Probabilistic Generalization of Backdoor Trees with Application to SAT

Alexander Semenov, Daniil Chivilikhin, Stepan Kochemazov, Ibragim Dzhiblavi

ITMO University, St. Petersburg, Russia
alex.a.semenov@itmo.ru, chivdan@gmail.com, stepan.kochemazov@itmo.ru, dzhiblavi@gmail.com

Abstract

The concept of Strong Backdoor Sets (SBS) for Constraint Satisfaction Problems is well known as one of the attempts to exploit structural peculiarities in hard instances. However, in practice, finding an SBS for a particular instance is often harder than solving it. Recently, a probabilistic weakened variant of the SBS was introduced: in the SBS, all subproblems must be polynomially solvable, whereas in the probabilistic SBS only a large fraction ρ of them should have this property. This new variant of backdoors called ρ-backdoors makes it possible to use the Monte Carlo method and metaheuristic optimization to find ρ-backdoors with ρ very close to 1, and relatively fast. Despite the fact that in a ρ-backdoor-based decomposition a portion of hard subproblems remain, in practice the narrowing of the search space often allows solving the problem faster with such a backdoor than without it. In this paper, we significantly improve on the concept of ρ-backdoors by extending this concept to backdoor trees: we introduce ρ-backdoor trees, show the interconnections between SBS, ρ-backdoors, and the corresponding backdoor trees, and establish some new theoretical properties of backdoor trees. In the experimental part of the paper, we show that moving from the metaheuristic search for ρ-backdoors to that of ρ-backdoor trees allows drastically reducing the time required to construct the required decompositions without compromising their quality.

Introduction

The concept of Backdoor sets to Constraint Satisfaction Problems (CSP) was introduced in the seminal paper (Williams, Gomes, and Selman 2003). Later, backdoors were actively researched in many papers, which studied both their theoretical properties, mainly from the point of view of structural and parameterized complexity (Kilby et al. 2005; Hemaspaandra and Narváez 2017, 2021; Fichte and Szeider 2011; Gaspers and Szeider 2012a,c,b; Misra et al. 2013; Gaspers and Kaploun 2022), and practical applications of backdoors for speeding up the solving of hard instances of combinatorial problems (Dilkina et al. 2009; Ferber et al. 2022; Khalil, Vaezipoor, and Dilkina 2022).

Here, the main practical interest is posed by Strong Backdoor Sets (SBS). The key element for defining an SBS is the notion of a *sub-solver*: a polynomial algorithm A that takes as input a variant of the original problem weakened by substitution of values of variables from some subset $B, B \subseteq X$ (here, X is the set of variables occurring in the considered set of constraints). If for each assignment of variables the algorithm A determines satisfiability/unsatisfiability (or consistency/inconsistency) of the corresponding set of constraints, then B is an SBS w.r.t. A.

Hereinafter, we will work with the Boolean Satisfiability Problem (SAT). It can be viewed as a special case of a CSP, where each variable from X has a domain of cardinality 2. Let C be a Boolean formula (usually, in Conjunctive Normal Form, CNF) over a set of variables X. Assume that we want to determine the satisfiability of C. It is clear that in this case the knowledge of some SBS B gives an upper bound on the hardness of C in the form of $p(|C|)\cdot 2^{|B|}$, where $p(\cdot)$ is some polynomial, and $|C|$ is the length of the binary description of C. However, the problem of finding an SBS with small cardinality is very hard (Dilkina, Gomes, and Sabharwal 2007; Hemaspaandra and Narváez 2017). Nevertheless, it is possible to formulate the problem of finding an SBS of minimum cardinality as the problem of minimization of a special function, which for an arbitrary $B \in 2^X$ checks whether this B is indeed an SBS, and if the answer is *yes* outputs the cardinality of B. Unfortunately, even computing the value of such a function on a particular input B is a very hard problem, since for each possible assignment of variables from B one has to evaluate whether substituting the corresponding values to the original formula results in a formula decidable by a sub-solver A.

One possible way to tackle this obstacle was described in (Semenov et al. 2022): it was proposed to search for such subsets B, for which not all formulas resulting from assigning values to variables from B in C are polynomially decidable, but their vast majority. In practice, one can use a complete SAT solver to solve the small fraction of simplified subproblems that cannot be solved by sub-solver A. The estimation of the fraction of polynomially decidable subproblems for a particular B can be done efficiently. As a result, one can consider the problem of finding such a probabilistic generalization of an SBS with relatively small cardinality and with a fraction of polynomially decidable subproblems close to 1. This problem can be viewed as an optimization problem for some pseudo-Boolean function which is not specified analytically (i.e. a black-box function).

The main goal of this paper is to adapt the concepts introduced in (Semenov et al. 2022) in order to increase the practical efficiency of finding probabilistic backdoors, since they provide a glimpse at some structural properties of original problems and can be used to increase the efficiency of their solving.

In the paper, we extend the probabilistic generalization of the SBS concept to tree-like representations of sets comprised of all possible assignments of variables from B. Here we mainly use the concept of *backdoor trees* introduced in (Samer and Szeider 2008). We formulate several new theoretical properties of backdoor trees which show that the combinatorial nature of backdoor sets and backdoor trees is surprisingly quite different. We then introduce probabilistic backdoor trees and study them from a theoretical perspective. In the experiments, we show that metaheuristic search for probabilistic backdoor trees is significantly faster than that for probabilistic variants of SBS thanks to potentially smaller number of calls to sub-solver A. The corresponding algorithms make it possible to efficiently find probabilistic backdoor trees for Boolean formulas with thousands of variables in mere seconds, and the resulting efficiency of solving with these backdoors is similar to that showcased in (Semenov et al. 2022), where several hours were used to find a single backdoor.

Thus, the main contributions of this paper are as follows: (I) we describe new combinatorial properties of backdoor trees; (II) we introduce ρ-backdoor trees, determine some of their properties and their connection to ρ-backdoors; (III) we show that it is possible to find good ρ-backdoor trees much faster than good ρ-backdoors, making the approach significantly more applicable to real-world problems.

Preliminaries

The Boolean Satisfiability Problem (SAT) is a well-known combinatorial problem, which can be viewed as a special case of the more general Constraint Satisfaction Problem (CSP). In the context of SAT, one operates with Boolean formulas, usually in Conjunctive Normal Form (CNF). A formula C in CNF is a conjunction of clauses, a clause is a disjunction of literals, and a literal is either a variable x or $\neg x$ (negation), where x is a Boolean variable (i.e. x has the range $\{0,1\}$). Let X be the set of all variables occurring in C. Assuming that an assignment of variables is defined in a standard way, see e.g. (Chang and Lee 1973), the assignment of variables from X such that C takes the value *True* on it is called a satisfying assignment and the corresponding formula is called *satisfiable*. If there are no assignments that satisfy C then it is called *unsatisfiable*.

Assume that B is an arbitrary subset of $X (B \subseteq X)$. Let us use the notation $\{0,1\}^{|B|}$ to denote the set of all possible assignments of variables from B. The notion of "backdoor" (in several aspects) was introduced in (Williams, Gomes, and Selman 2003). Hereinafter, we use the corresponding definition in the context of SAT. Let us present the following definition which precedes the backdoor concept.

