Записки научных
семинаров ПОМИ
Том 399, 2012 г.

D. M. Itsykson, D. O. Sokolov

THE COMPLEXITY OF INVERSION OF EXPLICIT GOLDREICH'S FUNCTION BY DPLL ALGORITHMS

Abstract. The Goldreich's function has n binary inputs and n binary outputs. Every output depends on d inputs and is computed from them by the fixed predicate of arity d. Every Goldreich's function is defined by it's dependency graph G and predicate P. In 2000 O. Goldreich formulated a conjecture that if G is an expander and P is a random predicate of arity d then the corresponding function is one way. In this paper we give a simple proof of the exponential lower bound of the Goldreich's function inversion by myopic DPLL algorithms. A dependency graph G in our construction may be based on an arbitrary expander, particulary it is possible to use an explicit expander; while all all previously known results are based on random dependency graphs. The predicate P may be linear or slightly nonlinear. Our construction may be used in the proof of lower bounds for drunken DPLL algorithms as well.

§1. Introduction

This work continues [1,2,17,18] and is devoted to lower bounds of DPLL (for Davis, Putnam, Logemann, and Loveland) algorithms on satisfiable formulas. DPLL 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. It returns the result "Satisfiable" if at least one of recursive calls returns "Satisfiable" (note that it is not necessary to make the second call if the first one was successful). Recursion stops if the input formula becomes trivial. That is, the algorithm is only allowed to backtrack when unsatisfiability in the current branch is proved. A DPLL algorithm is defined by simplification rules and two heuristics: the heuristic **A** chooses

Key words and phrases: DPLL algorithm, expander, one-way function, lower bounds.

Supported by Federal Target Programme "Scientific and scientific-pedagogical personnel of the innovative Russia" 2009-2013 and RFBR grant 11-01-00760-a) and the president grant NSh-5282.2010.1 The second author was also supported by Yandex Fellowship and CS-club scholarship.

a variable for splitting and the heuristic **B** chooses a value that will be investigated first.

The behaivior of DPLL algorithms on unsatisfiable formulas is equivalent to tree-like resolution proofs. Therefore lower bounds on DPLL algorithms on unsatisfiable formulas follow from lower bound for resolutions [9]. However the most interesting inputs are satisfiable formulas. Consider for example formulas that code the problem of inversion of one-way function. The most important case for practice is the case there one-way function indeed has preimage. There is no hope of proving a superpolynomial lower bound for all DPLL algorithms on satisfiable formulas since if $\mathrm{P} = \mathrm{NP}$, then the heuristic that chooses the value of a variable that would be investigated first may always choose the correct value.

Exponential lower bounds on running time of myopic and drunken DPLL algorithms on satisfiable formulas were proved in the paper [2]; these two classes of DPLL algorithms cover a lot of known DPLL algorithms. In myopic algorithms heuristics that choose a variable for splitting and that choose a value that will be investigated first have the following restrictions: they can see the formula with erased signs of negations and they also know the number of positive and negative occurrences of every variable and they also can request $K = n^{1-\varepsilon}$ clauses of the formula to read them precisely. In drunken algorithms the heuristic that chooses variable for splitting may be arbitrary, while the first substituted value is chosen at random with equal probabilities. Lower bounds for myopic algorithms were proved on the formulas that code the system of linear equations over $\mathbb{F}_2$ based on expander matrices; lower bounds for drunken algorithms were proved on artificial formulas that are based on hard examples for resolution.

The paper [1] gives a cryptographic view on [2]. Namely it was noted in [1] that the lower bound for myopic algorithms [2] was proved on the formulas that code the problem of inversion of Goldreich's function based on linear predicate. Goldreich's function [4] has n binary inputs and n binary outputs. Every output depends on d inputs and is computed from them by a fixed predicate of arity d. Goldreich conjectured that if the dependency graph is an expander and the predicate is random, then the resulting function is one-way. However, linear functions are not interesting from the cryptographic point of view since they can be easily inverted by Gaussian elimination. The main goal of [1] was the proof of lower bound for a function that is potentially hard to invert. J. Cook et al. consider Goldreich's function based on the predicate $x_1+x_2+\cdots+x_{d-2}+x_{d-1}x_d$ and a random graph (a random graph is an expander with high probability). They have proved the exponential lower bound for the weakened[1] variant of myopic algorithms. Recently Itsykson [17] and Miller [18] independently proved the lower bound on the complexity of inversion of Goldreich's function based on random graph and predicate of type $x_1+x_2+\cdots+x_{d-k}+Q(x_{d-k+1},\dots x_d)$, where Q is an arbitrary predicate of arity k and $k < d/4$ by drunken algorithms. We should note that the proof from [1] works for this type of predicates as well.

The construction of Goldreich's function in all papers listed above was randomized. In this paper we suggest an explicit construction of Goldreich's function based on expanders (for example the explicit expander from [16] fits our purposes). It is possible to use those formulas in the proof of exponential lower bound for drunken algorithms from [17]. In this paper we demonstrate the lower bound for myopic algorithms. Our proof is technically much simpler than proofs from [2] and [1]. We prove lower bound for the general notion of myopic algorithms (according to the definition from [2]) instead of the weakened variant that was used in [1].

Our Goldreich's function has the following structure: it is the sum of two Goldreich's functions: linear and nonlinear. The linear part is necessary for proving the lower bound for DPLL algorithms while the nonlinear part makes our function hard to invert in practice. The linear part is based on an expander, while nonlinear part may be almost arbitrary but it should depend only on $n^{\varepsilon/2}$ variables. Of course an adversary may guess the value of variables from nonlinear part and solve the resulting linear system by Gaussian elimination but the running time of such algorithm is $2^{n^{\varepsilon/2}}$ (still exponential), therefore we believe that there are hard invertible functions among our functions. We actually do not use in the proof the fact that the nonlinear part of predicate is the same for every bit of the output.

The plan of the proof is the following: first of all we slightly modify the expander from the linear part so that its adjacency matrix would have high rank. Since the nonlinear part of our function depends on very few variables we conclude that our Goldreich's function is almost a bijection. In order to prove the lower bound we first of all prove the lower bound for unsatisfiable formulas using lower bound techniques for resolutions from

[1]In contrast to [1, 2] did not allow the DPLL algorithm to use pure literal simplification rules and also myopic algorithms from [1] may read only constant (opposite to $n^{1-\varepsilon}$) number of clauses per step.

[5]. Using almost linearity and almost bijectivity we prove that with high probability the myopic algorithm makes the formula unsatisfiable during first several steps and we apply the lower bound for unsatisfiable formulas.

Our proof has one disadvantage compared to the proof from [2]; namely our proof works for expanders with degrees, that are large enough, while the technique from [2] works for degrees, that are at least 3. However the proof from [2] of the fact that a myopic algorithm with high probability makes the formula unsatisfiable during first several steps is complicated, while our proof is intuitive and based on the simple fact from elementary linear algebra: a dimension of a solution space of a satisfiable linear system $Ax = b$ does not depend on the right hand side b.

