Electronic Colloquium on Computational Complexity, Report No. 23 (2021)

$3.1n - o(n)$ Circuit Lower Bounds for Explicit Functions

Jiatu Li
IIIS, Tsinghua University
Beijing, China
lijt19@mails.tsinghua.edu.cn

Tianqi Yang
IIIS, Tsinghua University
Beijing, China
yangtq19@mails.tsinghua.edu.cn

February 20, 2021

Abstract

Proving circuit lower bounds has been an important but extremely hard problem for decades. Although one may show that almost every function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ requires circuit of size $\Omega(2^n/n)$ by a simple counting argument, it remains unknown whether there is an *explicit* function (for example, a function in NP) not computable by circuits of size $10n$. In fact, a $3n - o(n)$ explicit lower bound by Blum (TCS, 1984) was unbeaten for over 30 years until a recent breakthrough by Find et al. (FOCS, 2016), which proved a $(3 + \frac{1}{86})n - o(n)$ lower bound for *affine dispersers*, a class of functions known to be constructible in P.

In this paper, we prove a stronger lower bound $3.1n - o(n)$ for affine dispersers. To get this result, we strengthen the gate elimination approach for $(3 + \frac{1}{86})n$ lower bound, by a more sophisticated case analysis that significantly decreases the number of bottleneck structures introduced during the elimination procedure. Intuitively, our improvement relies on three observations: adjacent bottleneck structures becomes less troubled; the gates eliminated are usually connected; and the hardest cases during gate elimination have nice local properties to prevent the introduction of new bottleneck structures.

1 Introduction

Proving circuit lower bounds has been an important but extremely hard problem for decades. Indeed, strong lower bounds have already been shown since seventy years ago for circuits size even slightly smaller then optimal, say $\frac{2^n}{10n}$ by a simple counting argument [Sha49], but the proof does not give any *explicit* function (for example, computable in NP or even P) with large circuit complexity. Even though most people believe nowadays that $\mathsf{NP} \not\subseteq \mathsf{P}_{/\mathrm{poly}}$, we are still unable to prove much weaker lower bounds such as $\mathsf{NEXP} \not\subseteq \mathsf{TC}^0$ or $\mathsf{NP} \not\subseteq \mathsf{SIZE}[10n]$.

The attempts of finding functions computable in small uniform classes such as NP that require relatively large circuit size complexity go back to 1960s, starting from the $2n - O(1)$ lower bound for $\bigoplus(x_i \wedge x_j)$ proved by Kloss and Malyshev [KM65]. Another $2n - O(1)$ lower bound was given later by Schnorr [Sch74]. Soon after, this lower bound was pushed up to $2.5n - O(1)$ for certain symmetric functions by Stockmeyer [Sto77] and a slightly weaker $2.5n - o(n)$ for combinations of storage access functions by Paul [Pau77]. The latter one was then improved by Blum [Blu84] to $3n - o(n)$ with a slightly modified function, which stood unbeaten for over thirty years. It was not

ISSN 1433-8092

until 2015 when Find, Golovnev, Hirsch, and Kulikov [FGHK16] proved a $\left(3+\frac{1}{86}\right)n - o(n)$ lower bound for *affine dispersers* (which is a class of functions computable in polynomial time), following an alternative $3n - o(n)$ lower bound for the same function by Demenkov and Kulikov [DK11]. Based on a similar argument, Golovnev and Kulikov [GK16] proved that any *quadratic disperser* of appropriate parameters requires circuits of 3.11 size, but no explicit constructions are known for the parameters desired up to now.

Following the $\left(3+\frac{1}{86}\right)n - o(n)$ lower bound for *affine dispersers* on B_2 circuits by Find, Golovnev, Hirsch, and Kulikov [FGHK16], we strengthen the lower bound to $3.1n - o(n)$ with a more sophisticated case analysis building upon their proof.

Theorem 1.1 (Main theorem). Assume that C is a B_2 circuit computing an affine disperser for sublinear dimension, then C has size at least $3.1n - o(n)$. $\diamondsuit$

The key ingredient of our proof is a case study on the local topology of a *topological minimal* $\wedge$ gate in the circuit, and show that in each case the local part can be handled to meet our requirements. We will explain this intuition in Section 1.2 and make it formal in Section 4.

