A New Variable Ordering for In-processing Bounded Variable Elimination in SAT Solvers

Shuolin Li[1] , **Chu-Min Li**[1,2,*] , **Mao Luo**[3,*] , **Jordi Coll**[4] , **Djamal Habet**[1] and **Felip Manyà**[4]
[1]Aix Marseille Univ, Université de Toulon, CNRS, LIS, Marseille, France
[2]Université de Picardie Jules Verne, Amiens, France
[3]Hubei University of Technology, Wuhan, China
[4]Artificial Intelligence Research Institute, CSIC, Bellaterra, Spain
shuolin.li@etu.univ-amu.fr, chu-min.li@u-picardie.fr, luomao@hbut.edu.cn, jcoll@iiia.csic.es, Djamal.Habet@univ-amu.fr, felip@iiia.csic.es

Abstract

Bounded Variable Elimination (BVE) is an important Boolean formula simplification technique in which the variable ordering is crucial. We define a new variable ordering based on variable activity, called ESA (variable Elimination Scheduled by Activity), for in-processing BVE in Conflict-Driven Clause Learning (CDCL) SAT solvers, and incorporate it into several state-of-the-art CDCL SAT solvers. Experimental results show that the new ESA ordering consistently makes these solvers solve more instances on the benchmark set including all the 5675 instances used in the Crafted, Application and Main tracks of all SAT Competitions up to 2022. In particular, one of these solvers with ESA, Kissat_MAB_ESA, won the Anniversary track of the SAT Competition 2022. The behaviour of ESA and the reason of its effectiveness are also analyzed.

1 Introduction

The propositional satisfiability (SAT) problem in Conjunctive Normal Form (CNF) is the first problem shown to be NP-Complete [Cook, 1971]. Because of its expressive power and the progress made in SAT solving, modern Conflict-Driven Clause Learning (CDCL) [Silva and Sakallah, 1999; Moskewicz *et al.*, 2001] SAT solvers are widely used to solve real-world application problems nowadays. Since these problems are often of huge size, CNF formula simplification is an important element in the SAT solvers.

The CNF formula simplification techniques include variants of Bounded Variable Elimination (BVE) [Eén and Biere, 2005], addition or elimination of redundant clauses [Järvisalo *et al.*, 2010], elimination of redundant literals in a clause (clause vivification) [Luo *et al.*, 2017; Li *et al.*, 2020], detection of subsumed clauses [Eén and Biere, 2005], self-subsuming resolution (or subsuming resolvent [Ostrowski *et al.*, 2002]), equivalence reasoning, and so on, and suitable combinations of them [Järvisalo *et al.*, 2012]. These techniques can be applied to a CNF formula in a pre-processing, or in an in-processing interleaved with the CDCL search.

This paper focuses on in-processing BVE. Although BVE is an important pre- and in-processing technique in state-of-the-art SAT solvers, it is quite special: While other formula simplification techniques such as clause vivification, self-subsuming resolution and equivalence reasoning usually produce a formula smaller and easier to solve, BVE can produce a formula harder to solve [Reeves and Heule, 2021], depending on the subset of variables eliminated.

The state-of-the-art SAT solvers usually fix a variable ordering, which, together with some bounds on how much the formula is allowed to grow by eliminating a variable, implicitly decides the subset of variables eliminated, because eliminating a variable can increase the number of clauses in which other variables occur, possibly preventing other variables from being eliminated subject to the bounds. In order to eliminate as many variables as possible without exceeding the bounds, usual variable orderings are based on the number of clauses in which a variable occurs positively or negatively, in such a way that the variable with the smallest number of occurrences is eliminated first and the number of occurrences of other variables grows more slowly. In an in-processing BVE, the drawback of such an ordering is that it does not exploit the information learnt during the search.

In this paper, we propose a new variable ordering called ESA (variable Elimination Scheduled by Activity) for in-processing BVE based on variable activities measured according to their participation in recent conflicts. The ESA ordering is inspired from the fact that unrestricted resolution is exponentially stronger than regular resolution by allowing the variables to be resolved on many times [Huang and Yu, 1987; Goerdt, 1993], and from the intuition that the variables with frequent participation in conflicts should be protected from being eliminated to preserve the power of resolution in the CDCL search by allowing to resolve on them when needed.

We implemented the new ESA variable ordering in Kissat_MAB [Cherif *et al.*, 2021b], the winner of the Main track of SAT Competition 2021, and won the Anniversary track of the SAT Competition 2022 [Li *et al.*, 2022]. In this paper, we additionally implement it in the best two solvers of

[*]Corresponding author

the Main track of the SAT Competition 2022. As these winners are all based on Kissat [Biere *et al.*, 2020], we also implement ESA in the non-Kissat solver Cadical [Biere, 2017; Biere, 2018] which uses fewer decision heuristics but performs in-processing BVE more frequently than the three Kissat based solvers. Experimental results on all the 5675 different instances which have been used in the Application, Crafted, and Main tracks of all SAT Competitions up to 2022 show that the new ordering ESA consistently allows the base solver to solve more instances.

We note a prevailing opinion in SAT solving: simpler is better! For example, there is a hack track in SAT competitions since 2009 to identify the actual causes of improvement, in which a competitor must improve an existing solver with changes limited to 1000 characters. ESA is a simple technique in this context. Its effectiveness suggests that exploiting the difference between regular resolution and general resolution could be a promising direction to improve SAT solvers.

The structure of this paper is as follows. Section 2 presents the preliminaries. Section 3 reviews the application of BVE in modern SAT solvers. Section 4 defines the new variable ordering ESA for in-processing BVE. Section 5 empirically evaluates and analyzes ESA, providing an explanation of its effectiveness. Section 6 concludes.

2 Preliminaries

Given a set of propositional variables V, a literal is either a variable $x \in V$ or its negation $\neg x$, a clause c is a disjunction of literals (represented by a set of literals), and a CNF formula F is a conjunction of clauses (represented by a set of clauses).

A truth assignment is a mapping from V to $\{0, 1\}$, where 0 represents the truth value false and 1 represents the truth value true. A truth assignment to the variables occurring in a CNF formula F satisfies literal x ($\neg x$) if variable x is assigned 1 (0), satisfies a clause if it satisfies at least one literal in the clause, and satisfies F if it satisfies all its clauses. When a variable is assigned 1 or 0, we also say that the literals x and $\neg x$ are assigned, and are satisfied or falsified according to the value. An *empty*, *unit* and *binary* clause contains 0, 1 and 2 non-assigned literals, respectively, other literals, if any, being falsified. An empty clause cannot be satisfied and represents a *conflict*. A *tautology* is a clause containing both x and $\neg x$ and is always satisfied. A clause c_1 *subsumes* another clause c_2 if every literal of c_1 also occurs in c_2.

