Guiding CDCL SAT Search via Random Exploration amid Conflict Depression

Md Solimul Chowdhury Martin Müller Jia-Huai You

Department of Computing Science, The University of Alberta.

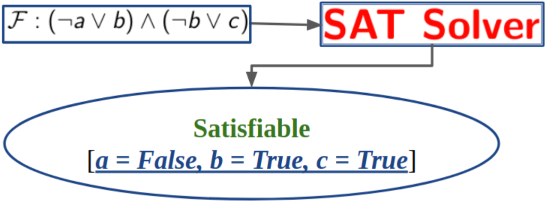
February 4, 2020

Outline

1. Introduction
2. Background
3. Empirical Observations
4. Proposed Framework
5. Empirical Evaluation
6. Analysis
7. Conclusions and Future Work

Introduction

- Boolean Satisfiability (SAT)
  - Given a SAT formula, determine assignments of the variables to satisfy a boolean formula, if one exists. Otherwise, unsatisfiable ....
    - A toy example:


$\mathcal{F} : (\neg a \vee b) \wedge (\neg b \vee c)$
SAT Solver
Satisfiable
[a = False, b = True, c = True]

- However, SAT solving is **NP-Complete** → Intractable, in general.
- Modern SAT solvers → Conflict Directed Clause Learning (CDCL).
  - **Applications in many practical domains**: Hardware design verification, Software testing, encryption, planning ..

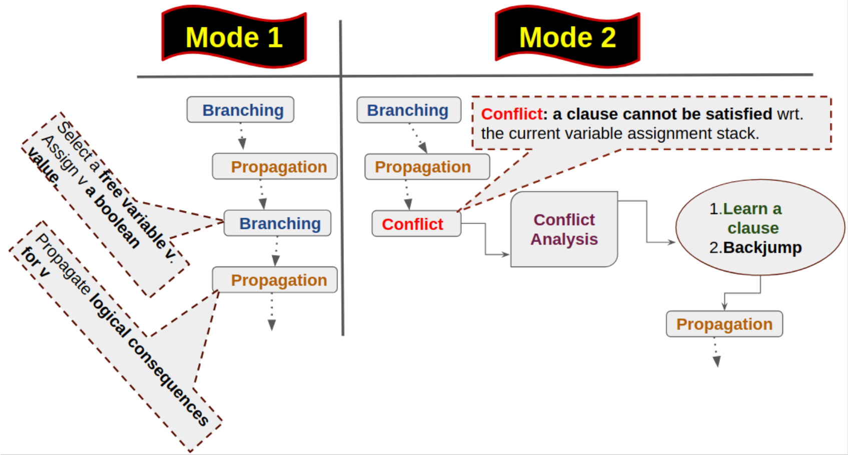
Contributions

In this work, we present

- **Contribution 1:** An empirical investigation of the CDCL SAT solving process to obtain insights on its conflict generation pattern.
- **Contribution 2:** A CDCL algorithmic extension, based on the obtained insights.
    - Employs random exploration, which is novel for CDCL SAT solving!
- **Contribution 3:** An extensive evaluation.

Background: How does a CDCL SAT solver works?

- Performs a backjumping tree-search to determine satisfiabilty.


Mode 1
Mode 2
Branching
Propagation
Branching
Propagation
Select a free variable v.
Assign v a boolean value.
Propagate logical consequences for v
Branching
Propagation
Conflict
Conflict: a clause cannot be satisfied wrt. the current variable assignment stack.
Conflict Analysis
1.Learn a clause
2.Backjump
Propagation

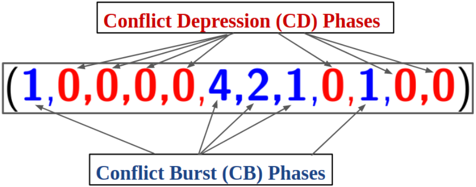
- **Restarts frequently**: abandons the current partial assignment and starts the search from the scratch.

