On OBDD-Based Algorithms and Proof Systems That Dynamically Change Order of Variables*

Dmitry Itsykson[1], Alexander Knop[2], Andrey Romashchenko[†3], and Dmitry Sokolov[4]

1 **St. Petersburg Department of V. A. Steklov Institute of Mathematics of the Russian Academy of Sciences, St. Petersburg, Russia**
dmitrits@pdmi.ras.ru

2 **St. Petersburg Department of V. A. Steklov Institute of Mathematics of the Russian Academy of Sciences, St. Petersburg, Russia**
aaknop@gmail.com

3 **LIRMM, University Montpellier and CNRS, Montpellier, France**
andrei.romashchenko@lirmm.fr

4 **St. Petersburg Department of V. A. Steklov Institute of Mathematics of the Russian Academy of Sciences, St. Petersburg, Russia**
sokolov.dmt@gmail.com

Abstract

In 2004 Atserias, Kolaitis and Vardi proposed OBDD-based propositional proof systems that prove unsatisfiability of a CNF formula by deduction of identically false OBDD from OBDDs representing clauses of the initial formula. All OBDDs in such proofs have the same order of variables. We initiate the study of OBDD based proof systems that additionally contain a rule that allows to change the order in OBDDs. At first we consider a proof system OBDD($\wedge$, reordering) that uses the conjunction (join) rule and the rule that allows to change the order. We exponentially separate this proof system from OBDD($\wedge$)-proof system that uses only the conjunction rule. We prove two exponential lower bounds on the size of OBDD($\wedge$, reordering)-refutations of Tseitin formulas and the pigeonhole principle. The first lower bound was previously unknown even for OBDD($\wedge$)-proofs and the second one extends the result of Tveretina et al. from OBDD($\wedge$) to OBDD($\wedge$, reordering).

In 2004 Pan and Vardi proposed an approach to the propositional satisfiability problem based on OBDDs and symbolic quantifier elimination (we denote algorithms based on this approach as OBDD($\wedge, \exists$)-algorithms). We notice that there exists an OBDD($\wedge, \exists$)-algorithm that solves satisfiable and unsatisfiable Tseitin formulas in polynomial time. In contrast, we show that there exist formulas representing systems of linear equations over $\mathbb{F}_2$ that are hard for OBDD($\wedge, \exists$, reordering)-algorithms. Our hard instances are satisfiable formulas representing systems of linear equations over $\mathbb{F}_2$ that correspond to some checksum matrices of error correcting codes.

1998 ACM Subject Classification F.2.2 [Nonnumerical Algorithms and Problems] Complexity of proof procedures, F.4.1 [Mathematical Logic] Proof Theory

Keywords and phrases Proof complexity, OBDD, error-correcting codes, Tseitin formulas, expanders

Digital Object Identifier 10.4230/LIPIcs.STACS.2017.43

* The research presented in Sections 3, 4 and 5 was supported by Russian Science Foundation (project 16-11-10123). The work of the third author on the research presented in Section 6 was partially supported by RFBR grant 16-01-00362.

† On leave from IITP RAS.

1 Introduction

An ordered binary decision diagram (OBDD) is a way to represent Boolean functions [3]. A Boolean function is represented as a branching program with two sinks such that on every path from the source to a sink variables appear in the same order. This restriction on the order of variables allows to handle the diagrams (compute binary Boolean operations on diagrams, compute projections, or test satisfiability) very efficiently.

Atserias, Kolaitis and Vardi [1] proposed an OBDD-based proof system. This system is meant to prove that a given CNF formula is unsatisfiable. For some order of variables π we represent clauses of the input formula as a π-ordered BDD; we may derive a new OBDD applying either the *conjunction* rule or the *weakening* rule. (The authors of [1] supplied the system with one more rule – the *projection*, which derives $\exists x D$ from D; we consider this rule as a special case of the weakening rule, so we do not need to allow it explicitly). A *proof* in this system is a derivation of an OBDD that represents the constant false function. We refer to this proof system as the OBDD($\wedge$, weakening)-proof system. The OBDD($\wedge$, weakening)-proof system simulates Cutting Plane with unary coefficients and thus it is stronger than Resolution. This proof system provides also short refutations for the formulas that represent unsatisfiable systems of linear equations over $\mathbb{F}_2$ [1], while linear systems are hard for Resolution. This observation motivates the study of the OBDD-based algorithms (notice that the popular DPLL and CDCL algorithms correspond to tree-like and DAG-like Resolutions).

Several exponential lower bounds are known for different versions of OBDD-proof systems. Segerlind [16] proved an exponential lower bound for the tree-like version of OBDD($\wedge$, weakening)-proof system using the communication complexity technique proposed in [2]. Krajíček [10] proved an exponential lower bound for the DAG-like version of it using monotone feasible interpolation. Several papers study the OBDD-based proof system that has only one inference rule – the conjunction rule (we refer to this system as OBDD($\wedge$)-proof system). Groote and Zantema [8] showed that Resolution does not simulate OBDD($\wedge$). Tveretina, Sinz and Zantema [18] proved the lower bound $2^{\Omega(n)}$ for the pigeonhole principle $\mathtt{PHP}_n^{n+1}$ in the OBDD($\wedge$)-proof system and proved that OBDD($\wedge$) does not simulate Resolution. Friedman and Xu [6] proved an exponential lower bound for the complexity of random unsatisfiable 3-CNF formulas in restricted versions of OBDD($\wedge$)-proof systems (with a fixed order of the variables) and an exponential lower bound for the running time on random unsatisfiable 3XOR formulas of restricted versions of OBDD($\wedge$)-proof systems (with fixed orders of application of rules).

An interesting approach to solving propositional satisfiability was suggested by Pan and Vardi [15]. They proposed an algorithm that chooses some order π on the variables of the input CNF formula F and creates the current π-ordered BDD D that initially represents the constant true function, and a set of clauses S that initially consists of all clauses of the formula F. Then the algorithm applies the following operations in an arbitrary order:

1. conjunction (or join): choose a clause $C \in S$, delete it from S and replace D by the π-ordered BDD representing $C \wedge D$;
2. projection (or $\exists$-elimination): choose a variable x that has no occurrence in the clauses from S and replace D by the π-ordered BDD representing $\exists x D$.

