On Probabilistic Generalization of Backdoors in Boolean Satisfiability

Alexander Semenov, Artem Pavlenko, Daniil Chivilikhin, Stepan Kochemazov

ITMO University, St. Petersburg, Russia

alex.a.semenov@itmo.ru, alpavlenko@itmo.ru, chivdan@itmo.ru, stepan.kochemazov@itmo.ru

Abstract

The paper proposes a probabilistic generalization of the well-known Strong Backdoor Set (SBS) concept applied to the Boolean Satisfiability Problem (SAT). We call a set of Boolean variables B a ρ-backdoor, if for a fraction of at least ρ of possible assignments of variables from B, assigning their values to variables in a Boolean formula in Conjunctive Normal Form (CNF) results in polynomially solvable formulas. Clearly, a ρ-backdoor with $\rho = 1$ is an SBS. For a given set B it is possible to efficiently construct an (ε, δ)-approximation of parameter ρ using the Monte Carlo method. Thus, we define an (ε, δ)-SBS as such a set B for which the conclusion "*parameter ρ deviates from 1 by no more than ε*" is true with probability no smaller than $1 - \delta$. We consider the problems of finding the minimum SBS and the minimum (ε, δ)-SBS. To solve the former problem, one can use the algorithm described by R. Williams, C. Gomes and B. Selman in 2003. In the paper we propose a new probabilistic algorithm to solve the latter problem, and show that the asymptotic estimation of the worst-case complexity of the proposed algorithm is significantly smaller than that of the algorithm by Williams et al. For practical applications, we suggest a metaheuristic optimization algorithm based on the penalty function method to seek the minimal (ε, δ)-SBS. Results of computational experiments show that the use of (ε, δ)-SBSes found by the proposed algorithm allows speeding up solving of test problems related to equivalence checking and hard crafted and combinatorial benchmarks compared to state-of-the-art SAT solvers.

Introduction

It is well known that the Boolean satisfiability problem (SAT) has an extremely wide range of practical applications. Despite the NP-hardness of SAT, modern SAT solvers successfully cope with Boolean formulas of large dimensions and, as a result, can be used in diverse application areas, such as symbolic verification, bioinformatics, cryptanalysis, explainable AI, etc. In view of this, the development of new and increasing the efficiency of known SAT solving algorithms is an important and relevant area of research.

One of the natural approaches to solving SAT is the one in which the original SAT instance (hereinafter we will assume that it is in Conjunctive Normal Form, CNF) is split into a family of simpler formulas which differ from the original in that the values of some variables are fixed. Some examples of this approach are the so-called Partitioning strategy (Hyvärinen, Junttila, and Niemelä 2010; Hyvärinen 2011) and Cube-and-Conquer (Heule et al. 2012). They both work exceptionally well when combined with parallel computing, because it is possible to solve the simplified problems independently of each other, making it possible to tackle even very hard problems, see e.g. (Heule, Kullmann, and Marek 2016; Heule 2018).

It is clear that we can always decompose a CNF formula C into a family of subformulas in such a way that each subformula can be solved by some polynomial algorithm. In the extreme case, if we use the whole set of variables in C for this purpose, then the subformulas will be easily solved by means of Unit Propagation (UP) rule (Dowling and Gallier 1984; Marques-Silva, Lynce, and Malik 2009). A more practical class of decompositions which use poly-time algorithms to solve subformulas was introduced in (Williams, Gomes, and Selman 2003) and relies on the concept of the so-called backdoor sets or "backdoors". In particular, a Strong Backdoor Set (SBS) is such a set of variables from C that the substitution of any of their assignments into C results in a formula for which SAT is solvable by a polynomial algorithm. Clearly, if there exists a small SBS, then it guarantees a significantly smaller hardness of the formula compared to $p(|C|) \cdot 2^{|X|}$, where X is the set of Boolean variables occurring in C and $p(\cdot)$ is some polynomial. Nontrivial SBSes arise in diverse contexts, e.g., they often can be identified in SAT encodings representing algorithms in symbolic verification or cryptography. In these cases, the variables encoding the input of a function specified by the considered algorithm form an SBS, and their share in the set X can be as small as tenths of a percent.

A large drawback of SBSes and their variants is that in the general case, the problem of finding such a backdoor is computationally hard. A number of results that fit these issues into the context of Structural Complexity can be found in (Kilby et al. 2005; Hemaspaandra and Narváez 2017, 2019, 2021). The works (Fichte and Szeider 2011; Gaspers and Szeider 2012a,b,c; Misra et al. 2013) demonstrate the relationship between backdoors and basic concepts of Parameterized Complexity. The article (Ansótegui et al. 2008) notes that backdoors can be naturally used to measure the hardness of CNF formulas, and also shows the relationship between "backdoor-hardness" and other hardness measures for SAT.

In the article (Semenov et al. 2021), based on the ideas from (Ansótegui et al. 2008), a new measure of hardness for CNF formulas was proposed w.r.t. an arbitrary complete deterministic SAT solver: the so-called *decomposition hardness*. The problem of evaluating the decomposition hardness in (Semenov et al. 2021) was reduced to the optimization of the pseudo-Boolean black-box function, which was carried out using evolutionary algorithms. In (Semenov et al. 2018), a special class of backdoor sets was introduced that enables one to estimate the hardness of cryptographic guess-and-determine attacks (Bard 2009).

Unfortunately, even if a CNF formula has a relatively small SBS, from (Williams, Gomes, and Selman 2003) it follows that finding such an SBS may take a colossal amount of time. On the other hand, the approaches which employ metaheuristic optimization to search for good decompositions (see e.g. (Semenov et al. 2021)) suffer from the fact that the fitness functions that are used in such schemes are quite costly. In the present paper we make an attempt to alleviate both listed drawbacks by introducing a probabilistic generalization of SBS. Let us briefly list the main contributions of the paper.

I. We introduce the probabilistic generalization of SBS. In particular, we define a ρ-backdoor for C w.r.t. a polynomial algorithm A as such a subset of variables $B \subseteq X$ that the fraction $\rho \in [0, 1]$ of $2^{|B|}$ subformulas resulting from fixing values of variables from B in C can be solved by algorithm A. The main attractiveness of this notion lies in the fact that the parameter ρ can be efficiently estimated via the Monte Carlo method. Clearly, an SBS in the sense of (Williams, Gomes, and Selman 2003) is a ρ-backdoor with $\rho = 1$. We define a special Monte Carlo test that allows constructing an interval estimation of the parameter ρ. For fixed $\varepsilon, \delta \in (0, 1)$ we define an (ε, δ)-SBS as a set that passes the aforementioned Monte Carlo test, and thus for this set the value of ρ deviates from 1 by no more than ε with probability no less than $1 - \delta$.

