Electronic Colloquium on Computational Complexity, Report No. 117 (2017)

Hard satisfiable formulas for splittings by linear combinations [*]

Dmitry Itsykson[†] Alexander Knop[†]

July 20, 2017

Abstract

Itsykson and Sokolov in 2014 introduced the class of DPLL($\oplus$) algorithms that solve Boolean satisfiability problem using the splitting by linear combinations of variables modulo 2. This class extends the class of DPLL algorithms that split by variables. DPLL($\oplus$) algorithms solve in polynomial time systems of linear equations modulo 2 that are hard for DPLL, PPSZ and CDCL algorithms. Itsykson and Sokolov have proved first exponential lower bounds for DPLL($\oplus$) algorithms on unsatisfiable formulas.

In this paper we consider a subclass of DPLL($\oplus$) algorithms that arbitrary choose a linear form for splitting and randomly (with equal probabilities) choose a value to investigate first; we call such algorithms drunken DPLL($\oplus$). We give a construction of a family of satisfiable CNF formulas Ψ_n of size poly(n) such that any drunken DPLL($\oplus$) algorithm with probability at least $1 - 2^{-\Omega(n)}$ runs at least $2^{\Omega(n)}$ steps on Ψ_n; thus we solve an open question stated in the paper [12]. This lower bound extends the result of Alekhnovich, Hirsch and Itsykson [1] from drunken DPLL to drunken DPLL($\oplus$).

1 Introduction

The Boolean satisfiability problem (SAT) is one of the most popular **NP**-complete problems. However, SAT solvers have different behaviors on satisfiable and unsatisfiable formulas. The protocol of a SAT solver on an unsatisfiable formula ϕ may be considered as a proof of unsatisfiability of ϕ. Therefore, the study of propositional proof systems is connected with the study of SAT solvers. It is well known that protocols of DPLL solvers on an unsatisfiable formula are equivalent to tree-like resolution proofs of this formula and protocols of CDCL solvers correspond to dag-like resolution proofs [3]. Thus lower bounds on the running time of DPLL and CDCL solvers on unsatisfiable instances follow from lower bounds on the size of tree-like and dag-like resolution proofs.

Satisfiable formulas are usually simpler for SAT solvers rather than unsatisfiable ones. Also, satisfiable instances are of much interest in the case where we reduce some **NP** search problem (for example, the factorization) to SAT. DPLL [8, 7] algorithm is a recursive algorithm. On each recursive call, it simplifies an input formula F (without affecting its satisfiability), chooses a variable v and makes two recursive calls on the formulas $F[v := 1]$ and $F[v := 0]$ in some order. Every DPLL algorithm is determined by a heuristic A that chooses a variable for splitting and by a heuristic B that chooses a value that should be investigated at first. If $\mathbf{P} = \mathbf{NP}$, then DPLL can solve all satisfiable instances in polynomial time: a heuristic B always chooses a correct value of a variable. Alekhnovich, Hirsch, and Itsykson [1] proved exponential lower bounds on satisfiable instances for two wide classes of DPLL algorithms: for myopic DPLL algorithms and drunken DPLL algorithms. In myopic DPLL the both heuristics have the following restrictions: they can see only the skeleton of the formula where all negation signs are erased, they also have access to the number of positive and negative occurrences of every variable and they are allowed to read $n^{1-\epsilon}$ of clauses precisely. Drunken algorithms have no restrictions on the heuristic A that chooses the variable, but the heuristic B chooses

[*]The research is partially supported by the Government of the Russia (grant 14.Z50.31.0030). The paper is to appear in Proceedings of SAT 2017.

[†]Steklov Institute of Mathematics at St. Petersburg, 27 Fontanka, St.Petersburg, 191023, Russia, dmitrits@pdmi.ras.ru, aaknop@gmail.com.

ISSN 1433-8092

a value at random with equal probabilities. There are also known lower bounds on the complexity of the inversion of Goldreich's one-way function candidate by myopic and drunken DPLL algorithms [5, 10, 11].