§2. Preliminaries

Let $X = \{x_1, x_2, \dots, x_n\}$ be the set of propositional variables.

A partial substitution is a function $\rho : X \to \{0, 1, *\}$, that maps a variable to its value or leaves it free. The set $\mathrm{Vars}(\rho) = \rho^{-1}(\{0, 1\})$ is the support of the substitution; we denote $|\rho| = |\mathrm{Vars}(\rho)|$.

If ρ_1 and ρ_2 are two partial substitutions with disjoint support then the substitution $\rho_1 \cup \rho_2$ can be defined by the natural way.

We say that a string $y \in \{0, 1\}^n$ is consistent with the partial substitution ρ (we denote it $y \sim \rho$) if for all x_j from the support of ρ the following is satisfied $y_j = \rho(x_j)$.

2.1. DPLL algorithms. We consider a wide class of SAT algorithms: DPLL (or backtracking) algorithms. A DPLL algorithm is defined by two *heuristics* (procedures): 1) Procedure **A** maps a CNF formula to one of its variables. (This is the variable for splitting). 2) Procedure **B** maps a CNF formula and its variable to $\{0, 1\}$. (This value will be investigated at first).

An algorithm may also use some syntactic *simplification rules*. Simplification rules may modify the formula without affecting its satisfiability and may also make substitutions to its variables if their values can be inferred from the satisfiability of the initial formula.

A DPLL algorithm is a recursive algorithm. Its input is a formula φ and a partial substitution ρ.

Algorithm 2.1. Input: formula φ and substitution ρ

- Simplify φ by means of simplification rules (assume that simplification rules change φ and ρ; all variables that are substituted by ρ should be deleted from φ).

- If current formula is empty (that is, all its clauses are satisfied by ρ), then return ρ. If formula contains an empty clause (unsatisfiable), then return "formula is unsatisfiable".
- $x_j := \mathbf{A}(\varphi)$; $c := \mathbf{B}(\varphi, x_j)$
- Make a recursive call with the input $(\varphi[x_j := c], \rho \cup \{x_j := c\})$ if the result is "formula is unsatisfiable", then make a recursive call with the input $(\varphi[x_j := 1 - c], \rho \cup \{x_j := 1 - c\})$ and return its result, otherwise return the result of the first recursive call.

Definition 2.1. *Myopic algorithms* [2] *are* DPLL *algorithms, where heuristics* **A** *and* **B** *have the following restrictions:*

- *They can see the whole formula with erased signs of negations.*
- *For every variable they know the number of its positive and the number of its negative occurrences.*
- *They may request to read $K = o(n)$ clauses to read precisely (with negation signs).*

Simplification rules: 1) Unit clause elimination*: if formula contains a clause with only one literal, then make a substitution that satisfies that clause.* 2) Pure literals rule*: if formula contains a variable that has only positive or only negative occurrences, then substitute it with the corresponding value.*

The running time of a DPLL algorithm for a given sequence of random bits (heuristics **A** and **B** may be randomized) is the number of recursive calls.

2.2. Expanders. All graphs that we consider in this paper are bipartite multigraphs with each part containing n vertices. The first part we denote by $X = \{x_1, x_2, \dots, x_n\}$ and the second we denote by $Y = \{y_1, y_2, \dots, y_n\}$. Every vertex from the set Y has an ordered list of its neighbours from the set X (repetitions are allowed). All considered graphs are d-regular: the degree of every vertex from Y is equal to d, where d is a constant.

Every graph has its adjacency matrix over $\mathbb{F}_2$. Rows of this matrix correspond to the set Y and columns correspond to the set X, the element with coordinates (y, x) contains the parity of the number of edges between y and x. We stress that such adjacency matrix does not uniquely determine a graph since it does not contain information about the number of parallel edges and about the order of edges; it only contains the information about the parity of the number of edges between two vertices.

Consider the example: let $X = \{x_1, x_2, x_3, x_4\}$ and $Y = \{y_1, y_2, y_3, y_4\}$, the lists of neighbours in graph G_0 are following: $\ell_{y_1} = \{x_1, x_2, x_2\}, \ell_{y_2} =$

$\{x_2, x_1, x_3\}, \ell_{y_3} = \{x_2, x_2, x_2\}, \ell_{y_4} = \{x_4, x_1, x_4\}$. The graph G_0 has the following adjacency matrix over $\mathbb{F}_2$:

	1	2	3	4
y_1	1	0	0	0
y_2	1	1	1	0
y_3	0	1	0	0
y_4	1	0	0	0

For set $A \subseteq Y$ we denote $\Gamma(A)$ (the set of neighbours of A) the set of vertices from X that are connected with at least one vertex from A; we denote $\delta(A)$ (the boundary of A) the set of vertices from X that have exactly one incoming edge from the set A.

Definition 2.2. *The graph G is a (r, d, c)-expander if* 1) *the degree of any vertex in Y is equal to d; 2) for any set $A \subseteq Y, |A| \leqslant r$ we have $|\Gamma(A)| \geqslant c|A|$. The graph G is called a (r, d, c)-boundary expander if the second condition is replaced by: 2) for any set $A \subseteq Y, |A| \leqslant r$ we have $|\delta(A)| \geqslant c|A|$.*

Lemma 2.1 (cf. [2], Lemma 1). *Every (r, d, c)-expander is also a $(r, d, 2c-d)$-boundary expander.*

Proof. Let $A \subseteq Y$, $|A| \leqslant r$, then $|\Gamma(A)| \geqslant c|A|$. The number of edges between A and $\Gamma(A)$ may be estimated:

$$d|A| \geqslant |\delta(A)| + 2|\Gamma(A) \setminus \delta(A)| = 2|\Gamma(A)| - |\delta(A)| \geqslant 2c|A| - |\delta(A)|.$$

Finally we get $|\delta(A)| \geqslant (2c - d)|A|$. □

We need boundary expanders; for this it is enough to have an expander with constant $c > d/2$. For example, a random graph is an appropriate expander.

Lemma 2.2 ([6], Lemma 1.9). *For $d \geqslant 32$, for all big enough n a random bipartite d-regular graph, where parts X and Y contain n vertices is a $(\frac{n}{10d}, d, \frac{5}{8}d)$-expander with probability* 0.9, *if for every vertex in Y d edges are chosen independently at random (with repetitions).*

Corollary 2.1. *In terms of Lemma* 2.2 *this graph is a $(\frac{n}{10d}, d, \frac{1}{4}d)$-boundary expander.*

Proof. Follows from Lemma 2.1. □

There are also explicit constructions of such expanders:

Lemma 2.3 ([16]). *For every constant $\epsilon > 0$ there is a constant d such that it is possible to construct a (r, d, c)-expander in polynomial of n time, where $c = (1-\epsilon)d$, $r = \Omega(n/d)$.*

Corollary 2.2. *This graph is a $(\Omega(n/d), d, (1-2\epsilon)d)$-boundary expander.*

2.3. Goldreich's function. O. Goldreich in the paper [4] introduces a function $f : \{0,1\}^n \to \{0,1\}^n$ defined by a graph G and a predicate $P : \{0,1\}^d \to \{0,1\}$. Every string from $\{0,1\}^n$ assignes some value to the variables from the set $X = \{x_1, x_2, \dots, x_n\}$. The value of $(f(x))_j$ (jth symbol of the string $f(x)$) is computed in the following way: if y_j has neighbours $x_{j_1}, x_{j_2}, \dots, x_{j_d}$, then $(f(x))_j = P(x_{j_1}, x_{j_2}, \dots, x_{j_d})$.