II. We define an (ε, δ)-analogue of minimum SBS to which we refer to as the minimum (ε, δ)-SBS with a probability of at least $1-\delta^k$, where $k \in \mathbb{N}$ is a constant which does not depend on the size of the CNF formula C. We propose an algorithm which constructs such an (ε, δ)-analogue of minimum SBS in time $\mathcal{O}\left(p(|C|) \cdot k \cdot \frac{16 \cdot \ln(2/\delta)}{\varepsilon^2} \cdot 2^{|X|}\right)$. At the same time, it is easy to see that the time complexity of the algorithm from (Williams, Gomes, and Selman 2003) when it is used for finding the minimum SBS is $\mathcal{O}(p(|C|) \cdot 3^{|X|})$, where $p(\cdot)$ is some polynomial.

III. For practical applications we propose a computational algorithm for finding ρ-backdoors, which are similar to minimum (ε, δ)-SBSes, and we refer to such backdoors as to *minimal* (ε, δ)*-SBSes*. The algorithm is based on the ideas used in metaheuristic black-box optimization. In particular, we consider the problem of finding a minimal (ε, δ)-SBS as the problem of minimization of a special pseudo-Boolean black-box function (*fitness function*). When defining this fitness function, we used the well-known penalty function method (Nocedal and Wright 2006). We use evolutionary algorithms to minimize this function.

IV. In the computational experiments we demonstrate the practical effectiveness of the developed black-box optimization algorithm in application to searching for minimal (ε, δ)-SBSes. Also, we show that by exploiting the found sets it is possible to significantly speed up state-of-the-art SAT solvers on hard SAT instances.

Preliminaries

The SAT problem consists in determining the satisfiability of an arbitrary Boolean formula. In the context of SAT it is usually convenient to consider Boolean formulas in CNF. A CNF is a conjunction of clauses, clauses are disjunctions of literals, and literals are formulas x and $\neg x$, where x is a Boolean variable. For an arbitrary CNF formula C over the set of Boolean variables X we will refer to an arbitrary mapping $\alpha\colon X \to \{0, 1\}$ as to an *assignment* of variables from X. An assignment α that makes C true is called a *satisfying assignment*. If there exists a satisfying assignment for C, then C is called *satisfiable*, otherwise it is called *unsatisfiable*. SAT w.r.t. an arbitrary CNF formula consists in answering the question whether C is satisfiable or not.

Consider an arbitrary CNF formula C over the set of Boolean variables X. For an arbitrary $B \subseteq X$ denote the set of all assignments of variables from B by $\{0, 1\}^{|B|}$. Next, denote by $C[\beta/B]$ the CNF formula obtained from C by substituting the values $\beta \in \{0, 1\}^{|B|}$ to variables from B.

It may turn out that for some B and $\beta \in \{0, 1\}^{|B|}$, SAT for $C[\beta/B]$ is decidable by some polynomial algorithm A. For example, this is the case if $C[\beta/B]$ belongs to some Schaefer's class (Schaefer 1978). In the general case, it is a very rare situation when A is able to solve SAT for $C[\beta/B]$ for any possible β. For example, consider the iterative application of UP as A. For some β-s, UP may output a satisfying assignment for C or prove unsatisfiability of $C[\beta/B]$, but for other β-s the result will be "Unknown", indicating that UP is not powerful enough for the considered CNF formula and cannot decide the satisfiability of corresponding $C[\beta/B]$.

Hereinafter, $C[\beta/B] \in S(A)$ denotes that SAT for $C[\beta/B]$ is solvable by some polynomial algorithm (often referred to as *subsolver*) A. The basic ideas used below go back to the article (Williams, Gomes, and Selman 2003) which introduced the concept of *backdoors* to Constraint Satisfaction Problems.

Definition 1 ((Williams, Gomes, and Selman 2003))**.** Let C be an arbitrary CNF formula over the set of Boolean variables X. We call the subset $B \subseteq X$ a strong backdoor set (SBS) for C w.r.t. polynomial algorithm A, if for any $\beta \in \{0, 1\}^{|B|}$ the following holds: $C[\beta/B] \in S(A)$.

The article (Williams, Gomes, and Selman 2003) proposed an algorithm for finding SBSes that can be used to solve SAT for an arbitrary CNF formula. This algorithm processes the subsets of the set of variables X of increasing cardinality. For each such subset $B \subseteq X$ it checks whether B is an SBS. In the worst case, the corresponding procedure needs to check if $C[\beta/B] \in S(A)$ for all $\beta \in \{0,1\}^{|B|}$. The complexity of solving SAT using the Williams et al. algorithm under the assumption that C has an SBS $B\colon |B| \le |X|/2$ is bounded from above by $p(|C|)\left(2|X|/\sqrt{|B|}\right)^{|B|}$, where $p(\cdot)$ is some polynomial.

In the paper (Ansótegui et al. 2008) it was noted that a particular SBS can be used to construct an estimate of CNF formula hardness. The following definition directly follows from the results of (Ansótegui et al. 2008).

Definition 2 (backdoor-hardness)**.** Let C be an arbitrary unsatisfiable CNF formula and B be an arbitrary SBS for C w.r.t. a polynomial algorithm A. Denote the total runtime of A on CNF formulas $\{C[\beta/B] \mid \beta \in \{0,1\}^{|B|}\}$ by $\tau_{B,A}(C)$. The backdoor hardness of C w.r.t. A is specified as $\tau_A(C) = \min_{B \in 2^X} \tau_{B,A}(C)$, where the minimum is taken among all possible SBSes for C w.r.t. A.

In the next section we describe the proposed probabilistic generalization of the SBS and its theoretical properties.

Probabilistic Backdoors

Let us start by introducing the notion of ρ-backdoors.

Definition 3 (ρ-backdoor)**.** Let C be a CNF formula over Boolean variables X and let A be some polynomial algorithm. The set $B \subseteq X$ is called a ρ-backdoor ($\rho \in [0,1]$) w.r.t. A if the fraction of CNF formulas $C[\beta/B]$ that belong to $S(A)$ over all possible $\beta \in \{0,1\}^{|B|}$ is at least ρ.

It is easy to see that a ρ-backdoor with $\rho = 1$ is an SBS in the sense of (Williams, Gomes, and Selman 2003). The issue with ρ is that to compute its value it is necessary to process all $2^{|B|}$ CNF formulas $C[\beta/B]$. However, below we show that to estimate this parameter, we can use the Monte Carlo method (Metropolis and Ulam 1949). More precisely, we build the so-called (ε,δ)-approximations of ρ by observing realizations of independent probabilistic experiments.