Itsykson and Sokolov [12] introduced a generalization of DPLL algorithms that split by the value of linear combinations of variables modulo 2; we call them DPLL($\oplus$) algorithms. DPLL($\oplus$) algorithms quickly solves formulas that encode linear systems over GF(2) (unsatisfiable linear systems are hard for resolution [19] and even for bounded-depth Frege [2]; satisfiable linear systems are hard for drunken and myopic DPLL [1, 10] and PPSZ [18, 16]) and perfect matching principles for graphs with odd number of vertices (these formulas are hard for resolution [17] and bounded-depth Frege [4]).

It is well known that the tree-like resolution complexity (and hence the running time of DPLL) of the pigeonhole principle PHP_n^{n+1} is $2^{\Theta(n \log n)}$ [6]. Itsykson and Sokolov [12] proved a lower bound $2^{\Omega(n)}$ and Oparin recently proved an upper bound $2^{O(n)}$ [14] on the running time of DPLL($\oplus$) algorithms on PHP_n^{n+1}. There are three other families of formulas that are hard for DPLL($\oplus$) algorithms proposed by Itsykson and Sokolov [12], Krajíček [13], and Garlík and Kołodziejczyk [9].

Our results. Itsykson and Sokolov [12] formulated the following open question: to prove a lower bound on satisfiable formulas for drunken DPLL($\oplus$) algorithms that arbitrary choose a linear combination and randomly with equal probabilities chooses a value that would be investigated at first. In this paper we answer to the question and give a construction of a family of satisfiable formulas Ψ_n in CNF of size poly(n) such that any drunken DPLL($\oplus$) algorithm with probability $1 - 2^{-\Omega(n)}$ runs at least $2^{\Omega(n)}$ steps on Ψ_n.

In order to construct Ψ_n we take the pigeonhole principle PHP_n^{n+1} and manually add one satisfying assignment to it. We prove that with high probability a drunken DPLL($\oplus$) algorithm will make incorrect linear assumption and the algorithm will have to investigate a large subtree without satisfying assignments. To show that this subtree is indeed large we extend the technique that was used for PHP_n^{n+1} [12].

Further research. The constructed family Ψ_n has clauses with large width. It would be interesting to prove lower bounds for formulas in $O(1)$-CNF. It is also interesting to prove a lower bound for myopic DPLL($\oplus$) algorithms.

2 DPLL($\oplus$) and parity decision trees

DPLL($\oplus$) algorithms are parameterized by two heuristics: A and B. The heuristic A takes a CNF formula and a system of linear equations and returns a linear combination (a DPLL($\oplus$) algorithm will use this linear combination for splitting). The heuristic B takes a CNF formula, a system of linear equations, and a linear combination and returns a value from $\{0, 1\}$ (this value would be considered at first by a DPLL($\oplus$) algorithm). The set of DPLL($\oplus$) algorithms is the set of algorithms $\mathcal{D}_{A,B}$ for all heuristics A and B that are defined below. An algorithm $\mathcal{D}_{A,B}$ takes on the input a CNF formula Φ and a system of linear equations F (we may omit the second argument if the system F is empty) and works as follows:

1. If the system F does not have solutions (it can be verified in polynomial time), then return "Unsatisfiable".
2. If the system F contradicts to a clause C of the formula Φ (a system G contradicts a clause $\ell_1 \vee \ldots \vee \ell_k$ iff for all $i \in [k]$ the system $G \wedge (\ell_i = 1)$ is unsatisfiable, hence this condition may be verified in polynomial time), then return "Unsatisfiable".
3. If the system F has the unique solution τ (in variables of Φ) and this solution satisfies Φ (it can also be verified in polynomial time), then return τ.
4. $f := A(\Phi, F)$.
5. $\alpha := B(\Phi, F, f)$.
6. If $\mathcal{D}_{A,B}(\Phi, F \wedge (f = \alpha))$ returns an assignment, then return it.
7. Return the result of $\mathcal{D}_{A,B}(\Phi, F \wedge (f = 1 - \alpha))$.