1.1 Related models

Computation models are essential when we talk about the complexity of concrete problems. In this paper, our interest is in single-output circuits with gates of fan-in 2 computing arbitrary functions out of $B_2 \triangleq \mathbb{F}_2 \times \mathbb{F}_2 \to \mathbb{F}_2$. For the sake of clarity, this model is called B_2 circuit.

Even if the usage of linear gates ($\oplus, \equiv$) is banned in the circuit (which are called U_2 circuits), significantly better lower bounds are still unknown. For this model, Schnorr [Sch76] first proved a $3n - O(1)$ lower bound for parity, which was later pushed up to $4n - O(1)$ by Zwick [Zwi91] on certain symmetric functions. Later, Lachish and Raz [LR01] gave a $4.5n - o(n)$ lower bound for *strongly two-dependent* functions, which was soon improved to $5n - o(n)$ by Iwama and Morizumi [IM02].

Since it is proven hard to show circuit lower bound for general models such as B_2 and U_2 circuit, efforts have been made on many more restricted models in the past few decades. For example, detecting the existence of a large clique in a graph is known to require exponential size of *monotone circuits* that only involve AND and OR gates of fan-in 2 [Raz85; AB87]. Note that it is a rather weak model, since monotone circuits can only compute *monotone functions*, i.e. $f(x_1, \ldots, x_n) \le f(y_1, \ldots, y_n)$ if $x_i \le y_i$ for all i.

Another restricted model of interest is the formulas, which are circuits such that each gate has fan-out at most 1. Explicit formula lower bounds have strong connections to the longstanding open problem P vs NC^1. For De Morgan formulas, Subbotovskaya [Sub61] proved an $\Omega(n^{1.5})$ lower bound for parity function, which was improved to $\Omega(n^2)$ later by [Khr71b] for the same function but via a different method. Khrapchenko [Khr71a] further formalize this method and get quadratic lower bounds for many different functions. However, it is hard to get a super-quadratic lower bound with it. To break this barrier, Andreev [And87] applied random restrictions on the model, and proved an $\Omega(n^{2.5-o(1)})$ lower bound for a cleverly constructed hard function known as Andreev function. By analyzing the *shrinkage exponent*, this lower bound was eventually improved to $\Omega(n^{3-o(1)})$ by [Tal14], following [IN93; PZ93]. Similar to the story of circuit lower bound, there is also a huge gap between what we believe ($\mathsf{P} \not\subseteq \mathsf{NC}^1$) and what we can prove (a function in P with $\Omega(n^{3-o(1)})$ formula size lower bound).

On low depth circuits, exponential lower bounds against AC^0 circuits have even been known for parity functions for a long time [Yao85; Hås86]. Generalizing this result, similar lower bounds can be shown for MOD_p functions against $\mathsf{AC}^0[q]$ circuits (AC^0 circuits with MOD_q gates) with $p \neq q$ being two primes [Raz87; Smo87]. However, their methods cannot be generalized to the cases when q is not prime. Indeed, the relationship between NEXP and even $\mathsf{AC}^0[6]$ is unknown for years. This was recently been solved by the seminal work by Williams [Wil13], who established a connection between circuit analysis algorithms and circuit lower bounds. By this *algorithmic approach*, Williams [Wil14] finally proved $\mathsf{NTIME}[2^n] \not\subseteq \mathsf{ACC}^0$ by showing an algorithm for #SAT of ACC^0 circuits, which was later strengthened to $\mathsf{NQP} \not\subseteq \mathsf{ACC}^0$ by Murray and Williams [MW20]. In fact, algorithmic lower bound applies to any well-behaved (for example, evaluatable in P, close under negation and conjunction) circuit class with efficient circuit analysis algorithm (for example, Circuit SAT or similar problems like Gap-UNSAT, see [MW20]). This gives us a novel way towards stronger lower bound for larger circuit classes, but we are still unable to get nontrivial circuit analysis algorithms even for many weak circuit classes such as TC^0.

1.2 Gate elimination

Gate elimination is currently our only weapon to prove unconditional explicit lower bounds for general Boolean circuits. The best lower bounds currently known for both B_2 and U_2 rely on gate elimination, while unconditional circuit lower bounds based on other methods seem too ad-hoc for further improvements. The intuition of gate elimination is to add restrictions to the input variables until we get a trivial circuit. If it is possible to find a function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ and a class of restrictions $\mathcal{R}$ to the inputs, such that any $r(n)$ restrictions cannot make the function trivial, and adding a restriction can eliminate δ gates for any circuit C computing f restricting to $\mathcal{R}$, we can then obtain a $\delta \cdot r(n)$ lower bound for this function by induction on n.