When S becomes empty, the algorithm stops and reports "unsatisfiable" if D represents the constant false function and "satisfiable" otherwise. Every particular instance of this algorithm uses its own strategies to choose an order of variables π and an order of application of the operations. We refer to these algorithms as OBDD($\wedge, \exists$)-algorithms.

The lower bounds for the OBDD($\wedge$, weakening)-proof systems mentioned above imply the same lower bounds for the OBDD($\wedge, \exists$)-algorithms.

Pan and Vardi [15] investigated some specific strategies and compared them with DPLL based SAT solvers and compared the strategies with SAT solvers based on OBDDs without quantifier eliminations (we call them OBDD($\wedge$)-algorithms). Experiments showed in particular that OBDD($\wedge, \exists$)-algorithms are faster than OBDD($\wedge$)-algorithms and DPLL based algorithms on many natural formulas.

One of these formulas was $\mathtt{PHP}_n^{n+1}$. The result of [4] implies that an OBDD($\wedge, \exists$)-algorithm can solve $\mathtt{PHP}_n^{n+1}$ in polynomial time in compare to OBDD($\wedge$)-algorithms and DPLL based algorithms.

1.1 Statement of the problem

It is known that changing the order of the variables in an OBDD can be performed in time polynomial in the sizes of the input and the output. So it seems to be very restrictive to use the same order of variables in all OBDDs in the proof. Hence we propose to supply the proof system with a supplementary rule that dynamically reorders the variables in OBDDs.

In OBDD-proofs, the reordering rule may be applied to an arbitrary OBDD from the proof but the conjunction rule may be applied only to OBDDs with the same order since the verification of the application of the conjunction to OBDDs in different orders is hard (this problem is indeed coNP-complete, see [14, Lemma 8.14]).

The first aim of this paper is to prove lower bounds for the OBDD-based algorithms and proof systems that use the reordering rule. The second aim is to show that reordering of the variables is really useful; we give examples of formulas that are hard for the OBDD-based proof systems without reordering and easy with reordering.

1.2 Our results

In Section 3 we consider the OBDD($\wedge$, reordering)-proof system. We prove two exponential lower bounds for the size of a OBDD($\wedge$, reordering)-derivation of the pigeonhole principle $\mathtt{PHP}_n^{n+1}$ and Tseitin formulas based on constant-degree algebraic expanders. The lower bound for pigeonhole principle extends the result of Tveretina et al. [18] from OBDD($\wedge$)-proofs to OBDD($\wedge$, reordering)-proofs. (Besides, we believe that our argument is simpler than the proof in [18]). The result for Tseitin formulas, to the best of out knowledge, was not known even for the more restrictive OBDD($\wedge$)-proofs. In both arguments we use the same strategy.

- At first step, we prove an exponential lower bound on the size of the OBDD-representation for an appropriate satisfiable formula. Assume for simplicity that the original unsatisfiable formula is minimally unsatisfiable. Roughly speaking, the satisfiable formula under consideration is equal to the original unsatisfiable formula with one canceled clause. For example, for the pigeonhole principle the appropriate satisfiable formula would be $\mathtt{PHP}_n^n$; for the Tseitin formulas such an appropriate formula is a satisfiable Tseitin formula. This part of the proof is quite cumbersome but it involves only rather elementary techniques of lower bounds for OBDD.
- Consider the last derivation step. It consists in the conjunction for F_1 and F_2 in the same order π. Our goal is to prove that at least one of F_1 and F_2 has an exponential size. Both F_1 and F_2 are satisfiable and they are conjunctions of different sets of clauses from the initial formulas. The idea is to construct partial substitutions ρ_1 and ρ_2 with the same support such that the formula $F_1|_{\rho_1} \wedge F_2|_{\rho_2}$ is isomorphic to the satisfiable formula from the first step. Then any OBDD representation of $F_1|_{\rho_1} \wedge F_2|_{\rho_2}$ has exponential size.

Hence, the size of either $F_1|_{\rho_1}$ or $F_2|_{\rho_2}$ is large for the order π. Thus, the size of either F_1 or F_2 is large for the order π.

Though our method has limitations, it may be applied to not minimally unsatisfiable formulas as well. Roughly speaking, our proof requires that all minimally unsatisfiable subformulas of the formula have simple structures. For example, our proof for $\texttt{PHP}_n^{n+1}$ transfers to $\texttt{onto-FPHP}_n^{n+1}$ almost literally since the unique minimal unsatisfiable subformula of $\texttt{onto-FPHP}_n^{n+1}$ is exactly $\texttt{PHP}_n^{n+1}$. We do not believe that the same technique applies to a random 3-XOR.

In Section 4 we construct an example of a family of formulas that are easy for the OBDD($\wedge$, reordering)-proof system but hard for the OBDD($\wedge$)-proof system.

In Section 5 we study OBDD($\wedge, \exists$, reordering)-algorithms. At first, we notice that there exists an OBDD($\wedge, \exists$)-algorithm that solves satisfiable and unsatisfiable Tseitin formulas in polynomial time. In contrast, we show that there exist formulas representing systems of linear equations over $\mathbb{F}_2$ that are hard for OBDD($\wedge, \exists$, reordering)-algorithms. More specifically, we describe a randomized construction of a family of satisfiable formulas F_n on n variables in $O(1)$-CNF such that every OBDD($\wedge, \exists$, reordering)-algorithm runs at least $2^{\Omega(n)}$ steps on F_n.

The plan of the proof is as follows.