The class of drunken DPLL($\oplus$) consists of all algorithms $\mathcal{D}_{A,rand}$, where A is an arbitrary heuristic and *rand* always returns a random element from $\{0,1\}$ with equal probabilities.

A *parity decision tree* for (Φ, F) is a rooted binary tree such that all its internal nodes are labeled with linear combinations of variables of Φ, for every internal node labeled with a linear form f one of its outgoing edge is labeled with $f = 0$ and the other one with $f = 1$. Let l be a leaf l of the tree, we denote by D_l the system of linear equations written on the edges of the path from the root to l. There are three kinds of leaves:

degenerate leaf: $F \wedge D_l$ does not have a solution;

satisfying leaf: $F \wedge D_l$ has the only solution in the variables of Φ and this solution satisfies Φ;

contradiction: $F \wedge D_l$ contradicts to a clause C of Φ.

If $\Phi \wedge F$ is unsatisfiable, then the recursion tree of $\mathcal{D}_{A,B}(\Phi, F)$ is a parity decision tree for (Φ, F) that does not contain satisfying leaves. Additionally the minimal size of a decision tree for (Φ, F) is a lower bound on the running time of $\mathcal{D}_{A,B}(\Phi, F)$. If $\Phi \wedge F$ is satisfiable then the recursion tree of $\mathcal{D}_{A,B}(\Phi, F)$ is a part of a parity decision tree since the execution stops when the algorithm $\mathcal{D}_{A,B}$ finds a satisfying assignment.

3 Lower bound

In this section we construct a satisfiable formula that is hard for all drunken DPLL($\oplus$) algorithms. Our hard example is based on the pigeonhole principle. The pigeonhole principle (PHP_n^m) states that it is possible to put m pigeons into n holes such that every pigeon is in at least one hole and every hole contains at most one pigeon. For every pigeon $i \in [m]$ and hole $j \in [n]$ we introduce a variable $p_{i,j}$; $p_{i,j} = 1$ iff i-th pigeon is in the j-th hole. The formula PHP_n^m is the conjunction of the following clauses:

short clauses: $\neg p_{i,k} \vee \neg p_{j,k}$ for all $i \neq j \in [m]$ and $k \in [n]$;

long clauses: $\bigvee_{k=1}^{n} p_{i,k}$ for all $i \in [m]$.

The formula PHP_n^m is unsatisfiable iff $m > n$. Let $\mathcal{P}_{m,n}$ denote the set of variables $\{p_{i,j} \mid i \in [m], j \in [n]\}$.

Let σ be a substitution to the variables $x_1, \ldots, x_n$ and Φ be a CNF formula on the variables $x_1, \ldots, x_n$. We denote by $\Phi + \sigma$ a CNF obtained from Φ in the following manner: for every clause C of Φ and variable x_i, the formula $\Phi + \sigma$ contains a clause $C \vee x_i^{\sigma(x_i)}$, where for every propositional variable x, x^0 denotes $\neg x$ and x^1 denotes x. Note that it is possible that $C \vee x_i^{\sigma(x_i)}$ is a trivial clause.

Proposition 3.1. *If Φ is unsatisfiable, then $\Phi + \sigma$ has the only satisfying assignment σ.*

Proof. It is straightforward that σ satisfies $\Phi + \sigma$. Assume that τ satisfies $\Phi + \sigma$ and $\tau \neq \sigma$. Consider $i \in [n]$ such that $\tau(x_i) \neq \sigma(x_i)$. Let for every CNF formula ϕ, variable x, and $\alpha \in \{0,1\}$ $\phi[x := \alpha]$ denote the result of the substitution $x := \alpha$ applied to ϕ. Note that the formula $(\Phi + \sigma)[x_i := \tau(x_i)]$ contains all clauses of $\Phi[x_i := \tau(x_i)]$, but Φ is unsatisfiable, hence $(\Phi + \sigma)[x_i := \tau(x_i)]$ is unsatisfiable and τ can not satisfy $\Phi + \sigma$. $\square$

We call an assignment σ to the variables $\mathcal{P}_{m,n}$ *proper* if it satisfies all short clauses of PHP_n^m, that is there are no two pigeons in one hole in σ.