Definition 1 (Williams, Gomes, and Selman 2003)**.** *In the context of SAT, for some Boolean formula C over variables X we refer to a deterministic polynomial algorithm A as to a sub-solver if the following requirements are satisfied.* ***1. Trichotomy****: given a CNF formula C, algorithm A either determines the satisfiability/unsatisfiability of formula C, or rejects C ("answer is not determined by A").* ***2. Trivial decidability****: A can recognize some trivial cases (such as, e.g., empty set of clauses).* ***3. Self-reducibility****: if A gives a solution of SAT for C, then A gives a solution of SAT for $C[\alpha/\{x\}]$, where $x \in X$ is any variable occurring in C, $\alpha \in \{0,1\}$ is any value of x, and $C[\alpha/\{x\}]$ is the CNF formula constructed by substituting the value α of x into C.*

As it was said above, several variants of the backdoor concept exist. Hereinafter, we will be interested only in the Strong Backdoor Set notion.

Definition 2 (Williams, Gomes, and Selman 2003)**.** *A set B, $B \subseteq X$, is called a Strong Backdoor Set (SBS) for formula C w.r.t. a sub-solver A, if for any $\beta \in \{0,1\}^{|B|}$ the sub-solver A recognizes the satisfiablity/unsatisfiability of formula $C[\beta/B]$ constructed from C by substituting into C the assignment β of all variables from B.*

The practical meaning of the concept of backdoors is well-illustrated by numerous practical examples. For instance, SAT encodings of symbolic verification and cryptanalysis problems often have very small SBSes (their size is fractions of a percent in relation to the total number of variables), if Unit Propagation is used as the sub-solver.

For example, such a situation takes place in the case when a CNF formula which encodes the logical equivalence (Drechsler, Junttila, and Niemelä 2021) of two circuits is constructed by applying Tseitin transformations (Tseitin 1970) to such circuits, augmented with a special functional block called a miter (Molitor and Mohnke 2007). Then, the set of variables corresponding to circuits' inputs forms an SBS w.r.t. Unit Propagation rule (Marques-Silva, Lynce, and Malik 2009), i.e. Strong Unit Propagation Backdoor Set (SUPBS).

The problem of finding the minimum or close to minimum SBS is very hard (Dilkina, Gomes, and Sabharwal 2007; Gaspers and Szeider 2012b; Hemaspaandra and Narváez 2017, 2021). And while we can employ metaheuristic algorithms for enumeration of different variants of B in the search space 2^X (power set of X), the barrier remains of having to call the sub-solver A on formulas $C[\beta/B]$ for all possible $\beta \in \{0,1\}^{|B|}$ in order to check if a set B is indeed an SBS. To overcome it, in (Semenov et al. 2022) the following probabilistic generalization of SBS was introduced.

Definition 3 (Semenov et al. 2022)**.** *Consider an arbitrary CNF formula C over variables X, a sub-solver A, and a fixed $\rho \in [0,1]$. An arbitrary set B is called a ρ-backdoor for C w.r.t. A, if A decides the satisfiability/unsatisfiability of $C[\beta/B]$ for a fraction of all possible $\beta \in \{0,1\}^{|B|}$ which is no smaller than ρ.*

We use the notation $C[\beta/B] \in S(A)$ for cases when A recognizes the satisfiability/unsatisfiability of $C[\beta/B]$, and write $C[\beta/B] \notin S(A)$ when A rejects $C[\beta/B]$. Let ρ^*_B be the exact fraction of $\beta \in \{0,1\}^{|B|}$ that $C[\beta/B] \in S(A)$. Then, following (Semenov et al. 2022), we can estimate ρ^*_B with any predetermined accuracy using a Monte Carlo test.

In more detail, let us connect with B a probability space $\Sigma_B = \langle \Omega, \mathfrak{U}, \Pr \rangle$, where the sample space Ω is the set $\{0,1\}^{|B|}$, $\mathfrak{U}$ is a σ-algebra which in our case is the set 2^{Ω} (power set of Ω), and $\Pr\colon \mathfrak{U} \to [0,1]$ is the probability function. Also we must require that Kolmogorov's axioms hold (see Feller 1971). Define on $\Omega = \{0,1\}^{|B|}$ a uniform probability distribution, i.e. assign to each elementary event $\beta \in \{0,1\}^{|B|}$ the probability $p(\beta) = 1/2^{|B|}$. Let us connect with Σ_B a random variable ξ_B, $\xi_B\colon \Omega \to \{0,1\}$, which is specified in the following manner: $\xi_B(\beta) = 1$ if $C[\beta/B] \in S(A)$, and $\xi_B(\beta) = 0$ if $C[\beta/B] \notin S(A)$. It is clear that ξ_B is a Bernoulli random variable, i.e. it has the range (spectrum) $\{1,0\}$, success probability ρ^*_B, and probability distribution $P(\xi_B) = \{\rho^*_B, 1-\rho^*_B\}$. Thus, for the expected value of ξ_B we have that $E[\xi_B] = \rho^*_B$.

Now, we can estimate ρ^*_B using independent observations $\xi_1, \ldots, \xi_N$ of random variable ξ_B and Chernoff's bound (see e.g. (Motwani and Raghavan 1995)). More precisely, we use the following form of this bound which can be found in, e.g. (Karp, Luby, and Madras 1989):

$$\Pr\left[\left|\frac{1}{N}\sum_{j=1}^{N}\xi_j - \rho^*_B\right| \le \varepsilon\right] \ge 1 - 2e^{-N\varepsilon^2/4}. \quad (1)$$

The relation (1) expresses some (ε,δ)-approximation of ρ^*_B: usually, this term is used regarding any relation of the form $\Pr[|\widetilde{\nu} - \nu| \le \varepsilon] \ge 1-\delta$, where ν is an estimated parameter and $\widetilde{\nu}$ is an estimation of ν calculated in some probabilistic experiment. Values ε and $1-\delta$ are usually referred to as *tolerance* and *confidence level*, respectively.

Using (1) it is not hard to see that for any fixed $\varepsilon, \delta \in (0,1)$ we can construct an (ε,δ)-approximation of ρ^*_B as $\widetilde{\rho}^*_B = \frac{1}{N}\sum\limits_{j=1}^{N}\xi_j$ if we take any $N\colon N \ge \frac{4\ln(2/\delta)}{\varepsilon^2}$. And if we want to obtain an approximation $\Pr[|\widetilde{\rho}^*_B - \rho^*_B| \le \varepsilon/2] \ge 1-\delta$, it is sufficient to choose $N = \left\lceil \frac{16\ln(2/\delta)}{\varepsilon^2} \right\rceil$.

Let us define the following Monte Carlo test. Fix $\varepsilon, \delta \in (0,1)$ and $N = \left\lceil \frac{16\ln(2/\delta)}{\varepsilon^2} \right\rceil$. Make N independent observations of random variable $\xi_B\colon \xi_1, \ldots, \xi_N$, compute $\widetilde{\rho}^*_B = \frac{1}{N}\sum\limits_{j=1}^{N}\xi_j$. If $\widetilde{\rho}^*_B \in [1-\frac{\varepsilon}{2}, 1]$, then conclude that B passes the test, otherwise B fails the test.