2.4. Formulas from Goldreich's function. Now we describe the way we code the problem of inversion of Goldreich's function as instance of CNF satisfiability problem.

Let $g : \{0,1\}^\ell \to \{0,1\}$, the canonical CNF representation of g is the following: for every $c \in \{0,1\}^\ell$ that satisfies $g(c) = 0$ we write the clause $x_1^{c_1} \vee x_2^{c_2} \vee \dots \vee x_\ell^{c_\ell}$, where $x_i^0 = x_i$ and $x_i^1 = \neg x_i$. The whole formula is the conjunction of all written clauses.

Let f be the Goldreich's function based on the graph G and the predicate P. We represent the equation $f(x) = b$ in the following way: for every vertex $y_j \in Y$ that has neighbours $x_{j1}, x_{j2}, \dots, x_{jd}$ we put down the canonical CNF representation of the equality $b_j = P(x_{j1}, x_{j2}, \dots, x_{jd})$ using variables $x_{j1}, x_{j2}, \dots, x_{jd}$. The conjunctions of all those formulas we denote $\Phi_{f(x)=b}$. The part of this formula that corresponds to the vertices from the set $A \subseteq Y$ we denote $\Phi^A_{f(x)=b}$.

Lemma 2.4. *If a function $g : \{0,1\}^\ell \to \{0,1\}$ is linear on at least two variables, then the canonical CNF representation of g has exactly $2^{\ell-1}$ clauses and every variable has an equal number of positive and negative occurrences.*

Proof. Let g have the following $\mathbb{F}_2$ representation $g(x_1, x_2, \dots, x_n) = x_1 + x_2 + h(x_3, \dots, x_\ell)$. Let us denote $T_0 = h^{-1}(0)$, $T_1 = h^{-1}(1)$. Then

$$g^{-1}(0) = \{00y \mid y \in T_0\} \cup \{11y \mid y \in T_0\} \cup \{01x \mid y \in T_1\} \cup \{10y \mid y \in T_1\}.$$

The latter shows that $|g^{-1}(0)| = 2^{l-1}$ and every variable has an equal number of positive and negative occurrences. □

Lemma 2.4 implies that if a myopic algorithm does not see negation signs, then it can't differ $g(x) = 0$ from $g(x) = 1$ when g is linear on at least 2 variables. Also we note that a canonical CNF formula is still canonical after substitution of the value of a variable.

§3. Almost bijective Goldreich's function

3.1. Linear function. Let G_1 be a d_1-regular graph and G_2 be a d_2-regular graph with the same sets X and Y. $G_1 + G_2$ is $(d_1 + d_2)$-regular graph such that for every vertex from Y the list of neighbours is a concatenation of lists of neighbours in graph G_1 and graph G_2. The adjacency matrix of G_1+G_2 is the sum of adjacency matrices of G_1 and G_2 modula 2.

Proposition 3.1. *If graph G is a (r, d, c)-expander and G' is a d'-regular graph, then $G + G'$ is a $(r, d + d', c)$-expander.*

Theorem 3.1. *Given a graph G it is possible to construct in polynomial of n time a 1-regular graph T such that the rank of adjacency matrix of $G + T$ is at least $n - 1$.*

Proof. First of all we prove the auxiliary lemma:

Lemma 3.1. *Let $a = (\alpha_1, \dots, \alpha_n) \in \mathbb{F}_2^n$. Then there are at least $n - 1$ linearly independent vectors among $b_i = (\alpha_1, \dots, \alpha_{i-1}, \alpha_i + 1, \alpha_{i+1}, \dots, \alpha_n)$, where $1 \leqslant i \leqslant n$.*

Proof. Let us consider the matrix A of size $n \times n$; all columns of A are equal to vector a. Vectors b_i are columns of the matrix $A+E$, where E is the identity matrix. Since the rank of the sum of matrices is less then or equal to the the sum of ranks we may conclude that $n = \operatorname{rk} E \leqslant \operatorname{rk}(A+E) + \operatorname{rk} A$. All columns of A are the same, hence $\operatorname{rk} A \leqslant 1$ and $\operatorname{rk}(A+E) \geqslant n-1$. □

Now we describe the construction of graph T. We start from an empty set of edges and we will add one edge per step. On the ith step for $1 \leqslant i \leqslant n - 1$ we add a neighbour to the vertex $y_i \in Y$ in such a way that the first i rows of $G + T$ are linearly independent. It can be done by the Lemma 3.1 (we apply the Lemma to the ith row of the matrix of graph G). We add an arbitrary neighbour to vertex y_n. □

Corollary 3.1. *If G is a (r, d, c)-expander, then the graph $G + T$ from the theorem is a $(r, d+1, c)$-expander and the Goldreich's function f based on $G + T$ and a linear predicate of arity $d + 1$ has the following property: for every $b \in \{0, 1\}^n$ the size of the set $f^{-1}(b)$ is at most 2.*

3.2. Slightly nonlinear Goldreich's function. Let $R \subseteq X$ be some subset of X. R-graph is a regular graph such that all vertices from $X \setminus R$ have degree 0.

Lemma 3.2. *Let G be a $(d-k)$-regular graph with an adjacency matrix of rank at least $n-1$ and H be a k-regular R-graph. Let f be Goldreich's function based on $G+H$ and predicate $x_1+x_2+\dots+x_{d-k}+Q(x_{d-k+1},\dots,x_d)$, where Q is an arbitrary predicate of arity k. Then for every $b \in \{0,1\}^n$ the size of the set $f^{-1}(b)$ is at most $2^{|R|+1}$.*

Proof. We consider the system of equalities $f(x) = b$ and fix values of all variables from the set R. We get the linear system whose matrix equals to the matrix of graph G after removing the columns from the set R. The matrix of G has rank $n-1$, therefore the resulting system has at most two solutions. Hence the initial system $f(x) = b$ has at most $2^{|R|+1}$ solutions. □

§4. Lower bound on unsatisfiable formulas

We say that a variable is *sensitive* if by changing its value we change the value of the formula (for every assignment of values of other variables). (The boolean function that corresponds to the formula is linear on all its sensitive variables).