Let f_1, f_2,..., f_k, and g be linear equations in variables $\mathcal{P}_{m,n}$. We say that f_1, f_2,..., f_k properly implies g iff every proper assignment that satisfies all f_1, f_2,..., f_k also satisfies g.

Let F be a linear system in variables $\mathcal{P}_{m,n}$. A *proper rank* of the system F is the size of the minimal set of equations from F such that linear equations from this set properly implies all other equations from F.

Notice that if F does not have a proper solutions, then its proper rank is the size of the minimal subsystem of F that has no proper solutions.

Proposition 3.2. *Let F and G be two linear systems in variables $\mathcal{P}_{m,n}$. Then the proper rank of $F \wedge G$ is at most the sum of the proper ranks of F and G.*

Proof. Let F' and G' be the minimal subsystems of F and G such that F' properly implies all equations from F and G' properly implies all equation from G. Hence $F' \cup G'$ properly implies all equations from $F \wedge G$. $\square$

Remark 3.1. *In contrast to the case of the common rank it is possible that a linear system F does not properly implies linear equation f but the proper rank of $F \wedge f$ does not exceed the proper rank of F. For example, $p_{1,3} + p_{2,3} = 1$ does not properly implies $p_{2,3} = 1$ but the proper rank of $(p_{1,3} + p_{2,3} = 1) \wedge (p_{2,3} = 1)$ equals 1 since $p_{2,3} = 1$ properly implies $p_{1,3} + p_{2,3} = 1$.*

Our goal is to prove the following theorem:

Theorem 3.1. *For every $m > n > 0$ and every proper assignment σ to the variables $\mathcal{P}_{m,n}$ the running time of any drunken $\mathrm{DPLL}(\oplus)$ algorithm $\mathcal{D}_{A,\mathrm{rand}}$ on the formula $\mathrm{PHP}_n^m + \sigma$ is at least $2^{\frac{n-1}{4}}$ with probability at least $1 - 2^{-\frac{n-1}{4}}$.*

In what follows we assume that $m > n$.

We use the following Lemma that was proposed in the paper of Itsykson and Sokolov [12]. We give its proof for the sake of completeness.

Lemma 3.1 ([12])**.** *Let us assume that a linear system $Ap = b$ in the variables $\mathcal{P}_{m,n}$ has at most $\frac{n-1}{2}$ equations and it has a proper solution. Then for every $i \in [m]$ this system has a proper solution that satisfies the long clause $p_{i,1} \vee \ldots \vee p_{i,n}$.*

Proof. Note that if we change 1 to 0 in a proper assignment, then it remains proper. Let the system have k equations; we know that $k \le \frac{n-1}{2}$. We consider a proper solution π of the system $Ap = b$ with the minimum number of ones. We prove that the number of ones in π is at most k. Let the number of ones is greater than k. Consider $k+1$ variables that take value 1 in π: $p_{r_1}, p_{r_2}, \ldots, p_{r_{k+1}}$. Since the matrix A has k rows, the columns that correspond to the variables $p_{r_1}, p_{r_2}, \ldots, p_{r_{k+1}}$ are linearly depended. Therefore, there exists a nontrivial solution π' of the homogeneous system $Ap = 0$ such that every variable with the value 1 in π' is from the set $\{p_{r_1}, p_{r_2}, \ldots, p_{r_{k+1}}\}$. The assignment $\pi' + \pi$ is also a solution of $Ap = b$ and is proper because $\pi' + \pi$ can be obtained from π by changing ones to zeros. Since π' is nontrivial, the number of ones in $\pi' + \pi$ is less than the number of ones in π and this statement contradicts the minimality of π.

