Exploring Conflict Generating Decisions: Initial Results (Extended Abstract)

Md Solimul Chowdhury[1], **Martin Müller**[2], **Jia-Huai You**[2]

[1]Department of Computer Science, Carnegie Mellon University
[2]Department of Computing Science, University of Alberta
solimul.chowdhury@gmail.com, mmueller@ualberta.ca, you@cs.ualberta.ca

Introduction

Boolean Satisfiability (SAT) is an NP-complete problem, indicating its inherent computational hardness. However, Conflict Driven Clause Learning (CDCL) SAT solvers efficiently tackle large instances from diverse domains. Efficient problem-solving with CDCL relies on rapid conflict identification, as conflicts leads to the search space pruning learning of clauses, which encode the underlying causes of the conflicts they originate from. CDCL decision heuristics prioritize variables that participated in recent conflicts, anticipating rapid conflict generation and expediting additional clause learning. In practice, only a fraction of decisions lead to conflicts, yet some decisions may yield multiple conflicts.

This paper delves into conflict-generating decisions in CDCL, distinguishing between single-conflict (sc) decisions, which generate only one conflict, and multi-conflict (mc) decisions, producing two or more conflicts. Our empirical analysis evaluates each decision type based on the quality of the learned clauses they produce. Our theoretical analysis suggests that in a mc decision, the learning of a clause depends on prior clause learning within that decision. This leads to the hypothesis that learned clauses in mc decisions share a more common set of literals compared to those in sc decisions. This hypothesis is empirically confirmed with our introduced concept of *learning proximity*. Finally, we propose *score reduction* (sr), a novel decision strategy that decreases the selection priority of specific variables from learned clauses in mc decisions. Evaluation of sr on over 1200 benchmarks demonstrates its effectiveness.

We assume familiarity with the CDCL SAT solving algorithms, and briefly introduce the concepts most relevant for this paper: (i) The Variable State Independent Decaying Sum (VSIDS) decision heuristic assigns an activity score $act[v]$ to each variable v, prioritizing those involved in recent conflicts. It exponentially increases the activity scores of conflict-involved variables during conflict analysis, prioritizing their selection in subsequent decisions. (ii) The state-of-the-art metric for measuring the quality of a learned clause is Literal Block Distance, or *LBD*, in short. It represents the count of literal blocks in a learned clause, where all the literals in a block are assigned at the same decision level. A lower *LBD* value indicates higher quality.

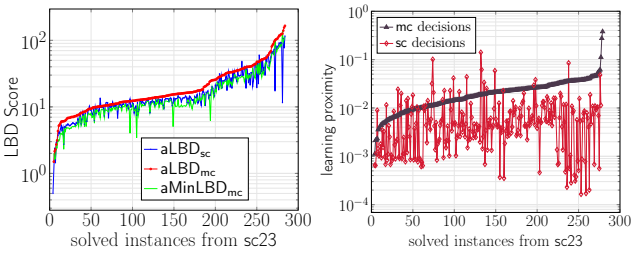

Figure 1: sc vs. mc decisions: Learned clause quality (left plot); *learning proximity* (right plot)

We denote a CDCL solver ψ running a SAT instance $\mathcal{F}$ as $\psi_{\mathcal{F}}$. We introduce three measures related to LBD scores. aLBD$_{sc}$ (average LBD for sc clauses) and aLBD$_{mc}$ (average LBD for mc clauses). For a given mc decision $\mathcal{M}$, min_LBD$_{\mathcal{M}}$ denotes the minimum LBD score among its learned clauses, and aMinLBD$_{mc}$ is the average of minimum LBD over mc decisions in $\psi_{\mathcal{F}}$.

Analysis of sc and mc Decisions

We analyze the frequency and quality of the clauses learned in sc and mc decisions, along with a comparison on the degree of commonality between the learned clauses in sequences of sc and mc decisions. We utilize sbvaCaDiCaL (Haberlandt, Green, and Heule 2023), the SAT Competition 2023 main track winner, on sc23, a set of 400 benchmarks submitted for the same track, with a 5000-second timeout per run, utilizing the StarExec cluster (Stump, Sutcliffe, and Tinelli 2012).

Distributions and Learned Clause Quality Recall that a decision in CDCL can yield 0 or multiple conflicts. This prompts questions on the percentage of conflict-producing decisions, and the distribution between sc and mc decisions. Across the 284 sc23 benchmarks solved by sbvaCaDiCaL within the timeout, on average, 6% of decisions are sc, and 28% are mc, totaling 34% of decisions resulting in conflicts. This suggests that, on average, 66% of decisions don't lead to conflicts. Notably, 82% of conflict-producing decisions belong to mc, emphasizing its prevalence over sc in occurrence frequency.

BS	Cnt	sbvaCaDiCaL			sbvaCaDiCaL^{sr}		
		S	U	C	S	U	C
parity	29	8	0	8	15	0	**15(+7)**
crypto	200	114	12	126	120	11	**131 (+5)**
bitvec	594	161	221	382	165	225	**390 (+8)**
sc23	400	121	163	284	124	164	**288 (+4)**
Overall	1223	404	396	800	424	400	**824 (+24)**

Table 1: Evaluation results for sbvaCaDiCaL and sbvaCaDiCaL^{sr} solver. BS: Benchmark-Set; Cnt:Total number of instances; S: # of solved Satisfiable instances; U: # of solved Unsatisfiable instances; C: S+U

Next, we compare the quality of learned clauses in sc and mc decisions for the same benchmark set. The left plot in Figure 1 provides per-instance values for three LBD-related measures: $\mathsf{aLBD}_{\mathsf{sc}}$, $\mathsf{aLBD}_{\mathsf{mc}}$, and $\mathsf{aMinLBD}_{\mathsf{mc}}$. On average, $\mathsf{aMinLBD}_{\mathsf{mc}}$ is consistently the lowest among these measures, followed by $\mathsf{aLBD}_{\mathsf{sc}}$, which is lower than $\mathsf{aLBD}_{\mathsf{mc}}$. Recall that the lower the LBD score of a clause, the better its quality. Hence, on average, mc decisions are learning-inefficient compared to sc decisions. However, on average, the *best quality* learned clause generated in a mc decision have better quality than the quality of a sc clause.