Theorem 4.1 ([17]). *Let f be a Goldreich's function based on G and P, where graph G is a (r,d,c)-boundary expander and predicate P contains at most k insensitive variables; ρ is a partial assignment to variables of X such that the formula $\Phi_{f(x)=b}|_\rho$ is unsatisfiable and for any set of vertices $A \subseteq Y$, $|A| < \frac{r}{2}$, the formula $\Phi^A_{f(x)=b}|_\rho$ is satisfiable. Then the running time of any DPLL algorithm (that does not use simplification rules) on the formula $\Phi_{f(x)=b}|_\rho$ is at least $2^{\frac{(c-k)r}{4}-|\rho|-d}$.*

Proof. See Appendix A. □

§5. Lower bound on satisfiable formulas

In this section G is a (r,d,c)-boundary expander where d is a constant, $r = \Omega(n)$ and $c > 5$. Let $2 < k < c-2$; we assume that graph G has the type $G_L + H$, where G_L is a $(d-k)$-regular, H is a k-regular R-graph and the rank of adjacency matrix of G_L is at least $n-1$. We also assume that $R = o\left(\frac{n}{K}\right)$, where K is the number of clauses that myopic algorithm may read with negation signs per step of recursion. Let $P(x_1, \dots, x_d) = x_1 + \dots + x_{d-k} + Q(x_{d-k+1}, \dots, x_d)$, where Q is arbitrary predicate of arity k. The Goldreich's function f is based on G and P and f is linear on variables $X \setminus R$.

Now we describe the construction of graph that suits properties above. We choose $\epsilon = \frac{1}{4k+10}$ and for given ϵ we construct an $(r, d, (1-\epsilon)d)$-expander H by Lemma 2.3. The constant d satisfies the inequality $d \geqslant 4k + 10$. By Theorem 3.1 we add to the constructed graph such 1-regular graph T that the resulting graph $H+T$ has the adjacency matrix with rank at least $n-1$. The resulting graph is a $(r, d+1, (1-\frac{1}{4k+10})d)$-expander. We choose the subset $R \subseteq X$ of size $o(n/K)$ and k-regular R-graph F. We define $G = H + T + F$; graph G is a $(r, d+1+k, (1-\frac{1}{4k+10})d)$-expander and hence a $(r, d+1+k, d(1-\frac{2}{4k+10}) - k - 1)$-boundary expander. For $k > 2$ the inequality $d(1-\frac{2}{4k+10}) - k - 1 > k + 2$ holds.

5.1. Closure. The next technical definition formalizes the following simple idea: suppose that the set J is removed from the part X of G and we want to remove a set I from Y (and also $\Gamma(I)$ from X) such that the resulting graph becomes a $(r/2, d, k+1)$-boundary expander. We construct such I step by step removing sets with small boundary from Y.

Definition 5.1. *Let $J \subseteq X$. The set of vertices $I \subseteq Y$ is called k-closure of the set J if there is a finite sequence of sets $I_1, I_2, \dots, I_m$ (we denote $C_\ell = \bigcup_{1 \leqslant i \leqslant \ell} I_i$, $C_0 = \varnothing$), such that the following properties are satisfied:*

- *$I_\ell \subseteq Y$ and $0 < |I_\ell| \leqslant \frac{r}{2}$ for all $1 \leqslant \ell \leqslant m$;*
- *$I_i \cap I_j = \varnothing$ for all $1 \leqslant i, j \leqslant m$;*
- *$|\delta(I_\ell) \setminus (\Gamma(C_{\ell-1}) \cup J)| \leqslant (1+k)|I_\ell|$; for all $1 \leqslant \ell \leqslant m$;*
- *for all $I' \subseteq Y \setminus C_m$ if $0 < |I'| \leqslant \frac{r}{2}$, then $|\delta(I') \setminus (\Gamma(C_m) \cup J)| > (1+k)|I'|$;*
- *$I = C_m$.*

The set of all k-closures of the set J we denote as $\mathrm{Cl}^k(J)$.

Lemma 5.1. (1) *For every set $J \subseteq X$ there exists a k-closure.* (2) *Let $J_1 \subseteq J_2$, then for every $I_1 \in \mathrm{Cl}^k(J_1)$ there exists $I_2 \in \mathrm{Cl}^k(J_2)$ such that $I_1 \subseteq I_2$.*

Proof. See Appendix B. □

Lemma 5.2 ([2]). *Let $|J| < \frac{(c-k-1)r}{2}$, then for every set $I \in \mathrm{Cl}^k(J)$ the inequality $|I| \leqslant (c-k-1)^{-1}|J|$ is satisfied.*

Proof. See Appendix B. □

Definition 5.2. *Let $f : \{0,1\}^n \to \{0,1\}^n$ be the Goldreich's function based on graph G and predicate P, $b \in \{0,1\}^n$. Partial substitution ρ is called locally consistent for the equation $f(x) = b$ if there exists a string $z \in \{0,1\}^n$ that is consistent to ρ and a set $I \in \mathrm{Cl}^k(\mathrm{Vars}(\rho))$ such that the equality $f(z)|_I = b|_I$ holds.*

Lemma 5.3 (cf. [2])**.** *If the partial substitution ρ is locally consistent for $f(x) = b$, then for all $Z \subseteq X$, $|Z| \leqslant \frac{r}{2}$ there exists a string $z \in \{0,1\}^n$ such that z is consistent with ρ and the equality $f(z)|_Z = b|_Z$ holds.*

Proof. Proof by contradiction. Consider the minimal $Z \subseteq Y$ such that $|Z| \leqslant \frac{r}{2}$ and for all z that are consistent to ρ the nonequality $f(z)|_Z \neq b|_Z$ holds. Let $I \in \mathrm{Cl}^k(\mathrm{Vars}(\rho))$ be from Definition 5.2. Partial substitution ρ is locally consistent therefore $Z \setminus I \neq \varnothing$.

By the definition of closure $|\delta(Z \setminus I) \setminus (\Gamma(I) \cup \mathrm{Vars}(\rho))| > (k+1)|Z \setminus I|$, therefore there exists $y \in Z \setminus I$ such that at least $k+1$ boundary vertices of set Z (not from the support of ρ and not connected with I) are connected with y. The minimality of Z implies that there exists $z \in \{0,1\}^n$, such that $z \sim \rho$ and $f(z)|_{Z\setminus\{y\}} = b|_{Z\setminus\{y\}}$. It is possible also to satisfy the equation corresponding to vertex y by flipping the z-value of one of the boundary neighbours of vertex y. Therefore there exists $z' \in \{0,1\}^n$ that is consistent with ρ and $f(z')|_Z = b|_Z$. Contradiction. □

5.2. Clever myopic algorithm. We assume that the myopic algorithm runs on the formula $\Phi_{f(x)=b}$, where $f^{-1}(b) \neq \varnothing$. We describe the clever myopic algorithm. A clever myopic algorithm is allowed to read more clauses precisely (equivalently it may open more bits of b). Besides, the clever algorithm doesn't make substitutions that obviously lead to unsatisfiable formulas. It is not hard to see that it is enough to proof the lower bound for clever myopic algorithms; the lower bound for all myopic algorithms will follow.

Now we describe the behavior of clever myopic algorithms more formally. A clever algorithm has a current partial substitution ρ and a set $I \in \mathrm{Cl}^k(\mathrm{Vars}(\rho))$. At the beginning $\rho = \varnothing$, $I = \varnothing$. On each step the clever algorithm simplifies the formula (probably increases ρ and extends the set I to the element of $\mathrm{Cl}^k(\mathrm{Vars}(\rho))$.