- We prove that if $\mathcal{C} \subseteq \{0,1\}^n$ is the set of codewords of a list-decodable code that allows to correct $\frac{2}{3}n$ of erasures, then every OBDD that represents the characteristic function of $\mathcal{C}$ ($\chi_{\mathcal{C}}$) has a big size (this size proves to be close to the number of all codewords in the code). Moreover, this property holds even for the projections of $\chi_{\mathcal{C}}$ onto any $\frac{1}{6}n$ coordinates. Notice that a similar result was known for error-correcting codes with large enough minimal distance (see for example the paper of Duris et al. [5]). Guruswami [9] showed that the codes with large enough minimal distance are erasure list-decodable but the opposite statement does not hold.
- We give a randomized construction of the required linear code. This construction is based on random checksum matrix. We use the codes of Gallager [7] that contain $O(1)$ ones per row. We prove that such a random code with a high probability enjoys the following expansion property: every $\frac{1}{6}n$ columns of the matrix contain ones in almost all rows.
- We consider the execution of an OBDD($\wedge, \exists$, reordering)-algorithm on a CNF formula that corresponds to the CNF representation of the checksum matrix of the code. We study two cases:
  1. the algorithm applies the projection rule less than $\frac{n}{6}$ times;
  2. the projection rule is applied at least $\frac{n}{6}$ times.
  In the first case, we focus on the OBDD at the end of the execution of the algorithm; its size should be exponential due to the properties of the code. In the second case, we consider the first moment in the computational process when we apply exactly $\frac{n}{6}$ projection operations. By the expansion property, OBDD D is a projection of almost the entire formula, thus its size should be close to the size of the OBDD of the characteristic function of the code. That is, the size of D should be large enough.

As we mentioned above, the previously known lower bounds for tree-like and DAG-like OBDD($\wedge$, weakening)-proofs imply lower bounds for OBDD($\wedge, \exists$)-algorithms. So, what is new in our results comparative to the lower bounds proven by Segerlind [17] and Krajíček [10]? First of all, our lower bound also works for the reordering operation. The second advantage of our construction is that we come up with quite a natural class of formulas (our formulas represent linear systems of equations that define some error correcting codes), while the constructions in [17] and [10] seem to be rather artificial. Further, we prove the lower

bound $2^{\Omega(n)}$ for a formula with n variables, whereas the previously known lower bounds are of the type 2^{n^ϵ} (for some $\epsilon < 1/5$). Besides, we proposed a new technique that might be applicable for other classes of formulas.

1.3 Open problems

The first open problem is to prove a superpolynomial lower bound for the OBDD($\wedge$, weakening, reordering)-proofs. The second open problem is to separate the OBDD($\wedge$, weakening, reordering) and the OBDD($\wedge$, weakening) proof systems.

2 Preliminaries

2.1 Ordered binary decision diagrams

An ordered binary decision diagram (OBDD) is a data structure that is used to represent Boolean function [3]. Let $\Gamma = \{x_1, \ldots, x_n\}$ be a set of propositional variables. A binary decision diagram is a directed acyclic graph with one source. Every vertex of the graph is labeled by a variable from Γ or by constants 0 or 1. If a vertex is labeled by a constant, then it is a sink (has out-degree 0). If a vertex is labeled by a variable, then it has exactly two outgoing edges: one edge is labeled by 0 and the other edge is labeled by 1. Every binary decision diagram defines a Boolean function $\{0,1\}^n \to \{0,1\}$. The value of the function for given values of $x_1, \ldots, x_n$ is computed as follows: we start a path at the source and on every step we go along the edge that corresponds to the value of the variable at the current vertex. Every such path reaches the vertex labeled by the constant; this constant is the value of the function.

Let π be a permutation of the set $\{1, \ldots, n\}$. A π-ordered binary decision diagram (BDD) is a binary decision diagram such that on every path from the source to a sink every variable has at most one occurrence and variable $x_{\pi(i)}$ can not appear before $x_{\pi(j)}$ if $i > j$. An ordered binary decision diagram (OBDD) is a π-ordered binary decision diagram for some permutation π. The size of an OBDD is the number of vertices in it.

OBDDs have the following nice property: for every order of variables every Boolean function has a unique minimal OBDD. For a given order π, the minimal OBDD of a function f may be constructed in polynomial time from any π-ordered BDD for the same f. There are also known polynomial-time algorithms that efficiently perform the operations of conjunction, negation, disjunction and projection (operation that maps diagram D computing Boolean function $f(x, y_1, \ldots, y_n)$ to diagram D' computing Boolean function $\exists x\ f(x, y_1, \ldots, y_n)$) to π-ordered ODDs [13]. There is an algorithm running in time polynomial in the size of the input and the output that gets as input a π-ordered diagram A, a permutation ρ and returns the minimal ρ-ordered diagram that represents the same function as A [13].

2.2 OBDD-proofs

If F is a CNF formula, we say that the sequence $D_1, D_2, \ldots, D_t$ is an OBDD-derivation of F if D_t is an OBDD that represents the constant false function, and for all $1 \le i \le t$, D_i is an OBDD that represents a clause of F or can be obtained by one of the following inference rules:

1. (conjunction or join) D_i is a π-ordered OBDD, that represents $D_k \wedge D_l$ for $1 \le l, k < i$, where D_k, D_l have the same order π;
2. (weakening) there exists a $j < i$ such that D_j and D_i have the same order π, and D_j semantically implies D_i, that is every assignment that satisfies D_j also satisfies D_i;
3. (reordering) D_i is an OBDD that is equivalent to an OBDD D_j with $j < i$.

We consider several different OBDD-proof systems with different sets of allowed rules. When we need to denote some specific proof system, we indicate the rules specific for this system in brackets. For example, the OBDD($\wedge$)-proof system uses only the conjunction rule and hence we may assume that all OBDDs in a proof have the same order. We use the notation π-OBDD($\wedge$)-proof if all diagrams in this proof have order π.

2.3 OBDD-algorithms for SAT

Pan and Vardi [15] proposed for the Boolean satisfiability problem the following family of algorithms based on OBDDs and symbolic quantifier elimination.

The algorithm gets as an input a CNF formula F, it chooses some order π on the variables and creates a π-ordered OBDD D (which initially is equal to the constant true function) and a set of clauses S (which initially consists of all clauses of the formula F). While S is not empty the algorithm applies one of the following three operations:

1. (join or conjunction) delete some clause C from S and replace D by a π-ordered BDD that represents the conjunction $D \wedge C$;
2. (projection) choose a variable x that has no occurrences in the clauses from S and replace D by a π-ordered BDD for the function $\exists x\ D$;
3. (reordering) choose a new order on variables π' and replace D by the equivalent π'-ordered diagram. Assign $\pi := \pi'$. (Note that Pan and Vardi did not consider this rule in [15]).

After every step of the algorithm, the following invariant is mainteined: F is satisfiable if and only if $\bigwedge_{C \in S} C \wedge D$ is satisfiable. After the termination of the algorithm the set S is empty; if the diagram D has a path from the source to a vertex labeled by 1, then the algorithm returns "Satisfiable" and returns "Unsatisfiable" otherwise.

