Proceedings of SAT Competition 2018: Solver and Benchmark Descriptions, volume B-2018-1 of Department of Computer Science Series of Publications B, University of Helsinki 2018.

Description of *expSAT* Solvers

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*—*expSAT* is a novel CDCL SAT solving method, which performs random-walk based explorations of the search space w.r.t the current search state to guide the search. It uses a new branching heuristics, called `expVSIDS`, which combines the standard variable selection heuristic VSIDS, which is based on search performance so far, with heuristic scores derived from random samples of possible future search states. This document describes the *expSAT* approach and four CDCL SAT solvers based on this approach, which we have submitted for the SAT competition-2018.**

I. The *expSAT* Approach

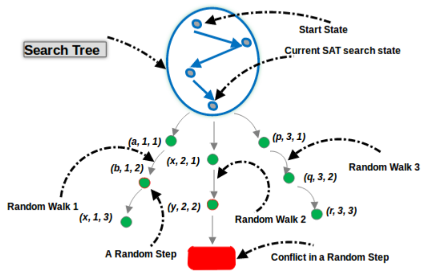
This section presents the *expSAT* approach, part of which is to appear in [1].

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 five *exploration parameters* $nW, lW, \theta_{stop}, p_{exp}$ and ω, where $1 \leq nW, lW \leq uVars(\mathcal{F})$, $0 < \theta_{stop}, p_{exp}, \omega \leq 1$. 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 the search-height, $\frac{|assign(\mathcal{F})|}{|vars(\mathcal{F})|} \leq \theta_{stop}$, 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. (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 VSIDS score and *expScore* of the unassigned variables. At the current state of the search, the variable bumping factor of VSIDS is g^z, where $g > 1$ and $z \geq 1$ is the count of conflicts in the search so far. To achieve a comparable scale for $expScore$ and VSIDS score, we scale up the *expScore* by g^z before adding these scores. 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: An exploration episode with $nW = 3$ walks and a maximum of $lW = 3$ random steps per walk. (v, i, j) represents that the variable v is randomly decided at the j^{th} step of i^{th} walk.



B. *Exploration Parameter Adaptation*

In *expSAT*, $(nW, lW, \theta_{stop}, p_{exp}, \omega)$, the set of exploration parameters, governs the exploration. The first two parameters dictate *how* much exploration to perform per episode, the third and fourth parameter dictate *when* to trigger an exploration episode. ω controls how exploration scores are computed.

How to adapt these parameters during the SAT search is an interesting question, which is not addressed in [1]. The *expSAT* based solvers submitted for this competition uses a simple local search algorithm to adapt the first four exploration parameters $P = (nW, lW, \theta_{stop}, p_{exp})$ to dynamically control when to trigger exploration episodes and how much exploration to perform in an exploration episode. This local search algorithm executes in parallel to the SAT search in *expSAT*.

a) The Adaptation Algorithm: The idea of this algorithm is to start with an initial value $val(P, 1) = (nW^1, lW^1, \theta^1_{stop}, p^1_{exp})$ of the exploration parameters and iteratively update the values between two consecutive restarts, based on the performance of exploration in the previous restarts.

Assume *expSAT* has just performed the j^{th} $(j \geq 2)$ restart. Let σ_j the performance of exploration between $(j-1)^{th}$ and j^{th} restarts. We define σ^j as follows:

$$\sigma^j = w_1 * \frac{c^j}{rSteps^j} + w_2 * \frac{gc^j}{rSteps^j} + w_3 * \frac{1}{\overline{rsLBD^j}}$$

Here, $rSteps^j$ is the number of random steps taken during the exploration episodes, which has occurred between $(j-1)^{th}$ and j^{th} restarts, c^j, gc^j, $\overline{rsLBD^j}$ are the number of conflicts, number of glue-clauses and the mean LBD value of the (learned) clauses identified in these $rSteps^j$ steps, respectively. w_1, w_2 and w_3 are three fixed weights.

After restart j, just before starting SAT search, the algorithm updates the exploration parameter values by comparing σ_j and σ_{j-1}. Let $val(P, j) = (nW^j, lW^j, \theta^j_{stop}, p^j_{exp})$ is the updated value of exploration parameters before the search begins, just after the $(j-1)^{th}$ restart.

- If $\sigma^j < \sigma^{j-1}$, then the performance of exploration deteriorates after the $(j-1)^{th}$ restart. In this case, we perform two operations on the exploration parameters after j^{th} restart:
  - **Decrement**: Let $dp \in P$ the parameter whose value was increased from x to x' after the $(j-1)^{th}$ restart. We attribute this deterioration of performance to this update. We revert the value of dp back to x from x'.
  - **Increment**: Randomly select a parameter $rp \in P$ and increase its value to y' from y, where $rp \neq dp$.
- If $\sigma^j - \sigma^{j+1} = 0$, then we only perform the **Increment** operation, as we do not know whom to blame for the stall.
  The updated value of these parameters remain effective until the $(j+1)^{th}$ restart.
- If $\sigma^j > \sigma^{j-1}$, then the performance of exploration is increasing after the $(j-1)^{th}$ restart and we do not change any parameter value as the current value of the exploration parameters leads to better performance.

For changing the value of a parameter $x \in P$, we associate a step size s_x with x. Also, in order to prevent the unbounded growth/shrink of the parameters we associate a lower and upper bound with each of the parameters. That is, for $x \in P$, we have a $[l_x, u_x]$. Whenever the value of x exceeds u_x OR the value of x is less than l_x, then the value of x is reset to its initial value x^1.

II. *expSAT* Solvers

We have submitted four CDCL SAT solvers based on the expSAT approach, which are implemented on top of Glucose, MapleCOMPSPS_LRB and MapleCOMPSPS. In the following, we describe our solvers:

a) ***expGlucose****:* expGlucose is an extension of Glucose, where we replace VSIDS by *expVSIDS* and and have kept everything else the same as in Glucose.

b) ***expMC_LRB_VSIDS_Switch****:* The corresponding baseline system MapleCOMPSPS_LRB switches between branching heuristics LRB and VSIDS in between restarts. In expMC_LRB_VSIDS_Switch, we replace VSIDS with *expVSIDS* and have kept everything else the same as in MapleCOMPSPS_LRB.

c) ***expMC_LRB_VSIDS_Switch_2500****:* The corresponding baseline system MapleCOMPSPS has three switches between VSIDS and LRB (i) VSIDS for initialization (first 50,000 conflicts), (ii) then run LRB for 2,500 seconds, and (iii) then switches to VSIDS for rest of the execution of the solver. In expMC_LRB_VSIDS_Switch_2500, we replace VSIDS with *expVSIDS* for (iii) and have kept everything else the same as in MapleCOMPSPS.

d) ***expMC_VSIDS_LRB_Switch_2500****:* This system is a variant of expMC_LRB_VSIDS_Switch_2500. It uses *expVSIDS* for the first 2,500 seconds and then switches to LRB for the rest of its execution.

References

[1] MS Chowdhury and M. Müller and J. You, “Preliminary results on exploration driven satisfiability solving (Student Abstract).”, AAAI-2018.