The fact that π has at most k ones implies that π has at least $n - k$ empty holes. From the statement of the Lemma we know that $n - k \ge k + 1$; we choose $k + 1$ empty holes with numbers $l_1, l_2, \ldots, l_{k+1}$. We fix $i \in [m]$; the columns of A that correspond to the variables $p_{i,l_1}, \ldots, p_{i,l_{k+1}}$ are linearly depended, therefore, there exists a nontrivial solution τ of the system $Ap = 0$ such that every variable with value 1 in τ is from the set $\{p_{i,l_1}, \ldots, p_{i,l_{k+1}}\}$. The assignment $\pi + \tau$ is a solution of $Ap = b$; $\pi + \tau$ is proper since holes with numbers $l_1, l_2, \ldots, l_{k+1}$ are empty in π, and τ puts at most one pigeon to them (if τ puts a pigeon in a hole, then this is the i-th pigeon). The assignment $\pi + \tau$ satisfies $p_{i,1} \vee p_{i,2} \vee \ldots \vee p_{i,n}$ because τ is nontrivial. $\square$

Corollary 3.1. *If a linear system F in variables $\mathcal{P}_{m,n}$ has a proper solution and the proper rank of F is at most $\frac{n-1}{2}$ then for every $i \in [m]$ the system F has a proper solution that satisfies the long clause $p_{i,1} \vee \ldots \vee p_{i,n}$.*

Proof. Let F' be the minimal subsystem of F that properly implies all equations from F. The number of equations in F' is the proper rank of F that is at most $\frac{n-1}{2}$. F' also has a proper solution since it is a subsystem of F. Thus by Lemma 3.1 F' has a proper solution that satisfies $p_{i,1} \vee \ldots \vee p_{i,n}$. This solution should also be a solution of F by the choice of F'. $\square$

Corollary 3.2. *Assume that a linear system $Ap = b$ in the variables $\mathcal{P}_{m,n}$ has a proper solution and its proper rank is at most $\frac{n-1}{2}$, then this system has at least two proper solutions.*

Proof. Let σ be a solution of $Ap = b$. Since PHP^m_n is unsatisfiable there is a long clause C such that σ falsify this clause. Though, by Corollary 3.1 for any clause there is a proper solution τ of $Ap = b$ that satisfies the clause C. Hence $\tau \neq \sigma$, thus τ and σ are different proper solutions of $Ap = b$. □

Lemma 3.2. *Let a system of linear equations $Ap = b$ in the variables $\mathcal{P}_{m,n}$ have a proper solution and let its proper rank be at most $\frac{n-1}{4}$. Then the size of any parity decision tree for $(\mathrm{PHP}^m_n, Ap = b)$ is at least $2^{\frac{n-1}{4}}$.*

Corollary 3.3. *Let a system $Ap = b$ of linear equations in the variables $\mathcal{P}_{m,n}$ have a proper solution and let its proper rank be at most $\frac{n-1}{4}$. Let σ be a proper assignment to variables $\mathcal{P}_{m,n}$ that does not satisfy $Ap = b$. Then the size of any parity decision tree for $(\mathrm{PHP}^m_n + \sigma, Ap = b)$ is at least $2^{\frac{n-1}{4}}$.*

Proof of Corollary 3.3. Consider a parity decision tree T for $(\mathrm{PHP}^m_n + \sigma, Ap = b)$. Since σ is the only satisfying assignment of $\mathrm{PHP}^m_n + \sigma$ and it does not satisfy $Ap = b$, there are no satisfying leaves in T. We claim that the tree T may be also considered as a parity decision tree for $(\mathrm{PHP}^m_n, Ap = b)$, and thus the size of T is at least $2^{\frac{n-1}{4}}$ by Lemma 3.2.