We refer to the algorithms of this type as OBDD($\wedge, \exists$, reordering)-algorithms. Besides, we use a similar notation for algorithms that use some of the rules: we just enumerate the used rules in the brackets. For example, the OBDD($\wedge$)-algorithms use only the conjunction rule and the OBDD($\wedge, \exists$)-algorithms use only the conjunction and projection rules.

Since join and projection for OBDDs may be performed in polynomial time and reordering may be performed in time polynomial on the sizes of the input and the output, the running time of an OBDD($\wedge, \exists$, reordering)-algorithm are polynomially related with the total sum of the sizes of all values in the diagram D. We ignore the time spent on choosing π and other operations with the permutation.

2.4 Error-correcting codes

By a *code* we mean a subset of binary strings with a fixed length. A code $\mathcal{C} \subseteq \{0,1\}^n$ has a relative distance δ if for any two codewords $c_1, c_2 \in C$ the Hamming distance between c_1 and c_2 is at least δn.

A *linear code* a set of all n-bits vectors $x = (x_1 \dots x_n)$ from some linear subspace in $\mathbb{F}_2^n$. If k is the dimension of this space, then the ratio k/n is called the *rate* of the code.

A linear code can be specified by a system of linear equations. For a code of dimension k this system should consist of $m \geq n - k$ linear equations involving n variables. The set of all solutions of the system should give exactly our code, so the rank of the system must be equal to $n - k$. If we require in addition that the equations in the system are linearly independent, then the number of equations is equal to $m = n - k$. The matrix of this linear system is called a *checksum matrix* of the code.

For a checksum matrix (h_{ij}) over $\mathbb{F}_2$ we say that a column i *intersects* a row j, if $h_{ij} = 1$. Further, we say that a tuple of columns $\langle i_1, \ldots, i_s \rangle$ intersects some row j if at least one of the columns $i_1, \ldots, i_s$ intersects row j.

We say that a code $\mathcal{C}$ recovers a fraction of ρ erasures by a list of size L (or C is (ρ, L)-erasure list-decodable) if for any $w \in \{0, 1, ?\}^n$ such that the number of "?" in w does not exceed ρn, there exist at most L elements in C that are consistent with w. A string $s \in \{0, 1\}^n$ is consistent with w if for all i, $w_i \in \{0, 1\}$ implies $s_i = w_i$.

▶ **Theorem 1** ([9, Lemma 2]). *If $\mathcal{C}$ is a code with relative distance δ, then for every $\epsilon > 0$ the code C is $((2 - \epsilon)\delta, \frac{2}{\epsilon})$-erasure list-decodable.*

3 Lower bounds for OBDD(∧, reordering)

3.1 Tseitin formulas

In this Section we prove an exponential lower bound on the size of OBDD(∧, reordering)-proofs of Tseitin formulas.

A Tseitin formula $TS_{G,c}$ is based on an undirected graph $G(V, E)$ with degree bounded by a constant d. Every edge $e \in E$ has the corresponding propositional variable p_e (in fact variables for loops are not used). There is a function $c : V \to \{0, 1\}$, we call it the labelling function. For every vertex $v \in V$ we write down a formula in CNF that encodes $\sum_{u \in V : (u,v) \in E, u \neq v} p_{(u,v)} \equiv c(v) \pmod 2$. The conjunction of the formulas described above is called a Tseitin formula. If $\sum_{v \in U} c(v) = 1$ for some connected component $U \subseteq V$, then the Tseitin formula is unsatisfiable. Indeed, if we sum up (modulo 2) all equalities stated in vertices from U we get $0 = 1$ since every variable has exactly 2 occurrences. If $\sum_{v \in U} c(v) = 0$ for every connected component U, then the Tseitin formula is satisfiable ([19, Lemma 4.1]).

Note that if formulas $TS_{G,c}$ and $TS_{G,c'}$ are satisfiable and $c \neq c'$ then $TS_{G,c}$ and $TS_{G,c'}$ are different functions since any satisfying assignment of $TS_{G,c}$ can not satisfy $TS_{G,c'}$.

In the following, we denote the number of connected components of a graph G by $\sharp G$.

▶ **Theorem 2.** *Let graph G with vertices V and edges E have the following property: for some $m \in [|E|]$ and $k > 0$ for all subsets $E' \subseteq E$ of the size m the inequality $\sharp G' + \sharp G'' \leq k$ holds, where G' and G'' are graphs with vertices V and edges E' and $E \setminus E'$ respectively. If $TS_{G,c}$ is satisfiable then any* OBDD *for $TS_{G,c}$ has at least $2^{|V|-k}$ vetices on the distance m from the source.*

Proof. Let us fix an order π of the variables of $TS_{G,c}$ (i.e. π orders the edges of G). Let E' be the set of the first m edges in this order. We show that there are at least $2^{|V|-k}$ substitutions to the variables from E' such that applying each of these substitutions to $TS_{G,c}$ results in $2^{|V|-k}$ different functions. It implies that the size of every OBDD representing $TS_{G,c}$ has at least $2^{|V|-k}$ vetices on the distance m from the source, since these substitutions correspond to paths in the OBDD with different endpoints.

Let $c' : V \to \{0, 1\}$ be a labeling function that corresponds to a partial substitution with support E': in every vertex v, $c(v)$ is the sum modulo 2 of the values of all edges from E' that are incident to v. Note that making a partial substitution to a Tseitin formula $TS_{G,c}$ gives as the resulting formula again a Tseitin formula – some formula $TS_{G'',c+c'}$, where G'' is a graph with vertices V and edges $E \setminus E'$, and $c + c'$ is the sum of the functions c and c' modulo 2.

We will prove a lower bound for the number of different c' such that they can be obtained by a substitution and $TS_{G'',c+c'}$ is satisfiable. The required properties of c' can be described

by a system of linear equations with variables $c'(v)$ for $v \in V$: for every connected component U of graph G' with vertices V and edges E' we put down the equation: $\sum_{v \in U} c'(v) = 0$ (this subsystem states that c' can be obtained by a substitution or in other words that $TS_{G',c'}$ is satisfiable) and for each connected component W of G'' we put down the equation: $\sum_{v \in W} c(v) + c'(v) = 0$ (this subsystem corresponds to the satisfiability of $TS_{G'',c+c'}$).