Let us suppose that B passes the described Monte Carlo test. But then ρ^*_B deviates from $\widetilde{\rho}^*_B$ by at most $\varepsilon/2$ with probability at least $1-\delta$, and thus with probability no smaller than $1-\delta$ the conclusion that $\rho^*_B \in [1-\varepsilon, 1]$ is correct.

As shown in (Semenov et al. 2022) using SAT as an example, to search for ρ-backdoors with ρ close to 1 one can use metaheuristic pseudo-Boolean optimization: among possible subsets of the set X we search for a set B that provides a minimum of an objective fitness function that depends on $\widetilde{\rho}^*_B$ and the cardinality of B. The value of this fitness function in each point $B \in 2^X$ is calculated efficiently: in the general case, in time limited by $p(|C|) \cdot \frac{16\ln(2/\delta)}{\varepsilon^2}$, where $p(\cdot)$ is some polynomial, and $|C|$ is the length of the binary encoding of the CNF formula C. Thus, the efficiency of the search depends mainly on the used metaheuristic optimization algorithm. In (Semenov et al. 2022), some variant of a genetic algorithm (Luke 2013) was used for this.

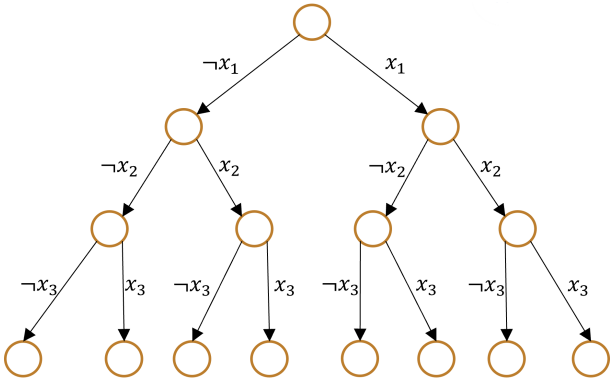

Figure 1: Example of a tree for set $B = \{x_1, x_2, x_3\}$

Since the estimation of ρ^*_B for a given set B can be constructed efficiently (using the described Monte Carlo test), one can employ metaheurisitic optimization to efficiently find ρ-backdoors with ρ close to 1 for formulas with thousands of variables. When some ρ-backdoor has been found, we can use the sub-solver A to solve $\rho^*_B \cdot 2^{|B|}$ problems $C[\beta/B] \in S(A)$, and apply some complete SAT solver to the remaining $(1-\rho^*_B)\cdot 2^{|B|}$ problems $C[\beta/B] \notin S(A)$.

Backdoor Trees and Some New Theoretical Facts About Them

The idea of working with a backdoor as a tree is contained in (Williams, Gomes, and Selman 2003): if B is some backdoor (e.g. SBS), we can branch on variables from B, selecting them in some fixed order. But in the explicit form, the backdoor tree concept was formulated in (Samer and Szeider 2008). Its main theoretical results are related to parameterized complexity: in particular, in that paper it was shown that the problems of checking for a specific CNF formula C and parameter k whether there exists a Horn-Backdoor tree or a 2CNF-Backdoor tree for C with at most k leaves are fixed-parameter tractable.

The main purpose of our research is to construct a computational algorithm which allows efficiently finding probabilistic backdoor trees for SAT instances. Along the way, we establish several surprising theoretical facts about the connection between SBSes and their tree-like counterparts.

Consider a CNF formula C over variables X and an arbitrary set $B \subseteq X\colon |B| = s$. It is clear that $\{0,1\}^{|B|}$ can be seen as a perfect (complete) binary tree in which all vertices except the leaves correspond to variables from B, and each path from the root to a leaf corresponds to a specific $\beta \in \{0,1\}^{|B|}$. An example of such a tree is shown in Fig. 1.

Let us put each leaf of the obtained tree in correspondence with the result of applying the algorithm A to the corresponding formula $C[\beta/B]$: if $C[\beta/B] \in S(A)$, then we assign the symbol "A" to the corresponding leaf, if $C[\beta/B] \notin S(A)$, then we attribute the symbol "$\times$" to this

leaf. We denote the resulting tree by T_B, and call its leaves *terminal vertices*. By virtue of the above, every path in T_B has length s, and T_B contains 2^s paths, thus the height of T_B is s. Also note that if B is an SBS, then all terminal vertices of T_B are assigned the symbol "A".

Note that in a tree T_B (such as the one shown in Fig.1), all vertices at a fixed depth are assigned the same variable from B, and thus a particular tree T_B corresponds to a particular order on B. We will refer to such a tree as to an *ordered tree*, and will use for it the notation $T_{B,\tau}$, where τ is a specific order. The above implies that any set $\{0,1\}^{|B|}$ can be represented in the form of a tree $T_{B,\tau}$. Thus, trees corresponding to different orders differ only in the labeling of their vertices (excluding leaves) and, therefore, there are $s!$ different trees $T_{B,\tau}$ for each set B. And though in the original paper (Samer and Szeider 2008) the backdoor trees are considered without connection with order, we further will deal only with ordered trees, since the order allows us to establish some important facts related to the probabilistic properties of the trees.

As it was noted in (Dilkina et al. 2009), for some orders it is possible that in some branches of the tree the sub-solver A will solve the problem obtained by substituting the first $r, r < s$, variables. Let τ be some order on B and $T_{B,\tau}$ be a tree corresponding to this order and representing $\{0,1\}^{|B|}$. We will traverse the tree from the root to the terminal vertices, and for each internal vertex v check whether $C[\beta_v/B] \in S(A)$, where β_v denotes a partial assignment of variables, which corresponds to a path from the root to vertex v in $T_{B,\tau}$. If $C[\beta_v/B] \in S(A)$, then the vertex becomes terminal and is marked with the symbol "A". Denote by $\widetilde{T}_{B,\tau}$ the tree obtained as a result of the described procedure. The following fact is obviously true (essentially, this is a slight reformulation of Lemma 2 from (Samer and Szeider 2008)).

Proposition 1. *The set B is an SBS for C w.r.t. A if and only if for any order τ on B, the tree $\widetilde{T}_{B,\tau}$ is a full binary tree (i.e. each non-terminal vertex has two children), each terminal vertex of which is assigned the symbol "A".*

The main result presented below is valid for the trees $\widetilde{T}_{B,\tau}$, in which some of the leaves are marked with symbol $\times$. In order to distinguish these trees from backdoor trees in the sense of (Samer and Szeider 2008), hereinafter let us refer to the latter as *SBS trees*.

Definition 4. *If B is some SBS, a tree $\widetilde{T}_{B,\tau}$ is called an SBS tree.*

The use of a tree $\widetilde{T}_{B,\tau}$ instead of an SBS B can speed up SAT solving for C due to fewer sub-solver calls. Thus, we would like to obtain a tree $\widetilde{T}_{B,\tau}$ for the SBS B, which can be considered minimal in some sense. First, we will be interested in the upper bounds for the complexity of finding such trees. And in this context, as we will see, the connection of the minimal/minimum tree $\widetilde{T}_{B,\tau}$ with the concept of the minimum SBS is important. It seems quite natural to define the minimum SBS in the following manner.