To make a trivial example, let us consider the parity function $\bigoplus_n(x_1, \ldots, x_n) = x_1 \oplus x_2 \oplus \cdots \oplus x_n$, which keeps non-constant even if we substitute constant values to $n-1$ of the variables. Together with the observation that substituting each variable by a constant will make its descendants useless[1], and can hence be removed from the circuit, we can prove that any circuit computing $\bigoplus_n$ needs at least $n-1$ gates. In fact, this lower bound can be improved to $3(n-1)$ for U_2 circuit with a slightly more involved case analysis.

One may notice that simple constant substitutions are unable to give us non-trivial lower bounds for B_2 circuits because of the existence of linear gates $\oplus$ and $\equiv$. As an extension of the parity function, *affine dispersers* are functions that keep non-constant even after many *affine substitutions* of form

$$x_j \leftarrow \bigoplus_{x_i \in F_j} x_i \oplus c$$

for some x_j, F_j and c. A $3n - o(n)$ lower bound for affine dispersers is given by Demenkov and Kulikov [DK11]. For short, they prove that three gates can be eliminated by cleverly performing an

[1]In fact, we shall formally name this case *degenerate* later in our paper, and the word *useless* will be used to represent a specific case. For details, see Section 3.3.

affine substitution to a variable, therefore an affine disperser that survives $n - o(n)$ substitutions has circuit complexity $3n - o(n)$.

Remark. Strong circuit lower bounds proved by gate elimination usually involve a tedious case analysis. Moreover, it seems hopeless to prove a superlinear circuit lower bound by gate elimination since it only concentrates on local structures of a circuit. This idea was formally shown recently in Golovnev, Hirsch, Knop, and Kulikov [GHKK18], who gave a limitation on the gate elimination method. Still, we have no better methods right now. One attempt to bypass this obstacle was recently presented by Golovnev, Kulikov, and Williams [GKW21], who showed a relationship between general circuits of small size and constant depth circuits of exponential size, inspired by the classical low depth reduction by Valiant [Val77]. In particular, they proved that a strong exponential lower bound against depth 3 circuits would imply a $3.9n$ lower bound against general B_2 circuits. Whether this approach can indeed lead to new lower bounds is still open.

1.3 The $(3 + \frac{1}{86})n - o(n)$ lower bound

The main obstacle for us to prove a lower bound greater than $3n$ for affine dispersers is the structure shown in Figure 1[2]. Simple substitutions to x and y will not allow us to remove more than 3 useless gates from the circuit. The pivotal idea in [FGHK16] is to explicitly take the number of such *troubled* structures into account. Following the notations in their paper, we call the gate G a *troubled gate*. This idea works based on two observations.

Ob1. There are not too many troubled gates in the circuits initially. More precisely, any circuit computing an sufficiently strong affine disperser contains at most $\frac{n}{2} + o(n)$ troubled gates.

Ob2. The number of troubled gates introduced while eliminating a gate from the circuit is bounded by a constant (which is 4 in [FGHK16]).

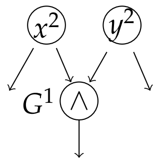


Figure 1: Main obstacle against the $3n$ barrier.

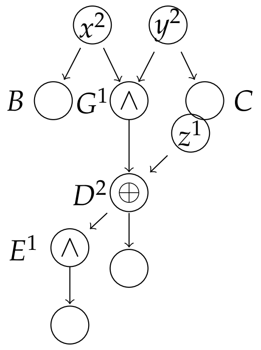


Figure 2: Even harder structure requiring quadratic substitution.

To properly utilize this, they define a quantity called *complexity measure*, which is the weighted sum of the size of the circuit and the number of troubled gates. Instead of directly counting the number of gates eliminated while adding each restriction, we calculate the decrement of complexity measure, i.e., we show that we can either reduce the size of the circuit a lot, or remove three

[2]For simplicity, we use superscripts after the labels of variables and gates to specify their *out degrees*.

gates while reducing the number of troubled gates. With the help of Ob2, a complexity measure lower bound for the original circuit can be obtained. Then by Ob1, we can translate this into a circuit size lower bound.

