heuristics to improve the performance of SAT solver

Lazy datastructure

- 2 watched literals
- Pure literals

Runtime choices

- Variable ordering ✓
- Restarts
- Learned clause deletion
- Phase saving.

Optimal storage

- variables
- clauses
- occurrence maps

Pre/In processing techniques

Variable ordering, Decision heuristics, Branching heuristics

1. # of variables occurrences in remaining unsatisfied clauses
   - ↦ different variants were studied in 90's

2. Dynamic heuristics
   - → focus on variables which were useful recently in deriving learned clauses.
   - → can be interpreted as reinforcement learning
   - → VSIDS : Variable State Independent Decaying Sum
     - ↦ different variants were studied

3. Look-ahead
   - ↦ spent more time in selecting good variables.

DLIS (Dynamic Largest Individual Sum)

- for any variable v:
  - $C_{v,P}$: # of unresolved (unsatisfied) clause in which v appears positively.
  - $C_{v,N}$: # of unresolved (unsatisfied) clause in which v appears negatively.
  - let 'a' be the literal for which $C_{a,P}$ is maximal
  - let 'b' be the literal for which $C_{b,N}$ is maximal
  - if $C_{a,P} > C_{b,N}$ then choose a and set to 1 else choose b & set to 0.

Implemented in Grasp

Requires # literals queries for each decision.

Jeroslow-Wang Method

- → Gives an exponentially higher weight to literals in shorter clauses.
- → for every literal l :-

$$J(l) = \sum_{l \in C_i,\, C \in F} 2^{-|C_i|}$$

MOM (Maximum Occurrence of clauses of Minimum size)

- decide a number w, such that if $|C| < w$ then clause C is considered to be the small.
- let $f^*(x)$ be the # of small clauses containing x. Choose x that maximizes.

$$(f^*(x) + f^*(\neg x)) \times 2^k + f^*(x) \times f^*(\neg x)$$

- k is choose heuristically.

- Give preference to satisfying small clauses.
- Among those, give preference to Balanced variables.
  - $f^*(x) = 3$ & $f^*(\neg x) = 3$ is preferred over $f^*(x) = 1$ & $f^*(\neg x) = 5$.

Variable State Independent Decaying Sum (VSIDS)

Implemented in Chaff.

- → Each literal (l) has a counter $S(l)$, initialized to zero.
- → For every new clause $C = [l_1, l_2, \ldots, l_n]$, $S(l_i)$ is incremented.
- → The unassigned variables of polarity with highest counter is choosen
- → Ties are broken randomly
- → Periodically (once in 256 conflict), all counter are halved.
  - 256 → can change

VSIDS example :-

Heuristic	Related data
literal	score
a	4
¬a	5
b	3
¬b	3
c	2
¬c	3
d	2
¬d	4
e	2
¬e	6
⋮	⋮

initial value; occurrences of 'a' in Formula F.

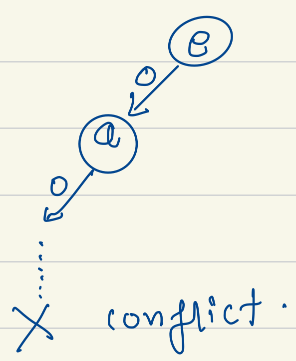
count literal appearances in formula F.

VSIDS example :-

Heuristic	Related data
literal	score
a	4
$\neg a$	5
b	3
$\neg b$	3
c	2
$\neg c$	3
d	2
$\neg d$	4
e	2
$\neg e$	6
⋮	⋮

initial value; occurrences of 'a' in Formula F.

count literal appearances in formula F.



$(\neg h \vee a \vee c \vee \neg b \vee k)$

Conflict clauses

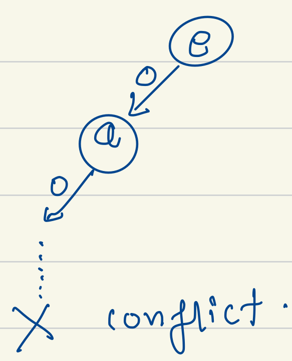
VSIDS example :-

#of conflict = 1

Heuristic	Related data
literal	score
a	4 +1
$\neg a$	5
b	3
$\neg b$	3 +1
c	2 +1
$\neg c$	3
d	2
$\neg d$	4
e	2
$\neg e$	6
⋮	⋮ +1 ⋮

initial value; occurrences of 'a' in Formula F.

count literal appearances in formula F.



