Exploiting Glue Clauses to Design Effective CDCL Branching Heuristics

Md Solimul Chowdhury   Martin Müller   Jia-Huai You

Department of Computing Science, The University of Alberta.

October 1, 2019

Outline

1. Introduction
2. Empirical Observations
3. Proposed Branching Heuristics
4. Empirical Evaluation
5. Additional Experiments
6. Conclusions and Future Work

Introduction

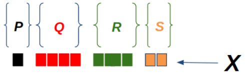
- In this work, I study Boolean Satisfiability (SAT)
  - Given a Boolean formula, the task is to **determine assignments** of the variables to satisfy that boolean formula, if one exists. Otherwise, report unsatisfiability ....
- SAT solving is **NP-Complete** $\rightarrow$ Intractable, in general.
- Modern SAT solvers $\rightarrow$ Conflict Directed Clause Learning (CDCL) Solvers.
  - **Applications in many domains**: Hardware design verification, Software testing, encryption, planning ..

Introduction

- Two basic SAT operations: **decision** and **propagation**.
- CDCL workflow:
  - decide $\rightarrow$ propagate $\rightarrow$ decide $\rightarrow$ propagate $\ldots$.
  - decide $\rightarrow$ propagate $\rightarrow$ **conflict**
    - **conflict**: a clause cannot be satisfied wrt. the current partial assignment.
  - **conflict** $\rightarrow$ conflict analysis $\rightarrow$ clause learning and back-jumping.
- Conflict Generation at a **fast rate is crucial** for CDCL SAT solvers.
  - conflict $\rightarrow$ learned clause $\rightarrow$ space pruning.
- A CDCL SAT solver **learns clauses at a fast rate**.
  - May affect the overall speed of a solver.
  - Learnt clause DB management is necessary $\rightarrow$ **periodic reduction**.

Introduction

- One criterion for clause DB management is **Literal Block Distance (LBD)** score of the learned clauses.
  - Number of distinct decision levels in a learned clause.
    - The learned clause **X** has 4 decision levels: *P*, *Q*, *R* and *S*.



  - Lower the better.
- **Glue Clause:** Learned clauses with LBD score 2.
  - are known to **possess high pruning power**.
- In this work, we relate **Glue clauses to branching decisions**.
  - At any given state of the search:
    - **Glue Variable**: a variable that appears in at least one glue clauses.
    - **NonGlue Variable**: never appears in any of the glue clauses.

Contributions

- **Contribution I:**
  - We empirically show that
    - Decisions with glue variables are **more conflict efficient**.
    - CDCL branching heuristics show a **clear bias** toward Glue variables.
- **Contribution II:**
  - Developed a structure aware variable bumping scheme - Glue Bumping (GB)
    - **prioritizes** selection of Glue variables
  - Empirically evaluated the GB method on **four state-of-the-art CDCL SAT solvers**.
- **Contribution III:**
  - Have introduced the **Glue to Learned (G2L)** metric
    - G2L: fraction of the learned clauses that are glue.
    - consistently explain the performance of the GB method.

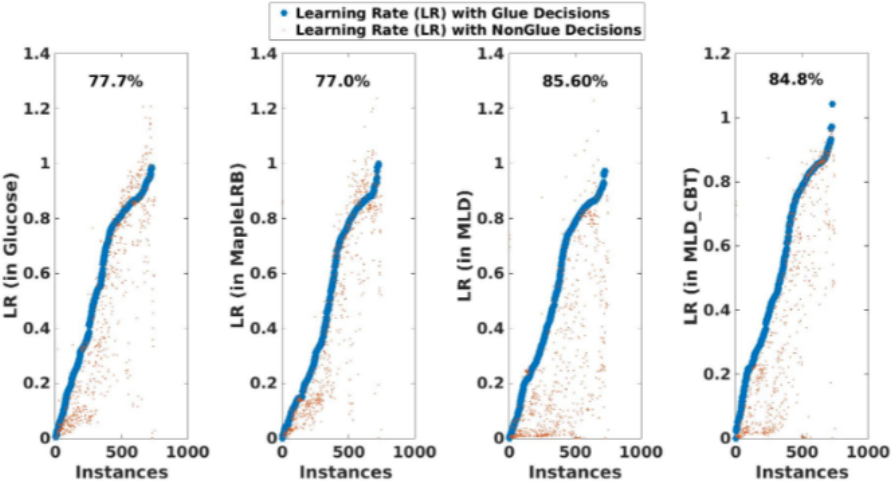
Notations

- For a run of a solver with a given SAT formula
  - **Learning Rate (LR)**
    - number of conflicts per decisions.
  - **Average LBD (aLBD)**
    - average LBD scores of the learned clauses derived from the generated conflicts.
  - **Glue** and **NonGlue decisions**
    The branching decision that selects
    - a Glue variable is called a **Glue decision**.
    - a NonGlue variable is called a **NonGlue decision**.