Consider any leaf of T. If this leaf corresponds to the situation where the system F contradicts to a clause C of $\mathrm{PHP}^m_n + \sigma$, then it also contradicts to some clause C' of PHP^m_n since every clause of $\mathrm{PHP}^m_n + \sigma$ is a superclause of a some clause of PHP^m_n Thus, tree T also may be considered as a parity decision tree for $(\mathrm{PHP}^m_n, Ap = b)$. □

In order to prove Lemma 3.2 we generalize Prover–Delayer games introduced by Pudlak and Impagliazzo [15].

Consider the following game with two players: Prover and Delayer. They are given a CNF formula Φ and a system of linear equations F such that formula $\Phi \wedge F$ is unsatisfiable. On each step Prover chooses a linear form f that depends on variables of formula Φ, then Delayer may choose a value $\alpha \in \{0, 1\}$ of f or return $*$. If Delayer returns $*$, then Prover chooses a value $\alpha \in \{0, 1\}$ of f by himself. We add the equality $f = \alpha$ in the system F. The game ends if the current linear system F is inconsistent or refutes some clause of Φ. Delayer earns a coin for every $*$. The goal of Delayer is to earn the maximum number of coins and the goal of Prover is to minimize the number of coins earned by Delayer.

Lemma 3.3 (similar to [15])**.** *Consider some CNF formula Φ and linear system F such that $\Phi \wedge F$ is unsatisfiable. Assume that there is a strategy for Delayer that allows Delayer to earn t coins, then the size of any parity decision tree for (Φ, F) is at least 2^t.*

Proof. Consider some parity decision tree T for (Φ, F). We construct a probabilistic distribution on the leaves of T that corresponds to the strategy of Delayer and the following randomized strategy of Prover. Prover uses the tree T, initially he asks the question for the linear form in the root, if Delayer returns $*$, Prover chooses a value at random with equal probabilities and go to the next vertex along an edge labeled with the chosen value. By the statement of the Lemma the probability that the game will finish in every particular leaf is at most 2^{-t}. Since with probability 1 the game will finish in a leaf, the number of leaves of T is at least 2^t. □

Proof of Lemma 3.2. Let us construct a strategy for Delayer that will guarantee that Delayer earns at least $\frac{n-1}{4}$ coins. Let G be the current linear system that consists of all equations that are already made by Prover or Delayer in the game and equations from the system $Ap = b$. At the beginning G equals $Ap = b$. The strategy of the Delayer the following: assume that Prover chooses a linear form f, then if G properly implies $f = \alpha$ for some $\alpha \in \{0, 1\}$, then Delayer returns α, otherwise Delayer returns $*$.

We prove by induction on the number of steps that the following invariant holds: the system G always has a proper solution. Basis case is true since $Ap = b$ has a proper solution. Assume that Prover chooses a linear form f. If G has a proper solution, then either $G \wedge (f = 0)$ or $G \wedge (f = 1)$ has the same proper solution. Assume that for some $\alpha \in \{0, 1\}$, $G \wedge (f = \alpha)$ does not have proper solutions. In this case G properly implies $f = 1 - \alpha$ hence Delayer chooses the value $1 - \alpha$ and $F \wedge G \wedge (f = 1 - \alpha)$ has a proper solution. Consider three situations at the end of the game.

- The system G becomes unsatisfiable. This situation is impossible since G has a proper solution.
- The system G contradicts a short clause. This situation is also impossible since G has a proper solution.
- The system G contradicts a long clause $p_{i,1} \vee \ldots \vee p_{i,n}$. Let G' be a subsystem of G that corresponds to answers $*$ of Delayer. By the construction every equation from G is properly implied from $(Ap = b) \wedge G'$. Hence, $(Ap = b) \wedge G'$ does not have proper solutions that satisfy $p_{i,1} \vee \ldots \vee p_{i,n}$. Corollary 3.1 implies that the proper rank of the system $(Ap = b) \wedge G'$ is greater than $\frac{n-1}{2}$. By Proposition 3.2 the rank of G' is greater than $\frac{n-1}{4}$, hence G' contains more than $\frac{n-1}{4}$ equations. Note that Delayer earns a coin for every equation in G', hence Delayer earns more than $\frac{n-1}{4}$ coins. Hence by Lemma 3.3 the size of any parity decision tree for $(\mathrm{PHP}^m_n, Ap = b)$ is at least $2^{\frac{n-1}{4}}$.