If the clever algorithm requests a clause that corresponds to the vertex $y_j \in Y$ we say that the algorithm opens j-th bit of output. We assume that all clauses corresponding to $y_j \in Y$ may be read by a clever algorithm for free.

Consider the heuristic **A** that choose variable x for splitting. Let Z be the set of all open bits of output (in particular Z includes K bits that were open before x was choosen). The clever algorithm extends the set I to the element of $\mathrm{Cl}^k(\mathrm{Vars}(\rho) \cup \{x\})$. The set of open bits is increased: $Z := Z \cup I$. The clever algorithm chooses the value of variable x in order to make the part of formula that corresponds to Z satisfiable.

Lemma 5.4. *For every clever myopic algorithm A there exists another clever myopic algorithm B such that B does not use pure literal and unit clause elimination rules and the running time of algorithm B on the formula $\Phi_{f(x)=b}$ is bounded by polynomial on the running time of algorithm A.*

Proof. If the current predicate in the vertex $y \in Y$ (taking into account ρ) is linear on at least two variables then Lemma 2.4 implies that there are no pure literals in the formula that corresponds to y. So predicates in vertices that contain pure literals have at most one linear variable. All such vertices are contained in $I \in \mathrm{Cl}^k(\mathrm{Vars}(\rho))$, hence all corresponding bits of output are open and the clever algorithm can see all pure literals. The clever algorithm B delays all pure literal substitutions until all other variables have been assigned: the clever algorithm should B should remember all pure literal substitutions it wants to make, but not assign them yet. There are two possible cases: the algorithm B backtracks before it starts to perform pure literal substitutions or the algorithm B gets a satisfiable formula that can be satisfied by several applications of pure literal rules. Similarly, if formula contains a unit clause, then the corresponding vertex is in I and a clever algorithm may choose the correct substitution by itself. Note that the susbstitution of the incorrect value to the variable form a unit clause leads to the immediate backtrack. $\square$

In the following we assume that clever myopic algorithms do not use simplification rules.

Let us denote $N = \lfloor \frac{(c-k-1)r}{4dK} \rfloor$, where K is the number of clauses that myopic algorithm may read with negation signs per step of recursion. Note that K is also the upper bound to the number of open bits of b per step.

Lemma 5.5 (cf. [2]). *After N steps of any clever myopic algorithm the number of open bits is at most $\frac{r}{2}$.*

Proof. The number of open bits is at most $K\frac{(c-k-1)r}{4dK} + |\operatorname{Cl}^k(\operatorname{Vars}(\rho))|$, where ρ is the current substitution. By Lemma 5.2,

$$|\operatorname{Cl}^k(\operatorname{Vars}(\rho))| \leqslant \frac{|\operatorname{Vars}(\rho)|}{c-k-1}.$$

Since $|\operatorname{Vars}(\rho)| \leqslant \frac{(c-k-1)r}{4}$ we may conclude

$$K\frac{(c-k-1)r}{4dK} + |\operatorname{Cl}^k(Z)| \leqslant \frac{(c-k-1)r}{4d} + \frac{r}{4} \leqslant \frac{r}{2}.$$

□

Corollary 5.1. *During the first N steps a clever myopic algorithm does not backtrack (backtracking corresponds to a leaf of the splitting tree) and ρ is locally consistent.*

Proof. During N steps the number of open bits is at most $\frac{r}{2}$. We prove by induction that the current substitution is locally consistent. It is trivial for the beginning. Induction step follows from the fact that the value of the variable is chosen in such a way that $\Phi_{f(x)=b}|_I$ is satisfiable. This is possible by Lemma 5.3 and by induction hypothesis. □

Our goal is to show that after N steps of a clever myopic algorithm the current formula will be unsatisfiable with high probability.

Lemma 5.6. *Let $b \in \{0,1\}^n$ and $J \subseteq X$. Let $y \in \{0,1\}^n$ and $Z \subseteq Y$, we define set $X_y = \{x \in \{0,1\}^n \mid \forall j \in (X \setminus J)\ x_j = y_j\}$ of all strings that are agree with y on $X \setminus J$ and set $S_y = \{x \in X_y \mid f(x)|_Z = b|_Z\}$. Then either $|S_y| \geqslant 2^{|J|-|Z|-|J\cap R|}$ or $|S_y| = 0$.*

We will apply Lemma 5.6 in the case $Z \in \operatorname{Cl}^k(J)$. In this case $|S_y|$ is the lower bound on the number of locally consistent substitutions with support J.

Proof. We have to estimate the number of $x \in X_y$ that satisfies the system of equalities $f(x)|_Z = b|_Z$. If we fix the values for variables x_j for $j \in J \cap R$ then the system becomes linear over variables x_j for $j \in (J \setminus R)$. The rank of the system does not exceed $|Z|$ and the number of variables is at least $|J| - |J \cap R|$ (it is not necessary for all those variables to have explicit occurrences in the system). Thus if a solution exists then the dimension of the solution space is at least $|J| - |J \cap R| - |Z|$. Since our system is over field $\mathbb{F}_2$ the number of solutions is at least $2^{|J|-|Z|-|J\cap R|}$ even for fixed values of x_j, $j \in J \cap R$. □

Let Z be the set of open bits b in the equation $f(x) = b$, ρ be some partial substitution; we denote $C_{\rho,Z,b}$ the set of $x \in \{0,1\}^n$ that are consistent with ρ and satisfy $f(x)|_Z = b|_Z$. Formally

$$C_{\rho,Z,b} = \{x \mid f(x)|_Z = b|_Z, x \sim \rho\}.$$

Lemma 5.7. *Let $Z \subseteq Y$, $|Z| < \frac{r}{2}$, $J \subseteq X$. Then for every two locally consistent substitutions ρ_1, ρ_2 with* $\mathrm{Vars}(\rho_1) = \mathrm{Vars}(\rho_2) = J$ *and for every $b \in \{0,1\}^n$ the following is satisfied:* $\frac{|C_{\rho_1,Z,b}|}{|C_{\rho_2,Z,b}|} \leqslant 2^{|R|}$.

In order to understand this lemma assume that $|R|$ is very small. This lemma says that for two locally consistent substitutions ρ_1 and ρ_2 the number of solutions of the system $f(x)|_Z = b|_Z$ that extends ρ_1 is approximately equal to the number of solutions of the system $f(x)|_Z = b|_Z$ that extends ρ_2.

Proof.

$$\frac{|C_{\rho_1,Z,b}|}{|C_{\rho_2,Z,b}|} = \frac{\sum\limits_{\sigma} |C_{\rho_1 \cup \sigma,Z,b}|}{\sum\limits_{\sigma} |C_{\rho_2 \cup \sigma,Z,b}|},$$

where the sum in both cases is over partial substitutions σ with support $\mathrm{Vars}(\sigma) = R \setminus J$.