The SAT problem is to find an assignment satisfying all clauses in a CNF formula F, or prove that such an assignment does not exist. In the former case, F is said to be *satisfiable*, and in the latter case, F is said to be *unsatisfiable*. Two formulas are *satisfiability equivalent* if both formulas are satisfiable or both are unsatisfiable.

Given two clauses $c = \{x, a_1, ..., a_n\}$ and $c' = \{\neg x, b_1, ..., b_m\}$ in F, the *resolution rule*, by *resolving c and c' on x*, derives a new clause $c \bigotimes_x c' = \{a_1, ..., a_n, b_1, ..., b_m\}$, called *resolvent* of c and c'. It is well-known that F is unsatisfiable if and only if there is a sequence of clauses $c_1, c_2, \ldots, c_k$, such that c_k is the empty clause and each c_i ($1 \leq i \leq k$) is a resolvent of two clauses in $F \cup \{c_1, \ldots, c_{i-1}\}$ [Robinson, 1965]. Given such a sequence of clauses, we can construct a binary resolution tree as follows: (1) every tree node is a clause in $F \cup \{c_1, \ldots, c_k\}$; (2) the root is c_k; (3) every non-leaf node c_i has two children in $F \cup \{c_1, \ldots, c_{i-1}\}$ whose resolvent is c_i; and (4) every leaf node is a clause in F. The resolution is said to be *regular* if no variable is resolved on more than once in any path from the root to a leaf in the resolution tree.

Given a CNF formula F, F_x ($F_{\neg x}$) denotes the set of all clauses in F containing literal x ($\neg x$). The resolution rule can be lifted to F_x and $F_{\neg x}$ to derive the set of clauses $F_x \bigotimes_x F_{\neg x} = \{c_1 \bigotimes_x c_2 \mid c_1 \in F_x, c_2 \in F_{\neg x}, c_1 \bigotimes_x c_2 \textit{ is not a tautology}\}$, called the *resolvent set* of F_x and $F_{\neg x}$.

In order to find a solution satisfying every clause in F or prove that F is unsatisfiable, the state-of-the-art SAT solvers apply the Conflict-Driven Clause Learning (CDCL) search. If F contains a unit clause l, CDCL executes unit propagation (UP): satisfy l (by assigning an appropriate truth value to its variable) and all the clauses containing l, and falsify $\neg l$, possibly resulting in new unit clauses or empty clauses. Unit propagation continues until there is no unit clause in F or an empty clause (i.e. a conflict) is produced. If UP finishes without producing a conflict, a new literal is picked using a decision heuristic and is satisfied by assigning an appropriate truth value to its variable (i.e., a *decision* is made), and UP is executed again. The number of decisions made so far is called the *current level*. If all variables are assigned without producing a conflict, a solution of F is found. If a conflict is produced, a conflict analysis is performed to derive a new clause by applying a sequence of resolution steps from the empty clause. The new clause is called a *learnt clause*.

Algorithm 1 depicts such a conflict analysis. The process to derive the learnt clause cl from the conflict is the so-called conflict-driven clause learning (CDCL). Then, the solver backtracks to the second highest level in cl (i.e., cancels all decisions made after the second highest level and the induced assignments), because cl becomes a unit clause in that level (i.e., all literals but one remain falsified after the backtracking), and the search continues from that level. We say that variables involved in the conflict analysis *participate* in the conflict, including the variables resolved on in Algorithm 1 and the variables in the learnt clause cl, The participation of a variable x in conflicts is said *activity* of x.

Algorithm 1: Analyse(cl, d), conflict analysis using 1UIP schema

```
Input: $cl$: an empty (or falsified) clause containing all
         its falsified literals,
       $d$: the level in which $cl$ is falsified
Output: $cl$: a falsified clause containing exactly one
          literal falsified in level $d$
begin
    while $cl$ contains at least two literals in level $d$ do
        $l \leftarrow$ the most recently assigned literal in $cl$;
        let $r = \{\neg l, a_1, \ldots, a_k\}$ be the unit clause
          falsifying $l$ ($a_1, \ldots, a_k$ are all falsified);
        $cl \leftarrow$ resolving $cl$ and $r$ on the variable of $l$;
```

From time to time, a CDCL solver cancels all its decisions, together with all assignments induced by them, and restarts. Such a CDCL search is shown to be equivalent to *general* (or unrestricted) resolution under a few assumptions [Beame *et al.*, 2003], which is exponentially stronger than regular resolution [Goerdt, 1993]. Before performing some restarts, a formula simplification procedure can be executed, such as clause vivification [Li *et al.*, 2020] and BVE. This is called *in-processing simplification.* We refer to [Biere *et al.*, 2021] for more details about CDCL solvers.

3 Review of Variable Elimination in Modern SAT Solvers

The well-known DP procedure [Davis and Putnam, 1960] solves a CNF formula F by eliminating variables one by one: to eliminate a variable x from F, remove $F_x \cup F_{\neg x}$ from F and add the resolvent set $F_x \bigotimes_x F_{\neg x}$ to F. The obtained formula is satisfiability equivalent to F. Note that $F_x \bigotimes_x F_{\neg x}$ does not contain x, nor any tautology. The resolution in the DP procedure is regular because when it derives an empty clause, a regular resolution tree can be constructed by tracing back the history of the empty clause. It is shown in [Rish and Dechter, 2000] that the DP procedure is very efficient for problems with low width, and that the advantages of DP and backtracking tree search can be combined using Bounded Variable Elimination (BVE). Inspired from this property, BVE was used as a pre-processing simplification technique in SAT solvers around twenty years ago [Subbarayan and Pradhan, 2004].

Example 1. *Let* F_x=$\{\{x, a_1, a_2\}, \{x, a_3\}, \{x, \neg b_1, \neg b_3\}\}$ *and* $F_{\neg x}$=$\{\{\neg x, b_1, b_2\}, \{\neg x, b_3\}\}$ *in F, eliminating x consists in replacing* $F_x \cup F_{\neg x}$ *in F by* $F_x \bigotimes_x F_{\neg x} = \{\{a_1, a_2, b_1, b_2\}, \{a_1, a_2, b_3\}, \{a_3, b_1, b_2\}, \{a_3, b_3\}\}$.