$\square$

Now we are ready to prove Theorem 3.1.

Proof of Theorem 3.1. We may assume that a heuristic A does not use random bits. Indeed otherwise we may prove the lower bound for fixed random bits of heuristic A and then apply the averaging principle to handle the case of randomized A. If A is deterministic we may consider the whole parity decision tree T of the algorithm $\mathcal{D}_{A,B}$ on the input $\mathrm{PHP}^m_n + \sigma$ that corresponds to all possible answers of heuristic B. For every execution of the algorithm $\mathcal{D}_{A,B}$, this algorithm bypasses only a part of the tree T until it finds a satisfying assignment. Tree T contains exactly one branch that correspond to the satisfying assignment σ; we call this branch a satisfying path. We prove that with high probability the algorithm will deviate from the satisfying path and will fall in a hard unsatisfiable subtree.

Assume that algorithm is in the state in the accepting path, that is the current linear system F is satisfied by σ. There are two possibilities to deviate from the satisfying path:

1. The algorithm chooses an equation $f = \alpha$ such that the system $F \wedge (f = \alpha)$ has no proper solutions. In this case $f = 1 - \alpha$ is properly implied by F. Thus the adding of $f = 1 - \alpha$ to F does not increase the proper rank.
2. The algorithm chooses an equation $f = \alpha$ and the system $F \wedge (f = \alpha)$ has proper solutions but σ is not a solution of $F \wedge (f = \alpha)$. In this case the proper rank of $F \wedge (f = \alpha)$ may be larger by one then the proper rank of F (but it is also possible that the proper rank of $F \wedge (f = \alpha)$ does not exceed the proper rank of F, see Remark 3.1). If the proper rank of $F \wedge (f = \alpha)$ is at most $\frac{n-1}{4}$, then by Corollary 3.3 the algorithm falls in an unsatisfiable subtree of size at least $2^{(n-1)/4}$.

Consider the leaf of the satisfying path; the linear system F in this leaf has the only solution σ. Hence by Corollary 3.2 the proper rank of F is greater than $\frac{n-1}{2}$. The value of the proper rank of F in the root is zero, the rank of F increases along the satisfying path. Consider the first nodes of the path when the proper rank equals 1, 2, …, $\frac{n-1}{4}$. The algorithm should visit this nodes, hence it should visit the predecessors of these nodes. In every of these predecessors algorithm have chance $\frac{1}{2}$ to deviate from the acceptance path. And since the proper rank increases, all this deviations correspond to the case 2 of the above. Thus, with probability at least $1 - 2^{-\frac{n-1}{4}}$ the algorithm goes to an unsatisfiable subtree of size at least $2^{\frac{n-1}{4}}$. $\square$

Acknowledgements. The authors are grateful to Dmitry Sokolov for fruitful discussions.

References

[1] Michael Alekhnovich, Edward A. Hirsch, and Dmitry Itsykson. Exponential Lower Bounds for the Running Time of DPLL Algorithms on Satisfiable Formulas. *J. Autom. Reason.*, 35(1-3):51–72, 2005.

[2] Boaz Barak. A Probabilistic-Time Hierarchy Theorem for "Slightly Non-uniform" Algorithms. In *RANDOM '02: Proceedings of the 6th International Workshop on Randomization and Approximation Techniques*, pages 194–208, London, UK, 2002. Springer-Verlag.

[3] Paul Beame, Henry A Kautz, and Ashish Sabharwal. Towards Understanding and Harnessing the Potential of Clause Learning. *J. Artif. Intell. Res. (JAIR)*, 22:319–351, 2004.