The system has a solution since $TS_{G,c}$ is satisfiable. There are $|V|$ variables and at most $\sharp G' + \sharp G'' \leq k$ equations. Hence there is at least $2^{|V|-k}$ solutions. Different solutions corresponds to different satisfiable formulas $TS_{G'',c+c'}$ and thus to different functions. ◀

We will apply Theorem 2 to algebraic expanders.

▶ **Definition 3.** A graph G with vertices V and edges E is an (n, d, α)-algebraic expander, if $|V| = n$, the degree of any vertex in V equals d and the absolute value of the second largest eigenvalue of the adjacency matrix of G is not greater than αd.

It is well known that for all $\alpha > 0$ and all large enough constants d there exists a family G_n of (n, d, α)-algebraic expanders. There are explicit constructions such that G_n can be constructed in poly(n) time [12]. Also, it is known that a random d-regular graph is a good enough expander with high probability.

▶ **Theorem 4.** *Let $G(V, E)$ be an (n, d, α)-algebraic expander for $\alpha < \frac{1}{2}$. Then for any $E' \subseteq E$ if $|E'| = \frac{n}{4d}$, then $\sharp G' + \sharp G'' \leq n(1 - \epsilon) + 2$, where G' is a graph with vertices V and edges E', G'' is a graph with vertices V and edges $E \setminus E'$ and $\epsilon = \frac{1}{8d(\alpha d+1)} - \frac{1}{d^2}$. Note that $\epsilon > 0$ if $\alpha < 1/32$ and $d \geq 4$.*

▶ **Corollary 5.** *If graph G is an (n, d, α)-algebraic expander for $\alpha < \frac{1}{32}$ and $d \geq 4$, and a Tseitin formula $TS_{G,c}$ is satisfiable, then the size of any* OBDD *for $TS_{G,c}$ is $2^{\Omega(n)}$.*

Proof. Follows from Theorems 2 and 4. ◀

▶ **Theorem 6.** *Let G be an (n, d, α)-algebraic expander with $d \geq 50$, $\alpha < \frac{1}{32}$. Then any* OBDD$(\wedge,$ reordering$)$*-proof of any unsatisfiable Tseitin formula $TS_{G,c}$ has the size at least $2^{\Omega(n)}$.*

Sketch of the Proof. We consider the last step of the proof: the conjunction of OBDDs F_1 and F_2 is the identically false function but both F_1 and F_2 are satisfiable. Both F_1 and F_2 are conjunctions of several clauses of $TS_{G,c}$. We use that a satisfiable Tseitin formula based on an expander has only exponential sized OBDDs. Moreover, if the underlying graph differs from some expander by $o(n)$ edges, then any of its OBDD representations has also an exponential size, since the number of connected component of graphs G' and G'' in Theorem 4 changes by at most $o(n)$.

Note that F_1 and F_2 together contains all clauses of $TS_{G,c}$. The main case is the following: there are two nonadjacent vertices u and v such that F_1 does not contain a clause C_u that corresponds to the vertex u and F_2 does not contain a clause C_v that corresponds to v. We consider two partial substitutions ρ_1 and ρ_2 that are defined on edges adjacent with u and v and on the edges of the shortest path p between u and v. The substitutions ρ_1 and ρ_2 assign opposite values to edges from the path p and are consistent on all other edges. The substitution ρ_1 satisfies C_v and refutes C_u and ρ_2 satisfies C_u and refutes C_v.

By the construction $F_1|_{\rho_1} \wedge F_2|_{\rho_2}$ is the satisfiable Tseitin formula based on the graph that is obtained from G by deletion of the vertices u and v and all edges from the path p (it is also possible that this formula does not contain some clauses for the vertices from p). The size of an OBDD representation of such a formula is exponential since the difference of the

underlying graph from the expander is at most $o(n)$. (Note that p is the shortest path in the expander, thus p contains at most $O(\log n)$ edges). Hence we get that either F_1 or F_2 has an exponential size in the given order. ◀

3.2 Pigeonhole principle

Let m and n be integers and $p_{i,j}$ be different variables; $p_{i,j}$ states whether the ith pigeon is in the jth hole or not. A formula $\mathtt{PHP}_n^m$ has two types of clauses. Clauses of the first type (long clauses) states that every pigeon is in at least one hole: $\bigvee_{j=1}^{n} p_{i,j}$ for all $i \in [m]$. Clauses of the second type (short clauses) states that in every hole there is at most one pigeon: $\neg p_{i,k} \vee \neg p_{j,k}$ for all $k \in [n]$ and all $i \neq j \in [m]$.

▶ **Theorem 7.** *Any* OBDD($\wedge$, reordering)*-proof of the pigeonhole principle formula* $\mathtt{PHP}_n^{n+1}$ *has size at least* $2^{\Omega(n)}$.

4 OBDD($\wedge$, reordering) is stronger than OBDD($\wedge$)

In this section we give an example of a family of unsatisfiable formulas Φ_n that have OBDD($\wedge$, reordering)-proofs of polynomial size while all OBDD($\wedge$)-proofs have size at least $2^{\Omega(n)}$.

▶ **Theorem 8.** *Let* $\Psi_n(x_1, x_2, \ldots, x_n)$ *be a family of unsatisfiable formulas of size* poly(n) *that satisfies the following conditions:*

- *there exists an order* τ *such that* Ψ_n *has* τ-OBDD($\wedge$) *refutation of the size poly*(n)*;*
- *there exists a polynomial* $p(n)$*, a function* $k : \mathbb{N} \to \mathbb{N}$ *with* $k(n) \leq \log p(n)$ *and permutations* $\sigma_1, \sigma_2, \ldots, \sigma_{2^{k(n)}} \in S_n$ *such that for any permutation* $\pi \in S_n$ *there exists* $i \in [2^{k(n)}]$ *such that any* $\pi\sigma_i$-OBDD($\wedge$)*-proof of* Ψ_n *has the size at least* $2^{\Omega(n)}$.