On (ε,δ)-approximations for ρ-backdoors

The approach developed further is conceptually close to the one used in (Karp and Luby 1983; Karp, Luby, and Madras 1989) for solving the Counting SAT (namely, #DNF) problem. In the latter paper, the term "probabilistic (ε,δ)-approximation algorithm" was used. Assume that we observe a random variable ξ, and want to estimate its characteristic θ (e.g. expected value). Suppose that θ is estimated using the value $\tilde{\theta}$, constructed on the basis of observations of ξ in independent probabilistic experiments. Then, for given $\varepsilon,\delta \in (0,1)$, the value $\tilde{\theta}$ is called an (ε,δ)-approximation of θ if the following relation holds:

$$\Pr\left[(1-\varepsilon)\cdot\theta \le \tilde{\theta} \le (1+\varepsilon)\cdot\theta\right] \ge 1-\delta, \tag{1}$$

or, equivalently, $\Pr\left[|\tilde{\theta}-\theta| \le \varepsilon\cdot\theta\right] \ge 1-\delta$. Values ε and $1-\delta$ are called *tolerance* and *confidence level*, respectively.

In some cases, for relating θ and $\tilde{\theta}$ it is convenient to use a condition that is, in general, weaker than (1):

$$\Pr\left[|\tilde{\theta}-\theta| \le \varepsilon\right] \ge 1-\delta. \tag{2}$$

If (2) holds for some fixed $\varepsilon,\delta \in (0,1)$, then we will call $\tilde{\theta}$ the *weak* (ε,δ)-approximation of θ.

As we will see later, the random variables for which we estimate parameters are Bernoulli variables: they take values from $\{0,1\}$. In this case, to construct specific estimates of the form (1) and (2), we will use the well-known Chernoff's inequality (Motwani and Raghavan 1995; Goldreich 2008; Arora and Barak 2009). It should be noted that the form of this inequality varies greatly from source to source. Further, we use Chernoff's inequality in a form that is close to the one considered in (Karp, Luby, and Madras 1989).

Theorem 1 (simple forms of Chernoff's bound[1])**.** *Let $\xi_1,\dots,\xi_N$ be independent, identically distributed Bernoulli random variables with success probability ρ. Then, for any $\varepsilon: 0 < \varepsilon \le 2$, the following inequalities hold:*

$$\Pr\left[\left|\frac{1}{N}\sum_{j=1}^{N}\xi_j - \rho\right| < \varepsilon\cdot\rho\right] \ge 1 - 2\cdot e^{-N\rho\varepsilon^2/4},$$

$$\Pr\left[\left|\frac{1}{N}\sum_{j=1}^{N}\xi_j - \rho\right| < \varepsilon\right] \ge 1 - 2\cdot e^{-N\varepsilon^2/4}.$$

Thus, the mean value over a series of independent observations of a Bernoulli random variable can be used as an (ε,δ)-approximation or a weak (ε,δ)-approximation of the probability of success (expected value) of this random variable.

A basic observation used below is that we can efficiently construct weak (ε,δ)-approximations for parameter ρ. Indeed, consider SAT for an arbitrary CNF formula C over the set of variables X and let $B, B \subseteq X$, be some ρ-backdoor w.r.t. some polynomial algorithm A. Let us define on $\{0,1\}^{|B|}$ a uniform distribution and associate with each $\beta \in \{0,1\}^{|B|}$ the value of the following random variable ξ_B:

$$\xi_B(\beta) = \begin{cases} 1, & \text{if } C[\beta/B] \in S(A); \\ 0, & \text{if } C[\beta/B] \notin S(A). \end{cases}$$

Clearly, ξ_B is a Bernoulli random variable with success probability $\rho = E[\xi_B]$. Further we will use the following fact which directly follows from Theorem 1: if we fix arbitrary $\varepsilon,\delta \in (0,1)$, then the runtime of the algorithm that calculates the value $\tilde{\rho}$ for which $\Pr\left[|\rho-\tilde{\rho}| < \varepsilon/2\right] \ge 1-\delta$ will be bounded from above by $poly(|C|)\cdot\frac{16\cdot\ln(2/\delta)}{\varepsilon^2}$.

It should be noted that one can attempt to use a ρ-backdoor B with ρ close to 1 in a similar fashion to an ordinary SBS: indeed, in this case, to solve SAT for the overwhelming majority of CNF formulas of the form $C[\beta/B] \in S(A)$ it is sufficient to use the polynomial algorithm A, and tackle the remaining "hard" CNF formulas $C[\beta/B] \notin S(A)$ by a complete SAT solving algorithm.

Thus, from a practical point of view, ρ-backdoors with ρ close to 1 appear to be the most useful, and below we focus on ways to construct them. Our first goal is to define an

[1]For proof, see https://github.com/ctlab/evoguess/releases/download/v2.0.0/AAAI22_Technical_appendix.pdf

(ε, δ)-analogue of SBS. From the above it appears sensible to define an (ε, δ)-SBS as such a ρ-backdoor B, $B \subseteq X$, for which the conclusion that $\rho \in [1-\varepsilon, 1]$ is valid with high probability.

Definition 4. For fixed $B \in 2^X$ let us define the following Monte Carlo test: fix $\varepsilon, \delta \in (0,1)$ and conduct N independent observations of random variable ξ_B: $\xi_1, \ldots, \xi_N$, where $N = \left\lceil \frac{16 \cdot \ln(2/\delta)}{\varepsilon^2} \right\rceil$. Let us calculate the value $\tilde{\rho} = \frac{1}{N}\sum_{j=1}^{N} \xi_j$. We will say that B passes the Monte Carlo test if $\tilde{\rho} \in [1-\varepsilon/2, 1]$.

Directly from Theorem 1 we have the following fact.

Corollary 1. If B passes the Monte Carlo test described above, then the conclusion that $\rho \in [1-\varepsilon, 1]$ is true with probability no smaller than $1-\delta$ (accordingly, the probability of it being wrong does not exceed δ).

Using the introduced notions let us give the definition of an (ε, δ)-SBS.

Definition 5. For fixed $\varepsilon, \delta \in (0,1)$ let us say that set B, $B \subseteq X$ is an (ε, δ)-SBS if B passes the Monte Carlo test described above and thus the conclusion that $\rho \in [1-\varepsilon, 1]$ is true with probability no smaller than $1-\delta$.

Below we consider the problem of searching for the minimum SBS which plays the crucial role in the definition of backdoor-hardness (see Definition 2), and the problem of searching for an (ε, δ)-approximation of such an SBS.

Finding the Minimum SBS

Let us refer to an SBS with minimum cardinality as to the *minimum SBS*. Although the problem of finding the minimum SBS was not considered directly in (Williams, Gomes, and Selman 2003), the algorithm described in that paper can be used to solve it. This algorithm sequentially iterates over all subsets in X of increasing cardinality: first, sets of one variable, then of two variables, etc. For each subset, the algorithm checks whether it is an SBS or not. Clearly, the first SBS to pass this check is the minimum SBS. The following fact holds.