[4] Paul Beame and Toniann Pitassi. An Exponential Separation Between the Parity Principle and the Pigeonhole Principle. *Annals of Pure and Applied Logic*, 80(3):195–228, 1996.

[5] James Cook, Omid Etesami, Rachel Miller, and Luca Trevisan. Goldreich's One-Way Function Candidate and Myopic Backtracking Algorithms. In Omer Reingold, editor, *Theory of Cryptography, 6th Theory of Cryptography Conference, TCC 2009, San Francisco, CA, USA, March 15-17, 2009. Proceedings*, volume 5444 of *Lecture Notes in Computer Science*, pages 521–538. Springer, 2009.

[6] Stefan S. Dantchev and Søren Riis. Tree Resolution Proofs of the Weak Pigeon-Hole Principle. In *Proceedings of the 16th Annual IEEE Conference on Computational Complexity, Chicago, Illinois, USA, June 18-21, 2001*, pages 69–75. IEEE Computer Society, 2001.

[7] Martin Davis, George Logemann, and Donald W Loveland. A machine program for theorem-proving. *Communications of the ACM*, 5(7):394–397, 1962.

[8] Martin Davis and Hilary Putnam. A Computing Procedure for Quantification Theory. *Journal of the ACM*, 7(3):201–215, 1960.

[9] Michal Garlík and Leszek A Kołodziejczyk. Some subsystems of constant-depth Frege with parity. Preprint, 2017.

[10] Dmitry Itsykson. Lower Bound on Average-Case Complexity of Inversion of Goldreich's Function by Drunken Backtracking Algorithms. *Theory of Computing Systems*, 54(2):261–276, 2014.

[11] Dmitry Itsykson and Dmitry Sokolov. The complexity of inverting explicit Goldreich's function by DPLL algorithms. *Journal of Mathematical Sciences*, 188(1):47–58, 2013.

[12] Dmitry Itsykson and Dmitry Sokolov. Lower Bounds for Splittings by Linear Combinations. In *Mathematical Foundations of Computer Science 2014 - 39th International Symposium, MFCS 2014, Budapest, Hungary, August 25-29, 2014. Proceedings, Part II*, pages 372–383, 2014.

[13] Jan Krajiček. Randomized feasible interpolation and monotone circuits with a local oracle. *CoRR*, abs/1611.0, 2016.

[14] Vsevolod Oparin. Tight Upper Bound on Splitting by Linear Combinations for Pigeonhole Principle. In Nadia Creignou and Daniel Le Berre, editors, *Theory and Applications of Satisfiability Testing - SAT 2016 - 19th International Conference, Bordeaux, France, July 5-8, 2016, Proceedings*, volume 9710 of *Lecture Notes in Computer Science*, pages 77–84. Springer, 2016.

[15] Pavel Pudlák and Russell Impagliazzo. A lower bound for DLL algorithms for k-SAT (preliminary version). In *Proceedings of the Eleventh Annual ACM-SIAM Symposium on Discrete Algorithms, January 9-11, 2000, San Francisco, CA, USA.*, pages 128–136, 2000.

[16] Pavel Pudlák, Dominik Scheder, and Navid Talebanfard. Tighter Hard Instances for PPSZ. *CoRR*, abs/1611.0, 2016.

[17] Alexander A Razborov. Resolution lower bounds for perfect matching principles. *Journal of Computer and System Sciences*, 69(1):3–27, 2004.

[18] Dominik Scheder, Bangsheng Tang, Shiteng Chen, and Navid Talebanfard. Exponential Lower Bounds for the PPSZ k-SAT Algorithm. In Sanjeev Khanna, editor, *Proceedings of the Twenty-Fourth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2013, New Orleans, Louisiana, USA, January 6-8, 2013*, pages 1253–1263. SIAM, 2013.

[19] Alasdair Urquhart. Hard Examples for Resolution. *Journal of the ACM*, 34(1):209–219, 1987.