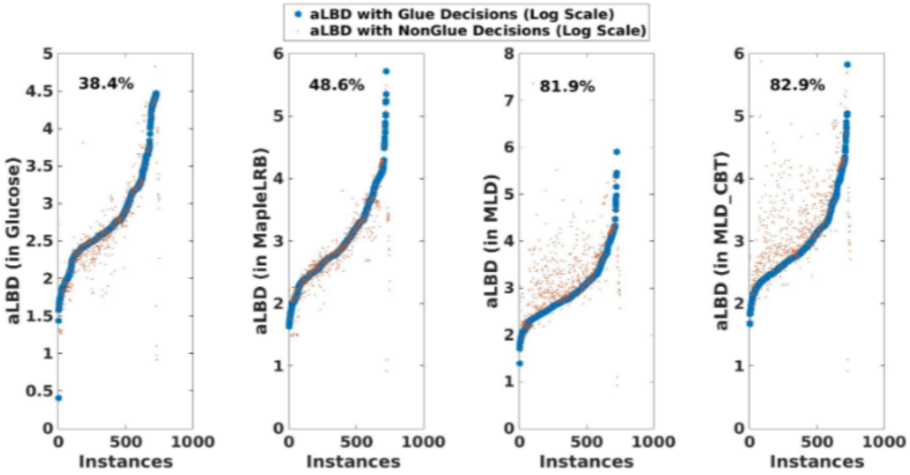
Contribution I: Conflict Efficiency of Glue Variables

- We study **LR** and **aLBD** over Glue and NonGlue decisions.
  - For all the maintrack instances from SAT-2017 and 2018 (750).
  - Using four state-of-the-art solvers:
    - **Glucose**,
    - **MaplePureLRB** (MapleLRB),
    - **MapleLCMDist** (MLD, winner of SAT-2017) and
    - **MapleLCMDistChronoBT** (MLD_CBT, winner of SAT-2018).
- For each run (time limit=5000s), we separately measure **LR** and **aLBD** over Glue and NonGlue decisions.

Conflict Efficiency of Glue Variables (LR)



Conflict Efficiency of Glue Variables (aLBD)



Biased Selection of Glue Variables

- For a run with a given solver for a given instance $\mathcal{F}$, we define
  - **Glue Percentage (GP):** $GP = \frac{\#Glue\_Variables\_in\_F}{\#Variable\_in\_F} *100$

(A) Systems	(B) Average for Glue Variable	
	GP (B1)	Glue Decisions % (B2)
Glucose	25.32%	65.43%
MapleLRB	21.8%	63.14%
MLD	22.05%	47.60%
MLD_CBT	22.19%	48.76%

Contribution II: The Glue Bumping (GB) method

- How can we exploit this empirical characteristics of glue variables?
  - Glue Bumping: which bumps the **activity score of glue variables**
  - based on **appearance count** of a variable in glue clauses and its **current activity score**.
- **Glue Level (gl):**
  - Let G be the **set of learned glue clauses** so far.
  - *gl(v)* of a variable *v* is the appearance count of *v* in the glue clauses in G.

Alg. 1: Increase Glue Level

Input: *A newly learned glue clause* θ

For $i \leftarrow 1$ *to* $|\theta|$
  $v \leftarrow varAt(\theta, i)$
  $gl(v) \leftarrow gl(v) + 1$
End

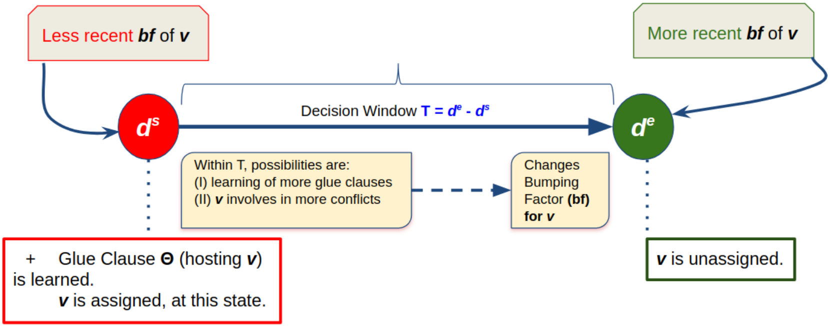
Alg. 2: Bump Glue Variable

Input: *A glue variable* v

$bf_v \leftarrow activity(v) * \left(\frac{gl(v)}{|G|}\right)$
$activity(v) \leftarrow activity(v) + bf_v$

Delayed Bumping in GB

- GB delays the bumping of **v** until it is **unassigned by backtracking**.
  - θ is the latest learned clause and every variables are currently assigned.
  - $T = d^e - d^s > 0$ be the **decision window**.



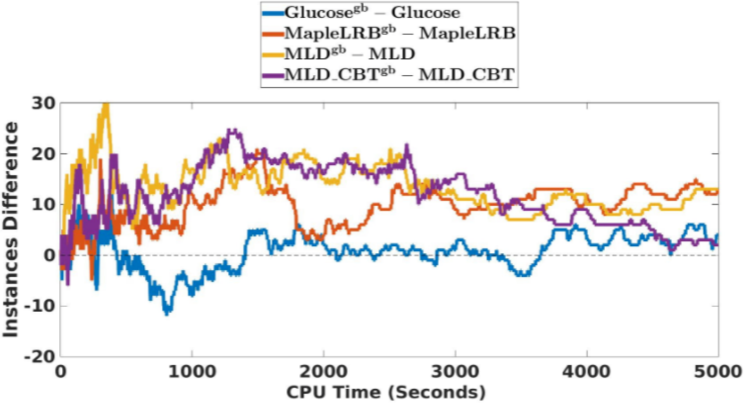
- Hence, GB method **delays the bumping until** d^e.