Variable elimination, as a simplification technique in a SAT solver, should be used carefully for four reasons: (1) Eliminating a variable x from F can increase the number of clauses in F by up to $|F_x \bigotimes_x F_{\neg x}| - |F_x| - |F_{\neg x}|$. So, repeating this procedure without limit can result in an exponential increase in the number of clauses. (2) The resolvent of two clauses can be longer than the two clauses. (3) After x is eliminated, it cannot be resolved on anymore, hurting the power of clause learning in the sequel. (4) when a variable is eliminated in an in-processing, all learnt clauses containing it are lost.

Variable elimination implemented in state-of-the-art SAT solvers usually only considers the first two reasons, by setting a bound denoted *clauseNbGrowthLimit* on $|F_x \bigotimes_x F_{\neg x}| - |F_x| - |F_{\neg x}|$ and/or a bound denoted *resolventLengthLimit* for the produced resolvents: x is not eliminated if $|F_x \bigotimes_x F_{\neg x}| - |F_x| - |F_{\neg x}| > clauseNbGrowthLimit$ or any resolvent in $F_x \bigotimes_x F_{\neg x}$ contains more than *resolventLengthLimit* literals. The variables are checked in a prefixed ordering and those satisfying the two bounds are eliminated one by one. Different variable orderings can lead to eliminating different subsets of variables, as illustrated in Example 2.

Example 2. *Assume that* $\{x, y, A_1\}$, $\{\neg x, y, A_2\}$, $\{\neg y, A_3\}$, $\{\neg y, A_4\}$ *and* $\{\neg y, A_5\}$ *are all the clauses containing x and y in a CNF formula F, where* A_1, A_2, A_3, A_4 *and* A_5 *are disjoint subsets of literals, and* clauseNbGrowthLimit $= 1$. *If we eliminate x first, all the clauses containing y in F are* $\{y, A_1 \cup A_2\}$, $\{\neg y, A_3\}$, $\{\neg y, A_4\}$ *and* $\{\neg y, A_5\}$. *Then y can also be eliminated successfully without violating* clauseNbGrowthLimit.

However, if we eliminate y first, all clauses containing x in F are $\{x, A_1 \cup A_3\}$, $\{x, A_1 \cup A_4\}$, $\{x, A_1 \cup A_5\}$, $\{\neg x, A_2 \cup A_3\}$, $\{\neg x, A_2 \cup A_4\}$ *and* $\{\neg x, A_2 \cup A_5\}$. *Then,* $|F_x \bigotimes_x F_{\neg x}| - |F_x| - |F_{\neg x}| >$ clauseNbGrowthLimit. *So, x cannot be eliminated after eliminating y.*

We review below some representative variable elimination procedures implemented in recent SAT solvers.

NiVER (Non-increasing Variable Elimination Resolution) [Subbarayan and Pradhan, 2004] is a pre-processor for eliminating variables before solving F by a CDCL solver. It implements the bounds *clauseNbGrowthLimit* and *resolventLengthLimit* using one constraint: the total number of literals in F should not increase when eliminating a variable x. In other words, x is eliminated only if the total number of literals in $\{F \setminus \{F_x \cup F_{\neg x}\}\} \cup \{F_x \bigotimes_x F_{\neg x}\}$ does not exceeds the total number of literals in F. Variables in F are checked in their natural integer order and eliminated one by one if the above condition is met. This process iterates until no variable can be removed.

MiniSAT 2.2.0 [Eén and Biere, 2005] eliminates variables with $clauseNbGrowthLimit = 0$ in a pre-processing. The *incremental variable elimination* [Nabeshima *et al.*, 2015] improves the BVE of MiniSAT 2.2.0 by relaxing *clauseNbGrowthLimit* gradually in iterative applications of BVE. In the first round, $clauseNbGrowthLimit = 0$. Then, it takes value $8, 16, \ldots$ in the following rounds, respectively. In each round, the variables are checked and eliminated in increasing order of their $|F_x| \times |F_{\neg x}|$. A variable x is not eliminated if $|F_x \bigotimes_x F_{\neg x}| - |F_x| - |F_{\neg x}| > clauseNbGrowthLimit$ or any resolvent in $F_x \bigotimes_x F_{\neg x}$ contains more than *resolventLengthLimit*(=20 by default) literals.

Unlike NiVER and MiniSAT, Kissat [Biere *et al.*, 2020] implements *in-processing* BVE to eliminate variables periodically (with increasing intervals) as the search proceeds. In each execution of BVE, variables are checked and eliminated in increasing order of $|F_x| \times |F_{\neg x}| + |F_x| + |F_{\neg x}|$ subject to the two bounds *clauseNbGrowthLimit* and *resolventLengthLimit*. As each in-processing BVE should be executed quickly, the *clauseNbGrowthLimit* bound is set to 0 and, then, if no variable can be eliminated in an execution, it is increased from 0 to 1, or multiplied by 2 in the next call to BVE, without exceeding 16. The initial *resolventLengthLimit* is set to 100 and increases gradually. In addition, a variable x is not eliminated if $|F_x| > 0$ and $|F_{\neg x}| > 0$, and $max(|F_x|, |F_{\neg x}|) > occlim$, where $occlim$ is initialized to 1000 and gradually increases. Finally, BVE stops if the total number of resolvents generated so far in this execution exceeds a limit.

The BVE procedure of MiniSAT and Kissat could be sketched in Algorithm 2, where F contains the original clauses in the input formula (irredundant clauses in the terminology of Kissat), eventually vivified, after removing those clauses already satisfied, and L contains the clauses learnt so far in the CDCL search (redundant clauses in the terminology of Kissat), which is empty in a pre-processing.