Then the formula

$$\Phi_n(w_1, w_2, \ldots, w_k, x_1, x_2, \ldots, x_n) = \bigwedge_{i=1}^{2^{k(n)}} \big((w = i-1) \to \Psi(x_{\sigma_i(1)}, x_{\sigma_i(2)}, \ldots, x_{\sigma_i(n)})\big)$$

has an OBDD($\wedge$, reordering)*-proof of the size* poly(n)*, but any* OBDD($\wedge$)*-proof has the size at least* $2^{\Omega(n)}$*. Here the equality* $w = i-1$ *means that* $w_1 w_2 \ldots w_{k(n)}$ *is a binary representation of the integer* $i-1$*. We assume that* Φ_n *is written in CNF as follows: we add a clause* $\neg(w = i-1)$ *to every clause of* $\Psi(x_{\sigma_i(1)}, x_{\sigma_i(2)}, \ldots, x_{\sigma_i(n)})$.

The similar construction was used by Segerlind [17] in order to show that tree-like OBDD($\wedge$, weakening) does not sumulate Resolution.

Sketch of the Proof. The lower bound. Consider an OBDD($\wedge$)-proof T of the formula Φ_n, let τ be the order of the variables $x_1, x_2, \ldots, x_n$ that is induced by the order from T. By the statement of the theorem, there exists $1 \leq i \leq 2^k$ such that all $(\tau\sigma_i)$-OBDD($\wedge$)-proofs of Ψ have at least exponential size. We make a substitution $w = i-1$ to the proof T. This substitution converts the proof of Φ_n into a proof of Ψ_n with the order $\tau\sigma_i$. Hence, T has an exponential size.

The upper bound. Since there exists a polynomial sized OBDD($\wedge$)-proof of Ψ_n, then for all i there is an order π_i (we may assume after the permutation π the variables w get the leftmost positions) such that there is a π_i-OBDD($\wedge$) derivation of the diagram representing $w \neq i-1$. From all such diagrams for different i we may construct a polynomial sized refutation of Φ_n, since w contains only $O(\log n)$ variables. ◀

Now we construct a family of unsatisfiable formulas Ψ_n that satisfies the conditions of Theorem 8. We use an argument similar to the proofs of the lower bounds for OBDD($\wedge$, reordering)-proofs. At first, we construct a function that has the sizes of OBDD representations in different orders close to the required sizes of proofs for Ψ_n.

Let $EQ_n : \{0,1\}^{2n} \to \{0,1\}$ be the equality predicate on pairs of n-bits strings. We denote it $EQ_n(x_1, x_2, \ldots, x_n, y_1, \ldots, y_n)$; in this notation the value of EQ_n is true iff $x_1 x_2 \ldots x_n = y_1 y_2 \ldots y_n$.

▶ **Proposition 9.** In the order $x_1, y_1, x_2, y_2, \ldots, x_n, y_n$ the function EQ_n has an OBDD representation of the size $3n + 2$.

Proof. The proof can be easily done by induction, using the following equation: $EQ_n(x_1, x_2, \ldots, x_n, y_1, \ldots, y_n) = (x_1 = y_1) \wedge EQ_{n-1}(x_2, \ldots, x_n, y_2, \ldots, y_n)$. ◀

The proof of the following lemma is similar to the proof of the $\Omega(n)$ lower bound on the best communication complexity of the shifted equality function [11, Example 7.9].

▶ **Lemma 10.** *Let σ_i for $i \in [n]$ be a cyclic permutation of the variables y that maps y_j to $y_{i+j \bmod n+1}$ for any $j \in [n]$. Formally $\sigma_i(j) = j$ for $j \in [n]$ and $\sigma_i(n + j) = n + (i + j - 1 \bmod n) + 1$ for $j \in [n]$. Then for any order π on $2n$ variables there exists $i \in [n]$ such that every $\pi\sigma_i$-*OBDD *representation of EQ_n has the size at least $2^{\Omega(n)}$.*

Now we are ready to construct a formula that may be used in Theorem 8. Consider a formula $\Psi_n(x, y, z)$ from $3n + 1$ variables (here x, y are vectors of n variables and z is a vector of $n + 1$ variables) that is the conjunction of CNF representations of the following conditions: $x_i = y_i$ for all $i \in [n]$; z_0; $(x_i = y_i) \to (z_{i-1} \to z_i)$ for all $i \in [n]$; $\neg z_n$. Note that $\Psi_n(x, y, z)$ is unsatisfiable since we have that $x_i = y_i$ for all i; it implies that $z_i = 1$ for all i, but z_n should be zero. The following statement is straightforward.

▶ **Proposition 11.** Ψ_n has a OBDD($\wedge$)-proof of polynomial size in the order $z_0, x_1, y_1, z_1, x_2, y_2, z_2, \ldots, x_n, y_n, z_n$.

▶ **Lemma 12.** *For any order π on the variables x, y, z the size of a π-*OBDD($\wedge$)*-proof of Ψ_n is at least $\frac{1}{10}\sqrt{S}$, where S is the size of the shortest π'-*OBDD *representation of $EQ_n(x, y)$, where π' is the order induced by π on the variables x, y.*

▶ **Theorem 13.** *There exists an unsatisfiable CNF formula Φ_n of size poly(n) such that there exists a polynomial size* OBDD($\wedge$, reordering)*-proof of Φ_n but every* OBDD($\wedge$)*-proof of Φ_n has the size $2^{\Omega(n)}$.*

Proof. By Lemma 12 for every order π a size of π-OBDD($\wedge$)-proof of Ψ_n is at least $\frac{1}{10}\sqrt{S}$, where S is the size of the shortest π'-OBDD representation of EQ_n and π' is the order induced by π. By Lemma 10 there exists a family of permutations σ_i of $[2n]$ for $i \in [n]$ such that for every order τ there exists an $i \in [n]$ such that the size of any $\tau\sigma_i$-OBDD($\wedge$)-proof of Ψ_n is at least $2^{\Omega(n)}$. By Proposition 11 there exists a required order τ and a τ-OBDD($\wedge$)-proof of Ψ_n of size poly(n). Theorem 8 gives a construction of the desired formula Φ_n. ◀

5 OBDD($\wedge$, $\exists$, reordering)-algorithms

It is known that OBDD($\wedge$, $\exists$)-algorithms can prove $\mathtt{PHP}_n^{n+1}$ in polynomial time [4]. Now we show that the Tseitin formulas are also easy for OBDD($\wedge$, $\exists$)-algorithms.