Definition 5. *Let C be a CNF formula over the set of variables X. An SBS for C of the minimum possible cardinality is called a minimum SBS.*

To search for a minimum SBS, one can use the algorithm described in the article (Williams, Gomes, and Selman 2003) (hereinafter referred to as the WGS-algorithm). It enumerates all possible subsets of X of sequentially increasing cardinality (starting from sets of cardinality 1) and for each $B, B \subseteq X$, checks for all $\beta \in \{0,1\}^{|B|}$ if $C[\beta/B]$ is solvable by sub-solver A. The first SBS found in this way is a minimum one. In the worst-case scenario for each $B \subset X$, the situation $C[\beta/B] \in S(A)$ will take place for the first $2^{|B|}-1$ assignments β (in some order on $\{0,1\}^{|B|}$), and for the last assignment β we will have $C[\beta/B] \notin S(A)$. And only when $B = X$ will we get $C[\beta/B] \in S(A)$ for each $\beta \in \{0,1\}^{|X|}$. Therefore, in the worst-case scenario of this algorithm, the sub-solver A is called $\sum_{k=1}^{n} \binom{n}{k} 2^k = \mathcal{O}(3^n)$ times, where $n = |X|$. However, if C has a relatively small minimum SBS, then, as shown in (Williams, Gomes, and Selman 2003), the WGS-algorithm can solve SAT asymptotically faster than in time $poly(|C|) \cdot 2^n$.

It is well known that the main parameters by which a tree is usually evaluated are its height and the number of vertices. It is easy to establish the validity of the following fact.

Proposition 2. *The minimum height of an SBS tree among all SBSes is $s^* = |B^*|$, where B^* is the minimum SBS.*

Proof. Let $B^*\colon |B^*| = s^*$, be some minimum SBS. Obviously, $h(\widetilde{T}_{B^*,\tau}) = s^*$ for any order τ on B^*, where $h(\cdot)$ denotes the height of the considered tree. Indeed, if we assume the opposite, then there exists such an order τ' that each path in the tree $\widetilde{T}_{B^*,\tau'}$ has length smaller than s^*, and, thus, there exists an SBS $B'\colon B' \subset B^*$, but this contradicts the fact that B^* is a minimum SBS. For the same reasons there cannot exist an SBS $B'\colon |B'| > |B^*|$ such that for some order τ on B' it would hold that $h(\widetilde{T}_{B',\tau}) < s^*$. □

In the context of the above, the following definition seems practically reasonable.

Definition 6. *Denote by $|T|$ (size of T) the number of all vertices in T, where T is an arbitrary tree. For an arbitrary $B, B \subseteq X$ (which is not necessarily an SBS) let us call $\widetilde{T}_{B,\tau}$ the smallest tree for B, if it has the minimum size $|\widetilde{T}_{B,\tau}|$ across all possible orders τ on B. Denote the smallest tree for B as $\widetilde{T}^*_B$.*

Note that following (Samer and Szeider 2008) we could have defined the size of a tree as the number of its leaves. However, the definition we employ is more convenient for our purposes because later we will estimate the number of calls to A at each vertex of our tree (except its root).

Let us consider the problem of finding the smallest tree among all possible SBS-trees. Denote such a tree as $\widetilde{T}^*$. Let B^* be an SBS of the smallest cardinality (minimum SBS) and $s^* = |B^*|$. Proposition 2 implies that $h(\widetilde{T}^*) \geq h(\widetilde{T}^*_{B^*})$ and thus the following inequalities from (Samer and Szeider 2008) hold:

$$2s^* + 1 \leq |\widetilde{T}^*| \leq 2^{s^*+1} - 1. \qquad (2)$$

As we have seen, the minimum SBS tree has the minimal height among all possible SBS trees. Surprisingly, we are not aware of any similar statements about the interconnection between the values $|\widetilde{T}^*|$ and $|\widetilde{T}^*_{B^*}|$ which would give us more explicit information than (2). Indeed, firstly, for different minimum SBSes (with the same cardinality) their smallest trees can have different sizes. Moreover, there are no obstacles for the existence of an SBS $B\colon |B| > |B^*|$, $B^* \not\subset B$ such that $|\widetilde{T}^*_B| < |\widetilde{T}^*_{B^*}|$.

As implied from the above, we can hardly say anything definite about the complexity of searching for the smallest SBS tree $\widetilde{T}^*$. And taking this into account, we further consider the problem of searching for a smallest tree of the minimum SBS (i.e. for $\widetilde{T}_{B^*}$). The corresponding result is given by the following theorem which is in spirit of Theorem 4.1 from (Williams, Gomes, and Selman 2003).

Theorem 1. *Let C be some set of constraints over Boolean variables X, $|X| = n$, and A be some sub-solver. For any fixed $q \geq 2$, let us suppose that there exists an SBS for C w.r.t. A, such that $|B| \leq n/q$. Then, the smallest tree for the minimum SBS can be found in time:*

$$\mathcal{O}^*\Big(2^{\frac{n}{q}(\log n - \log q - \log(e/2))}\Big), \tag{3}$$

where $\mathcal{O}^*$ means "big $\mathcal{O}$ up to some polynomial factor".

Proof sketch. First, we find the minimum SBS using the WGS-algorithm. Then, we construct trees $\widetilde{T}_{B^*,\tau}$ by enumerating all possible orders on B^*. Denote by $r(B^*)$ the number of times we check whether $C[\beta/B] \in S(A)$ when searching for B^* with the WGS-algorithm. By virtue of the assumptions in the condition of the theorem and the properties of binomial coefficients, we have:

$$r(B^*) \leq \sum_{i=1}^{\lceil n/q \rceil} \binom{n}{i} \cdot 2^i \leq \left\lceil \frac{n}{q} \right\rceil \cdot \binom{n}{\lceil n/q \rceil} \cdot 2^{\lceil n/q \rceil}. \tag{4}$$

After carefully estimating the value on the right-hand side of (4) with the use of Stirling's formula, we have the following estimation (here, $\log x$ denotes the binary logarithm):

$$r(B^*) \leq p(n) \cdot 2^{n\left(\frac{1}{q} + \log q - \frac{q-1}{q}\log(q-1)\right)}, \tag{5}$$

where $p(\cdot)$ is some polynomial.

There are $s!$, $s = |B|$, possible trees $\widetilde{T}_{B,\tau}$. The sub-solver A is called no more than $2^{s+1} - 1$ times at the tree edges. We again apply the Stirling's formula to $s!$ and construct the following upper bound for the complexity of finding the smallest tree for B^* when all possible orders on B^* are tried:

$$p(n) \cdot 2^{\frac{n}{q}(\log n - \log q - \log(e/2))}. \tag{6}$$

Taking into account the fact that (6) grows faster than the right-hand side of (5) with the increase of n, we can conclude that the statement of the theorem is true. □

Probabilistic Backdoor Trees

Our main practical interest is to transfer the concept of probabilistic backdoors to their tree-like counterparts. However, at first glance, it is not entirely clear how to determine the probability space in such a case. In fact, we must do this in such a way that the resulting distribution corresponds to some easily reproducible probabilistic experiment.