However, there is still a case shown in Figure 2 that even this argument does not work, since we can neither eliminate sufficiently many gates nor reduce the number of troubled gates. To handle this case, [FGHK16] tries to use the *quadratic substitution* of form

$$x_j \leftarrow ((x_i \oplus c_1) \wedge (x_k \oplus c_2)) \oplus c_3.$$

The restrictions applied to the circuit and the function therefore consist of both affine equations and quadratic equations with some structural constraints (in fact, the quadratic ones can be viewed as delayed affine substitutions). Formally, they define *read-once depth-two quadratic source* (or *rdq-source* for short) to represent this structure. By taking the number of quadratic equations into account in the complexity measure, we can finally resolve this issue.

1.4 Our improvements

The main bottleneck of their proof is the number of troubled gates introduced in one elimination (see Ob2). Find, Golovnev, Hirsch, and Kulikov [FGHK16] showed that each elimination can only introduce at most 4 troubled gates, which is in fact over-estimated. By a much more sophisticated analysis, we can significantly strengthen this upper bound on the number of troubled gates introduced, resulting in a stronger circuit lower bound. Generally speaking, our proof builds upon the following three observations.

Pairing troubled gates The first essential observation is that troubled gates will no longer bother us if they are adjacent. Indeed, if two troubled gates share a common variable and form the structure in Figure 3, we can handle them easily. We call this structure *troubled pairs*. We notice that if these cases are ruled out, then each elimination can only introduce 2 troubled gates.

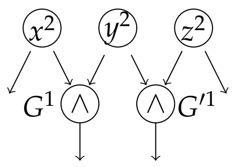


Figure 3: A pair of adjacent troubled gates.

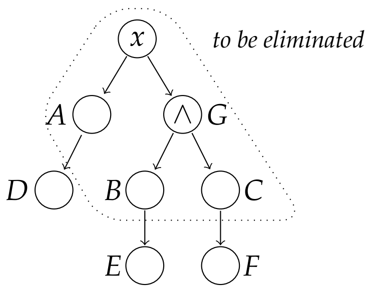


Figure 4: Eliminated gates are local.

The gates eliminated are connected [FGHK16] argued the introduction of troubled gates for each elimination *separately*. However, we observe that the gates to be eliminated *in a single substitution* usually form a highly local structure, and have good properties such as connectivity. Take Figure 4 for example, which we will formally handle by Case 3 later in our proof, we can substitute a constant to x to eliminate the gates G, A, B and C. By noticing that these gates form a connected subgraph, many troubled gates will not be introduced in each elimination. By a more rigorous analysis, almost every elimination will only introduce one troubled gate.

Bottleneck structures have good local properties By looking into the cases, the major bottleneck preventing us from better lower bounds is the case shown in Figure 2 (which introduces quadratic substitutions), and the cases eliminating quadratic substitutions. However, these cases all have very special local structures, so that we can either argue that some places will never introduce troubled gates, or we can rule out the cases when too many troubled gates are formed by showing that these cases can be easily handled in other ways. For example, in Figure 2, by doing a quadratic substitution, although 5 gates are eliminated, we can show that at most *one* troubled gate can be introduced in overall, which happens while eliminating the descendent of E.

Since the first two observations are graph-theoretical properties independent of the concrete function computed, we formalize them as the *normalization lemma* formally presented in Lemma 3.11. With this lemma and the third observation, we reformulate the case analysis for $(3 + \frac{1}{86})n$ circuit lower bound and resolve several bottleneck cases. This leads to the $3.1n - o(n)$ circuit lower bound if we choose optimal parameters to define the complexity measure.

1.5 Outline

In Section 2 we define basic concepts and operations for gate elimination. In Section 3 we formally define troubled gates, troubled pairs and the *normalization lemma*. Finally we present the complete proof of the $3.1n - o(n)$ circuit lower bound in Section 4.

2 Preliminaries

In this section we define basic concepts and operations required by gate elimination. The definitions and propositions in this section has been introduced in [FGHK16] for $(3 + \frac{1}{86})n$ circuit lower bound. For the convenience of the reader, we keep their notations on circuit, substitutions, etc. Readers that are not familiar with complexity theory are referred to [AB09] for the definition of standard notations like P, NP, AC^0, etc.

