CS 433 Automated Reasoning 2024

Lecture 8: CDCL - optimizations

Instructor: Ashutosh Gupta

IITB India

Compile date: 2024-02-08

More heuristics

More heuristics that may improve the performance of SAT solvers

- Runtime choices
  1. Variable ordering
  2. Restarts
  3. Learned clause deletion
  4. Phase saving
- Pre/In-processing

Topic 8.1

Runtime choices

Runtime choices

Let us study the following runtime choices available to the solvers

1. Variable ordering
2. Restarts
3. Learned clause deletion
4. Phase saving

Decision ordering

After each backtrack, we may choose a different order of assignment.

There are many proposed strategies for the decision order.

Desired property: allow different order after backtracking and less overhead

The following are two widely used strategies:

1. Select a literal with maximum occurrences in unassigned clauses
2. Variable state independent decaying sum

Very popular

Exercise 8.1

What is the policy in Z3?

Variable state independent decaying sum(VSIDS)

Each literal has a score. The highest-scored unassigned literal is the next decision, tie is broken randomly

- Initial score is the number of occurrences of the literals
- Score of a literal is incremented whenever a learned clause contains it
- In regular intervals, divide the scores by a constant (loop over all the variables)

decay

VSIDS is almost deterministic. Some solvers occasionally make random decisions to get out of potential local trap.

Exercise 8.2

Find the used decay rate, increment value, and the interval of update of scores in a solver.

Commentary: Variable state independent decaying sum gives greater weight to the occurrence in the later learned clauses. In some implementations, the weights of variables that appear in the conflict graph after the cut are also incremented.

VSIDS is effective

The principle of exploitation and exploration

- exploitation : decide literals that have participated in conflicts (local search)
- exploration : due to the decay, move on to search somewhere else (global search)

Exercise 8.3

Can we reduce effort of resizing scores of all the variables?

Commentary: Variants of VSIDS: [EènS örensson'03/'06]
Following the similar principal: Liang, J.H., Ganesh, V., Poupart, P., and Czarnecki, K. Conflict-history Based Branching Heuristic for SAT Solvers. AAAI 2016

Restart

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

Solution: restart CDCL with a different variable ordering

- Keep learned clauses across restarts
- Increase the interval of restarts such that tool becomes a complete solver
- To avoid trap of long restarts, we need a strategy to keep having short restarts

Exercise 8.4
Suggest a design of a parallel sat solver.

Heavy tail restarts

Heavy tail behavior : often short restarts, but with significant chances of long restarts

Example 8.1

Let us consider a well know strategy called Luby restart [LubySinclairZuckerman93].

Let u be a unit number of conflicts. At ith run, we restart after $t_i u$, where

$$t_i = \begin{cases} 2^{k-1}, & \text{if } i = 2^k - 1 \\ t_{i-2^{k-1}+1}, & \text{if } 2^{k-1} \leq i < 2^k - 1. \end{cases}$$

Exercise 8.5

a. Plot t_i for first 70 points.

b. Show that v_n in the following reluctant doubling sequence is equal to t_n.[Knuth'12]

$(u1, v1) = (1, 1)$ *and* $(u_{n+1}, v_{n+1}) = (u_n \& - u_n) = v_n ? (u_n + 1, 1) : (u_n, 2v_n)$

Learned clause deletion

CDCL may learn a lot of clauses.

The solvers time to time delete some learned clauses.

The solver remains sound with deletions. However, the completeness may be compromised.

For completeness, reduce deletion of clauses over time.

Exercise 8.6
After learning how many clauses, we should start deleting?

(estimate via common sense; Imaging yourself in an interview!!!)

`https://arxiv.org/pdf/1402.1956.pdf` Gluecose `http://www.ijcai.org/Proceedings/09/Papers/074.pdf`

Deletion strategies

A solver may adopt a combination of the following choices.

Which clauses to delete?

- Delete long clauses with higher probability
- Never delete binary clauses
- Never delete active clauses, i.e., are participating in unit propagation

When to delete?

- At restart
- After crossing a threshold of number of the learned clauses; clauses involved in unit propagation can not be deleted

Deletion measure for clauses : Literal block distance

Definition 8.1

Literal block distance(LBD) = number of decisions in a learned clause

Larger LBD implies more likely to be deleted

LBD is a popular technique

Phase saving

- During CDCL run, the partial assignments satisfies a part of formula
- After restarts, we may want to use the same last partial assignment
- We save the last assigned phase of a variable. In future decisions, we use the same phase.
- Works well with rapid restarts

SAT solving: algorithm, science, or art

Algorithm:
We can not predict the impact of the optimizations based on theory. The current theoretical understanding is limited.

Science:
We need to run experiments to measure the performance.

Art:
Only SAT solving elders can tell you what strategy of solving is going to work on a new instance.