So, let C be an arbitrary CNF formula over variables X, A be some sub-solver, $B, |B| = s$, be an arbitrary subset of X, and $\widetilde{T}_{B,\tau}$ be some tree with an order τ, which was constructed in the way described above (not necessarily an SBS-tree). Recall that $\widetilde{T}_{B,\tau}$ is a full binary tree with height $h(\widetilde{T}_{B,\tau}) \leq s$. Link with each path π from the root to a terminal vertex in $\widetilde{T}_{B,\tau}$ the number $1/2^{L(\pi)}$, where $L(\pi)$ is the length of the path π in the standard sense (see e.g. (Cormen, Leiserson, and Rivest 1990)). Let $\beta(\pi)$ be an assignment of variables from B w.r.t. an order τ which corresponds to the path π. Define: the sample space as $\Omega = \{\beta(\pi)\}_{\pi \in \widetilde{T}_{B,\tau}}$, where the probability of an elementary event $\beta(\pi)$ is defined as $p(\beta(\pi)) = 1/2^{L(\pi)}$; the σ-algebra as $\mathfrak{U} = 2^\Omega$, and the probability function as $\Pr\colon \mathfrak{U} \to [0,1]$. Let us establish the following fact.

Proposition 3. $\Sigma_{\widetilde{T}_{B,\tau}} = \langle \Omega, \mathfrak{U}, \Pr \rangle$ *is a probability space.*

Proof sketch. In order to prove the proposition, we have to establish that all Kolmogorov's axioms (Feller 1971) hold. Axioms 1 and 3 hold obviously. Let us show that $\sum_{\pi \in \widetilde{T}_{B,\tau}} p(\beta(\pi)) = 1$. Let us prove this fact by induction on the height h of tree $\widetilde{T}_{B,\tau}$: $B = \{x_1^B, \ldots, x_s^B\}$, $\tau\colon x_1^B \prec \cdots \prec x_s^B$, $s \geq 1$. A full binary tree of height 1 is unique (contains two paths of length 1), each path is assigned a probability $1/2$, respectively, so for this case the conjecture is obviously valid. Let us assume that the conjecture is true for an arbitrary $h = k$ ($k \geq 1$) and consider the case $h = k + 1$. Suppose that the root of the tree $\widetilde{T}_{B,\tau}$ is assigned by variable x_1^B. If one of the children of the root is a leaf, then it is obvious that the path from root to this leaf is assigned the probability 1/2. Let us assume that one of two sub-trees of the considered tree has height $h = k$, i.e. it is a full binary tree $\widetilde{T}_{B',\tau'}\colon B' = B \setminus \{x_1^B\}$, where $\tau'\colon x_2^B \prec \cdots \prec x_s^B$. If we had constructed the space $\Sigma_{\widetilde{T}_{B',\tau'}}$, then by the inductive assumption the relation $\sum_{\pi \in \widetilde{T}_{B',\tau'}} p(\beta(\pi)) = 1$ would hold. However, any path from $\widetilde{T}_{B',\tau'}$ of length L corresponds to a path of length $L+1$ in tree $\widetilde{T}_{B,\tau}$ and, therefore, the sum of values $p(\beta(\pi))$ over all such paths in $\widetilde{T}_{B,\tau}$ is $1/2$. But then $\sum_{\pi \in \widetilde{T}_{B,\tau}} p(\beta(\pi)) = 1$ for the case $h = k + 1$. □

Denote by $\pi_A(\widetilde{T}_{B,\tau}) \in \mathfrak{U}$ (where $\mathfrak{U}$ is the σ-algebra of space $\Sigma_{\widetilde{T}_{B,\tau}}$) the event that the path π in $\widetilde{T}_{B,\tau}$ ends with the symbol "A". It is obvious that the probability $\Pr[\pi_A(\widetilde{T}_{B,\tau})]$ of such an event within the space $\Sigma_{\widetilde{T}_{B,\tau}}$ is the sum of values $p(\beta(\pi))$ over all paths π ending with the symbol "A".

Definition 7. *A ρ-backdoor tree $\widetilde{T}_{B,\tau}$ (B is not necessarily an SBS) is a tree with* $\mathrm{Pr}[\pi_A(\widetilde{T}_{B,\tau})] \geq \rho$, $\rho \in [0,1]$.

As noted above, the value ρ^*_B can be considered as the probability that an assignment β selected from $\{0,1\}^{|B|}$ w.r.t. the uniform distribution gives a formula $C[\beta/B]$: $C[\beta/B] \in S(A)$. But then, the analogue of ρ^*_B in the context of the above distribution on $\widetilde{T}_{B,\tau}$ (w.r.t. some fixed τ) is the value $\mathrm{Pr}[\pi_A(\widetilde{T}_{B,\tau})]$. Let us establish the following fact.

Theorem 2. *Let B be some ρ-backdoor for C w.r.t. A. Then, for an arbitrary fixed τ, the following equality holds:*

$$\rho^*_B = \mathrm{Pr}[\pi_A(\widetilde{T}_{B,\tau})].$$

Proof sketch. Probability ρ^*_B is defined in the probability space comprised by sample space $\{0,1\}^{|B|}$ with uniform distribution defined on it, and ρ^*_B can be expressed as:

$$\rho^*_B = 1 - \mathrm{Pr}[C[\beta/B] \notin S(A)].$$

But it is obvious that

$$\mathrm{Pr}[C[\beta/B] \notin S(A)] = |\{\beta : C[\beta/B] \notin S(A)\}| \cdot 1/2^{|B|},$$

i.e. this is the sum of probabilities assigned to all such β's for which $C[\beta/B] \notin S(A)$. For probability $\mathrm{Pr}[\pi_A(\widetilde{T}_{B,\tau})]$ we can write a similar equation:

$$\mathrm{Pr}[\pi_A(\widetilde{T}_{B,\tau})] = 1 - \mathrm{Pr}[\pi_\times(\widetilde{T}_{B,\tau})],$$

where $\pi_\times(\widetilde{T}_{B,\tau})$ denotes the event comprised of all paths in $\widetilde{T}_{B,\tau}$ which end with "$\times$". But the number of such paths is exactly $|\{\beta \in \{0,1\}^{|B|} : C[\beta/B] \notin S(A)\}|$, and each such path has length $|B|$ in the tree $\widetilde{T}_{B,\tau}$ and, thus, this path has probability $1/2^{|B|}$ in the probability space $\Sigma_{\widetilde{T}_{B,\tau}}$. From the above, we conclude that the theorem statement holds. □

Note that this theorem looks quite surprising: in fact, it means that the probability $\mathrm{Pr}[\pi_A(\widetilde{T}_{B,\tau})]$ does not depend on the order τ. At the same time it should be noted that the number of calls to A during the traversal of a tree $\widetilde{T}_{B,\tau}$ can be significantly smaller than when we call A on $C[\beta/B]$ for each $\beta \in \{0,1\}^{|B|}$.