2.1 Basic concepts

A *circuit* is an acyclic directed graph, in which each vertex is either a *variable* or a *gate*. For variables, the corresponding vertices must have in-degree 0. Each gate is labelled with an arbitrary function out of a class of boolean functions $\mathcal{C}$, usually called the *basis* of the circuit. The *size* of the circuit means the number of *gates* in it. The basis we are interested in, called B_2, contains all $\mathbb{F}_2^2 \to \mathbb{F}_2$ functions. Hence in the rest of the paper, we will abuse the notation 'circuit' to mean B_2 circuits, if no other clarifications are specified.

Based on the functionality, we can classify all 16 kinds of gates in B_2 circuits into 4 types:

Trivial gate The outputs of these functions are independent of the inputs, i.e., $f(x, y) = c$. They are constant functions, and hence can in fact be replaced by a constant. However, to better illustrate our proof, we explicitly keep them in the circuit.

Degenerate gate These functions depend on *exactly* one of the inputs, i.e., $f(x,y) = (x \wedge c_1) \oplus (y \wedge \neg c_1) \oplus c_2$. A degenerate gate outputs one of the inputs or its negation, hence we can remove the gate and modify its descendants so that the circuit is simplified. Nevertheless, we keep them for the same reason.

$\oplus$-type gate These functions are affine functions depending on both inputs, i.e., $f(x,y) = x \oplus y \oplus c$.

$\wedge$-type gate These functions are quadratic functions depending on both inputs, i.e., $f(x,y) = ((x \oplus c_1) \wedge (y \oplus c_2)) \oplus c_3$.

A gate (or a variable) is called a k-gate (or a k-variable), if it has out-degree *exactly* k. We add a superscript $+$ after k (say k^+ gate) to mean that it has out-degree *at least* k. k^- gates and k^- variables are defined similarly.

For the convenience of case analysis, we will extensively use the following graphical representation of circuit (see Figure 5). The input variables are labeled with $x_1, x_2, \ldots, x_n$ inside the circle. The type of function computed by a gate is also marked inside the circle. The name and out-degree are included beside a node if necessary. Note the different circles in the graph may represent the same gate or input variable, unless we clarify it explicitly.

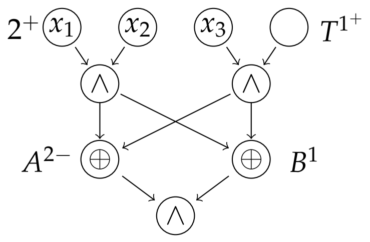


Figure 5: Local topology of a B_2 circuit.

2.2 Affine disperser

The hard functions we utilize are those which keeps non-constant when many affine constraints such as $x_1 \oplus x_3 \oplus x_7 = 1$ are introduced. We formally define such functions below.

Definition 2.1 (Affine disperser). A function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ is called an *affine disperser* for dimension d, if it is not a constant function when the inputs are restricted to any d dimensional affine subspace of $\mathbb{F}_2^n$. $\diamondsuit$

An affine disperser for dimension d is not constant when inputs are restricted to the solutions of a system of linear equations with d free variables. It can be considered as a generalization of the xor function, which is not trivial after assigning *constant* values to $n-1$ input variables.

For our lower bounds, affine dispersers for sublinear dimension are enough. Explicit constructions of these functions are already given in [Li11; BK12; Li15]. More precisely, there exists a polynomial time algorithm $A(n,x)$ such that for each n, $A(n,\cdot):\mathbb{F}_2^n\to\mathbb{F}_2$ is an affine disperser for sublinear dimension.

2.3 Read-once depth-two quadratic source

In order to perform gate elimination, we need to add restrictions to inputs to simplify the circuit. To simplify our proof, we use, instead of affine restrictions, a slightly stronger class of restriction called *read-once depth-two quadratic source* (or *rdq-source*) which was introduced in [FGHK16].

Definition 2.2 (Rdq-source). Let $X=\{x_1,\ldots,x_n\}$ be the set of variables and F, L, Q be a disjoint partition of X, containing *free*, *linear* and *quadratic* variables, respectively. An *rdq-source* is defined as the partition (F, L, Q) together with the following system of equations.

1. For each quadratic variable $x_j\in Q$, there is a quadratic equation
$$x_j=((x_i\oplus c_1)\wedge(x_k\oplus c_2))\oplus c_3,$$
where x_i, x_k are free variables and c_1, c_2, c_3 are constants. In addition, the variables at the right-hand side of the quadratic equations are *disjoint*.
2. For each linear variable $x_j\in L$, there is an affine equation
$$x_j=\bigoplus_{x_i\in F_j}x_i\oplus\bigoplus_{x_k\in Q_j}x_k\oplus c_1,$$
where $F_j\subseteq F$, $Q_j\subseteq Q$ and c_1 is a constant. Note that we do not require F_j or Q_j to be disjoint here. Linear variables are also called *affine variables* in this paper.