Algorithm 2: BVE($F, L, O, growth, length$), a generic BVE procedure for CDCL solvers

```
Input: $F$: a CNF formula not containing learnt clauses, $L$: a set of learnt clauses, $O$: a variable ordering, $growth$: clauseNbGrowthLimit, $length$: resolventLengthLimit
Output: $F \cup L$: a simplified CNF formula
begin
    $V \leftarrow$ list of variables in $F$ sorted in ordering $O$;
    while $V$ is not empty do
        $x \leftarrow$ the first variable in $V$;
        Remove $x$ from $V$;
        if $|F_x \bigotimes_x F_{\neg x}| - |F_x| - |F_{\neg x}| \leq growth$ and no resolvent in $F_x \bigotimes_x F_{\neg x}$ contains more than $length$ literals then
            $F \leftarrow \{F \setminus \{F_x \cup F_{\neg x}\}\} \cup \{F_x \bigotimes_x F_{\neg x}\}$;
            for each variable $y$ in $F_x \bigotimes_x F_{\neg x}$ do
                if $y$ is not in $V$ then
                    insert $y$ into $V$ and sort $V$ in $O$;
            $L \leftarrow L \setminus \{c | c \in L, x \in c \text{ or } \neg x \in c\}$;
            Remove subsumed clauses from $F \cup L$;
    return $F \cup L$;
```

O is the increasing order of $|F_x| \times |F_{\neg x}|$ in MiniSAT and $|F_x| \times |F_{\neg x}| + |F_x| + |F_{\neg x}|$ in Kissat, respectively.

Note that eliminating a variable x can change $|F_y|$ and $|F_{\neg y}|$ of a variable y. Consequently, if y was checked before x, but was not eliminated because of the *clauseNbGrowthLimit* and *resolventLengthLimit* bounds, it is inserted into V to be checked again if $|F_y|$ or $|F_{\neg y}|$ is decreased by the elimination of x. Note that when x is eliminated, all learnt clauses containing x or $\neg x$ are lost.

4 ESA: Variable Elimination Scheduled by Activity

It is well-known that the variables in a CNF formula are not equally important for solving it. In this section, we define a new variable ordering called *ESA* (variable Elimination Scheduled by Activity) to eliminate first the variables of smaller importance, which can reinforce and prevent the variables of greater importance from being eliminated by increasing the number of clauses containing them. We first describe the principle of ESA, and then its implementation in Cadical and three Kissat based solvers.

4.1 The Principle of ESA

Since CDCL under a few assumptions is shown to be equivalent to general (or unrestricted) resolution and BVE belongs to a process based on regular resolution, a CDCL solver implementing BVE could be thought as hybridizing or switching between unrestricted resolution and regular resolution. Recall that unrestricted resolution is exponentially stronger than regular resolution by allowing to resolve on the same variables several times in any path of a resolution tree. So, in theory, BVE could hurt the power of CDCL, because the learnt clauses containing the eliminated variables are lost and the eliminated variables cannot be resolved on anymore.

In practice, however, not all variables need to be resolved on several times to produce a short resolution proof when solving a large CNF formula. Indeed, we observe that a state-of-the-art CDCL solver could never need to resolve on a subset of variables when solving some CNF formulas: these variables never participate in any conflict nor in any learnt clause.

But what are the variables a state-of-the-art CDCL solver will resolve on several times? The answer is probably those variables participating frequently in recent conflicts, i.e., the variables with high activities. There are two reasons: (1) the solver usually selects the next decision variable among these variables; (2) they have tight relations with recently learnt clauses, because these learnt clauses contain them or have been obtained by resolving on them. Consequently, these variables have a high probability to participate in future conflicts to be resolved on during the conflict analysis.

Therefore, we define the ESA ordering to be the increasing order of the activities of the variables in recent conflicts. In practice, a decision heuristic in a state-of-the-art solver usually already quantifies these activities as a score. So, the ESA ordering is obtained by sorting the variables in the increasing order of their score induced by the decision heuristic. In the case where the solver switches between several decision heuristics, the score used to sort the variables is given by the current heuristic when BVE is called.

When there is no decision heuristic that can effectively quantify the participation of a variable in recent conflicts (e.g. in a pre-processing), the ESA ordering is the increasing order of the value $|F_x| \times |F_{\neg x}|$ in MiniSAT and $|F_x| \times |F_{\neg x}| + |F_x| + |F_{\neg x}|$ in Kissat.

We implement the ESA ordering in Algorithm 2, so that the low activity variables satisfying the two bounds *clauseNbGrowthLimit* and *resolventLengthLimit* are eliminated first, which may reinforce the high activity variables by increasing the number of clauses they occur in, possibly preventing them from being eliminated.

Note that if the number of clauses in which a high activity variable x occurs is not increased enough by the elimination of the low activity variables, x is still eliminated if it satisfies the bounds. So, the ESA ordering is a soft (or indirect) protection of the high activity variables, which is different from a hard protection (or direct) that, e.g., forbids the elimination of any variable with an activity higher than a threshold. Note that such a threshold probably should be instance-dependent and deserves future study.

4.2 Implementation of ESA in Cadical and Kissat Based Solvers

Cadical [Biere, 2018] is the CDCL solver that won the UNSAT Main track of the SAT Competition 2018. We use version 1.4.1, which is the specified version for the Hack track of the SAT Competition 2022. It switches between focused and stable modes. The decision heuristics VMTF (Variable Move-to-Front) [Biere and Fröhlich, 2015] and VSIDS (Variable State Independent Decaying Sum) [Moskewicz *et al.*, 2001] are used to select decision literals in focused mode and stable mode, respectively. The VMTF heuristic gives the highest score to the variables participating in the most recent conflict, breaking ties using their (non-)participation in the second most recent conflict, the remaining ties being broken using their (non-)participation in the third most recent conflict, etc.... The VSIDS heuristic initializes the score of every variable to 0. Let inc be a real such that $0 < inc < 1$. The VSIDS score s of a variable participating in the conflict number i is updated by $s \leftarrow s + (1/inc)^i$.

The interval between two calls to BVE in Cadical is measured in the number of conflicts, which grows in $O(n)$, where n is the number of calls to BVE so far.

Cadical_ESA is Cadical except that BVE is called with the ESA ordering in the stable mode where VSIDS is used as decision heuristic. In other words, the variables are sorted in the increasing order of their VSIDS score in the BVEs called during the stable mode. In the focused mode where VMTF is used as decision heuristic, the original increasing ordering of $|F_x| \times |F_{\neg x}| + |F_x| + |F_{\neg x}|$ remains unchanged in BVE.

Kissat is originally an optimized C version of Cadical by the same author with some changes to data structures. As it is improved, its difference with Cadical becomes bigger and bigger, especially after other authors add new solving techniques to it. We implemented ESA in the in-processing BVE of Kissat_MAB (KM for short), the winner of the Main track in the SAT Competition 2021; Kissat_MAB_HyWalk [Zheng *et al.*, 2022] (KM_HW for short) and Kissat_Inc [Chen *et al.*, 2022] (K_Inc for short), the best two solvers in the Main track of the SAT Competition 2022.