Another important property of the probability space $\Sigma_{\widetilde{T}_{B,\tau}}$ is that we can naturally associate a computationally reproducible probabilistic experiment with this space and, due to this, determine an observable random variable on $\Sigma_{\widetilde{T}_{B,\tau}}$. Let us establish this fact.

Without loss of generality, consider the order $\tau\colon x^B_1 \prec \cdots \prec x^B_s$ on the set $B = \{x^B_1, \ldots, x^B_s\}$. Let us choose a value of x^B_1 by tossing a fair coin: in this experiment, $\mathrm{Pr}[x^B_1 = 0] = \mathrm{Pr}[x^B_1 = 1] = 1/2$. A specific value $x^B_1 = \alpha$ corresponds to the choice of a specific branch of the tree $\widetilde{T}_{B,\tau}$ emanating from the root, that is, its specific full sub-tree. The root v of this sub-tree is assigned a variable x^B_2. Let us call the sub-solver A for the formula $C[\alpha/x^B_1]$, and if A rejects the corresponding formula, we will carry out a similar independent experiment at vertex v: we will choose the value of the variable x^B_2. Let us continue moving along the tree $\widetilde{T}_{B,\tau}$ in this way until either calling A solves the corresponding formula, or there are no variables from B which are not assigned a value. Since all coin tossing experiments are independent, an arbitrary path π in the tree $\widetilde{T}_{B,\tau}$ corresponds to the product of the probabilities of single coin toss experiments carried out on the edges of the tree $\widetilde{T}_{B,\tau}$ that form the path π. Thus, an arbitrary path π corresponds to the probability $p(\beta(\pi)) = 1/2^{L(\pi)}$ and, therefore, the described experiment corresponds to the probability space $\Sigma_{\widetilde{T}_{B,\tau}}$.

Associate with probability space $\Sigma_{\widetilde{T}_{B,\tau}}$ a random variable $\xi_{\widetilde{T}_{B,\tau}} : \Omega \to \{0,1\}$, which we define as follows. Consider an arbitrary elementary event from Ω, that is, a path π in $\widetilde{T}_{B,\tau}$. We will traverse it (w.r.t. order τ), calling at each vertex (excluding the root) the sub-solver A. If at some vertex of π the sub-solver A solves the corresponding CNF formula, then $\xi_{\widetilde{T}_{B,\tau}}(\pi) = 1$, and $\xi_{\widetilde{T}_{B,\tau}}(\pi) = 0$ otherwise. Thus, $\xi_{\widetilde{T}_{B,\tau}}$ is a Bernoulli random variable with success probability $\mathrm{Pr}[\pi_A(\widetilde{T}_{B,\tau})]$, and, thus, $\mathrm{Pr}[\pi_A(\widetilde{T}_{B,\tau})] = \mathrm{E}[\xi_{\widetilde{T}_{B,\tau}}]$.

One observation of random variable $\xi_{\widetilde{T}_{B,\tau}}$ is the result of a number of independent fair coin tosses, with a sub-solver A called after each toss. Thus, the value $\xi_{\widetilde{T}_{B,\tau}}$ is observed as the outcome of a computationally reproducible probabilistic experiment, and, therefore, the described scheme for estimating the expectation of random variable using Monte Carlo sampling is suitable for estimation of $\mathrm{E}[\xi_{\widetilde{T}_{B,\tau}}]$.

Let us say several words about how we can seek ρ-backdoor trees which would be considered as good for solving SAT for a specific formula. Following (Semenov et al. 2022), we associate with this problem some pseudo-Boolean fitness function whose value gives some estimation of the usefulness of the considered ρ-backdoor tree.

When building such a function, we will take into account the properties of ρ-backdoor trees determined above. First, note that, by virtue of Theorem 1, the main contribution to the complexity of finding the smallest backdoor tree is made by the enumeration of possible orders on a specific set B. Based on this, in the experiments, at the backdoor search stage, we used some fixed order on X, which was chosen before starting the search based on heuristic considerations (see details in the next section). The second point is the fitness function definition. This function should reflect our decisions about such a tree: we prefer trees over a set B whose cardinality $|B|$ is small, and whose parameter ρ is close to 1. Taking this into account, we will consider the minimization problem for the following function:

$$F_{C,A,\widetilde{T}_{B,\tau}}\colon \{0,1\}^{|X|} \to \mathbb{R}. \tag{7}$$

Function (7) works as follows. Its input is an arbitrary vector λ_B of length $|X|$ that defines the set B: the 1's in the vector λ_B point to variables from X that belong to B; then on set B the order τ is fixed and the random traversal process for tree $\widetilde{T}_{B,\tau}$ described above is started using fair coin tossing. Traversal of each random path in $\widetilde{T}_{B,\tau}$ corresponds to one observation of the random variable $\xi_{\widetilde{T}_{B,\tau}}$. If the set B has small cardinality (see detail in the experiments de-

scription), then the tree $\widetilde{T}_{B,\tau}$ is explored completely, and the exact value of the probability $\Pr[\pi_A(\widetilde{T}_{B,\tau})]$ is calculated. In other cases, we have an estimate of the said probability $\Pr[\pi_A(\widetilde{T}_{B,\tau})]$ as the expected value $\mathrm{E}[\xi_{\widetilde{T}_{B,\tau}}]$ calculated as a result of independent observations of $\xi_{\widetilde{T}_{B,\tau}}$.

We make the following basic assumptions in relation to function (7): 1) its value must increase with the increase of s as $\mathcal{O}(q^s)$ for some $q > 1$; 2) the larger the difference of $\Pr[\pi_A(\widetilde{T}_{B,\tau})]$ (or its estimated value) from 1, the more significant should be the growth of the function. With that said, we used the following fitness function:

$$F_{C,A,\widetilde{T}_{B,\tau}} = \tilde{\rho}_{\widetilde{T}_{B,\tau}} \cdot 2^{|B|} + (1 - \tilde{\rho}_{\widetilde{T}_{B,\tau}}) \cdot 2^{\gamma}, \qquad (8)$$

where $\gamma > 0$ is a parameter (we used $\gamma = 20$) and $\tilde{\rho}_{\widetilde{T}_{B,\tau}}$ is the probability $\Pr[\pi_A(\widetilde{T}_{B,\tau})]$ or its estimation. The term $(1 - \tilde{\rho}_{\widetilde{T}_{B,\tau}}) \cdot 2^{\gamma}$ is a penalty function (Nocedal and Wright 2006) which rapidly increases when $\tilde{\rho}_{\widetilde{T}_{B,\tau}}$ deviates from 1.

Experiments

In all the experiments we used the Unit Propagation rule as the polynomial sub-solver. Close to the ideas of (Khalil, Vaezipoor, and Dilkina 2022), in the experiments when optimizing the fitness function (8), we used a reduced search space, which was built as follows. Recall that we use Unit Propagation as the polynomial sub-solver, and our goal is to find a subset B such that it will result in a large portion of $C[\beta/B]$ solved by UP. A particular $C[\beta/B]$ being solved by UP means that when propagating the values from β assigned to variables from B, UP derives conflicting literal assignments. Clearly, the more unit literals are propagated when we assign a value to a variable x_i, the greater is the potential gain from the inclusion of this variable into a backdoor set. So, if we want to reduce the search space and nevertheless be able to find good ρ-backdoor trees, we can use the greedy strategy. Thus, we choose the top k variables sorted by the measure computed for x_i as the sum of the numbers of literals propagated by UP for $C[0/x_i]$ and $C[1/x_i]$. In all experiments, we used $k = 200$. The set X' was thus formed by 200 variables chosen in the aforementioned manner.