Background: Importance of Fast Conflict Generation

- Conflict Generation at a **fast rate is crucial** for CDCL SAT solvers.
- **Conflict** ⟶ **Clause learning** ⟶ **Pruning** ⟶ **Faster Solving.**
- CDCL branching heuristics are **conflict-greedy**.
  - Example:
    Variable State Independent Decaying Sum (VSIDS)
    Learning Rate Based (LRB)
    - based on **look-back principle**
    - selection priority is based on **recent conflict involvements**.
    - **intuition**: such selection will generate more conflicts.
- CDCL branching heuristics **generate conflicts at a fast rate**.
  - On average, <u>1 conflict in 2 decisions</u>. (Liang 2017 et. al.)

Notions and Definitions

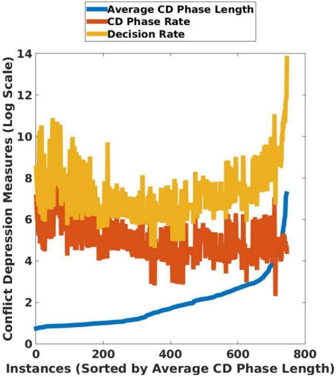
- We formulate **two novel notions:**
  - **Conflict Depression**: Sequence of one or more consecutive decisions with no conflict.
  - **Conflict Burst**: Sequence of one or more consecutive decisions with at least one conflict.

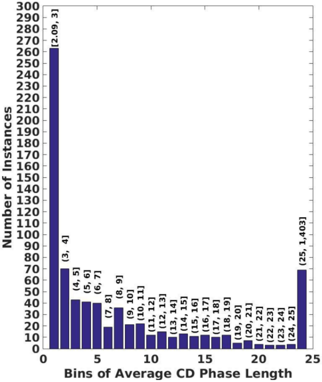


- Some Measures:
  - **Decision Rate**: number of decisions per restart.
  - **CD phase Rate**: number of CD phases per restarts.
  - **Average CD phase Length**

Contribution 1: CD phases with VSIDS

- We study CD phases with VSIDS.
  - CDCL solver: Glucose (uses VSIDS exclusively).
  - 750 maintrack instances from SAT-2017, 2018.
- Observations:
  - **1:** CD phases occur frequently with VSIDS. (left)
  - **2:** For many instances, avg. CD phase length are high! (right)





Propagation Depression during a CD phase

- During a CD phase, VSIDS decisions are ineffective to create conflicts and **only create truth value propagation**.
  - But, how much propagation ?

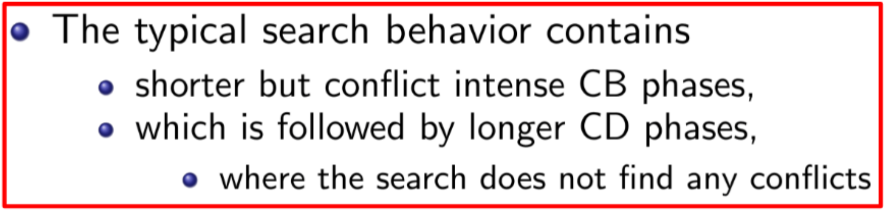
Observation 3:

- Propagtions in a CD phase is **10 times lower** than Propagations in a CB phase.
  ⟶ During a CD phase, VSIDS branching decisions **go through a propagation depression** as well !

Bursts of Conflict Generation

- **Observation 4:** On average,
  - Only 25% of the decisions produce at least one conflict.
    - Some of which produces **more than one conflicts.** How many of them?
  - 61% of the total conflict producing decisions, produces more than one conflict.
- **Conflict Burst (CB)** phases are **short**, but **conflict intense**.
  - Many conflicts within a short span of consecutive decisions.