Theorem 2. *The time complexity of the algorithm for finding the minimum SBS from (Williams, Gomes, and Selman 2003) is $\mathcal{O}(p(|C|) \cdot 3^{|X|})$ for some polynomial $p(\cdot)$.*

Proof. The worst case corresponds to the situation when the whole set X is the only SBS. In this case, the algorithm sequentially enumerates all sets with cardinality from 1 to $|X|$. Suppose that for each set B of cardinality $i, i < |X|$, the algorithm checks almost all vectors in $\{0,1\}^i$ before verifying that B is not an SBS. In this case, the number of performed operations will be close to the following value:

$$p(|C|) \cdot \sum_{i=1}^{|X|} \binom{|X|}{i} \cdot 2^i = \mathcal{O}\left(p(|C|) \cdot (2+1)^{|X|}\right),$$

where $p(\cdot)$ is a polynomial that bounds the complexity of substituting an assignment of variables from B into C. $\square$

At first glance it is not clear how to define an (ε, δ)-analogue of a minimum SBS. Intuitively, it should be an (ε, δ)-SBS for which there are some arguments that justify the nonexistence of a smaller (ε, δ)-SBS. In order to construct such arguments we require the following auxiliary notion.

Definition 6. Let us refer to a set B, $B \subseteq X$, as to an (ε, δ)-SBS candidate if for fixed $\varepsilon, \delta \in (0,1)$ and $N = \left\lceil \frac{16 \cdot \ln(2/\delta)}{\varepsilon^2} \right\rceil$ the following condition holds:

$$\Pr\left[1 - \frac{1}{N} \cdot \sum_{j=1}^{N} \xi_j \leq \varepsilon/2\right] \geq 1-\delta.$$

Now let us consider the following algorithm which is essentially an analogue of the algorithm from (Williams, Gomes, and Selman 2003). Consider the subsets of X of increasing cardinality in the role of B: first all subsets of size 1, then the subsets of size 2, etc.

Let us fix $\varepsilon, \delta \in (0,1)$ and $N = \left\lceil \frac{16 \cdot \ln(2/\delta)}{\varepsilon^2} \right\rceil$. If for $B \in 2^X$ we have $N \geq 2^{|B|}$ then we check if for every $\beta \in \{0,1\}^{|B|}$ it holds that $C[\beta/B] \in S(A)$. If in the first run the algorithm found an SBS B such that $2^{|B|} \leq N$, then it terminates and outputs B as an answer. This is the trivial case.

Next, suppose that an SBS $B : 2^{|B|} \leq N$ does not exist. For each $B : 2^{|B|} \geq N$ we conduct the Monte Carlo test described in Definition 4. If B passes this test, i.e. B is an (ε, δ)-SBS, then the algorithm outputs B as an answer and terminates.

Let us perform k runs of the described algorithm. Assume that $B_1, \ldots, B_k$ are the constructed (ε, δ)-SBSes. Choose among them an SBS of minimum cardinality and denote it as B_*. We can make the following conclusion: that there are no (ε, δ)-SBS candidates with cardinality smaller than $|B_*|$. Assume that this conclusion is wrong and there exists an (ε, δ)-SBS candidate B': $|B'| < |B_*|$. However, in this case the probability that the set B' does not pass the Monte Carlo test is smaller than δ, and the probability that B' does not pass k independent tests of this kind is smaller than δ^k. This is exactly the probability that the conclusion is wrong. Thus the set B_* has the following properties: it is an (ε, δ)-SBS and with probability no smaller than $1-\delta^k$ there are no (ε, δ)-SBS candidates with cardinality smaller than $|B_*|$.

Definition 7. Let us say that the set B_* found by the algorithm described above is a minimum (ε, δ)-SBS with a probability of at least $1-\delta^k$.

Since the total number of Monte Carlo tests within a single run of the described algorithm does not exceed $2^{|X|}$, the following theorem holds.

Theorem 3. *The time complexity of the algorithm for constructing a set B_* which is a minimum (ε, δ)-SBS with a probability of at least $1-\delta^k$ is $\mathcal{O}\left(p\left(|C|\right) \cdot k \cdot \frac{16 \cdot \ln(2/\delta)}{\varepsilon^2} \cdot 2^{|X|}\right)$ for some polynomial $p(\cdot)$.*

Using Several ρ-Backdoors to Solve SAT

If B is some ρ-backdoor with ρ close to 1, e.g. an (ε, δ)-SBS, then a natural way of using B to increase the efficiency of solving SAT for formula C is the following: we solve most of the problems $C[\beta/B]$ (for which $C[\beta/B] \in S(A)$) using a polynomial algorithm A, and to the remaining small number of "hard" problems $C[\beta/B] \notin S(A)$ we apply some complete SAT solver. However, we do not know anything in advance about the "hard" problems. They may well turn out to be too difficult and then the use of the backdoor may not be beneficial.

Imagine now that we have a set of several different (ε, δ)-SBSes: $\Delta = \{B_1, \ldots, B_s\}$. For each $i \in \{1, \ldots, s\}$ we can solve SAT using the polynomial algorithm A for CNF formulas $C[\beta/B_i] \in S(A)$. Denote by Γ_i the set of all $\beta \in \{0,1\}^{|B_i|}$ such that $C[\beta/B_i] \notin S(A)$. Construct the Cartesian product of all such sets Γ_i: $\Gamma = \Gamma_1 \times \ldots \times \Gamma_s$, and consider an arbitrary CNF formula $C[\gamma]$, $\gamma \in \Gamma$, derived from C by substituting the values γ of variables from Δ.

Note that, in general, much more information is substituted into $C[\gamma]$ than into each formula $C[\beta/B_i]$, $i \in \{1, \ldots, s\}$: indeed, if $B_i \cap B_j = \emptyset$ for any $i, j \in \{1, \ldots, s\}$, $i \neq j$, then each CNF formula $C[\gamma]$, $\gamma \in \Gamma$, is derived from C by substituting values of $\sum_{i=1}^{s} |B_i|$ variables. It is not hard to see that C is unsatisfiable if and only if all formulas $C[\beta/B_i] \in S(A)$ for all $i \in \{1, \ldots, s\}$ are unsatisfiable, and also all formulas $C[\gamma]$, $\gamma \in \Gamma$, are unsatisfiable. If $|\Gamma|$ is relatively small (for small ε), the total complexity of checking all of the above cases can be significantly smaller than the SAT solving time for the original CNF formula.

Also note that in practice backdoors may intersect: there may exist such B_i and B_j that $B_i \cap B_j \neq \emptyset$ for some $i, j \in \{1, \ldots, s\}$, $i \neq j$. In this case, Γ will contain some assignments γ with contradictory literals of the common variables. However, these cases are almost instantly processed by a SAT solver and do not require any special treatment.

In fact, one can view the set of variables $\tilde{B} = \bigcup_{i=1}^{s} B_i$ as some ρ-backdoor, for which we construct the set of problems $C[\beta/\tilde{B}] \in S(A)$ in a compound fashion because the size of $\tilde{B}$ is too large to do it the usual way.