The three Kissat based solvers switch between focused mode, in which the VMTF decision heuristic is used, and stable mode, in which the solvers switch further between two decision heuristics VSIDS and CHB(Conflict History-based Branching) [Liang *et al.*, 2016] to select decision literals, by means of MAB (Multi-Armed Bandit) borrowed from reinforcement learning [Cherif *et al.*, 2021a], which is different from Cadical that uses only VSIDS in stable mode.

The CHB score $q(x)$ of a variable x is initialized to 0. After a unit propagation, $q(x)$ is updated, for each variable x assigned in this unit propagation, using the following Exponential Recency Weighted Average (ERWA) function, where $0 < \alpha < 1$ is a parameter and $r(x)$ is a reward:

$$q(x) \leftarrow (1-\alpha) \times q(x) + \alpha \times r(x) \tag{1}$$

Reward $r(x)$ is defined as follows:

$$r(x) = multiplier/(i - lastConflict(x) + 1) \tag{2}$$

where i is the number of conflicts so far, $lastConflict(x)$ is the last conflict number in which x participated, and $multiplier$ is 1 if the unit propagation that just terminated produces a conflict, otherwise it is 0.9.

The variables participating in recent conflicts are favored for two reasons: (1) $multiplier$ is greater for these variables, (2) $(i - lastConflict(x) + 1)$ is smaller.

The interval between two calls to BVE in the three Kissat based solvers is also measured in the number of conflicts, which grows in $O(n * log^2(n))$, where n is the number of calls to BVE so far. In addition, some planned BVEs are canceled if there are too many clauses. So, BVE is executed less frequently in the Kissat based solvers than in Cadical.

Solver	Solved	Avg(s)	PAR2
Cadical	4222	443	2890
Cadical_ESA	4230	441	2875
Kissat_MAB_HyWalk	4252	388	2798
Kissat_MAB_HyWalk_ESA	4269	410	2786
Kissat_Inc	4282	410	2764
Kissat_Inc_ESA	4291	405	2745
Kissat_MAB	4283	415	2767
Kissat_MAB_ESA	4302	420	2738

Table 1: Results of Solvers with and without ESA

KM_ESA (KM_HW_ESA, K_Inc_ESA) is KM (KM_HW, K_Inc) except that BVE is called with the ESA ordering in the stable mode. When the BVE procedure is called at the moment VSIDS (CHB) is used as decision heuristic, the ESA ordering is the increasing order of the VSIDS (CHB) score.

As in Cadical, when the three Kissat based solvers are in the focused mode where VMTF is used as decision heuristic, the original variable ordering by the value $|F_x| \times |F_{\neg x}| + |F_x| + |F_{\neg x}|$ remains unchanged in BVE. In other words, ESA does not use the VMTF score to sort the variables in BVE.

Note that ESA cannot be implemented in MiniSAT based solvers such as Glucose [Simon and Audemard, 2009] and MergeSat [Manthey, 2021], because these solvers do not implement in-processing BVE.

5 Experimental Results

We first show the effectiveness of ESA on the set of the 5355 instances of the Anniversary track in the SAT Competition 2022, augmented with the 320 instances from the Main track in the SAT Competition 2022 that were not included in the Anniversary track. In other words, the benchmark set includes all instances used in the Application, Crafted, and Main tracks of all SAT Competitions up to 2022, a total of 5675 distinct instances, so that the results can give a global performance evaluation. All experiments were run with Intel Xeon CPUs E5-2680@2.40GHz under Linux with 31GB of memory. The time limit for solving each instance is 5000s for each solver, as in the SAT Competition.

Then we conduct experiments to analyze the behavior of ESA and the reasons of its effectiveness.

5.1 Effectiveness of the ESA Procedure

Table 1 gives the number of instances solved by Cadical, KM, KM_HW, K_Inc and their versions with ESA, the average time to solve these solved instances and the PAR2 score (computed as in SAT competitions as the average solving time of all the 5675 instances, the solving time of an unsolved instance counted as 2×timeout, i.e. 10000s). As can be seen, a solver with ESA consistently solves more instances than the base solver without ESA. Recall that the only difference between each solver with ESA and its baseline without ESA is the variable ordering used in BVE, which indicates the effectiveness and the robustness of ESA, as these results show the compatibility of ESA with different techniques implemented in KM, KM_HW, K_Inc and Cadical. It is worth noting that

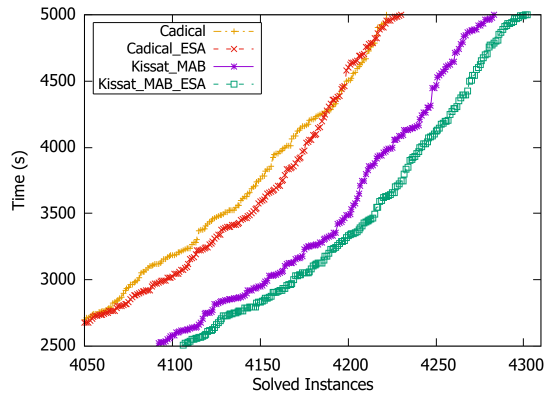


Figure 1: Cactus plots of different solvers. Each point (N, T) in a curve indicates the number N of instances solved within T secs by the corresponding solver. Recall that the frequence of BVE and decision heuristics are different in Cadical and KM.

KM_ESA and Cadical_ESA participated in the SAT Competition 2022: KM_ESA won the Anniversary track, and Cadical_ESA solved more instances than all other variants of Cadical in the Anniversary track of the competition.

More specifically, KM_ESA solves 57(38) instances more(less) than its baseline KM, among which 2,2,2,3 and 4 more instances in families Ptn, tseitingrid, fphp, preimage and hcp, respectively; and 2 and 2 fewer instances in LABS and mchess, respectively. The difference does not exceed 1 on other families of instances. No significant difference is observed between sat and unsat instances.

Figure 1 gives the cactus plots of Cadical_ESA, KM_ESA and their baselines, showing that Cadical_ESA solves more instances than Cadical at most points, and KM_ESA solves more instances than KM with any timeout.