Latest trends in SAT solving

- Portfolio solvers
- Machine learned solver configuration
- Optimizations for applications, e.g., maxsat, unsatcore, etc.
- solving cryptography constraints

Exercise 8.7

Visit the latest SAT conference website. Read a paper and write a comment(400 chars max).

Topic 8.2

Pre(in)-processing

Pre(in)-processing

Simplify input before CDCL

- Eliminate tautologies/Unit clauses/Pure literal elimination
- Subsumption/Self-subsuming resolution
- Blocked clause elimination
- Literal equivalence
- Bounded variable elimination/addition
- Failed literal probing/Vivification
- Stamping
- ....

http://theory.stanford.edu/~barrett/summerschool/soos.pdf

Source of Lingeling (http://fmv.jku.at/lingeling/)

Obvious eliminations

- Eliminate tautologies
  - Remove clauses like $p_1 \vee \neg p_1 \vee ....$
- Assign unit clauses
  - Unit propagation at 0th decision level.
- Pure literal elimination
  - Remove all the clauses that contain the literal

Exercise 8.8

a. *What is the cost of eliminating tautologies?*

b. *What is the cost of pure literal elimination?*

Commentary: Sorted clause make tautology detection efficient. Pre-computing of occurrence while parsing helps identifying pure literals.

Subsumption

Remove clause C' if $C \subset C'$ is present.

- Use backward subsumption: for a C search for weaker clauses
- Only search using short C
- Iterate over the occurrence list of the literal in C that has the smallest occur size.
- Containment check is sped up using bloom filter.

Example 8.2

$p \vee q \vee r$ is a redundant clause if $p \vee q$ is present.

Subsumption algorithm

The fingerprint used in Lingeling for Bloom filter.

$$fingerPrint(C) = |_{\ell \in C}(1 << (atom(\ell)\&31))$$

$atom(\ell)$ returns the atom in literal ℓ.

Algorithm 8.1: Subsumption(F)

```
for $C \in F$ such that $|C| < shortLimit$ do
    $sigC := fingerPrint(C)$;
    $\ell :=$ literal in $C$ with smallest $|OccurList(\ell)|$;
    for $C' \in OccurList(\ell)$ such that $C' \neq C$ do
        if $sigC$ ?? $fingerPrint(C')$ then
            if $C \subset C'$ then
                $F := (F - \{C'\})$;
```

Exercise 8.9

Complete the missing operator '??'.

Self-subsumption (Strengthening)

Replace clause $C' \vee \ell$ by C' if for some $C \subset C'$, $C \vee \neg\ell$ is present.

Example 8.3

$p \vee q \vee r \vee \neg s$ should be replaced by $p \vee q \vee r$ if $r \vee s$ is present.

Algorithm 8.2: SelfSubsumption(F)

```
for $C \in F$ such that $|C| < shortLimit$ do
    $sigC := fingerPrint(C)$;
    $\ell :=$ literal in $C$ with smallest $|OccurList(\ell) \cup OccurList(\neg\ell)|$;
    for $C' \in OccurList(\ell) \cup OccurList(\neg\ell)$ such that $C' \neq C$ do
        if $sigC$ ?? $fingerPrint(C')$ then
            if $D' \vee \neg\ell' = C'$ and $D \vee \ell' = C$ and $D \subset D'$ then
                $F := (F - \{C'\}) \cup D'$;
```

Commentary: Same answer for ?? as in the previous slide.

Blocked clause elimination

Now, we will look at a more general condition than pure literal to remove clauses.

Definition 8.2

called blocking literal

A clause $C \in F$ is a blocked clause in F, if there is a literal $\ell \in C$ such that for each $C' \in F$ with $\neg\ell \in C'$, there is a literal ℓ' such that $\ell' \in C$ and $\neg\ell' \in C' \setminus \{\neg\ell\}$.

Claim:

We can safely disable blocked clauses, without affecting satisfiability.

Example: blocked clause elimination

Example 8.4

In the following clauses, p_1 is a blocking literal in the blocking clause C_1.

$C_1 = (p_1 \lor p_2 \lor \neg p_3) \land$
$C_2 = (\neg p_3 \lor \neg p_2) \land$
$C_3 = (\neg p_1 \lor \neg p_2) \land$
$C_4 = (p_1 \lor \neg p_5) \land$
$C_5 = (\neg p_1 \lor p_3 \lor p_4)$

Only, C_3 and C_5 contain $\neg p_1$.

$p_3 \in C_5$ is helping p_1 to become blocked literal in C_1, since negation of p_3 is present in C_1.

Exercise 8.10

Which literal in C_3 helping p_1 to become blocked literal in C_1?

Soundness of blocking clause elimination

Theorem 8.1

If C is a blocking clause in F, then F and $F \setminus C$ are equisatisfiable.