Searching for Probabilistic Backdoors via Black-Box Optimization

In this section, we formulate the basic ideas behind the algorithms for seeking (ε, δ)-SBSes which can be applied to practical SAT instances. The algorithm for finding the minimum (ε, δ)-SBS described above is of mostly theoretical interest. For practical applications of the probabilistic generalization of SBS introduced above, we need to make the following steps. First, instead of enumerating all possible backdoors of increasing size, we will employ the strategies used in metaheuristic optimization (Luke 2015). Second, we introduce a special *fitness function* which uses the statistical estimation of ρ. Then, we can minimize this function over a Boolean hypercube using metaheuristic optimization algorithms. This process can be interrupted once the number of iterations exceeds some limit. Finally, we refer to a set B (viewing it as a point of a hypercube) with the minimal value of the fitness function w.r.t fixed $\varepsilon, \delta \in (0,1)$ and $N = \left\lceil \frac{16 \cdot \ln(2/\delta)}{\varepsilon^2} \right\rceil$ as to a *minimal* (ε, δ)*-SBS*.

Consider 2^X, the set of all subsets of X. Each $B \in 2^X$ can be represented as a Boolean vector λ_B of length $|X|$. Thus, the search space $\{0,1\}^{|X|}$ consisting of all such vectors is formed. If some $x_i \in X$ belongs to B, then the i-th coordinate in vector λ_B equals 1, otherwise it equals 0.

Fitness Function

In our case, the fitness function takes as input the CNF formula C, the polynomial algorithm A, and the vector λ_B representing a ρ-backdoor, for which we can estimate the value of ρ. In the problem of finding a minimal (ε, δ)-SBS, we have two optimization criteria of equal importance, the size of the set and the (estimated) value of ρ ($\tilde{\rho}$): the smaller the size of the set and the larger the value of $\tilde{\rho}$, the better. In addition, for backdoors of equal size we want to introduce a heavy penalty for the ones with lower estimated values of ρ.

Summing up, our fitness function is computed in the following way. For an arbitrary λ_B we first construct the set B specified by this vector, and generate the random sample $\beta_1, \ldots, \beta_N \in \{0,1\}^{|B|}$ w.r.t. the uniform distribution defined on $\{0,1\}^{|B|}$. Next, we calculate $\tilde{\rho}_B = (\sum_{i=1}^{N} \xi_i)/N$ using algorithm A. The fitness function has the form:

$$F_{C,A,N}(\lambda_B) = \tilde{\rho}_B \cdot 2^{|B|} + G_{C,A,N}(\lambda_B). \tag{3}$$

Here, $G_{C,A,N}(\lambda_B)$ is a penalty function (Nocedal and Wright 2006) whose value sharply increases if the random sample $\beta_1, \ldots, \beta_N$ contains at least one $\beta \in \{0,1\}^{|B|}$ such that $C[\beta/B] \notin S(A)$. When $\rho_B = 1$, i.e. when B is an SBS, the value of (3) must be equal to the number of all possible assignments of variables from this SBS. Also, if for all $\beta_1, \ldots, \beta_N$ we have $C[B/\beta_j] \notin S(A)$, $j \in \{1, \ldots, N\}$, then it is reasonable to consider the corresponding λ_B as unpromising. Taking this into account, in the experiments we used penalty functions of the following form:

$$G_{C,A,N}(\lambda_B) = \begin{cases} (1 - \tilde{\rho}_B) \cdot 2^{\omega |X|}, & \text{if } \tilde{\rho}_B > 0; \\ \infty, & \text{if } \tilde{\rho}_B = 0. \end{cases}, \tag{4}$$

where $\omega \in [0,1]$ is a parameter which can be heuristically selected for each specific CNF formula.

Additionally note that (3) is a multivalued function: for λ_B we can generate different random samples and the corresponding values of (3) may differ.

Used Black-Box Optimization Algorithms

The fitness function (3) is a pseudo-Boolean black-box function for which no analytical properties are known. Thus, to minimize it, one can apply any algorithms used in metaheuristic optimization (Luke 2015). In our experiments, we used the well-known (1+1)-evolutionary algorithm ((1+1)-EA) (Mühlenbein 1992; Droste, Jansen, and Wegener 2002) and also one variation of the genetic algorithm (GA).

$(1+1)$-EA is based on the idea of random mutation. The mutated individual is an arbitrary Boolean vector $\alpha \in$

$\{0,1\}^n$, in which each bit is independently flipped with a fixed probability. Usually, the probability of mutation is set to $p = \frac{1}{n}$. In this case, the expected value of the number of bits flipped during a single mutation of α is 1, and thus, on average, the $(1+1)$-EA performs in a similar fashion to the Hill Climbing local search algorithm (Russell and Norvig 2010). However, unlike Hill Climbing, $(1+1)$-EA has a non-zero probability of moving to an arbitrary point from $\{0,1\}^n$ in one step.

$(1+1)$-EA is extremely inefficient in the worst-case scenario (Droste, Jansen, and Wegener 2002). However, in many practical cases it can work surprisingly well. There are a number of modifications of $(1+1)$-EA which have significantly better worst-case estimations. One of such well-known modifications is the $(1+1)$-Fast Evolutionary Algorithm ($(1+1)$-FEA) described in (Doerr et al. 2017).

In our computational experiments, the best results were obtained using one variant of a GA, which used a mutation operator proposed in (Doerr et al. 2017). The GA works with a population consisting of several vectors $\lambda_B \in \{0,1\}^{|X|}$ representing some ρ-backdoors. Let $P_{\text{curr}} = \left\{\lambda_{B_1}, \ldots, \lambda_{B_Q}\right\}$ be the current population of the GA. The next population P_{new} such that $|P_{\text{curr}}| = |P_{\text{new}}| = Q$ is constructed in the following way. The population P_{curr} is associated with a distribution $D_{\text{curr}} = \{p_1, \ldots, p_Q\}$:

$$p_i = \frac{1/F_{C,A,N}(\lambda_{B_i})}{\sum_{j=1}^{Q}\left(1/F_{C,A,N}(\lambda_{B_j})\right)}, i \in \{1, \ldots, Q\}.$$

The algorithm selects individuals from P_{curr} randomly and independently according to the distribution D_{curr}, and applies the standard two-point crossover (Luke 2015) to each selected pair, producing a pair of child individuals. Afterwards, a mutation operator is applied to both children. Then, the constructed set of individuals is extended with H individuals from P_{curr} which have the best values of the fitness function. This step corresponds to the elitism concept (Luke 2015). At the same time, we need to guarantee that the following condition holds: $G + H = Q$. As a result, we have a new population P_{new}. In the experiments, we used $Q = 8$, $H = 2$.