Note that KM and Cadical are already highly optimized and it is extremely hard to make them solve even one more instance. As a matter of fact, the best two solvers in the Main track of the SAT Competition 2022, namely, KM_HW and K_Inc, solve exactly the same number of instances in that track. Only a few seconds in solving time allowed to distinguish them in the main track. Our results show that ESA would allow them to win more clearly.

Table 2 further confirms the effectiveness of ESA by comparing KM and KM_ESA with the following variants:

Kissat_MAB_NE (KM_NE for short), it is KM with BVE disabled (i.e., no elimination);

Kissat_MAB_ESRall (KM_ESRall), it is KM, but variables are checked and eliminated in a random ordering in BVE both in focused and stable mode;

Kissat_MAB_ESN (KM_ESN), it is KM, but the variables are sorted for BVE in the natural integer ordering in stable mode and in the increasing order of $|F_x| \times |F_{\neg x}| + |F_x| + |F_{\neg x}|$ in focused mode;

Kissat_MAB_ESNall (KM_ESNall), it is KM, but the variables are sorted for BVE in the natural integer ordering in both focused and stable modes;

Solver	Solved	Avg(s)	PAR2
Kissat_MAB_NE	4202	460	2936
Kissat_MAB_ESRall	4262	415	2802
Kissat_MAB_ESNall	4275	406	2773
Kissat_MAB_ESN	4277	419	2780
Kissat_MAB	4283	415	2767
Kissat_MAB_ESAfull	4287	426	2768
Kissat_MAB_Bin	4296	423	2750
Kissat_MAB_ESA	4302	420	2738

Table 2: Results of Variant Solvers

Kissat_MAB_Bin (KM_Bin), it is KM, but when a variable x is eliminated, let L_x^2 ($L_{\neg x}^2$) denote the set of binary learnt clauses containing x ($\neg x$), add $L_x^2 \bigotimes_x L_{\neg x}^2$ into L after removing all learnt clauses containing x or $\neg x$ from L.

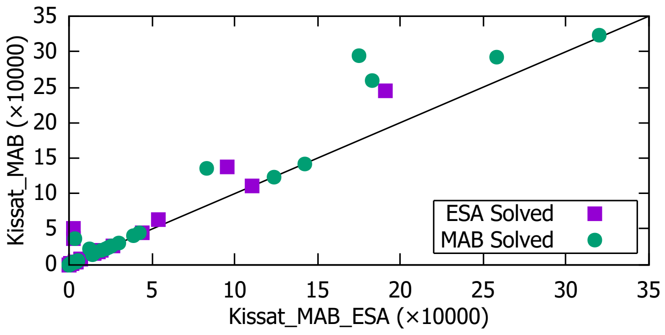
Kissat_MAB_ESAfull (KM_ESAfull), it is KM_ESA, but when BVE is called in focused mode where VMTF is used as decision heuristic, the variables are sorted in the increasing ordering of their VMTF score in BVE instead of $|F_x| \times |F_{\neg x}| + |F_x| + |F_{\neg x}|$.

A number of observations can be made from Table 2.

- In-processing BVE is effective, because KM solves 81 instances more than KM_NE.
- The natural integer ordering is better than the random ordering, but is worse than the $|F_x| \times |F_{\neg x}| + |F_x| + |F_{\neg x}|$ ordering, because KM solves 6 (8) instances more than KM_ESN (KM_ESNall).
- By keeping a part of resolution results so far, KM_Bin is significantly better than KM, but it does not reach the performance of KM_ESA. Indeed, the impact of an eliminated high activity variable x could be divided into two phases by the moment t it is eliminated. KM_Bin catches up a part of the impact of x before t, but loses its impact after t because it is eliminated. KM_ESA can have all its impact before and after t, because KM_ESA probably would not eliminate it.
- The VMTF score is not suitable to sort the variables in BVE, because KM_ESAfull is significantly worse than KM_ESA. It appears that not all effective decision heuristics in a CDCL solver are also effective to sort the variables in BVE, and only those decision heuristics reflecting accurately the activity of a variable in all recent conflicts are suitable to sort variables in BVE, which is not the case for VMTF.

5.2 Analyzing the Behaviour of ESA

In the Anniversary track of the SAT Competition 2022, KM_ESA solved 16 instances more than KM (named Kissat_MAB_UCB in the results) from a total of 5355 instances: KM_ESA solved 62 instances that KM did not solve (ESA Solved), and KM solved 46 instances that KM_ESA did not solve (MAB Solved). Figure 2 (left) compares the number of variables eliminated by the first BVE of the two solvers when solving these 108 instances. Since the two solvers have exactly the same behavior and follow the same search trajectory until the first BVE, the unique reason for the difference



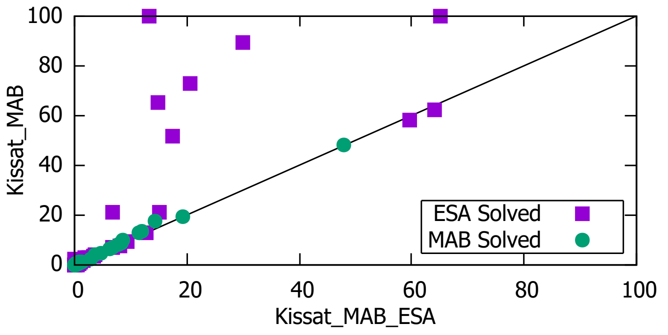
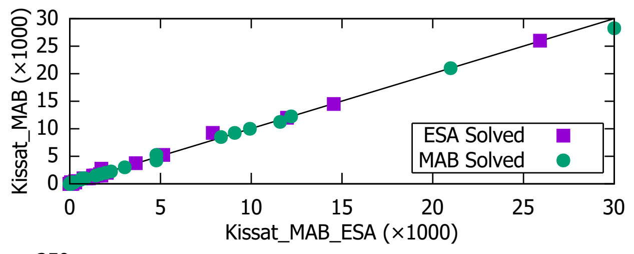


Figure 2: Scatter plot of the number of eliminated variables (left), and scatter plot of the number of learnt clauses lost per variable (right) by the 1st BVE of Kissat_MAB_ESA and Kissat_MAB.

in the first BVE is the variable ordering in BVE. It is clear that the ESA ordering eliminates fewer variables in the first BVE than the original variable ordering in MAB.

When a variable is eliminated in an in-processing, the learnt clauses containing this variable are lost. Figure 2 (right) compares the average number of lost learnt clauses per variable in KM_ESA and KM in the first BVE. KM_ESA loses fewer learnt clauses than KM when eliminating variables, which might partly explain the effectiveness of ESA.