The *dimension* of an rdq-source is defined as the number of free variables. $\diamondsuit$

An rdq-source of dimension d defines a subset of $\mathbb{F}_2^n$ of size 2^d, since the values of linear and quadratic variables are uniquely determined by the values of free variables. Intuitively one may consider an rdq-source by a depth-two circuit containing a quadratic layer and an affine layer (see Figure 6).

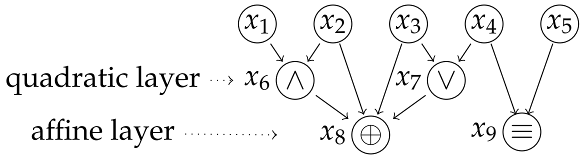


Figure 6: Rdq-source as depth-two circuit.

Rdq-source is said to be *read-once* since the variables at the right-hand sides of quadratic equations are disjoint, or equivalently, each free variable is read at most once by quadratic equations. A free variable is called *protected* if it occurs at the right-hand side of an quadratic equation, and the free variables that are not protected are called *unprotected*. The two (protected) variables occur in the same quadratic equation is said to be *coupled* with each other.

Rdq-source is said to be a *source* since it specify a subset of $\mathbb{F}_2^n$ on which the circuit should agree with the function it computes. Let $f : \mathbb{F}_2^n \to \mathbb{F}_2$ be a function and C be a circuit. Let R be an rdq-source on n variables. We say C compute f restricting to R (or C computes $f|_R$), if the following two conditions hold.

1. Linear and quadratic variables in R are 0-variables in C.
2. C computes f on all tuples of inputs satisfying the system of linear equations in R.

Proposition 2.3. An rdq-source of dimension $2d$ contains an affine subspace dimension d. Hence an affine disperser for dimension d is not constant restricting to rdq-source of dimension $2d$. $\diamondsuit$

Proof. An rdq-source of dimension $2d$ contains $2d$ free variables and at most d quadratic equations, since each quadratic equation contains two free variables. For each affine equation depending on free variables x_i and x_k, we further assign $x_i = 0$ such that it becomes an affine equation. We then obtain an rdq-source of dimension $\geq d$ without quadratic variable, which is exactly an affine subspace of dimension $\geq d$. $\square$

2.4 Substitutions

Let C be a circuit computing $f|_R$ for a function f and an rdq-source R, we can further restrict R to a smaller source R' by adding or modifying equations, and obtain a smaller circuit C' computing $f|_{R'}$ by removing redundant gates from C. This operation is called a *substitution*. If the restriction is denoted by $x_j = g(x_{i_1}, x_{i_2}, \ldots, x_{i_k})$ for a function g, we call it a g-substitution to x_j.

In order to define the modification of circuit after substitutions, we slightly extend the definition of circuits to involve *constants* of in-degree 0. In such case, a gate can be fed by gates, variables or constants[3]. In particular, a gate fed by a constant is either trivial or degenerate. Constants will be removed by normalization rules, which will be described in Section 3.3.

In this paper, we will use the following four types of substitutions.

Constant substitution. Let x_j be an arbitrary free variable, we can substitute $x_j \leftarrow c_1$ for constant c_1. For the circuit, all the descendants of x_j are rewired to be fed by a constant c_1. Since x_j is now a 0-variable, we can make it a linear variable by introducing an affine equation $x_j = c_1$. All its occurrence in linear and quadratic equations are replaced by the constant c_1, so that the system of equations remains to be a valid rdq-source.

Affine substitution to couples. Let x_j, x_k be a couple of free (protected) variables that occurs in the same quadratic equations for $x_i \in Q$. We can substitute $x_j \leftarrow x_k \oplus c_1$ for constant c_1, if it is possible to make x_j a 0-variable after appropriate rewiring[4]. All the occurrence of x_j in linear

[3]Note that the size of a circuit still refers to the number of gates involved; the constants as well as the variables are not counted.