Computational Experiments

In all computational experiments, we used the Unit Propagation rule as the polynomial algorithm A for identifying subproblems $C[\beta/B]$ such that $C[\beta/B] \in S(A)$, and used modern CDCL SAT solvers to solve SAT for CNF formulas $C[\beta/B] \notin S(A)$. To find good (ε,δ)-SBSes (ρ-backdoors with ρ close to 1), we minimized the function (3) using the algorithm described in the previous section.

We experimented with two approaches to selecting the initial candidate solution: 1) start the search from the set $B = X$, and 2) start the search from an empty set $B = \emptyset$. In the first case, the algorithm always starts discarding some variables, lowering the cardinality of B (during a series of initial iterations, $\tilde{\rho} = 1$). In the second case, on the contrary, the algorithm starts adding new variables to the current backdoor, and thus, for some initial iterations we have $\tilde{\rho} = 0$.

The strategy 2 is well-adapted to cases when a small (ε,δ)-SBS exists: such a backdoor can be found quite rapidly. The other strategy requires more computational resources, but sometimes allows finding backdoors with ρ much closer to 1 than in the case of strategy 2.

Note that the use of weak (ε,δ)-approximation allows one to a priori guarantee any level of estimation accuracy for ρ, regardless of the size of the considered backdoor. Indeed, e.g., for making probabilistic conclusions of the mentioned form with parameters $\varepsilon = 0.01$ and $\delta = 0.1$, it is sufficient to use a random sample size of at least $10^4 \cdot 16 \cdot \ln 20 \approx 4.8 \cdot 10^5$. In the reported experiments we started the search from the empty backdoor (strategy 2), and initially used random samples of size of $N = 4000$: for small backdoors B ($|B| \leq 11$) that are generated in the early stages of the algorithm, this value provides an exact computation of ρ. We also doubled the value of N if for the current value we had $\tilde{\rho}_B = 1$, at the same time keeping N within the theoretical bound calculated above. When the algorithm terminated, we ensured the exact calculation of ρ for any constructed backdoor B by solving all $2^{|B|}$ subformulas with a UP solver (in the experiments, the resulting $|B|$ was quite small).

Note that a considerable advantage of the proposed approach over other conceptually similar ones, e.g. (Semenov and Zaikin 2016; Kochemazov and Zaikin 2018; Semenov et al. 2021), is that computing functions (3) is cheap: in the experiments, strategy 2 gave good results even for formulas with several thousand variables.

Implementation Details

The proposed approach was implemented in Python in the form of a multi-threaded application `EvoGuess`[2], using PySAT (Ignatiev, Morgado, and Marques-Silva 2018) for interfacing with SAT solvers. We used incremental SAT solvers that are available in PySAT: namely, we mainly ran Glucose 3.0 (`g3`), Glucose 4.1 (`g4`), and CaDiCaL 1.0.3 (`cd`), though we also used Minisat 2.2 (`m22`) in one experiment. Preliminary experiments were done using one node of the HPC-cluster "Academician V.M. Matrosov"[3] (with two 18-core Intel Xeon E5-2695 CPUs). For main experiments, we used one node of a computing cluster in ITMO University equipped with an Intel Xeon Gold 6248R CPU @ 3.00 GHz.

Benchmarks

In the experiments, we considered several classes of unsatisfiable CNF formulas. This choice is motivated by the fact that for satisfiable formulas the behavior of SAT solvers is highly irregular: in some cases, the algorithm can get "lucky" and find a satisfying assignment very quickly, and in other ones can run much longer. The first set of benchmarks belongs to the general class of equivalence checking instances (Kuehlmann and Krohm 1997; Molitor and Mohnke 2007). Essentially, they consist in the following. Two discrete functions are considered: $f : \{0,1\}^n \to \{0,1\}^m$ and $g : \{0,1\}^n \to \{0,1\}^m$. Assume that they are defined by Boolean circuits S_f and S_g. The goal is to prove either that

[2]https://github.com/ctlab/EvoGuess/releases/tag/v2.0.0

[3]Irkutsk Supercomputer Center of SB RAS, http://hpc.icc.ru

Instance	$\|X\|$	g3	g4	cd
$\mathtt{PvS}_{4,7}$	1213	936	1440	736
$\mathtt{BvP}_{4,8}$	1315	3232	3325	1287
$\mathtt{BvP}_{6,7}$	1558	1225	1533	526
$\mathtt{BvS}_{7,7}$	2007	2342	1621	606
pmg12	190	> 72h	> 72h	41915
par9	162	> 24h	> 24h	39541
$\mathtt{sgen}_{100}^{150}$	150	11365	2365	3139
$\mathtt{PHP}_{13,12}$	156	8496	14196	164
$\mathtt{PHP}_{15,14}$	210	> 24h	> 24h	2543

Table 1: Solving times in seconds of the considered CNF formulas by SAT solvers g3, g4, cd

f and g implement the same function, i.e. that $f \cong g$ (pointwise equality is implied), or to refute this assumption. In the first case, S_f and S_g are called equivalent. We considered the equivalence checking problem for circuits S_f, S_g, where f and g are different algorithms for sorting d arbitrary l-bit natural numbers, i.e. $f, g : \{0,1\}^k \to \{0,1\}^k$, where $k = d \cdot l$. In the role of f and g, we used the functions defined by sorting algorithms: Bubble sorting, Selection sorting (Cormen, Leiserson, and Rivest 1990), and Pancake sorting (Gates and Papadimitriou 1979). The corresponding circuits were constructed in form of And-Inverter Graphs (over the basis $\{\neg, \wedge\}$). Below, we refer to the constructed instances as to $\mathtt{PvS}_{\mathtt{d,l}}$ when the considered problem encodes the equivalence of Pancake sorting and Selection sorting, $\mathtt{BvP}_{\mathtt{d,l}}$ for Bubble sorting vs Pancake sorting, and $\mathtt{BvS}_{\mathtt{d,l}}$ in case of Bubble sorting vs Selection sorting.

We also considered some crafted tests: sgen (Spence 2015, 2017), Pigeonhole Principle formulas (PHP), Parity principle (par9), and pmg12 from SAT competition. Table 1 shows solving times of the considered CNF formulas with selected SAT solvers.

Finding Probabilistic Backdoors

In this section, we report the main experimental results on finding ρ-backdoors with ρ close to 1. In each experiment, for each CNF formula we 1) simplified the formula with SatELite/Minisat, 2) ran the backdoor search using the proposed metaheuristic algorithm (initializing with the empty backdoor, i.e. strategy 2), and then 3) solved the weakened subformulas $C[\beta/B]$ for the best found (according to $\tilde{\rho}$) ρ-backdoor B using different SAT solvers.

As a result of the last step, we calculated for each SAT solver A and each ρ-backdoor B the ratio $r_{B,A}$ further referred to as the *decomposition rate*: the time used to solve the formula with the backdoor B (via solving all weakened formulas $C[\beta/B]$ with solver A) divided by the the time used to solve the original CNF formula with the same solver. Cases when $r_{B,A} < 1$ indicate situations when solving with the backdoor is faster than solving the original formula with a conventional SAT solver. Note that since the backdoor search does not depend on the used SAT solver, we may search for a backdoor once and then use it to solve the original CNF formula with any available SAT solver.