The ρ-backdoor trees were built on subsets of X', and at the search stage, a fixed natural tree traversal order τ^* was used: that is, the values of variables from X' with a large value of the measure are selected earlier.

The search algorithm for ρ-backdoor trees was implemented in C++[1]. To optimize the function (8), we used the (1+1) Fast Evolutionary Algorithm (Doerr et al. 2017) with parameter $\beta = 3$, in which the initial solution consists of one randomly selected variable from X'. Calculation of the fitness function is based on a modification of the `propagate` function from the Minisat solver (Eén and Sörensson 2004). This modification consists in the ability to efficiently propagate a set of literals (values of backdoor variables) one by one, stop when a conflict has been detected, and use the conflict information in future UP calls for the considered backdoor.

[1]https://github.com/ctlab/itmo_parsat

Instance	$\lvert X\rvert$	ks	cd
$PvS_{7,4}$	1213	392	617
$BvP_{7,6}$	1558	313	574
$BvP_{8,4}$	1315	554	1675
$BvS_{7,7}$	2007	556	901
$PHP_{13,12}$	156	16967	$> 36\ h$
$par9$	162	33864	11227
$pmg12$	190	15734	23047
$sgen$	150	1127	1751

Table 1: Solving times (in seconds) for the considered CNF formulas by SAT solvers Kissat (ks) and CaDiCaL (cd)

All experiments were run on a computer with a 32-Core Intel(R) Xeon(R) 106 CPU @ 1.99 GHz and 128 GB of RAM. For each considered SAT instance, we ran the search for ρ-backdoor trees using a time limit of three seconds, utilizing one thread for each run. The use of such a small time limit is motivated by the need to actually solve the resulting hard subproblems $C[\beta/B] \notin S(A)$ after they are determined by the found ρ-backdoor trees. So, in order for the backdoor-based SAT solving to be competitive against existing SAT solvers, the search procedure must be fast.

In the search algorithm, for all ρ-backdoor trees with $|B| < 16$, the value ρ was calculated exactly, and for larger backdoors the corresponding random sampling was used during fitness function evaluation. We used a sample of size 10^3 in these cases. The number is computed using Chernoff bounds in a way similar to (Semenov et al. 2022). Ten independent runs of the algorithm for each instance were performed, resulting in ten different independent ρ-backdoor trees. Afterwards, we used a SAT solver to solve the hard subproblems for each ρ-backdoor tree, and measured the execution time of the solver. We considered two solvers: Kissat (Biere 2022) as one of the fastest existing sequential SAT solvers, and CaDiCaL (Biere 2021) was selected for comparing with the results from (Semenov et al. 2022).

Data on the instances used in the experiments is shown in Table 1 (for comparison purposes, we used the instances that were considered in (Semenov et al. 2022), the instances were downloaded from the GitHub repository (Pavlenko 2022)). For each instance, the table shows the number of variables in the corresponding instance, and the solving times (in seconds) by solvers Kissat (ks) and CaDiCaL (cd).

The main experimental results are presented in Table 2. The principal measured parameter for each instance and solver is the *decomposition rate* $r_{B,M}$, which in (Semenov et al. 2022) is calculated for a backdoor B and SAT solver M as the time used to solve all hard subproblems $C[\beta/B] \notin S(A)$ associated with backdoor B divided by the time used to solve the original formula with solver M. In our experiments, the nominator of the expression for $r_{B,M}$ is supplemented with the time used to find the backdoor B (three seconds); this way, in contrast with (Semenov et al. 2022), the value $r_{B,M}$ expresses the ratio of the total amount of work used to solve a formula with the backdoor divided by the total amount of work for solving the formula without the

C	$\|B\|$	ρ	$r_{B,ks}$	$r_{B,cd}$
$PvS_{7,4}$	11.4	0.997 ± 0.001	0.62 ± 0.08	0.68 ± 0.10
$BvP_{7,6}$	11.4	0.997 ± 0.001	0.85 ± 0.12	0.72 ± 0.11
$BvP_{8,4}$	11.2	0.997 ± 0.002	0.87 ± 0.14	0.71 ± 0.11
$BvS_{7,7}$	11.4	0.997 ± 0.002	0.99 ± 0.16	0.78 ± 0.11
$PHP_{13,12}$	12.0	0.997 ± 0.000	0.55 ± 0.00	$< 0.36 \pm 0.04$
$par9$	12.2	0.995 ± 0.004	0.65 ± 0.02	0.48 ± 0.04
$pmg12$	13.4	0.969 ± 0.014	0.12 ± 0.01	0.10 ± 0.01
$sgen^{100}_{150}$	13.3	0.978 ± 0.011	0.29 ± 0.04	0.21 ± 0.03

Table 2: Main experimental results

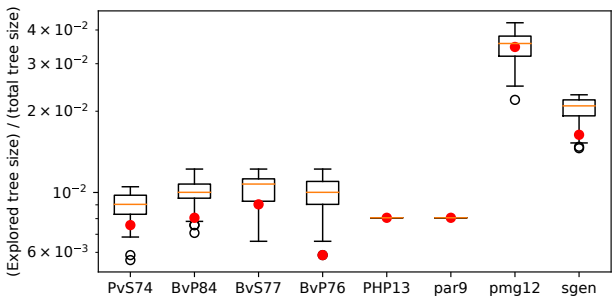

Figure 2: Ratios of explored size of the backdoor trees

backdoor.

For each SAT instance C, Table 2 shows the average cardinality $|B|$ of the found backdoors, ρ for these backdoors in the form (mean $\pm$ standard deviation), and the rates $r_{B,M}$ calculated based on execution times of SAT solvers Kissat (ks) and CaDiCaL (cd) in the form (mean $\pm$ relative standard deviation) over 10 independently generated backdoors. Note that cases when $r_{B,M} < 1$ mean that all subproblems corresponding to the backdoor are *sequentially* solved faster than with the SAT solver M.

The decomposition rates for backdoors found by our algorithm are well in line with the results reported in (Semenov et al. 2022) for CaDiCaL, but with an important difference: the time used by our algorithm for finding the backdoor was 3 s (using one thread), whereas the approach from (Semenov et al. 2022) required 0.5–6 h using 16 threads.

Finally, consider the plots in Fig. 2. For each formula, we took one backdoor $\widetilde{T}_{B,\tau}$ found for some fixed $\tau = \tau^*$, and calculated the size of the subtree explored when calculating the value of ρ_B; this value was divided by the total number of vertices in $\widetilde{T}_{B,\tau}$, which is $2^{s+1} - 1$, where $s = |B|$. The value of this ratio for τ^* is depicted with a red dot for each formula. In addition, we sampled 100 different orders $\tau \neq \tau^*$, measured this ratio for each such order, and plotted the results in the form of a boxplot. Thus, the plot for each formula demonstrates the influence of a specific order τ on the efficiency of evaluating ρ_B. From the figure, we see that the specific heuristic order τ^* that we used in the experiments most often gives good results, allowing to explore only a small portion of the whole tree. As a last remark, the explored backdoor tree size ratios for backdoor trees mentioned in Table 2 are below 0.01 for all instances except $sgen^{100}_{150}$ and *pmg12*, for which the mean ratios are 0.014 and 0.035 correspondingly.