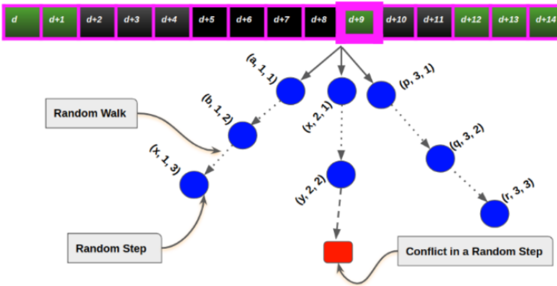
Summary of Empirical Observations



- The typical search behavior contains
  - shorter but conflict intense CB phases,
  - which is followed by longer CD phases,
    - where the search does not find any conflicts

Contribution 2: Random Exploration amid CD phases

- Can we do better amid a CD phase? VSIDS variable re-ranking?
- Formulated an exploration based CDCL framework named *expSAT*.
- **Main Idea:** Amid a substantial CD phase, **with a non-zero probability** ,
    - perform **exploration episodes** to identify conflict friendly variables.
    - Exploration Episode: a fixed number of **random walks**, with a fixed number of **steps per walk**.



Random Walk

Random Step

Conflict in a Random Step

Step Variables	Exploration Score
a	0
b	0
x	**bonus**
y	**bonus**
p	0
q	0
r	0

- **expVSIDS**: VSIDS+exploration score.
    - Selects the variable that maximizes the combined score.

Contribution 3: Empirical Evaluation

- Implemented expSAT on top of 4 CDCL solvers:
    - **glucose 4.2.1** (gLCM)
    - **MapleCOMSPS** (MplCOMSPS)
    - **Maple_CM** (MplCM)
    - **MapleLCMDist_ChronoBT** (MplCBT)
- Two test sets:
    - **Competition Benches**: 750 maintrack instances from SAT-2017 and 2018.
        - Time Out: 5,000 seconds
    - **SATCoin Benches**: 52 hard SATCoin benchmark instances generated by an instance generator submitted for SAT Competition 2018.
        - Time Out: 36,000 secs

Experimental Results

- Competition Benches Results: **Good-to-Strong** gains.

Solver Name	**Solved by Baseline**	**Solved by expSAT Extension**	**PAR2 Score Decrement**
gLCM	372	**379 (+7)**	1.59%
MplCOMSPS	412	**428 (+16)**	7.52%
MplCM	442	**443 (+1)**	0.3%
MplCBT	442	**451 (+9)**	2.88%

- Results with SATCoin Benches: **Very Strong** gains

Solver Name	**Solved by Baseline**	**Solved by expSAT Extension**
gLCM	7	**12 (+5)**
MplCOMSPS	4	**13 (+9)**
MplCM	1	**10 (+9)**
MplCBT	41	**43 (+2)**

paramAdapt: Exploration Parameter Adaptation

- **paramAdapt**
  - During the course of the expSAT search, it dynamically controls
    - **frequency of** exploration episodes,
    - **how much exploration** to perform in an exploration episode.
- **Experiments:**
  - Repeated the same experiments with both test sets.
  - **paramAdapt** implemented on top of our expSAT solvers.
- **Results:**
  - Compared **adaptive version** with **non-adaptive version** .
    - **mixed performance** over SATComp Benches.
    - **outstanding gains** over SATCoin Benches!

      - Biggest improvement: **baseline: 1, it's adaptive extension: 23**!

Analysis of the Solving Efficiency

- Analyze the experimental data for **glucose and eGLCM (non-adaptive)** to reveal further insights.
- Observation for **conflict efficiency**: In general,
    - Better solver for a subset of a problem **is more conflict efficient**.
        - Produces conflict at a fast rate, from which high quality clauses are learned.
- Observation for **average CD phase Length** : In general,
    - Better solver for a subset of a problem **reduces average CD phase length**.
    - exploration helps a solver **to escape from CD phases expeditiously**.

Conclusions and Future Work

- **Conclusions**:
    - Showed that VSIDS frequently undergoes the **pathological phase of CD**, in which branching decisions **are ineffective**.
    - To combat CD phases, **we proposed expSAT** that performs random exploration in the SAT search space.
    - **Empirically showed the effectiveness** of the expSAT approach.
- **Future Work**:
    - Integrate expSAT to **LRB based systems**.
    - Study exploration as in expSAT **to guide polarity selection**, e.g., by extending the phase-saving heuristic.
    - **Identify characteristics of SAT domains** which influence the effectiveness of exploration.