Preliminary Results on Exploration-Driven Satisfiability Solving

Md Solimul Chowdhury, Martin Müller and Jia-Huai You

Department of Computing Science, The University of Alberta, Edmonton, Alberta, Canada.

Introduction

General Issues with Heuristics Search

- In heuristics search, heuristics are often inaccurate.
  - Can lead to **early mistakes** → **inefficient search.**
- Exploration can ***compensate*** → ***more robust search***.
- Successful Application of Exploration: Game-playing and Deterministic planning.

Contributions:

- study exploration of SAT search space via random sampling → a novel CDCL SAT solver named ***expSAT***.
- *expSAT* uses intermittent exploration of the search space to find ***valuable statistics*** to guide the search.
- *expSAT* uses a novel branching heuristic ***expVSIDS***.
- *expVSDIS* ***combines*** standard VSIDS score and exploration score from expSAT statistics.
- Empirical evaluation shows ***performance gain***.

Background

The Satisfiability Problem

A SAT formula: Conjunction of ***clauses***

- a clause is disjunction of Boolean ***literals***,
- a literal is either a ***variable or its negation***.

The SAT problem: Given a formula of Boolean variables, is there an ***assignments of those variables,*** which satisfies that formula?

Satisfiability (SAT) Solving

- Modern SAT solvers employ ***tree search*** to solve a given formula.
  - Branch → Unit Propagation → Branch → …
  - Branch → Unit Propagation → Conflict → Backtrack → Branch → Unit Propagation →…
- Dominant SAT solvers are ***Conflict Driven Clause Learning (CDCL)***
  - branching heuristics, conflict analysis, clause learning , back jumping.

Branching Heuristics

- ***Which variable to selec***t from the set of unassigned variables?
- Dominant CDCL branching heuristics: **VSIDS** (and recently LRB)
- VSIDS:
  - Select variable, which ***participated in recent conflicts***.

Clause Learning

- If unit propagation derives ***an empty clause***, then a conflict occurs.
- ***conflict analysis*** identifies the ***root cause*** of that conflict and ***learns*** a clause
  - a learned clause prunes the future search space.

LBD Score and Goal of Exploration

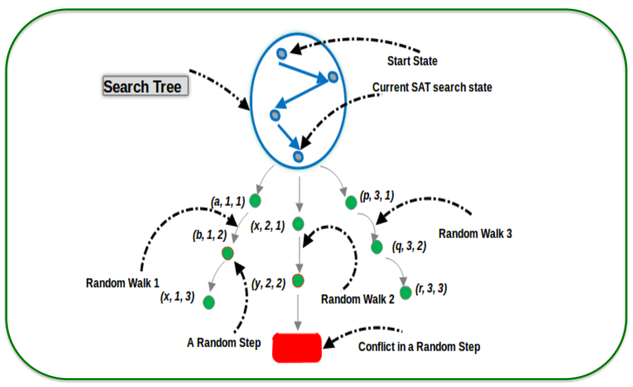
- **Literal Block Distance (LBD)** score of a ***learned clause c***.
  - Number of ***distinct decision*** levels in c.
  - ***Lower*** LBD score means ***better quality*** clause.
- **The goal of exploration** in expSAT is to identify branching variables, which quickly lead to *high-quality* conflicts.
  - expVSIDS exploration scores are derived from ***sampling future search states***, not from the tree.
  - expVSIDS ***rewards*** a variable based ***on the quality of conflicts*** it generates during sampling.
  - It ***favors*** variables assignment of which lead to generation of ***high-quality clauses*** with low LBD in sampling.

The *expSAT* Solver

Random Exploration and Statistics Collection

- In upper part of the search tree, before ***some branching decisions***, with a probability p_{exp}, *expSAT* performs ***an exploration episode***.
- Exploration consists of fixed (***lW)*** number of ***random walks***.
- Random Walk consists of ***random steps***.
  - Random step → random selection of variable + UP
- A Walk ***terminates***, if ***mS*** random steps are taken or a conflict occurs.
- ***Statistics*** for walks that terminate with a conflict :

  LBD of the learned from that conflict



Computation of Exploration Score

- Exploration score ***expScore*** is computed for each variable that participated in the last exploration episode.
- *expScore* of v is the ***average of the walk scores*** *ws(v)* of the walks, where v appears as step variable.
- *ws(v)* is computed as follows:

> ***ws(v)* = 0, if the walk ended without a conflict.**
> **Otherwise, *ws(v) = ω^d / lbd(c),***
> - **d >= 0 is the decision distance**
> - **Example: d=1 for the node (x,2,2)**
> - ***ω < 1* is the exponential decay factor.**

expVSIDS:

- ***expVSIDS***: new heuristics, combines *VSIDS score* and *expScore* of v.

> **expVSIDS scroe of v = g^z * expScore(v) + VSIDS(v).**

- For branching, selects v^* with ***highest expVSIDS score***.

Empirical Evaluation

Implementation:

- Two prototype solver implementations:
  - ***MiniSAT → expMiniSAT***
  - ***Glucose → expGlucose***

Setting Exploration Parameters:

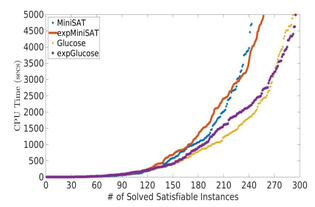
- Parameter Values:

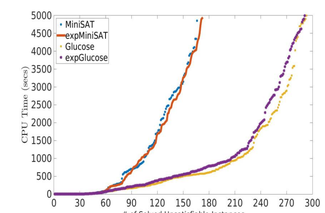
Parameters	Values
nW	5,7,8,10,15
lW	4,5
θ_{stop}	0.5, 0.6, 0.7, 0.8,0.9
p_{exp}	2, 3, 4, 5, 6
ω	0.81,0.90

- Parameters are obtained from experiments with ***a small subset of satisfiable instances*** from SATRACE -2015.
- Constitute a set of ***500 parameter settings*** for these parameter values.
- We have selected ***20 parameter settings out of 500*** parameter settings to run experiment.
- **755 application** instances from SAT Competition-2014, SATRACE-2015 and SAT Competition-2016.

Experimental Results:

- ***expMiniSAT*** solves 21 (+5.31%) more instances than ***MiniSAT***.





- ***expGlucose*** performs par with ***Glucose***.
  - Overall, expGlucose
    (I) Solves 1 (-0.17%) less Instance.
    (ii) Scales well for harder instances.

Discussion and Future Work

Discussion:

- Performance is dependent on ***parameter settings***.
- ***Faster conflict generation rate*** with expVSIDS
- ***More*** improvement on ***satisfiable instances*** than on unsatisfiable instances.

Future Work:

- ***Make parameter settings adaptive*** during the search.
- Uniform random selection ➔ use other methods, such ***soft-max method for step variable*** selection.