$(\neg h \vee a \vee c \vee \neg b \vee k)$

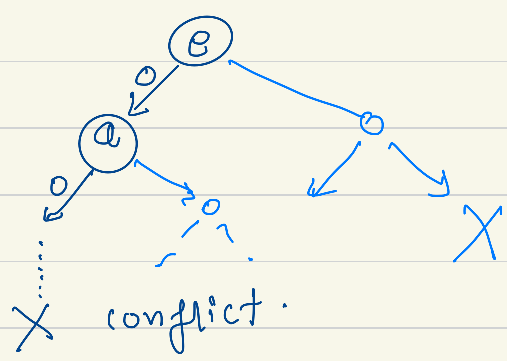
Conflict clauses

VSIDS example :-

Heuristic literal	Related data score
a	$10 \to 5$
$\neg a$	$12 \to 6$
b	$18 \to 9$
$\neg b$	$6 \to 3$
c	$12 \to 6$
$\neg c$	$11 \to 5$
d	$6 \to 3$
$\neg d$	$2 \to 1$
e	$16 \to 8$
$\neg e$	$6 \to 3$
⋮	⋮

# of conflict = 256

Reset the counter



conflict .

$(\neg h \vee a \vee c \vee \neg b \vee k)$

Conflict clauses

Why VSIDS was a breakthrough?

→ Pre-chaff static heuristics
  → Go over all clauses that are not satisfied & compute some function f(a) for each literal "a".

→ VSIDS
  → extremely low overhead
  → dynamic & local
    → conflict driven
  → focuses the search to learn from the local content.

EVSIDS (Exponential VSIDS '03)

→ dynamically adjust increment : $\delta' = \delta \cdot 1/f$

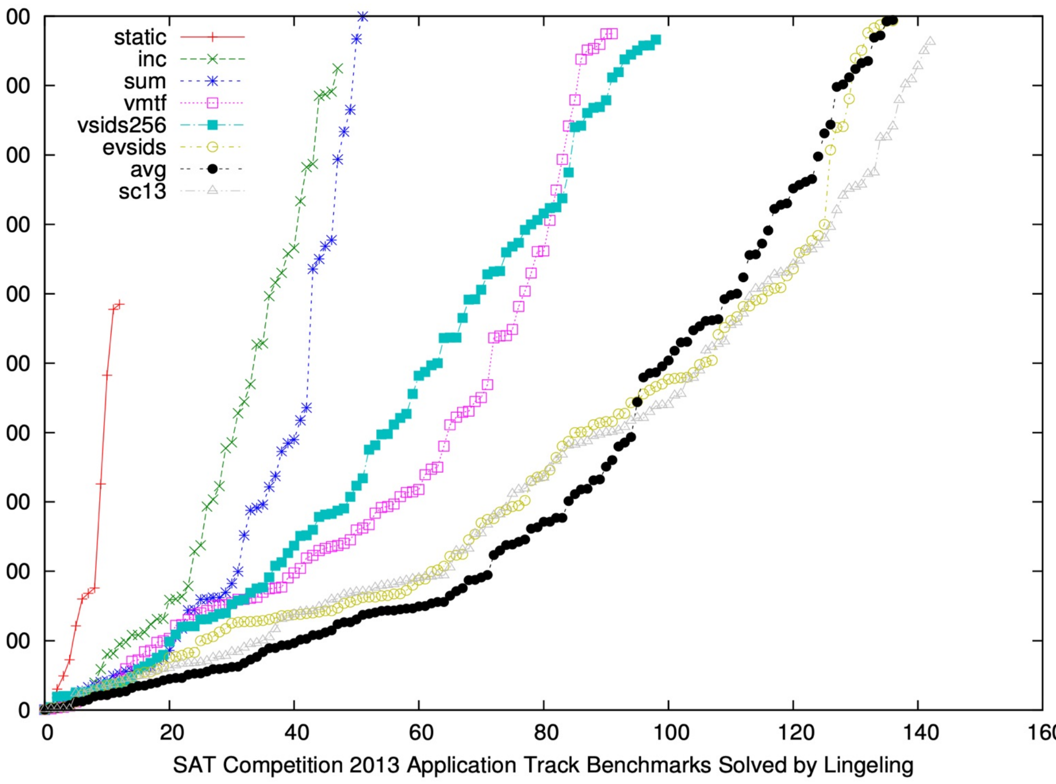
δ : is score of a literal

f : need to be choosen

typically 0.95

→ Rescale when score for any variables becomes higher than 10^{100}.