Assume that KM_ESA (KM) calls k_1 (k_2) times BVE in total for solving an instance and $k = min(k_1, k_2)$. The above and below plots in Figure 3 compare respectively the average number of eliminated variables per BVE call, and the average number of lost learnt clauses per variable in these k BVE.



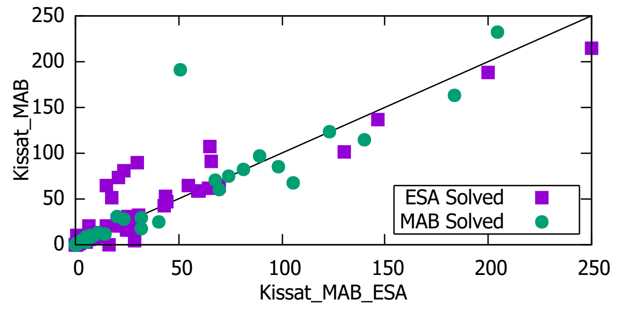


Figure 3: Scatter plot of the average number of eliminated variables per BVE call (above), and scatter plot of the average number of learnt clauses removed per variable among the first k BVE calls (below).

Although KM eliminates more variables in the first BVE, Figure 3 (above) shows that KM and KM_ESA eliminate roughly the same average number of variables per BVE call, meaning that KM_ESA eliminates more variables in the subsequent BVEs. This is reasonable for ESA, because it is probable that the important variables are not yet fully identified before the first BVE, so that it is better that the first BVE does not eliminate too many variables, and leave more variables to eliminate in subsequent BVEs for which the important variables are better recognized with their activities. Figure 3 (below), together with Figure 2 (right) and the good results of KM_Bin, suggests that one reason for which KM fails to solve an ESA solved instance might be that KM lost too many learnt clauses per eliminated variable.

6 Conclusions

While in-processing BVE is very useful in solving SAT, it currently has two drawbacks: (1) when a variable is eliminated, all learnt clauses containing it are also removed, so that a part of the results in the CDCL search so far is lost; (2) an eliminated variable cannot be resolved on anymore in the subsequent CDCL search, hurting the power of resolution in CDCL. In this paper we propose a new variable ordering, called ESA (variable Elimination Scheduled by Activity), to remedy the above two drawbacks. In practice, the ESA ordering is obtained by using the decision heuristic of the solver that is able to reflect accurately the participation of the variables in recent conflicts. Consequently, the low activity variables satisfying the bounds are eliminated first in BVE, providing a soft protection to the high activity variables.

The ESA ordering is implemented in Cadical and three winning solvers based on Kissat which call in-processing BVE differently and implement different decision heuristics and other techniques. Experimental results show that ESA consistently allows these solvers to solve more instances in the set of all the 5675 Crafted, Application and Main track instances of all SAT Competitions up to 2022, demonstrating the effectiveness and robustness of ESA. An empirical analysis is carried out to analyze the reasons of the effectiveness and robustness of ESA.

The principle behind ESA (i.e., high activity variables should be protected and reinforced) could be used in the future to improve other solving techniques in a CDCL solver. For example, a CDCL solver has to periodically remove a subset of learnt clauses. A heuristic could be defined to prevent removing the learnt clauses not containing any low activity variable.

Acknowledgments

This work has been supported by the French Agence Nationale de la Recherche, via AI chair ANR-19-CHIA-0013-01 (Massal'IA) and project ANR-20-ASTR-0011 (POSTCRYPTUM), by grants PID2019-111544GB-C21 and TED2021-129319B-I00 funded by MCIN/AEI/ 10.13039/501100011033, and partially supported by Archimedes Institute, Aix-Marseille University. We specially thank Université de Picardie Jules Verne for providing the Matrics Platform.

References

[Beame *et al.*, 2003] Paul Beame, Henry Kautz, and Ashish Sabharwal. Understanding the power of clause learning. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence, IJCAI*, pages 1194–1201, 2003.

[Biere and Fröhlich, 2015] Armin Biere and Andreas Fröhlich. Evaluating CDCL variable scoring schemes. In *Proceedings of the 18th International Conference on Theory and Applications of Satisfiability Testing, SAT*, pages 405–422, 2015.

[Biere *et al.*, 2020] Armin Biere, Katalin Fazekas, Mathias Fleury, and Maximillian Heisinger. CaDiCaL, Kissat, Paracooba, Plingeling and Treengeling entering the SAT Competition 2020. In *Proc. of SAT Competition 2020: Solver and Benchmark Descriptions*, pages 51–53. University of Helsinki, 2020.

[Biere *et al.*, 2021] Armin Biere, Marijn Heule, Hans van Maaren, and Toby Walsh, editors. *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*. IOS Press, 2021.

[Biere, 2017] Armin Biere. CaDiCaL, Lingeling, Plingeling, Treengeling, YalSAT Entering the SAT Competition 2017. In *Proc. of SAT Competition 2017: Solver and Benchmark Descriptions*, pages 14–15. University of Helsinki, 2017.

[Biere, 2018] Armin Biere. CaDiCaL, Lingeling, Plingeling, Treengeling and YalSAT Entering the SAT Competition 2018. In *Proc. of SAT Competition 2018: Solver and Benchmark Descriptions*, pages 13–14. University of Helsinki, 2018.

[Chen *et al.*, 2022] Zhihan Chen, Xindi Zhang, Shaowei Cai, and Pinyan Lu. CDCL Solvers with Improved Local Search Cooperation and Pre-processing. In *Proc. of SAT Competition 2022 – Solver and Benchmark Descriptions*, pages 37–38. University of Helsinki, 2022.

[Cherif *et al.*, 2021a] Mohamed Sami Cherif, Djamal Habet, and Cyril Terrioux. Combining VSIDS and CHB using restarts in SAT. In *Proceedings of the 27th International Conference on Principles and Practice of Constraint Programming, CP*, pages 20:1–20:19, 2021.

[Cherif *et al.*, 2021b] Mohamed Sami Cherif, Djamal Habet, and Cyril Terrioux. Kissat MAB: Combining VSIDS and CHB through multi-armed bandit. In *Proc. of SAT Competition 2021 – Solver and Benchmark Descriptions*, page 15. University of Helsinki, 2021.