We show that the size of the set $C_{\rho_i \cup \sigma,Z,b}$ is either 0 or some fixed value and not dependant on σ and $i \in \{1,2\}$.

The size of the set $C_{\rho_i \cup \sigma,Z,b}$ equals the number of solutions of the system of equations $f(x)|_Z = b|_Z$ if some bits of x are fixed by substitution $\rho_i \cup \sigma$. This fixation makes the system linear. Note that the rank of this system does not depend on substitutions ρ_i and σ (since ρ_i and σ influence only the column of constants in the system). Therefore, if such system has a solution then the number of solutions does not depend on i and σ.

Since the substitution ρ_i is locally consistent and $|Z| < \frac{r}{2}$, Lemma 5.3 implies that there exists such substitution σ_i with support $\mathrm{Vars}(\sigma_i) = R \setminus J$ that $C_{\rho_i \cup \sigma_i,Z,b} \neq \varnothing$.

$$\frac{|C_{\rho_1,Z,b}|}{|C_{\rho_2,Z,b}|} \leqslant \frac{2^{|R|}|C_{\rho_1 \cup \sigma_1,Z,b}|}{|C_{\rho_2 \cup \sigma_2,Z,b}|} = 2^{|R|}.$$

□

Theorem 5.1. *Let ρ be the current substitution after N steps of a clever myopic algorithm running on the fomula $\Phi_{f(x)=b}$ for some $b \in f(\{0,1\}^n)$*

and Z is the set of open bits. Then $\Pr_{y\leftarrow U(\{0,1\}^n)}[\exists x : x \sim \rho,\ f(x) = f(y) \mid f(y)|_Z = b|_Z] \leqslant 2^{-\Omega(\frac{n}{K})}$.

Before giving a formal prove we informally describe the main idea. For simplicity we assume that $R = \varnothing$ therefore the predicate P is linear . We consider a clever myopic algorithm after N steps (i.e. $|\rho| = N$). In this moment the size of $I \in \mathrm{Cl}^k(\mathrm{Vars}(\rho))$ does not exceed $(1-\varepsilon)N$ for some positive ε by Lemma 5.2. We apply Lemma 5.6 for $J = \mathrm{Vars}(\rho)$ and $Z = I$, Lemma 5.6 states that the number of locally consistent substitutions is at least $2^{|\mathrm{Vars}(\rho)|-|I|} = 2^{\Omega(N)}$.

Lemma 5.3 and Lemma 5.5 imply that every local consistent partial substitution may be extended to the full substitution that is consistent with open bits of the right hand side. Lemma 5.7 states that the number of such extensions is the same for every locally consistent substitution if $R = \varnothing$. A myopic algorithm has no chance to find one substitution among all locally consistent substitutions since they all have equal chances to be correct. Since our linear system has at most two solutions (if $R = \varnothing$), there are at most two locally consistent substitutions that can be extended to the solution of the system. Therefore the probability of correct substitution is at most $2^{-\Omega(N)} = 2^{-\Omega(\frac{n}{K})}$.

Proof of Theorem 5.1. Corollary 5.1 implies that during N steps the algorithm does not backtrack and $|\rho| = N$.

We apply Lemma 5.6 for $J = \mathrm{Vars}(\rho)$ and $Z = I$, where $I \in \mathrm{Cl}^k(\mathrm{Vars}(\rho))$ is from definition of a clever myopic algorithm after step N. Since $b \in f(\{0,1\}^n)$ there exists $y \in \{0,1\}^n$ such that $S_y \neq \varnothing$ (S_y is defined in the Lemma 5.6) and the inequality $|S_y| \geqslant 2^{|\mathrm{Vars}(\rho)|-|I|-|R|}$ holds. Therefore at least $2^{|\mathrm{Vars}(\rho)|-|I|-|R|}$ substitutions with support $\mathrm{Vars}(\rho)$ are locally consistent.

$$\begin{aligned}
&\Pr_{y\leftarrow U(\{0,1\}^n)}[\exists x : x \sim \rho,\ f(x) = f(y) \mid f(y)|_Z = b|_Z]\\
&\qquad = \Pr_{y\leftarrow U(\{0,1\}^n)}[f^{-1}(f(y)) \cap C_{\rho,Z,b} \neq \varnothing \mid f(y)|_Z = b|_Z]\\
&\qquad \leqslant \max_y |f^{-1}(f(y))| \cdot \Pr_{y\leftarrow U(\{0,1\}^n)}[y \in C_{\rho,Z,b} \mid f(y)|_Z = b|_Z]
\end{aligned}$$

By Lemma 3.2, the first term may be estimated as

$$\max_y |f^{-1}(f(y))| \leqslant 2^{|R|+1}.$$

Let us estimate the second term:

$$\Pr_{y \leftarrow U(\{0,1\}^n)}[y \in C_{\rho,Z,b} \mid f(y)|_Z = b|_Z] \leqslant \frac{\max_\sigma |C_{\sigma,Z,b}|}{\sum_\sigma |C_{\sigma,Z,b}|},$$

where σ goes through all locally correct substitutions with the support $\operatorname{Vars}(\rho)$. By Lemma 5.7,

$$\frac{\max_\sigma |C_{\sigma,Z,b}|}{\sum_\sigma |C_{\sigma,Z,b}|} \leqslant 2^{|R|} \frac{\min_\sigma |C_{\sigma,Z,b}|}{2^{|\operatorname{Vars}(\rho)|-|I|-|R|} \min_\sigma |C_{\sigma,Z,b}|} = 2^{2|R|+|I|-|\operatorname{Vars}(\rho)|}.$$

Altogether:

$$\Pr_{y \leftarrow U(\{0,1\}^n)}[\exists x : x \sim \rho,\ f(x) = f(y) \mid f(y)|_Z = b|_Z] \leqslant 2^{3|R|+|I|-|\operatorname{Vars}(\rho)|+1}.$$

Since $I \in \mathrm{Cl}^k(\operatorname{Vars}(\rho))$ the Lemma 5.2 implies $|I| \leqslant (c-k-1)^{-1}|\operatorname{Vars}(\rho)|$. The statement of the theorem follows from $\operatorname{Vars}(\rho) = \Omega(\frac{n}{K})$, $R = o(n/K)$ and $c > k+2$. □

Theorem 5.2. *For every myopic algorithm A the following inequality holds:*

$$\Pr_{y,s}[t_A(\Phi_{f(x)=f(y)}) \geqslant 2^{\Omega(n)}] \geqslant 1 - 2^{-\Omega(\frac{n}{K})},$$

where $t_A(x)$ denotes the running time of A on input x and s is a string of random bits used by A.

Proof. Lemma 5.4 implies that it is enough to prove the Theorem for clever myopic algorithms that do not use simplification rules.

We fix the string of random bits s and prove that for algorithms that use s instead of random bits the following holds:

$$\Pr_{y}[t_A(\Phi_{f(x)=f(y)}) \geqslant 2^{\Omega(n)}] \geqslant 1 - 2^{-\Omega(\frac{n}{K})},$$

and the theorem follows.