▶ **Proposition 14.** There exists an OBDD$(\wedge, \exists)$-algorithm that solves Tseitin formulas in polynomial time.

Sketch of the Proof. Notice that the projection of two linear equations over the common variable is just the sum of these equations. Since every variable has exactly two occurrences in Tseitin formulas, we can sum up all equations in every connected component. ◀

Now we show that there are satisfiable linear systems over $\mathbb{F}_2$ that are hard for OBDD $(\wedge, \exists, \text{reordering})$. At first we notice that every OBDD for a characteristic function of a good enough code has at least exponential size.

▶ **Theorem 15.** *Let $\mathcal{C} \subseteq \{0,1\}^n$ be a $(\frac{1}{2}+\epsilon, L)$-erasure list decoding code. Then every* OBDD *representing the characteristic function of $\mathcal{C}$ has the size at least $\frac{|\mathcal{C}|}{L^2}$.*

Moreover, for every tuple of $k \in [2\epsilon n]$ different indices $i_1, \dots, i_k \in [n]$ every OBDD *for the Boolean function $\exists x_{i_1} \dots \exists x_{i_k}\ \chi_{\mathcal{C}}(x_1, \dots, x_n)$ has the size at least $\frac{|\mathcal{C}|}{L^2}$.*

▶ **Lemma 16.** *Let A be an $m \times n$ checksum matrix of a (ρ, L)-erasure list decodable code. The matrix A' is the result of deleting r rows from A. Then A' is a checksum matrix of a $(\rho, 2^r L)$-erasure list-decodable code.*

▶ **Theorem 17.** *Let $\mathcal{C} \subseteq \{0,1\}^n$ be a linear code with relative distance $\frac{1}{3}$ such that the checksum matrix H of the code $\mathcal{C}$ has the following properties:*

- *H has size $\alpha n \times n$, where $\alpha \in (0;1)$ is a constant;*
- *every row of H contains at most $t(n)$ ones, where t is some function;*
- *every $\frac{1}{6}n$ columns of H intersect (contains ones in) at least $(\alpha - \delta)n$ rows, where $\delta \in (0, \frac{1-\alpha}{2})$ is a constant.*

Let the formula F_n be the standard representation of $H(x) = 0$ as a t-CNF of size at most $\alpha n 2^{t(n)-1} t(n)$ (every equation is represented in a straightforward way, without any additional variables). Then every OBDD$(\wedge, \exists, \text{reordering})$*-algorithm runs on the formula F_n for at least $2^{\Omega(n)}$ steps.*

Proof. The code $\mathcal{C}$ has the relative distance $\frac{1}{3}$. Hence, $\mathcal{C}$ is $(\frac{7}{12}, 8)$-erasure list-decodable by Theorem 1, choosing $\epsilon = \frac{1}{4}$. We consider the execution of an OBDD$(\wedge, \exists)$-algorithm on the formula F_n. We prove that at some moment size of a diagram D will be at least $2^{\Omega(n)}$. Assume that the algorithm during its execution applies the projection operation at least $k = \frac{1}{6}n$ times. We consider the diagram D just after the first moment when the algorithm applies the projection operation k times. Let D represent a function of type $\exists y_1, \dots, y_k\ \phi(x_1, \dots, x_{n-k}, y_1, \dots, y_k)$, where ϕ is the conjunction of several clauses from F_n. The projection operation on a variable x can be applied only if all clauses from S do not depend on x. Then all clauses corresponding to the linear equations with the variable x must be among clauses of ϕ. By the assumption of the theorem any k columns of H have ones in at least $(\alpha - \delta)n$ rows. Thus, ϕ contains all clauses from the representation of $(\alpha - \delta)n$ equations and possibly several other clauses from other equations.

Lemma 16 implies that ϕ is a characteristic function of a $(\frac{7}{12}, 8 \cdot 2^{\delta n})$-erasure list-decodable code. The number of satisfying assignments of ϕ is at least the number of solutions of the system $Hx = 0$, hence size of the code defined by ϕ is at least $2^{(1-\alpha)n}$. By Theorem 15 size of every OBDD for $\exists y_1, \dots, y_k\ \phi(x_1, \dots, x_{n-k}, y_1, \dots, y_k)$ is at least $2^{(1-\alpha)n} 2^{-2\delta n} \frac{1}{16} > 2^{(1-\alpha-2\delta)n-4}$ for every order of variables.

If the algorithm applies the projection operations less than $\frac{1}{6}n$ times, the argument is similar; we just need to consider the last diagram D. ◀

In the next section, we will show that there exist linear codes matching the requirements of Theorem 17. These constructions together with Theorem 17 imply the following result:

▶ **Corollary 18.** *For all large enough n there exists a CNF formula of size $O(n)$ with n Boolean variables, such that every* OBDD$(\wedge, \exists, \text{reordering})$*-algorithm runs on this formula at least $2^{\Omega(n)}$ steps.*

6 Code construction

In this section, we use Low Density Parity Codes (LDPC) of Gallager [7] with quite standard parameters (a constant number of ones in each row and in each column of the checksums matrix). We supplement the usual definition of LDPCs with a rather nonconventional property of uniformity. In what follows we prove that most random Gallager's codes with suitably chosen parameters enjoy this property.

First of all we recall the classic construction of random LDPC from [7]. Let us fix some integer parameters t, r, and n (assuming that t divides n). Define the "basic" matrix A of size $(n/t) \times n$ as a concatenation of t copies of the identity matrix $(n/t) \times (n/t)$.

Notice that each column of A contains one non-zero element; in each row of A there are exactly t non-zero elements. Further, we consider the family of all matrices H of size $(rn/t) \times n$ that consist of r horizontal "blocks" of size $(n/t) \times n$, where each block is obtained as a permutation of columns of A. It is easy to see that in each column of this matrix there are r ones, and in each row there are exactly t ones. We introduce the uniform distribution on all matrices of this type. We can interpret these matrices H as checksums matrices of some linear codes. Gallager proved that most of these codes have rather large minimal distance.

