Four CDCL solvers based on expLRB, expVSIDS and Glue Bumping

Md Solimul Chowdhury
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada
mdsolimu@ualberta.ca

Martin Müller
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada
mmueller@ualberta.ca

Jia-Huai You
Department of Computing Science
University of Alberta
Edmonton, Alberta, Canada
jyou@ualberta.ca

***Abstract*—This document describes 4 CDCL SAT solvers: kissat_gb, kissat_crvr_gb, cms_expV_gbL and CaDiCaL_hack_gb which are entering the SAT Competition-2021. These solvers are based on three new ideas: 1) Guidance of Learning Rate Based (LRB) and Variable State Independent Decaying Sum (VSIDS) branching heuristics via random exploration amid pathological phases of conflict depression 2) Activity score bumping of variables which appear in the glue clauses, and 3) Common Reason decision Variable score Reduction (CRVR).**

I. Guidance of CDCL Branching Heuristics via Random Exploration during Conflict Depression

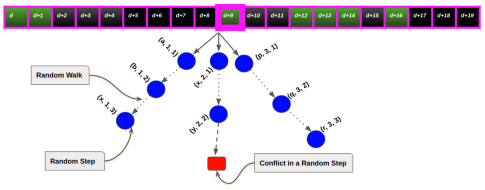
This approach is based on our observation that CDCL SAT solving entails clear non-random patterns of bursts of conflicts followed by longer phases of *conflict depression (CD)* [1]. During a CD phase a CDCL SAT solver is unable to generate conflicts for a consecutive number of decisions. To correct the course of such a search, we propose to use exploration to combat conflict depression. We therefore design a new SAT solver, called *expSAT*, which uses random walks in the context of CDCL SAT solving. In a conflict depression phase, random walks help identify more promising variables for branching. As a contrast, while exploration explores *future* search states, LRB and VSIDS relies on conflicts generated from the *past* search states.

A. expSAT algorithm

Given a CNF SAT formula $\mathcal{F}$, let $vars(\mathcal{F})$, $uVars(\mathcal{F})$ and $assign(\mathcal{F})$ denote the set of variables in $\mathcal{F}$, the set of currently unassigned variables in $\mathcal{F}$ and the current partial assignment, respectively. In addition to $\mathcal{F}$, *expSAT* also accepts four *exploration parameters* nW, lW, p_{exp} and ω, where $1 \leq nW, lW \leq uVars(\mathcal{F})$, $0 < p_{exp}, \omega \leq 1$. These parameters control the exploration aspects of *expSAT*. The details of these parameters are given below.

Given a CDCL SAT solver, *expSAT* modifies it as follows: (I) Before each branching decision, if a **substantially large CD phase** is detected then with probability p_{exp}, *expSAT* performs an *exploration episode*, consisting of a fixed number *nW* of random walks. Each walk consists of a limited number of *random steps*. Each such step consists of (a) the uniform random selection of a currently unassigned *step variable* and assigning a boolean value to it using a standard CDCL *polarity* heuristic, and (b) a followed by Unit Propagation (UP). A walk terminates either when a conflict occurs during UP, or after a fixed number *lW* of random steps have been taken. Figure 1 illustrates an exploration episode amid a CD phase. (II) In an exploration episode of nW walks of maximum length lW, the *exploration score expScore* of a decision variable v is the average of the *walk scores* $ws(v)$ of all those random walks within the same episode in which v was one of the randomly chosen decision variables. $ws(v)$ is computed as follows: (a) $ws(v) = 0$ if the walk ended without a conflict. (b) Otherwise, $ws(v) = \frac{\omega^d}{lbd(c)}$, with decay factor $0 < \omega \leq 1$, $lbd(c)$ the LBD score of the clause c learned for the current conflict, and $d \geq 0$ the *decision distance* between variable v and the conflict which ended the current walk: If v was assigned at some step j during the current walk, and the conflict occurred after step $j' \geq j$, then $d = j' - j$. We assign credit to all the step variables in a walk that ends with a conflict and give higher credit to variables closer to the conflict. (III) The novel branching heuristic `expVSIDS` adds VSDIS score and *expScore* of the variables that participated in the most recent exploration episode. For `expVSIDS`, a variable v^* with maximum combined score is selected for branching. (IV) All other components remain the same as in the underlying CDCL SAT solver.

Fig. 1: The 20 adjacent cells denote 20 consecutive decisions starting from the d^{th} decision, with $d > 0$, where a green cell denotes a decision with conflicts and a black cell denotes a decision without conflicts. Say that amid a CD phase, just before taking the $(d+9)^{th}$ decision, *expSAT* performs an exploration episode via 3 random walks each limited to 3 steps . The second walk ends after 2 steps, due to a conflict. A triplet (v, i, j) represents that the variable v is randomly chosen at the j^{th} step of the i^{th} walk.