MiniSAT uses EVSIDS



static
inc
sum
vmtf
vsids256
evsids
avg
sc13
00
0
0
20
40
60
80
100
120
140
160
SAT Competition 2013 Application Track Benchmarks Solved by Lingeling

Learned clause deletion

- → CDCL may learn a lot of clauses.
- → In terms of storage solver needs to delete some clauses periodically.
  - → How does it effect the soundness & completeness of a CDCL based SAT solver?

clause deletion

- which clause to delete?
  - delete longer clauses with higher prob.
  - never delete unit clause
  - never delete "active clause", clauses which are participating in unit propagation.
- when to delete a clause?
  - At restart
  - if # of learned clause = predefine threshold.

clause deletion

- → MiniSAT reduce (deletes) half of the clauses.
- → keep the most active, then shortest, then youngest (FIFO) clauses.

Restarts :-

→ SAT solvers are likely to get stuck in a local search space.

→ restart CDCL with a different variable ordering.

→ keep learned clauses across restart.

→ slowly increase the intervals of restarts such that tools becomes a complete solver.

→ Usually depends on # of conflict clauses or # of decision levels.

Phase Saving & Rapid Restarts

→ polarity of a variable.

"phase saving" → pick the phase of last assignment
→ if not forced, don't change.

rapid restarts :- theoretically shown that it avoids local minima

→ practically works well with phase-saving.

Pre (in) Processing

- Eliminate tautologies / unit clauses / Pure literal elimination.
- Subsumption / Self-subsuming resolution.
- Blocked clause elimination.
- literal equivalence.
- Bounded variable addition / Elimination
-

first used in kissat

Blocked clause Elimination (BCE)

	one clause $C \in F$ with l.	all clauses with $\neg l$
F formula	$a \vee b \vee l$	$\neg l \vee \neg a \vee c$
		$\neg l \vee \neg b \vee d$

Resolution with C, all the resolvents of C on l.
tautological

BCE :

A clause $C \in F$ is a blocked clause in F, if there is literal $\ell \in C$ such that for each $C' \in F$ with $\neg \ell \in C'$, the resolvent $(C \setminus \{\ell\} \cup C' \setminus \{\neg \ell\})$ obtained from resolving C & C' on ℓ is a tautology.

$$F = (a \vee b) \wedge (a \vee \neg b \vee \neg c) \wedge (\neg a \vee c)$$

$F = (a \vee b) \wedge (a \vee \neg b \vee \neg c) \wedge (\neg a \vee c)$

1st clause
- $a \longrightarrow c_1 \,\&\, c_3 \longrightarrow b \vee c$ (not tautology)
- $b \rightarrow c_1 \,\&\, c_2 \rightarrow a \vee \neg c$ (not tautology)

2nd clause
- $a \longrightarrow c_2 \,\&\, c_3 \longrightarrow (\neg b \vee c \vee \neg c)$ tautology
- $b \rightarrow c_1 \,\&\, c_2 \rightarrow (a \vee \neg c)$ not tautology
- $c \rightarrow c_2 \,\&\, c_3 \rightarrow (a \vee \neg a \vee \neg b)$ tautology

3rd clause
- $\neg a$
  - $c_1 \,\&\, c_3$ $b \vee c$ $\rightarrow$ not tautology
  - $c_2 \,\&\, c_3$ $\neg b \vee c \vee \neg c$ $\rightarrow$ tautology.
- $c \rightarrow c_2 \,\&\, c_3$ $(a \vee \neg a \vee \neg b)$ tautology

$F = (a \vee b) \wedge (a \vee \neg b \vee \neg c) \wedge (\neg a \vee c)$

→ only first clause is not blocked

→ second clause has two blocked literals
  └ a & ¬c

→ third clause has c has blocked literals.

$F = (a \vee b) \wedge (a \vee \neg b \vee \neg c)$

Now, all clauses are blocked, hence all clauses can be removed

SAT solving is algorithm, science or art

* Current theoretical understanding is limited

need to run experiments to measure preformance.

what work?
why works?