▶ **Proposition 19** (see [7])**.** Most (say, at least 90%) of matrices in Gallager's family define a linear code with parameters approaching Gilbert–Varshamov bound. More precisely, for every $\delta \in (0, \frac{1}{2})$ and for every t there exists $r = r(t)$ such that for large enough n most matrices from the defined family have the minimal distance $\geq \delta n$. Moreover, the ratio $r(t)/t$ approaches $h(2\delta)$ as t goes to infinity, where $h(x) = -x \log x - (1-x)\log(1-x)$ (the binary entropy). This means that the rate of the code can be made arbitrarily close to $1 - h(2\delta)$ (i.e., to the Gilbert–Varshamov bound).

A family of linear codes defined above can be specified by parameters r, t, n. However, it is more convenient to specify these codes by another triple of numbers – by (δ, t, n) (assuming that the value $r = r(\delta, t, n)$ is defined implicitly as the minimal integer such that most codes of the family have the minimal distance greater than δn). Now we can state the main technical lemma of this section.

▶ **Lemma 20.** *For all $\beta \in (0, 1)$, $\gamma < 1$, and $\delta \in (0, \frac{1}{2})$, for all large enough t most (say, at least 90%) of linear codes from Gallager's family with parameters (δ, t, n) satisfy the following property: every βn columns in the checksum matrix of the code intersect at least a fraction γ of all rows of the matrix.*

▶ **Corollary 21.** *For the distribution defined above, the system of linear equations $Hx = 0$ with probability close to 1 can be represented as a CNF of size $O(n)$.*

Thus, we obtain Corollary 18 for CNF of size $O(n)$ (with n Boolean variables). In other words, for every N there exists a CNF formula of size N such that every OBDD$(\wedge, \exists, \text{reordering})$-algorithm runs on this formula at least $2^{\Omega(N)}$ steps.

Acknowledgements. The authors are grateful to Sam Buss for fruitful discussions and to the anonymous reviewers for useful comments.

References

1. Albert Atserias, Phokion G. Kolaitis, and Moshe Y. Vardi. Constraint propagation as a proof system. In Mark Wallace, editor, *Principles and Practice of Constraint Programming – CP 2004*, volume 3258 of *Lecture Notes in Computer Science*, pages 77–91. Springer, 2004. doi:10.1007/978-3-540-30201-8_9.
2. Paul Beame, Toniann Pitassi, and Nathan Segerlind. Lower Bounds for Lovász–Schrijver Systems and Beyond Follow from Multiparty Communication Complexity. *SIAM J. Comput.*, 37(3):845–869, 2007. doi:10.1137/060654645.
3. Randal E. Bryant. Symbolic manipulation of boolean functions using a graphical representation. In *Proceedings of the 22nd ACM/IEEE conference on Design automation, DAC 1985, Las Vegas, Nevada, USA, 1985.*, pages 688–694, 1985. doi:10.1145/317825.317964.
4. Wěi Chén and Wenhui Zhang. A direct construction of polynomial-size OBDD proof of pigeon hole problem. *Information Processing Letters*, 109(10):472–477, 2009. doi:10.1016/j.ipl.2009.01.006.
5. Pavol Duris, Juraj Hromkovic, Stasys Jukna, Martin Sauerhoff, and Georg Schnitger. On multi-partition communication complexity. *Inf. Comput.*, 194(1):49–75, 2004. doi:10.1016/j.ic.2004.05.002.
6. Luke Friedman and Yixin Xu. Exponential Lower Bounds for Refuting Random Formulas Using Ordered Binary Decision Diagrams. In Andrei A. Bulatov and Arseny M. Shur, editors, *CSR 2013*, volume 7913 of *Lecture Notes in Computer Science*, pages 127–138. Springer, 2013. doi:10.1007/978-3-642-38536-0_11.
7. Robert G. Gallager. Low-density parity-check codes. *Information Theory, IRE Transactions on*, 8(1):21–28, 1962.
8. Jan Friso Groote and Hans Zantema. Resolution and binary decision diagrams cannot simulate each other polynomially. *Discrete Applied Mathematics*, 130(2):157–171, 2003. doi:10.1016/S0166-218X(02)00403-1.
9. Venkatesan Guruswami. List decoding from erasures: bounds and code constructions. *Information Theory, IEEE Transactions on*, 49(11):2826–2833, 2003. doi:10.1109/tit.2003.815776.
10. Jan Krajíček. An exponential lower bound for a constraint propagation proof system based on ordered binary decision diagrams. *Journal of Symbolic Logic*, 73(1):227–237, 2008. doi:10.2178/jsl/1208358751.
11. Eyal Kushilevitz and Noam Nisan. *Communication complexity*. Cambridge University Press, 1997.
12. A. Lubotzky, R. Phillips, and P. Sarnak. Ramanujan graphs. *Combinatorica*, 8(3):261–277, 1988.
13. Christoph Meinel and Anna Slobodova. On the complexity of Constructing Optimal Ordered Binary Decision Diagrams. *Proceedings of Mathematical Foundations of Computer Science*, 841:515–524, 1994.
14. Christoph Meinel and Thorsten Theobald. *Algorithms and Data Structures in VLSI Design: OBDD – Foundations and Applications*. Springer, 1998.
15. Guoqiang Pan and Moshe Y. Vardi. Search vs. Symbolic Techniques in Satisfiability Solving. *7th International Conference on Theory and Applications of Satisfiability Testing, SAT 2004, Revised Selected Papers*, 3542:235–250, 2005. doi:10.1007/11527695_19.
16. Nathan Segerlind. Nearly-Exponential Size Lower Bounds for Symbolic Quantifier Elimination Algorithms and OBDD-Based Proofs of Unsatisfiability. *CoRR*, abs/cs/070:40, 2007. arXiv:0701054.

17 Nathan Segerlind. On the relative efficiency of resolution-like proofs and ordered binary decision diagram proofs. In *Proceedings of the 23rd Annual IEEE Conference on Computational Complexity, CCC 2008, 23-26 June 2008, College Park, Maryland, USA*, pages 100–111, 2008. doi:10.1109/CCC.2008.34.

18 Olga Tveretina, Carsten Sinz, and Hans Zantema. An Exponential Lower Bound on OBDD Refutations for Pigeonhole Formulas. *Proceedings Fourth Athens Colloquium on Algorithms and Complexity*, 4(Acac):13–21, 2009. arXiv:0909.5038, doi:10.4204/EPTCS.4.2.

19 Alasdair Urquhart. Hard Examples for Resolution. *JACM*, 34(1):209–219, 1987. doi:10.1145/7531.8928.