor \bar{x}_1 \lor \bar{x}_5) \land (x_8 \lor \bar{x}_9 \lor x_3) \land (x_2 \lor x_1 \lor x_3) \land (\bar{x}_1 \lor x_8 \lor x_4) \land \\
(\bar{x}_9 \lor \bar{x}_6 \lor x_8) \land (x_8 \lor x_3 \lor \bar{x}_9) \land (x_9 \lor \bar{x}_3 \lor x_8) \land (x_6 \lor \bar{x}_9 \lor x_5) \land \\
(x_2 \lor \bar{x}_3 \lor \bar{x}_8) \land (x_8 \lor \bar{x}_6 \lor \bar{x}_3) \land (x_8 \lor \bar{x}_3 \lor \bar{x}_1) \land (\bar{x}_8 \lor x_6 \lor \bar{x}_2) \land \\
(x_7 \lor x_9 \lor \bar{x}_2) \land (x_8 \lor \bar{x}_9 \lor x_2) \land (\bar{x}_1 \lor \bar{x}_9 \lor x_4) \land (x_8 \lor x_1 \lor \bar{x}_2) \land \\
(x_3 \lor \bar{x}_4 \lor \bar{x}_6) \land (\bar{x}_1 \lor \bar{x}_7 \lor x_5) \land (\bar{x}_7 \lor x_1 \lor x_6) \land (\bar{x}_5 \lor x_4 \lor \bar{x}_6) \land \\
(\bar{x}_4 \lor x_9 \lor \bar{x}_8) \land (x_2 \lor x_9 \lor x_1) \land (x_5 \lor \bar{x}_7 \lor x_1) \land (\bar{x}_7 \lor \bar{x}_9 \lor \bar{x}_6) \land \\
(x_2 \lor x_5 \lor x_4) \land (x_8 \lor \bar{x}_4 \lor x_5) \land (x_5 \lor x_9 \lor x_3) \land (\bar{x}_5 \lor \bar{x}_7 \lor x_9) \land \\
(x_2 \lor \bar{x}_8 \lor x_1) \land (\bar{x}_7 \lor x_1 \lor x_5) \land (x_1 \lor x_4 \lor x_3) \land (x_1 \lor \bar{x}_9 \lor \bar{x}_4) \land \\
(x_3 \lor x_5 \lor x_6) \land (\bar{x}_6 \lor x_3 \lor \bar{x}_9) \land (\bar{x}_7 \lor x_5 \lor x_9) \land (x_7 \lor \bar{x}_5 \lor \bar{x}_2) \land \\
(x_4 \lor x_7 \lor x_3) \land (x_4 \lor \bar{x}_9 \lor \bar{x}_7) \land (x_5 \lor \bar{x}_1 \lor x_7) \land (x_5 \lor \bar{x}_1 \lor x_7) \land \\
(x_6 \lor x_7 \lor \bar{x}_3) \land (\bar{x}_8 \lor \bar{x}_6 \lor \bar{x}_7) \land (x_6 \lor x_2 \lor x_3) \land (\bar{x}_8 \lor x_2 \lor x_5)
\end{array}
$$

Does there exists a truth assignment satisfying all clauses?

Search for a satisfying assignment (or prove none exists)

# Motivation

- From 100 variables, 200 constraints (early 90s) to 1,000,000 vars. and 20,000,000 cls. in 20 years.
- Applications:
  Hardware and Software Verification, Planning, Scheduling, Optimal Control, Protocol Design, Routing, Combinatorial problems, Equivalence Checking, etc.
- SAT used to solve many other problems!

# SAT Problem
## Satisfiability problem in propositional logic



Definitions:

- Formula in propositional logic: well-formed string that may contain
  - propositional variables $x_1, x_2, \ldots, x_n$;
  - truth values $\top$ ('true'), $\bot$ ('false');
  - operators $\neg$ ('not'), $\wedge$ ('and'), $\vee$ ('or');
  - parentheses (for operator nesting).

# SAT Problem

## Satisfiability problem in propositional logic

Definitions:

- Formula in propositional logic: well-formed string that may contain
  - propositional variables $x_1, x_2, \ldots, x_n$;
  - truth values $\top$ ('true'), $\bot$ ('false');
  - operators $\neg$ ('not'), $\wedge$ ('and'), $\vee$ ('or');
  - parentheses (for operator nesting).

- Model (or satisfying assignment) of a formula $F$: Assignment of truth values to the variables in $F$ under which $F$ becomes true (under the usual interpretation of the logical operators)

# SAT Problem

## Satisfiability problem in propositional logic

Definitions:

- Formula in propositional logic: well-formed string that may contain
  - propositional variables $x_1, x_2, \ldots, x_n$;
  - truth values $\top$ ('true'), $\bot$ ('false');
  - operators $\neg$ ('not'), $\wedge$ ('and'), $\vee$ ('or');
  - parentheses (for operator nesting).
- Model (or satisfying assignment) of a formula $F$: Assignment of truth values to the variables in $F$ under which $F$ becomes true (under the usual interpretation of the logical operators)
- Formula $F$ is satisfiable iff there exists at least one model of $F$, unsatisfiable otherwise.