Learning Proximity over sc and mc Decisions Given a sequence of n learned clauses $\mathcal{L} = \langle L_1, \ldots, L_n \rangle$, the *learning proximity* $\mathsf{lp}_{\mathcal{L}}$ between the clauses in $\mathcal{L}$ is defined as $\frac{|\mathsf{C}_{\mathcal{L}}|}{|\mathsf{D}_{\mathcal{L}}|}$, where $\mathsf{C}_{\mathcal{L}}$ and $\mathsf{D}_{\mathcal{L}}$ are the common and distinct sets of literal blocks in $\mathcal{L}$, respectively. For any two given learned clause sequences $\mathcal{L}$ and $\mathcal{L}^{'}$ with $|\mathcal{L}| = |\mathcal{L}^{'}|$, if $\mathsf{lp}_{\mathcal{L}} > \mathsf{lp}_{\mathcal{L}'}$, then the learned clauses in $\mathcal{L}$ exhibit a higher degree of commonality in literal blocks compared to those in $\mathcal{L}^{'}$.

In a given mc decision that learns $x \geq 2$ clauses within a single decision, we claim that these x learned clauses are locally connected, meaning that the learning of the j^{th} clause is necessitated by the learning of the previous $j-1 < x$ clauses within that mc decision, forming a chain of learned clauses. Based on this claim, we hypothesize that on average, the $x > 2$ learned clauses which are generated within a given mc decision, exhibit a higher degree of commonality in literal blocks compared to the learned clauses generated in the last x sc decisions.

We empirically validate this hypothesis using sbvaCaDiCaL as the solver and sc23 as the benchmark set. When a mc decision with $x \geq 2$ learned clauses occurs, we compute the *learning proximity* for the following learned clause sequences: (i) for x learned clauses in that mc decision and (ii) for learned clauses from the last x sc decisions. The right plot of Figure 1 confirms that the average *learning proximity* for clauses over mc decisions (black line) are higher than those over sc decisions (red line) for almost all benchmarks. This confirms our hypothesis that clauses learned over mc decisions tend to exhibit a higher degree of commonality in literal blocks compared to conflicts generated in sc decisions. Since learned clauses explicitly encode the root cause of the conflicts they arise from, the validity of this hypothesis implies that, on average, for $x \geq 2$ conflicts reached within a given mc decision are caused by a more common set of literal blocks compared to conflicts reached in the x most recent sc decisions.

The Score Reduction Strategy

In a mc decision, multiple learned clauses are generated. A poor mc decision is identified when the LBD score of the best-quality learned clause within that decision surpasses the moving average LBD of all learned clauses up to that point in the search. The shared decision variables (sdv) across multiple learned clauses in a poor mc decision collectively contribute to the generation of low-quality learned clauses within that decision. Does the suppression of such sdv for future decisions help the search achieve better efficiency?

We address this question by designing a decision strategy named *score reduction* (sr). If the current decision turns out to be a poor mc decision with learned clause sequence $\mathcal{L}$, sr marks all the sdv variables $y \in \mathsf{C}_{\mathcal{L}}$ as poor. While the sr strategy aims to reduce the score of each marked variable y, it defers the actual score reduction until the backjumping operation unassigns y. This delay ensures that the score reduction for y occurs at the *earliest* point in time, when y becomes a free variable and can be considered for making a new decision. Before unassigning y, the backjump procedure in sr decreases the activity score of y by a factor of $Q \in (0, 1)$, followed by the unmarking of y. Following backjumping, the free variable y is then reconsidered for a decision with its reduced score.

We implemented sr on top of sbvaCaDiCaL, resulting in the extended solver sbvaCaDiCaL^{sr}. After an initial experiment, we set Q to 0.1, reducing the activity score of a poor sdv by 10%. We evaluate the performance of both solvers across four benchmark sets[1]—Minimum Disagreement Parity (parity), Cryptography (crypto), Bitvector (bitvec), and sc23—comprising 1223 instances, each with a timeout of 5000 seconds. Table 1 shows the result of the evaluation. Our extension employing sr demonstrates performance gains over sbvaCaDiCaL, especially for SAT instances. sbvaCaDiCaL^{sr} solves 824 instances (424 SAT, 400 UNSAT), outperforming sbvaCaDiCaL solving 800 instances (404 SAT, 396 UNSAT), marking a 24-instance improvement with our extension. These results showcase the achieved efficiency with our proposed approach.

References

Haberlandt, A.; Green, H.; and Heule, M. J. H. 2023. Effective Auxiliary Variables via Structured Reencoding. In *SAT-2023*, 11:1–11:19.

Stump, A.; Sutcliffe, G.; and Tinelli, C. 2012. Introducing StarExec: a Cross-Community Infrastructure for Logic Solving. In *Proceedings of the 1st International Workshop on Comparative Empirical Evaluation of Reasoning Systems*, volume 873 of *CEUR Workshop Proceedings*, 2. CEUR-WS.org.

[1] Available in https://benchmark-database.de/