II. Glue Variable Bumping

Let a CDCL SAT solver M is running a given SAT instance $\mathcal{F}$ and the current state of the search is S. We call the variables that appeared in at least one glue clause up to the current state S *Glue Variables*. We design a structure-aware variable score bumping method named *Glue Bumping* (GB) [2], based on the notion of *glue centrality (gc)* of glue variables. Given a glue variable v_g, glue centrality of v_g dynamically measures the fraction of the glue clauses in which v_g appears, until the current state of the search. Mathematically, the glue centrality of v_g, $gc(v_g)$ is defined as follows:

$$gc(v_g) \leftarrow \frac{gl(v_g)}{ng}$$

, where ng is the total number of glue clauses generated by the search so far. $gl(v_g)$ is the glue level of v_g, a count of glue clauses in which v_g appears, with $gl(v_g) \leq ng$.

A. The GB Method

The GB method modifies a CDCL SAT solver M by adding two procedures to it, named *Increase Glue Level* and *Bump Glue Variable*, which are called at different states of the search. We denote by M^{gb} the GB extension of the solver M.

Increase Glue Level: Whenever M^{gb} learns a new glue clause g, before making an assignment with the first UIP variable that appears in g, it invokes this procedure. For each variable v_g in g, its glue level, $gl(v_g)$ is increased by 1.

Bump Glue Variable: This procedure bumps a glue variable v_g, which has just been unassigned by backtracking. First a bumping *factor (bf)* is computed as follows:

$$bf \leftarrow activity(v_g) * gc(v_g)$$

, where $activity(v_g)$ is the current activity score of the variable v_g and $gc(v_g)$ is the glue centrality of v_g. Finally, the activity score of v_g, $activity(v_g)$ is bumped as follows:

$$activity(v_g) \leftarrow activity(v_g) + bf$$

III. Common Reason decision Variable score Reduction (CRVR)

During a CDCL search, a single decision can generate more than one conflicts, from which multiple clauses are learned. We refer decisions with multiple conflicts as `mc` decisions. Let a `mc` decision $\mathcal{M}$ generates n conflicts, from which it learns a sequence of clauses $\mathcal{L} = (L_1 \dots L_n)$. For any clause $L_i \in \mathcal{L}$, let $L_i = R_i \vee \neg f$, where R_i be the reason clause and f be the unique implication point for the conflict that generates L_i.

We call the set of decision variables in $R_1 \cap \dots \cap R_i \cap \dots \cap R_n$ as Common Decision Variables (CRVs) for $\mathcal{M}$. CRVs are the common decision variables over the reason clauses in $\mathcal{M}$.

The CRVR scheme decreases the activity scores of those CRVs, which correspond to `mc` decisions with average LBD score higher (i.e., have lower quality learned clauses) than the recent search average.

IV. Solvers Description

We have submitted four CDCL SAT solvers to SAT Competition-2021, which are based on four combinations of the three approaches described in the previous sections. Our solvers are implemented on top of the solver kissat_sat (kissat with `sat` configuration) [3], CaDiCaL1.4.0 (base solver for the CaDiCaL hack track) [4], and CryptoMiniSAT5.8.0 [5]. In the following, we describe our solvers:

a) ***kissat_gb****:* This solver implements the GB method on top of kissat_sat. kissat_sat employs two branching heuristics: VSIDS and VMTF. In kissat_gb, the GB scheme is kept active only when VSIDS is active.

b) ***kissat_crvr_gb****:* This solver implements the GB and CRVR method on top of kissat_sat. In kissat_crvr_gb, the GB and CRVR schemes are kept active only when VSIDS is active.

c) ***cms_expV_gbL****:* The baseline CryptoMiniSAT5.8.0 employs a combination of three branching heuristics: LRB, VSIDS and VMTF. This system extends this baseline by implementing the GB method on top of LRB, and by replacing VSIDS with `expVSIDS`.

d) ***CaDiCaL_hack_gb****:* This system implements the GB method on the top of CaDiCal1.4.0, only when VSIDS is active in the baseline system. CaDiCaL_hack_gb is submitted to the CaDiCaL hack track of the SAT compeition-2021.

References

[1] Md Solimul Chowdhury and Martin Müller and Jia You, Guiding CDCL SAT Search via Random Exploration amid Conflict Depression, To appear in Proceedings of AAAI-2020.

[2] Md. Solimul Chowdhury, Martin Müller, Jia-Huai You, Exploiting Glue Clauses to Design Effective CDCL Branching Heuristics. In Proceedings of CP 2019: 126-143.

[3] Armin Biere Katalin Fazekas Mathias Fleury Maximilian Heisinge. CADICAL, KISSAT, PARACOOBA, PLINGELING and TREENGELING Entering the SAT Competition 2020, In Proceedings of SAT Competition 2020:50-52.

[4] CaDiCal 1.4, https://github.com/arminbiere/cadical/tree/rel-1.4.0, access date: 02-April-2021.

[5] CryptoMiniSat 5.8.0, https://github.com/msoos/cryptominisat/releases, access date: 02-April-2021.