We consider a clever myopic algorithm after N steps on the formula $\Phi_{f(x)=f(y)}$. Let Z_y be the set of open bits of output by this moment. Note that for a fixed string s the behavior of algorithm during the first N steps is the same for all $y' \in \{0,1\}^n$ such that $f(y')|_{Z_y} = f(y)|_{Z_y}$ (in this case $Z_{y'} = Z_y$). Thus the set of all $y \in \{0,1\}^n$ may be split on the finite number of classes of equivalence $S_1, S_2, \ldots, S_m$ such that for all y and for all $y' \in S_y$ the values of $Z_{y'}$ are the same and the values of $f(y)|_{Z_y}$ are the same, and this is not true for different classes

$$\Pr_{y}[t_A(\Phi_{f(x)=f(y)}) \geqslant 2^{\Omega(n)}]$$

$$= \sum_{i=1}^{m} \Pr_y \left[t_A(\Phi_{f(x)=f(y)}) \geqslant 2^{\Omega(n)} \mid y \in S_i\right] \Pr_y[y \in S_i].$$

By Theorem 5.1, after N steps of a clever myopic algorithm

$$\Pr_y[\Phi_{f(x)=f(y)}|_\rho \text{ is unsatisfiable } \mid y \in S_i] \geqslant 1 - 2^{-\Omega(\frac{n}{K})},$$

where ρ is the current substitution that is locally consistent. Finally Theorem 4.1 implies that

$$\Pr_y[t_A(\Phi_{f(x)=f(y)}) \geqslant 2^{\Omega(n)} \mid y \in S_i] \geqslant 1 - 2^{-\Omega(\frac{n}{K})}.$$

The theorem follows from the last inequality. □

It may be easily verified that we do not use the fact that the predicate Q is the same for all vertices of the set Y.

Appendix §A. Behavior of DPLL algorithms on unsatisfiable formulas

Behavior of DPLL algorithms on unsatisfiable formulas is closely connected with the resolution proof system. The resolution proof system is used for proving the unsatisfiability of CNF formulas. The proof of unsatisfiability of formula φ in the resolution proof system is a sequence of clauses, every clause in this sequence is either a clause of φ or a result of application of the resolution rule to two previous clauses; and the last clause in the sequence is an empty clause (a contradiction). The resolution of two clauses $(l_1 \vee l_2 \vee \dots \vee l_n)$ and $(l'_1 \vee l'_2 \vee \dots \vee l'_m)$ where $l'_m = \neg l_n$ is the clause $(l_1 \vee \dots \vee l_{n-1} \vee l'_1 \vee \dots \vee l'_{m-1})$. The proof is called treelike if every inferred clause is used as the premise of the resolution rule at most once.

The running of every DPLL algorithm (that does not use symplification rules) on the unsatisfiable formula corresponds to the splitting tree. Vertices of the tree are marked with variables that are chosen for splitting. There are two outgoing edges from every vertex except leaves; one of the edges is marked with 0, the other edge is marked with 1. In every leaf at least one of clauses of initial formula is refuted. The running time of a DPLL algorithm is the size of the splitting tree (note that if formula is unsatisfiable then the algorithm should investigate the whole tree and it's number of steps is the same for all heuritics **B**).

The following statement is well known.

Proposition A.1. *The running time of DPLL algorithm (that does not use symplification rules) on unsatisfiable formula is at least the size (number of clauses) of the shortest treelike resolution proof.*

Proof. By induction of the size of the tree it is easy to show that if unsatisfiable formula φ has splitting tree of size k then φ has resolution refutation of size k. The base of induction is the spitting tree with only one vertex, such formula should contain an empty clause therefore the size of resolution refutation is 1. Induction step. Note that the tree necessarily contains two leaves u and v with the same parent w. Let x_i be the splitting variable in the vertex w, the leaf u corresponds to the assignment $x_i = 1$ and the leaf v corresponds to the assignment $x_i = 0$. Two clauses that are refuted in the vertices v and u contain the variable x_i with different signs. The resolvent (the result of an application of the resolution rule) of this two clauses C must be refuted in the vertex w. We construct new splitting tree: cut leaves u and w and add clause C to vertex w. Now we get a correct splitting tree for a formula that is obtained from the initial formula by adding a resolvent of two clauses. And we apply induction hypothesis to the resulting tree (the number of vertices is decreased by 1). □

Ben-Sasson and Wigderson in [5] introduced the notion of width of the proof. The width of a clause is the number of literals in it. The width of a CNF formula is the width of its widest clause. The width of a resolution proof is the width of its widest clause.

Theorem A.1 ([5], corollary 3.4)**.** *The size of a treelike resolution refutation of the formula φ is at least 2^{w-w_φ}, where w is the minimum width of the resolution refutation of φ and w_φ is the width of φ.*

Let G be a boundary (r, d, c)-expander. We associate a proposition variable with every vertex in set X. Let every vertex y_j in set Y have a CNF formula that depends on variables adjacent to y_j. We denote the formula in the vertex y_j as φ_j. Obviously the width of φ_j is at most d. The conjunction of all formulas that correspond to the vertices Y we denote Φ. For any subset $A \subseteq Y$ the conjunction of all formulas that correspond to the vertices in A we denote as Φ^A.

Theorem A.2. *Let every formula φ_j contain at most k insensible variables; ρ is a partial assignment to variables of X such that formula $\Phi|_\rho$ is unsatisfiable and for any set of vertices $A \subseteq Y$, $|A| < \frac{r}{2}$, the formula*

$\Phi^A|_\rho$ is satisfiable. Then any resolution refutation of $\Phi|_\rho$ has width at least $\frac{(c-k)r}{4} - |\rho|$.

Proof. We consider Ben-Sason–Wigderson measure μ that is defined on the clauses of resolution proof of $\Phi|_\rho$. $\mu(D)$ is the size of the minimal set of vertices A such that clause D is a semantic implication of $\Phi^A|_\rho$ (it means that every satisfying assignment of $\Phi^A|_\rho$ also satisfies D). The measure μ is semiadditive: if clause D is a resolvent of clauses C_1 and C_2, then $\mu(D) \leqslant \mu(C_1) + \mu(C_2)$. Since for every set $A \subseteq Y$ such that $|A| < \frac{r}{2}$, formula $\Phi^A|_\rho$ is satisfiable, then the measure of an empty clause is at least $\frac{r}{2}$. Semiadditivity implies that there exists a clause C such that $\frac{r}{2} > \mu(C) \geqslant \frac{r}{4}$ for r large enough. Let A be the minimal set of vertices such that $\Phi^A|_\rho$ semantically implies C, i.e. $|A| = \mu(C) \geqslant \frac{r}{4}$. Since G is a (r, d, c)-boundary expander we have $\delta(A) \geqslant c|A|$. $\delta(A)$ is a set of variables that have exactly one occurrence in the formulas corresponding to the set A. There are at least $(c-k)|A|$ variables among them that are sensible for at least one vertex of A. There are at least $(c-k)|A| - |\rho|$ sensible variables in the formula $\Phi^A|_\rho$. Now we will show that the clause C contains all sensible variables. Suppose for contradiction that there is a variable x_j that is sensible for a vertex $v \in A$ and the clause C does not contain x_j. Consider the set $A \setminus \{v\}$. It doesn't semantically imply C, therefore there exists such an assignment that satisfies all formulas for $A \setminus \{v\}$ and doesn't satisfy C. We may change the value of x_j in this assignment in such way that the resulting assignment satisfies all formulas in A and doesn't satisfy C. The later contradicts the fact that C is a semantic implication of A. □

Corollary A.1. *The size of the splitting tree of $\Phi|_\rho$ is at least $2^{\frac{(c-k)r}{4} - |\rho| - d}$.*

Proof. Follows from the Theorem A.2, Theorem A.1 and Proposition A.1. □

Appendix §B. Closure

Lemma B.1.