[Cook, 1971] Stephen A. Cook. The complexity of theorem-proving procedures. In Michael A. Harrison, Ranan B. Banerji, and Jeffrey D. Ullman, editors, *Proceedings of the 3rd Annual ACM Symposium on Theory of Computing, May 3-5, 1971, Shaker Heights, Ohio, USA*, pages 151–158. ACM, 1971.

[Davis and Putnam, 1960] Martin Davis and Hilary Putnam. A computing procedure for quantification theory. *J. ACM*, 7(3):201–215, 1960.

[Eén and Biere, 2005] Niklas Eén and Armin Biere. Effective preprocessing in SAT through variable and clause elimination. In *Proceedings of the 8th International Conference on Theory and Applications of Satisfiability Testing, SAT*, pages 61–75, 2005.

[Goerdt, 1993] Andreas Goerdt. Regular resolution versus unrestricted resolution. *SIAM Journal on Computing*, 22(4):661–683, 1993.

[Huang and Yu, 1987] Wenqui Huang and Xiangdong Yu. A DNF without regular shortest consensus path. *SIAM J. Comput.*, 16(5):836–840, 1987.

[Järvisalo *et al.*, 2010] Matti Järvisalo, Armin Biere, and Marijn Heule. Blocked clause elimination. In Javier Esparza and Rupak Majumdar, editors, *Tools and Algorithms for the Construction and Analysis of Systems, 16th International Conference, TACAS 2010, Held as Part of the Joint European Conferences on Theory and Practice of Software, ETAPS 2010, Paphos, Cyprus, March 20-28, 2010. Proceedings*, volume 6015 of *Lecture Notes in Computer Science*, pages 129–144. Springer, 2010.

[Järvisalo *et al.*, 2012] Matti Järvisalo, Marijn Heule, and Armin Biere. Inprocessing rules. In Bernhard Gramlich, Dale Miller, and Uli Sattler, editors, *Automated Reasoning - 6th International Joint Conference, IJCAR 2012, Manchester, UK, June 26-29, 2012. Proceedings*, volume 7364 of *Lecture Notes in Computer Science*, pages 355–370. Springer, 2012.

[Li *et al.*, 2020] Chu-Min Li, Fan Xiao, Mao Luo, Felip Manyà, Zhipeng Lü, and Yu Li. Clause vivification by unit propagation in CDCL SAT solvers. *Artif. Intell.*, 279, 2020.

[Li *et al.*, 2022] Shuolin Li, Jordi Coll, Chu-Min Li, Mao Luo, Djamal Habet, and Felip Manyà. Solvers Cadical ESA and Kissat MAB ESA in 2022 SAT competition. In *Proc. of SAT Competition 2022 – Solver and Benchmark Descriptions*, pages 33–34. University of Helsinki, 2022.

[Liang *et al.*, 2016] Jia Hui Liang, Vijay Ganesh, Pascal Poupart, and Krzysztof Czarnecki. Exponential recency weighted average branching heuristic for SAT solvers. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence, AAAI*, pages 3434–3440, 2016.

[Luo *et al.*, 2017] Mao Luo, Chu-Min Li, Fan Xiao, Felip Manyà, and Zhipeng Lü. An effective learnt clause minimization approach for CDCL SAT solvers. In Carles Sierra, editor, *Proceedings of the Twenty-Sixth International Joint Conference on Artificial Intelligence, IJCAI*

2017, Melbourne, Australia, August 19-25, 2017, pages 703–711. ijcai.org, 2017.

[Manthey, 2021] Norbert Manthey. The MergeSat Solver. In Chu-Min Li and Felip Manyà, editors, *Theory and Applications of Satisfiability Testing - SAT 2021 - 24th International Conference, Barcelona, Spain, July 5-9, 2021, Proceedings*, volume 12831 of *Lecture Notes in Computer Science*, pages 387–398. Springer, 2021.

[Moskewicz *et al.*, 2001] Matthew W. Moskewicz, Conor F. Madigan, Ying Zhao, Lintao Zhang, and Sharad Malik. Chaff: Engineering an efficient SAT solver. In *Proceedings of the 38th Design Automation Conference, DAC*, pages 530–535, 2001.

[Nabeshima *et al.*, 2015] Hidetomo Nabeshima, Koji Iwanuma, and Katsumi Inoue. Glueminisat 2.2. 10 & 2.2. 10-5. *SAT Race*, 2015.

[Ostrowski *et al.*, 2002] Richard Ostrowski, Éric Grégoire, Bertrand Mazure, and Lakhdar Sais. Recovering and exploiting structural knowledge from CNF formulas. In Pascal Van Hentenryck, editor, *Principles and Practice of Constraint Programming - CP 2002, 8th International Conference, CP 2002, Ithaca, NY, USA, September 9-13, 2002, Proceedings*, volume 2470 of *Lecture Notes in Computer Science*, pages 185–199. Springer, 2002.

[Reeves and Heule, 2021] Joseph E. Reeves and Marijn J. H. Heule. The impact of bounded variable elimination on solving pigeonhole formulas. *Proceedings of Pragmatics of SAT*, 2021.

[Rish and Dechter, 2000] Irina Rish and Rina Dechter. Resolution versus search: Two strategies for SAT. *J. Autom. Reason.*, 24(1/2):225–275, 2000.

[Robinson, 1965] John Alan Robinson. A machine-oriented logic based on the resolution principle. *J. ACM*, 12(1):23–41, 1965.

[Silva and Sakallah, 1999] João P. Marques Silva and Karem A. Sakallah. GRASP: A search algorithm for propositional satisfiability. *IEEE Trans. Computers*, 48(5):506–521, 1999.

[Simon and Audemard, 2009] Laurent Simon and Gilles Audemard. Predicting learnt clauses quality in modern SAT solvers. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence, IJCAI*, pages 399–404, 2009.

[Subbarayan and Pradhan, 2004] Sathiamoorthy Subbarayan and Dhiraj K. Pradhan. NiVER: non increasing variable elimination resolution for preprocessing SAT instances. In *Proceedings of the 7th International Conference on Theory and Applications of Satisfiability Testing, SAT*, page 276–291, 2004.

[Zheng *et al.*, 2022] Jiongzhi Zheng, Kun He, Zhuo Chen, Jianrong Zhou, and Chu-Min Li. Combining Hybrid Walking Strategy with Kissat MAB, CaDiCaL, and LStech-Maple. In *Proc. of SAT Competition 2022 – Solver and Benchmark Descriptions*, pages 20–21. University of Helsinki, 2022.