SAT Problem (decision problem, search variant):

- **Given:** Formula $F$ in propositional logic
- **Task:** Find an assignment of truth values to variables in $F$ that renders $F$ true, or decide that no such assignment exists.

SAT Problem (decision problem, search variant):

- **Given:** Formula $F$ in propositional logic
- **Task:** Find an assignment of truth values to variables in $F$ that renders $F$ true, or decide that no such assignment exists.

SAT: A simple example

- **Given:** Formula $F := (x_1 \vee x_2) \wedge (\neg x_1 \vee \neg x_2)$
- **Task:** Find an assignment of truth values to variables $x_1, x_2$ that renders $F$ true, or decide that no such assignment exists.

Definitions:

- A formula is in conjunctive normal form (CNF) iff it is of the form

$$\bigwedge_{i=1}^{m} \bigvee_{j=1}^{k_i} l_{ij} = (l_{11} \vee \ldots \vee l_{1k_1}) \wedge \ldots \wedge (l_{m1} \vee \ldots \vee l_{mk_m})$$

  where each literal $l_{ij}$ is a propositional variable or its negation. The disjunctions $c_i = (l_{i1} \vee \ldots \vee l_{ik_i})$ are called clauses.

- A formula is in $k$-CNF iff it is in CNF and all clauses contain exactly $k$ literals (*i.e.*, for all $i$, $k_i = k$).

Definitions:

- A formula is in conjunctive normal form (CNF) iff it is of the form

$$\bigwedge_{i=1}^{m} \bigvee_{j=1}^{k_i} l_{ij} = (l_{11} \vee \ldots \vee l_{1k_1}) \wedge \ldots \wedge (l_{m1} \vee \ldots \vee l_{mk_m})$$

  where each literal $l_{ij}$ is a propositional variable or its negation. The disjunctions $c_i = (l_{i1} \vee \ldots \vee l_{ik_i})$ are called clauses.

- A formula is in $k$-CNF iff it is in CNF and all clauses contain exactly $k$ literals (*i.e.*, for all $i$, $k_i = k$).

- In many cases, the restriction of SAT to CNF formulae is considered.
- For every propositional formula, there is an equivalent formula in 3-CNF.

Example:

$$
\begin{aligned}
F := \quad & \wedge (\neg x_2 \vee x_1) \\
& \wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \\
& \wedge (x_1 \vee x_2) \\
& \wedge (\neg x_4 \vee x_3) \\
& \wedge (\neg x_5 \vee x_3)
\end{aligned}
$$

- $F$ is in CNF.
- Is $F$ satisfiable?

### Example:

$$F := \begin{aligned} &\wedge (\neg x_2 \vee x_1) \\ &\wedge (\neg x_1 \vee \neg x_2 \vee \neg x_3) \\ &\wedge (x_1 \vee x_2) \\ &\wedge (\neg x_4 \vee x_3) \\ &\wedge (\neg x_5 \vee x_3) \end{aligned}$$

- $F$ is in CNF.
- Is $F$ satisfiable?
  Yes, *e.g.*, $x_1 := x_2 := \top$, $x_3 := x_4 := x_5 := \bot$ is a model of $F$.

### MAX-SAT (optimization problem)

Which is the maximal number of clauses satisfiable in a propositional logic formula $F$?

# Pre-processing

Pre-processing rules: low polynimial time procedures to decrease the size of the problem instance.

Typically applied in cascade until no rule is effective anymore.

# Examples

1. eliminate duplicate literals
2. tautologies: $x_1 \vee \neg x_1 \ldots$
3. subsumed clauses
4. pure literals
5. unit clauses
6. unit propagation

# Construction heuristics

- Variable selection heuristics
  aim: minimize the search space
  plus: could compensate a bad value selection

# Construction heuristics

- Variable selection heuristics
  aim: minimize the search space
  plus: could compensate a bad value selection

- Value selection heuristics
  aim: guide search towards a solution (or conflict)
  plus: could compensate a bad variable selection

# Construction heuristics

- Variable selection heuristics
  aim: minimize the search space
  plus: could compensate a bad value selection

- Value selection heuristics
  aim: guide search towards a solution (or conflict)
  plus: could compensate a bad variable selection

- Restart strategies
  aim: avoid heavy-tail behavior [GomesSelmanCrato'97]
  plus: focus search on recent conflicts when combined with dynamic heuristics

# Variable selection heuristics

- Maximal Occurrence in clauses of Minimal Size (Jeroslow-Wang)
- Variable State Independent Decaying Sum (VSIDS) original idea (zChaff): for each conflict, increase the score of involved variables by 1, half all scores each 256 conflicts [MoskewiczMZZM2001]
- improvement (MiniSAT): for each conflict, increase the score of involved variables by $\delta$ and increase $\delta := 1.05\delta$ [EenS''orensson2003]

# Value selection heuristics

- Based on the occurrences in the (reduced) formula
  examples: Jeroslow-Wang, Maximal Occurrence in clauses of Minimal Size (MOMS), look-aheads

- Based on the encoding / consequently
  negative branching (early MiniSAT)

- Based on the last implied value (phase-saving)