Empirical Evaluation

- Extended Glucose, MapleLRB, MLD and MLD_CBT with the GB.
- Performed experiments (timeout=5000s) → Apple-to-Apple comparison.
  - 13 additional instances solved by both MapleLRB^{gb} and MLD^{gb}.

Systems	SAT Comp-2017 and 2018			
	SAT	UNSAT	Total	PAR-2
Glucose	180	191	371	4167
Glucose^{gb}	**182 (+2)**	**193 (+2)**	**375 (+4)**	**4141**
MapleLRB	194	190	384	3966
MapleLRB^{gb}	**204 (+10)**	**193 (+3)**	**397 (+13)**	**3851**
MLD	235	207	442	3442
MLD^{gb}	**246 (+11)**	**209 (+2)**	**455 (+13)**	**3318**
MLD_CBT	238	215	453	3365
MLD_CBT^{gb}	**240 (+2)**	215 (+0)	**455 (+2)**	**3295**

Solve Time Comparison



Solve time comparisons. For any point above 0 in the vertical axis, our extensions solve more instances than their baselines at the time point in the horizontal axis.

Surprising observation for GLR and aLBD

- Better branching heuristics have **higher GLR and lower aLBD**, on average (Liang 2017 et. al.)
- We take two subsets (Extreme cases) into considerations:
  - $\textbf{GB}_{\textbf{exclusive}}$: Instances are solved by the GB extension, not by its baseline.
  - $\textbf{Baseline}_{\textbf{exclusive}}$: Instances are solved by the baseline, not by its GB extension.
- Expectations:
  - For $\textbf{GB}_{\textbf{exclusive}}$, our GB extensions achieve **higher GLR** and **lower aLBD**, on average.
  - For $\textbf{Baseline}_{\textbf{exclusive}}$, our GB extensions achieve **lower GLR** and **higher aLBD**, on average.
- We observe **almost opposite scenario**:
  - Average GLR does not hold the expectations, at all.
  - Average aLBD is also inconsistent for these two subsets.

Contribution III: G2L- A new measure for performance

(A) Systems	(B) Employed Heuristics	(C) $GB_{exclusive}$				(D) $Baseline_{exclusive}$			
		#inst	avg. GLR	avg. aLBD	avg. G2L	#inst	avg. GLR	avg. aLBD	avg. G2L
Glucose	{VSIDS}	33	**0.56**	28.60	0.0005	29	0.59	**18.52**	**0.0015**
$Glucose^{gb}$	$\{VSIDS\}^{gb}$		0.53	**24.69**	**0.0016**		**0.62**	20.14	0.00078
MapleLRB	{LRB}	27	**0.50**	26.06	0.00073	14	0.47	**30.75**	**0.00046**
$MapleLRB^{gb}$	$\{LRB\}^{gb}$		0.46	**20.38**	**0.00126**		**0.48**	32.02	0.00037
MLD	{Dist/VSIDS/LRB}	28	**0.55**	**23.60**	0.00029	15	0.53	26.70	**0.0011**
MLD^{gb}	$\{Dist/VSIDS/LRB\}^{gb}$		0.51	26.04	**0.00032**		**0.58**	**23.21**	0.0009
MLD_CBT	{Dist,VSIDS,LRB}	26	**0.49**	**26.08**	0.0006	24	0.51	29.64	**0.00065**
MLD_CBT^{gb}	$\{Dist/VSIDS/LRB\}^{gb}$		0.43	36.24	**0.0011**		**0.55**	**25.42**	0.00037

$$G2L = \frac{\#glue\_clauses}{\#learned\_clauses}$$

- **Better heuristic** for **an instance set** consistently achieves higher G2L.

Peculiarity of Glucose

- Lowest gains with **Glucose**. $\rightarrow$ why?
- **Glucose** already increases the score of some of the (glue) variables during conflict analysis.
- Hypothesis: GB in **Glucose$^{\mathbf{gb}}$** creates imbalance.
- We lower the bumping factor with **high normalizing factor** $\rightarrow$ improved performance with **Glucose$^{\mathbf{gb}}$**.
    - Solves **11** additional instances.
        - In comparison, the version with **lower normalization factor** solves 4 additional instances.

Conclusions and Future Work

- Conclusions:
  - Decisions with Glue variables are conflict efficient.
  - GB method with delayed bumping of Glue variables.
  - Empirical evaluation shows performance gain.
  - G2L correlates well with performance.
- Future Work:
  - Relationships between normalized glue level and other centrality measures.
  - Design clause deletion heuristics based on the notion of glue level?
  - New branching heuristics based on G2L?