Proof.

Wlog, let $C = \ell_1 \vee \cdots \vee \ell_k$ and ℓ_1 be the blocking literal.
Let us suppose $m \models F \setminus C$ and $m \not\models C$, otherwise proof is trivial.
Therefore, $m(\ell_i) = 0$.

Claim: $m[\ell_1 \mapsto 1] \models F$
Choose $C' \in F$. Now three cases.

1. $\neg\ell_1 \in C'$: there is ℓ_i for $i > 1$(Why?) such that $\ell_i \in C$ and $\neg\ell_i \in C'$.
   Since $m(\ell_i) = 0$, $m[\ell_1 \mapsto 1] \models C'$.(Why?)
2. $\ell_1 \in C'$: Since $m[\ell_1 \mapsto 1] \models C'$, $m[\ell_1 \mapsto 1] \models C'$.
3. $\{\ell_1, \neg\ell_1\} \cap C' = \emptyset$: trivial.(Why?) $\square$

Commentary: A näive implementation will be inefficient. To find an efficient implementation, read `http://fmv.jku.at/papers/JarvisaloBiereHeule-TACAS10.pdf`

Vivification

Let us suppose $C = (\ell_1 \vee \ldots \vee \ell_n) \in F$ and

$$\textsc{UnitPropagation}(\emptyset, F \wedge \neg\ell_1 \wedge \cdots \wedge \neg\ell_{i-1} \wedge \neg\ell_{i+1} \wedge \cdots \wedge \neg\ell_n)$$

results in conflict.

We replace C in F by $C \setminus \{\ell_{i-1}\}$.

Implemented in many state of the art solvers.

Compaction

The pre-processing changes the set of variables and clauses.

Before running CDCL,

- the solvers rename all the variables with contiguous numbers and
- clause lists are also compacted.

This increases cache locality, and fewer cache misses.

Topic 8.3

Problems

SAT solver for the following problems

- Please download the following SAT solver

  `https://github.com/arminbiere/cadical`

- Install the solver as instructed in the source code.
- Benchmarks: Download 400 main Track instances from:

  `https://satcompetition.github.io/2021/downloads.html`

- The goal of the following problems would be to change A PARAMETER regarding certain optimization and draw the cactus plot for various choices of the value of the parameter. There may be multiple parameters for some optimization. Choose one that makes most sense to you. Please write one to two paragraph report to explain your results. (`-h` option will give you descriptions of the parameters.)

Play with exponential VSIDS (EVSIDS)

Exercise 8.11

Please modify the following parameters related to EVSIDS and draw cactus plot.

```
    score, scorefactor
```

Please follow the instructions at the start of problem section to do the above exercise.

Play with restarts

Exercise 8.12

Please modify the following parameters related to restarts and draw cactus plot.

```
    restart, restartint, restartmargin, restartreusetrail,
    reluctant, reluctantmax
```

Please follow the instructions at the start of problem section to do the above exercise.

Play with clause deletion

Exercise 8.13
Please modify the following parameters related to clause deletion and draw cactus plot.

```
reduce, reduceint, reducetarget, reducetier1glue, reducetier2glue,
emagluefast, emaglueslow
```

Please follow the instructions at the start of problem section to do the above exercise.

Play with phase saving

Exercise 8.14

Please modify the following parameters related to phase saving and draw cactus plot.

```
    forcephase, phase, rephase, rephaseint,
    stabilize, stabilizefactor, stabilizeint, stabilizemaxint,
    stabilizeonly, stabilizephase,
```

Please follow the instructions at the start of problem section to do the above exercise.

Play with chronological backtracking (Clause learning)

Exercise 8.15

Please modify the following parameters related to chronological backtracking and draw the cactus plot.

```
chrono, chronoalways, chronolevelim, chronoreusetrail,
```

Please follow the instructions at the start of problem section to do the above exercise.

Topic 8.4

More problems

Play with vivifications

Exercise 8.16

Please modify the following parameters related to vivification and draw cactus plot.

```
vivify, vivifymaxeff, vivifymineff, vivifyonce, vivifyredeff, vivifyreleff
```

Please follow the instructions at the start of problem section to do the above exercise.

Play with failed literal probing

Exercise 8.17

Please modify the following parameters related to failed literal probing and draw the cactus plot.

```
probe, probehbr, probeint, probemaxeff, probemineff, probereleff, proberounds
```

Please follow the instructions at the start of problem section to do the above exercise.

Play with blocked clause elimination

Exercise 8.18

Please modify the following parameters related to blocked clause elimination and draw the cactus plot.

```
    block, blockmaxclslim, blockminclslim, blockocclim,
```

Please follow the instructions at the start of problem section lecture to do the above exercise.

End of Lecture 8