Conclusion

In this paper, we explored backdoor trees for SAT and found a number of their features that do not appear to have been previously published. We introduced a probabilistic generalization of backdoor trees (ρ-backdoor trees) and established a number of their basic properties, including their relationship to the corresponding ρ-backdoor sets. We also described a practically efficient algorithm for finding small ρ-backdoor trees, and showed that the use of these trees in some cases significantly reduces the running time of the SAT solver.

The main practical contribution of this paper is that the proposed algorithm and its implementation are so efficient, that the concept of backdoors is deemed as close as never before to be practically applicable to general SAT solving. Indeed, one may envisage a parallel/cloud SAT solver that would quickly search for some ρ-backdoor trees, and then solve the corresponding hard subproblems in parallel.

Acknowledgements

We are thankful to the anonymous reviewers for their insightful comments that made it possible to significantly improve the paper.

This work was supported by the Analytical Center for the Government of the Russian Federation (IGK 000000D730321P5Q0002), agreement No. 70-2021-00141.

References

Biere, A. 2021. CaDiCaL SAT solver, commit ca9bff0. https://github.com/arminbiere/cadical. Accessed: 2022-08-10.

Biere, A. 2022. Kissat SAT solver, commit 97917dd. https://github.com/arminbiere/kissat. Accessed: 2022-08-10.

Chang, C.-L.; and Lee, R. C.-T. 1973. *Symbolic Logic and Mechanical Theorem Proving*. Academic Press, Inc.

Cormen, T.; Leiserson, C.; and Rivest, R. 1990. *Introduction to Algorithms*. MIT Press.

Dilkina, B.; Gomes, C. P.; Malitsky, Y.; Sabharwal, A.; and Sellmann, M. 2009. Backdoors to Combinatorial Optimization: Feasibility and Optimality. In van Hoeve, W.-J.; and Hooker, J. N., eds., *CPAIOR*, 56–70. Berlin, Heidelberg: Springer Berlin Heidelberg.

Dilkina, B.; Gomes, C. P.; and Sabharwal, A. 2007. Tradeoffs in the Complexity of Backdoor Detection. In Bessière, C., ed., *Principles and Practice of Constraint Programming*, 256–270. Berlin, Heidelberg: Springer Berlin Heidelberg.

Doerr, B.; Le, H. P.; Makhmara, R.; and Nguyen, T. D. 2017. Fast Genetic Algorithms. In *GECCO*, 777–784.

Drechsler, R.; Junttila, T. A.; and Niemelä, I. 2021. Non-Clausal SAT and ATPG. In Biere, A.; Heule, M.; van Maaren, H.; and Walsh, T., eds., *Handbook of Satisfiability - Second Edition*, volume 336 of *Frontiers in Artificial Intelligence and Applications*, 1047–1086. IOS Press.

Eén, N.; and Sörensson, N. 2004. An Extensible SAT-solver. In *SAT*, 502–518.

Feller, W. 1971. *An Introduction to probability theory and its applications*, volume 2. John Wiley & Sons, Inc., 2 edition.

Ferber, A. M.; Song, J.; Dilkina, B.; and Yue, Y. 2022. Learning Pseudo-Backdoors for Mixed Integer Programs. In Schaus, P., ed., *CPAIOR*, volume 13292 of *Lecture Notes in Computer Science*, 91–102. Springer.

Fichte, J. K.; and Szeider, S. 2011. Backdoors to Tractable Answer-Set Programming. In *IJCAI*, 863–868.

Gaspers, S.; and Kaploun, A. 2022. Faster Algorithms for Weak Backdoors. In *AAAI*, 3741–3748. AAAI Press.

Gaspers, S.; and Szeider, S. 2012a. Backdoors to Acyclic SAT. In *ICALP*, 363–374.

Gaspers, S.; and Szeider, S. 2012b. *Backdoors to Satisfaction*, 287–317. Berlin, Heidelberg: Springer Berlin Heidelberg.

Gaspers, S.; and Szeider, S. 2012c. Strong Backdoors to Nested Satisfiability. In *SAT*, 72–85.

Hemaspaandra, L. A.; and Narváez, D. E. 2017. The Opacity of Backbones. In *AAAI*, 3900–3906.

Hemaspaandra, L. A.; and Narváez, D. E. 2021. Existence versus exploitation: the opacity of backdoors and backbones. *Progress in Artificial Intelligence*, 10: 297–308.

Karp, R. M.; Luby, M.; and Madras, N. 1989. Monte-Carlo Approximation Algorithms for Enumeration Problems. *J. Algorithms*, 10(3): 429–448.

Khalil, E. B.; Vaezipoor, P.; and Dilkina, B. 2022. Finding Backdoors to Integer Programs: A Monte Carlo Tree Search Framework. In *AAAI*, 3786–3795. AAAI Press.

Kilby, P.; Slaney, J.; Thiébaux, S.; and Walsh, T. 2005. Backbones and Backdoors in Satisfiability. In *AAAI*, 1368–1373.

Luke, S. 2013. *Essentials of Metaheuristics*. Lulu, second edition.

Marques-Silva, J.; Lynce, I.; and Malik, S. 2009. Conflict-Driven Clause Learning SAT Solvers. In *Handbook of satisfiability*, volume 185 of *Frontiers in Artificial Intelligence and Applications*, 131–153.

Misra, N.; Ordyniak, S.; Raman, V.; and Szeider, S. 2013. Upper and Lower Bounds for Weak Backdoor Set Detection. In *SAT*, volume 7962 of *LNCS*, 394–402.

Molitor, P.; and Mohnke, J. 2007. *Equivalence checking of digital circuits: fundamentals, principles, methods*. Springer.

Motwani, R.; and Raghavan, P. 1995. *Randomized Algorithms*. Cambridge University Press.

Nocedal, J.; and Wright, S. 2006. *Numerical Optimization*. Springer.

Pavlenko, A. 2022. EvoGuess. https://github.com/ctlab/EvoGuess/releases/tag/v2.0.0. Accessed: 2022-08-10.

Samer, M.; and Szeider, S. 2008. Backdoor Trees. In *AAAI*, volume 1, 363–368. AAAI Press.

Semenov, A.; Pavlenko, A.; Chivilikhin, D.; and Kochemazov, S. 2022. On Probabilistic Generalization of Backdoors in Boolean Satisfiability. In *AAAI*, volume 36, 10353–10361.

Tseitin, G. S. 1970. On the Complexity of Derivation in Propositional Calculus. *Studies in Constructive Mathematics and Mathematical Logic, Part II, Seminars in mathematics,*, 115–125.

Williams, R.; Gomes, C. P.; and Selman, B. 2003. Backdoors To Typical Case Complexity. In *IJCAI*, 1173–1178.