As a preliminary experiment, we ran the GA and $(1+1)$-EA on the $\mathtt{PvS}_{4,7}$ instance, which is the simplest (in terms of SAT solving time) formula considered. Each algorithm was independently run five times, each run was limited to 2 h using 16 threads. For each run of each algorithm, we selected the ρ-backdoor with the best value of $\tilde{\rho}$ and calculated the ratio $r_{B,A}$. We observed that $r_{B,A}$ of backdoors generated by the GA were about two times smaller than of the ones generated by the $(1+1)$-EA. Therefore, in all further experiments we only used the GA.

Instance	$\|X\|$	$\|B\|$	ρ	Solver	$r_{B,A}$
$\mathtt{PvS}_{4,7}$	1213	12	0.9960	g3	0.45
		13	0.9968	g4	0.30
		12	0.9960	cd	0.68
		12	0.9941	m22	0.39
$\mathtt{BvP}_{4,8}$	1315	12	0.9987	g3	0.81
		12	0.9987	g4	0.98
		11	0.9926	cd	0.77
$\mathtt{BvP}_{6,7}$	1558	12	0.9970	g3	0.74
		12	0.9970	g4	0.69
		12	0.9970	cd	0.91
pmg12	190	13	0.9882	cd	0.23
$\mathtt{sgen}_{100}^{150}$	150	13	0.9816	g3	0.25
		13	0.9824	g4	0.27
		13	0.9648	cd	0.38
$\mathtt{PHP}_{13,12}$	156	9	0.9804	cd	0.34
$\mathtt{PHP}_{15,14}$	210	9	0.9804	cd	0.91

Table 2: Decomposition rates of ρ-backdoors

As it was mentioned above, the value of $\omega|X|$ should be empirically selected for each particular problem. Common sense suggests: if we want to find a ρ-backdoor B with $|B|$ close to $a \in \mathbb{N}$ and with ρ close to 1, then the value $\omega|X|$ should be close to a. Otherwise, the rapidly growing value of penalty will drive the search from the points with values close to 2^a. Taking this into account, in our experiments we used $\omega|X| \in \{15, 20\}$, so that the size of the found ρ-backdoors allowed to solve all weakened subformulas $C[\beta/B]$ and $C[\gamma]$.

Experimental results are summarized in Table 2: for each SAT instance, it shows its number of variables after simplification, the size $|B|$ of the found ρ-backdoor, its ρ value, and the ratio $r_{B,A}$ for different SAT solvers. Each GA run was allotted 0.5–6 hours of cluster time (depending on the formula) using 16 threads. Data shown in Table 2 indicates that the proposed approach allows finding ρ-backdoors with ρ very close to 1, and that these backdoors allow speeding up state-of-the-art SAT solvers.

Solving SAT with Several ρ-backdoors

In this section, we describe the experiments on solving SAT using several ρ-backdoors with the method proposed above. For a given CNF formula, we launched s searches for ρ-backdoors, each for the same 0.5–6 hours. As a result, we got a set of s different ρ-backdoors $\Delta = \{B_1, \ldots, B_s\}$.

For each resulting set of s backdoors, we 1) determined all hard problems $C[\beta/B_i] \notin S(A)$, $i \in \{1, \ldots, s\}$ (by ap-

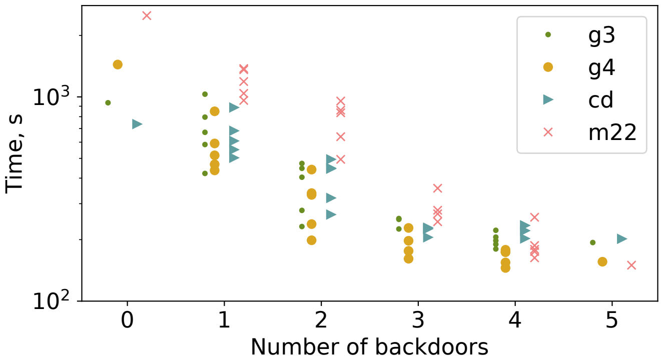

Figure 1: Solving $\texttt{PvS}_{4,7}$ with combinations of several ρ-backdoors using solvers g3, g4, cd, m22

Instance	$\lvert X\rvert$	s	$\lvert\Gamma\rvert$	Solver	$r_{\Delta,A}$
$\texttt{PvS}_{4,7}$	1213	4	6×10^5	g3	0.19
		5	1×10^6	g4	0.11
		4	1×10^5	cd	0.27
		5	1×10^6	m22	0.06
$\texttt{BvS}_{7,7}$	2005	3	2×10^5	g3	0.59
		2	352	g4	0.54
pmg12	190	4	3.1×10^7	g3	< 0.01
		4	3.1×10^7	g4	< 0.1
		4	3.1×10^7	cd	0.34
		2	4536	cd	0.21
par9	162	8	1×10^8	g3	< 0.77
		7	1×10^7	cd	0.36
$\texttt{sgen}_{100}^{150}$	150	2	2×10^5	g3	0.20
		2	2×10^5	g4	0.26
$\texttt{PHP}_{13,12}$	156	7	1×10^7	g3	0.74
		6	1×10^6	g3	0.54
		6	1×10^6	g4	0.12
		5	1×10^5	cd	0.12
$\texttt{PHP}_{15,14}$	210	6	1×10^6	cd	0.36

Table 3: Decomposition rates for sets of ρ-backdoors

plying a UP solver), 2) built the Cartesian product Γ, and 3) solved all formulas $C[\gamma]$ using a SAT solver. The plot in Fig. 1 shows results of experiments with $\texttt{PvS}_{4,7}$, where we did $s = 5$ runs of the backdoor search, and then considered all possible k-combinations ($1 \leq k \leq 5$) of these 5 backdoors. The plot shows the time used to solve SAT with different solvers for $\texttt{PvS}_{4,7}$: each point corresponds to the solving time with a specific set Δ. Results indicate that combinations of backdoors provide a significant advantage over using individual backdoors in terms of total solving time.

Table 3 shows the results on solving other formulas using several ρ-backdoors. To represent the efficiency gain from using several backdoors Δ, we introduce the ratio $r_{\Delta,A}$. For each instance, the table shows the number of combined backdoors s, the cardinality of the set Γ, and the resulting ratio $r_{\Delta,A}$ for different solvers.

Most notably, in some cases we found sets Δ that allowed solving very hard SAT instances. In particular, for pmg12 neither g3 nor g4 found a solution in more than 72 h, whereas a set of four ρ-backdoors allowed finding a solution in a matter of minutes for g4 and hours for g3, despite that the corresponding set Γ contained more than 3.1×10^7 assignments. The same goes for par9 with g3. This inspiring result allows one to expect that the proposed method may extend the area of applicability of SAT solvers, at least in some domains involving very hard unsatisfiable formulas.