1. *For every set $J \subseteq X$ there exists a k-closure.*
2. *Let $J_1 \subseteq J_2$, then for every $I_1 \in Cl^k(J_1)$ there exists $I_2 \in Cl^k(J_2)$ such that $I_1 \subseteq I_2$.*

Proof. 1. A k-closure may be obtained as a result of the following algorithm $\mathcal{C}$ on the input $(J, \varnothing)$.

Algorithm B.1. Algorithm $\mathcal{C}(J, I_0)$

```
(1) $I := I_0$ (the variable $I$ means some subset of $Y$)
(2) While there exists $I' \subseteq Y \setminus I$ such that
        $0 < |I'| \leqslant \frac{r}{2}, \quad |\delta(I') \setminus (\Gamma(I) \cup J)| \leqslant (1+k)|I'|$
      • $I := I \cup I'$
(3) Return $I$.
```

2. Let $I_1 \in Cl^k(J_1)$, then we can get $I_2 \in Cl^k(J_2)$ as a result of the algorithm $\mathcal{C}$ on the input (J_2, I_1). The condition $I_1 \subseteq I_2$ is satisfied. □

Lemma B.2 ([2]). *Let $|J| < \frac{(c-k-1)r}{2}$, then for every set $I \in Cl^k(J)$ the inequality $|I| \leqslant (c-k-1)^{-1}|J|$ is satisfied.*

Proof. Proof by contradiction. Let $I_1, I_2, \ldots, I_m$ be the sequence corresponding to the k-closure I, $C_\ell = \bigcup_{1 \leqslant i \leqslant \ell} I_\ell$. Let t be the minimal number such that the inequality $|C_t| > (c-k-1)^{-1}|J|$ is satisfied, then $|C_t| \leqslant (c-k-1)^{-1}|J| + \frac{r}{2} \leqslant r$. Then

$$|\delta(C_t)| \geqslant c|C_t| > |J| + (k+1)|C_t|.$$

By induction on ℓ we will show that $|\delta(C_\ell) \setminus J| \leqslant (k+1)(|C_\ell|)$, and it contradicts the above inequality for $l = t$. If $l = 1$ the inequality follows from $|\delta I_1 \setminus J| \leqslant (k+1)|I_1|$.

$$\begin{aligned} |\delta(C_\ell) \setminus J| &\leqslant |\delta(I_1 \cup \cdots \cup I_{\ell-1}) \setminus J| + |\delta(I_\ell) \setminus (J \cup \Gamma(C_{\ell-1}))| \\ &\leqslant (k+1)((|C_{\ell-1}|)) + (k+1)|I_\ell|. \end{aligned}$$

□

References

1. J. Cook, O. Etesami, R. Miller, L. Trevisan, *Goldreich's One-Way Function Candidate and Myopic Backtracking Algorithms.* In: Proceedings of TCC, Springer-Verlag (2009), pp. 521–538,
2. M. Alekhnovich, E. A. Hirsch, A. Edward, D. Itsykson, *Exponential lower bounds for the running time of DPLL algorithms on satisfiable formulas.* — J. Autom. Reason. **35**, No. 1–3, (2005), 51–72.
3. L. Levin, *One-way functions and pseudorandom generators.* — Combinatorica **7**, No. 4 (1987), 357–363.
4. O. Goldreich, *Candidate One-Way Functions Based on Expander Graphs.* — Electronic Colloquium on Computational Complexity, No. 00-090 (2000).
5. E. Ben-Sasson, A. Wigderson, *Short proofs are narrow – resolution made simple.* — J. ACM **48**, No. 2 (2001), 149–169.

6. S. Hoory, N. Linial, A. Wigderson, *Expander graphs and their applications.* — Bull. Amer. Math. Soc. **43** (2006), 439–561,
7. N. Nisan, A. Wigderson, *Hardness vs. randomness.* — J. Computer System Sciences **49** (1994), 149–167.
8. I. Mironov, L. Zhang, *Applications of SAT Solvers to Cryptanalysis of Hash Functions.* — In: A. Biere, C. P. Gomes (eds.) Theory and Applications of Satisfiability Testing–SAT **4121**, Lect. Notes Comput. Sci., Springer (2006), pp. 102–115.
9. G. S. Tseitin, *On the complexity of derivation in the propositional calculus.* — Zap. Nauchn. Semin. LOMI **8** (1968), 234–259.
10. N. Eén, A. Biere, *Effective Preprocessing in SAT Through Variable and Clause Elimination.* — Theory and Applications of Satisfiability Testing (2005), 61–75.
11. N. Een, N. Sorensson, *An Extensible SAT-solver.* — Lect. Notes Computer Science, Springer (2003), 502–518.
12. M. Alekhnovich, E. Ben-Sasson, A. A. Razborov, A. Wigderson, *Pseudorandom generators in propositional proof complexity.* — In: Proceedings of the 41st Annual Symposium on Foundations of Computer Science, IEEE Computer Society, Washington, DC, USA (2000), p. 43.
13. A. Urquhart, *Hard Examples for Resolution.* — JACM **34**, No. 1 (1987), 209–219.
14. M. Davis, G. Logemann, D. Loveland, *A machine program for theorem-proving.* — Communications ACM **5** (1962), 394–397.
15. M. Davis, H. Putnam, *A computing procedure for quantification theory.* — JACM **7** (1960), 201–215.
16. M. Capalbo, O. Reingold, S. Vadhan, A. Wigderson, *Randomness conductors and constant-degree lossless expanders.* — In: Proceedings of the 34th Annual ACM Symposium on Theory of Computing (2002), pp. 659–668.
17. D. Itsykson, *Lower bound on average-case complexity of inversion of Goldreich function by drunken backtracking algorithms.* — In: Proceedings of International Computer Science Symposium in Russia Lecture Notes in Computer Science, Springer (2010) pp. 204–215.
18. R. Miller, *Goldreich's one-way function candidate and drunken backtracking algorithms.* — University of Virginia (2009).

St.Petersburg Department
of the Steklov Mathematical Institute,
Fontanka 27,
St.Petersburg 191023,
Russia
E-mail: dmitrits@pdmi.ras.ru
sokolov.dmt@gmail.com

Поступило 31 июля 2011 г.