Discussion & Conclusion

In this paper, we defined and studied a new form of backdoor set in the context of the Boolean Satisfiability problem. We defined a ρ-backdoor as such a subset of the set of variables of the formula that a fraction of at least ρ of assignments of variables from B, when substituted to the original formula, result in formulas solvable by a polynomial algorithm. We also proposed an efficient (ε, δ)-approximation algorithm to estimate ρ, and also an algorithm for finding (ε, δ)-approximations of Strong Backdoor Sets in the sense of (Williams, Gomes, and Selman 2003) of minimum cardinality. The proposed algorithm has a significantly better upper bound compared to the algorithm by Williams et al., if we use the latter to search for the minimum SBS. To find backdoors in practice, we proposed to use metaheuristic algorithms that minimize a specially formulated fitness function. Experiments showed that the proposed algorithm allows finding ρ-backdoors with ρ close to 1 for hard unsatisfiable SAT instances in several hours of runtime of a single cluster node. We used the found ρ-backdoors to solve the original SAT instance by first processing all possible assignments of variables via Unit Propagation rule to identify the hard subproblems, to which we then applied CDCL SAT solvers. In the majority of cases, the total runtime of a solver when using such a backdoor to produce and solve subformulas $C[\beta/B]$ is significantly smaller than its runtime on the original formula. We also described a method that allows using several found backdoors simultaneously to gain an even larger speedup.

In the future, one could replace the UP rule by a complete SAT solver in a limited setting, e.g. so that the number of conflicts does not exceed some constant. The concepts of SBS, ρ-backdoors, and other theoretical results from the paper can easily be transferred to this case. In particular, we can analyze the problem of finding a minimum (ε, δ)-approximation of a backdoor in such a form and prove a result similar to Theorem 3. The proposed metaheuristics can also be adapted to finding backdoors of this type.

Acknowledgements

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

References

Ansótegui, C.; Bonet, M. L.; Levy, J.; and Manyà, F. 2008. Measuring the Hardness of SAT Instances. In *AAAI*, 222–228.

Arora, S.; and Barak, B. 2009. *Computational Complexity: A Modern Approach*. Cambridge University Press.

Bard, G. V. 2009. *Algebraic Cryptanalysis*. Springer Publishing Company, Incorporated, 1st edition.

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

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

Dowling, W. F.; and Gallier, J. H. 1984. Linear-time algorithms for testing the satisfiability of propositional Horn formulae. *The Journal of Logic Programming*, 1(3): 267 – 284.

Droste, S.; Jansen, T.; and Wegener, I. 2002. On the Analysis of the (1+1) Evolutionary Algorithm. *Theor. Comput. Sci.*, 276(1–2): 51–81.

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

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

Gaspers, S.; and Szeider, S. 2012b. Backdoors to Satisfaction. In *The Multivariate Algorithmic Revolution and Beyond*, 287–317.

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

Gates, W. H.; and Papadimitriou, C. H. 1979. Bounds for sorting by prefix reversal. *Discrete Mathematics*, 27(1): 47–57.

Goldreich, O. 2008. *Computational Complexity: A Conceptual Perspective*. Cambridge University Press.

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

Hemaspaandra, L. A.; and Narváez, D. E. 2019. Existence Versus Exploitation: The Opacity of Backdoors and Backbones Under a Weak Assumption. In *SOFSEM*, 247–259.

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

Heule, M. J. H. 2018. Schur Number Five. In *AAAI*, 6598–6606.

Heule, M. J. H.; Kullmann, O.; and Marek, V. W. 2016. Solving and Verifying the Boolean Pythagorean Triples Problem via Cube-and-Conquer. In *SAT*, 228–245.

Heule, M. J. H.; Kullmann, O.; Wieringa, S.; and Biere, A. 2012. Cube and Conquer: Guiding CDCL SAT Solvers by Lookaheads. In *Hardware and Software: Verification and Testing*, 50–65.

Hyvärinen, A. E. J. 2011. Grid Based Propositional Satisfiability Solving. PhD thesis. Aalto University publication series.

Hyvärinen, A. E. J.; Junttila, T.; and Niemelä, I. 2010. Partitioning SAT Instances for Distributed Solving. In *LPAR*, 372–386.

Ignatiev, A.; Morgado, A.; and Marques-Silva, J. 2018. PySAT: A Python Toolkit for Prototyping with SAT Oracles. In *SAT*, 428–437.

Karp, R. M.; and Luby, M. 1983. Monte-Carlo algorithms for enumeration and reliability problems. In *FOCS*, 56–64.

Karp, R. M.; Luby, M.; and Madras, N. 1989. Monte-Carlo approximation algorithms for enumeration problems. *Journal of Algorithms*, 10(3): 429–448.

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

Kochemazov, S.; and Zaikin, O. 2018. ALIAS: A Modular Tool for Finding Backdoors for SAT. In *SAT*, 419–427.

Kuehlmann, A.; and Krohm, F. 1997. Equivalence Checking Using Cuts and Heaps. In *DAC*, 263–268.

Luke, S. 2015. *Essentials of Metaheuristics*. 2 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.

Metropolis, N.; and Ulam, S. 1949. The Monte Carlo Method. *Journal of the American Statistical Association*, 44(247): 335–341.

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.

Mühlenbein, H. 1992. How Genetic Algorithms Really Work: Mutation and Hillclimbing. In *PPSN*, 15–26.

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

Russell, S.; and Norvig, P. 2010. *Artificial Intelligence – A Modern Approach*. Pearson Education, 3 edition.

Schaefer, T. J. 1978. The Complexity of Satisfiability Problems. In *STOC*, 216–226.

Semenov, A.; Chivilikhin, D.; Pavlenko, A.; Otpuschennikov, I.; Ulyantsev, V.; and Ignatiev, A. 2021. Evaluating the Hardness of SAT Instances Using Evolutionary Optimization Algorithms. In *CP*, 47:1–47:18.

Semenov, A.; and Zaikin, O. 2016. Algorithm for finding partitionings of hard variants of boolean satisfiability problem with application to inversion of some cryptographic functions. *SpringerPlus*, 5(1). Article no. 554.

Semenov, A.; Zaikin, O.; Otpuschennikov, I.; Kochemazov, S.; and Ignatiev, A. 2018. On cryptographic attacks using backdoors for SAT. In *AAAI*, 6641–6648.

Spence, I. 2015. Weakening Cardinality Constraints Creates Harder Satisfiability Benchmarks. *ACM J. Exp. Algorithmics*, 20.

Spence, I. 2017. Balanced random SAT benchmarks. In *SAT Competition 2017*, volume B-2017-1, 53–54.

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