[4]For rewiring we mean that, in order to make x_j a 0-variable, all its descendants in the original circuit should be fed by another variables. For instance, if an affine substitution $x_j \leftarrow x_k \oplus 1$ is performed, we can use the negation of x_k to denote x_j. More precisely, we can rewire all the gates fed by x_j to be fed by x_k, and change the function computed by these gates accordingly, to replace x_j by $x_k \oplus 1$.

and quadratic equations are replaced by $x_k \oplus c_1$, and in particular, the quadratic equation for x_i becomes an affine equation, hence x_i becomes a linear variable.

Affine substitution. Let x_j be an unprotected free variable and x_k be a free variable. We can substitute $x_j \leftarrow x_k \oplus c_1$ for constant c_1, and make x_j a 0-variable by appropriate rewiring. By this, we means to unfold x_j to be $x_k \oplus c_1$ at all its occurrence in the original affine equations. Clearly x_j becomes an affine variable with the equation $x_j = x_k \oplus c_1$, the validity of having no occurrence in quadratic equations is guaranteed since it is originally unprotected.

Quadratic substitution. Let x_j, x_i, x_k be unprotected free variables. We can substitute

$$x_j \leftarrow ((x_i \oplus c_1) \wedge (x_k \oplus c_2)) \oplus c_3$$

for constants c_1, c_2 and c_3, if it is possible to make x_j a 0-variable after appropriate rewiring. Similarly, x_j becomes a quadratic variable. It is easy to see that the equations containing x_j at the right-hand sides are still valid.

Since each type of substitution eliminates exactly one free variable, with Proposition 2.3, we immediately get the following fact.

Proposition 2.4. An affine disperser for dimension d if not trivial after performing $n - 2d$ substitutions. Hence an affine disperser for sublinear dimension is not trivial after performing $n - o(n)$ substitutions. $\diamondsuit$

Note that by performing a substitution, we will *always* disconnect the newly introduced non-free variable from the circuit by appropriate rewiring. So throughout the process, all variables with non-zero out-degrees should be free variables.

2.5 Cyclic circuit

Substitution to variables will make their descendants trivial or degenerate so that can be removed from the circuit. However, the local topology near the inputs may not be good enough to eliminate sufficiently many gates. For example, if the inputs only feed $\oplus$-type gates and are all 3^--variables, it will be hard to eliminate more than three gates. In general, it will be hard to eliminate more gates if every $\wedge$-type gate is far from the inputs.

The solution to this issue, proposed by Find, Golovnev, Hirsch, and Kulikov [FGHK16], is to generalize the circuit so that we can substitute constant values to *gates* apart from variables. This will allow us to substitute a constant value to the input of gates we want even if it is a gate, in order to trivialize it and further remove its descendants. To support this operation, we allow the circuit to contain cycles of $\oplus$-type gates.

Definition 2.5 (Cyclic xor-circuit and semicircuit). A *cyclic xor-circuit* is a directed graph of $\oplus$-type gates and input variables, which may involve cycles. A cyclic xor-circuit may have arbitrarily many output gates. Furthermore, we define a *semicircuit* to be the composition of a cyclic xor-circuit C_1 and a circuit C_2 (i.e. the outputs of C_1 may feed C_2). $\diamondsuit$

A cyclic circuit is a reasonable computation model only if its behavior (i.e. the output of each gate) is uniquely determined given any inputs. Let C be a cyclic xor-circuit containing gates $\{G_1, G_2, \dots, G_m\}$, such that G_i is fed by I_i and J_i and computes a function $f_i \in \{\oplus, \equiv\}$. Note that I_i and J_i may be constants, variables or gates. This circuit corresponds to a system of linear equations $\{G_i = f_i(I_i, J_i)\}|_{i=1}^{m}$, where the input variables are viewed as constants. It can be represented as $\mathbf{Ag} = \mathbf{b}$, where $\mathbf{g} = [G_1, G_2, \dots, G_m]^\top$ and $\mathbf{b}$ consists of constants and variables. The cyclic xor-circuit C is said to be *fair* if $\mathbf{A}$ is of full rank, i.e. this system of linear equation has unique solution for any assignment of input variables. A semicircuit is *fair* if the cyclic xor-circuit is fair.

Proposition 2.6. Let C be a fair semicircuit and R be an rdq-source. A g-substitution (out of the four types in Section 2.4) to x_j that does not introduce cycles is valid. That is, it produces an rdq-source R' that further restrict R to the subset where $x_j = g(x_i, x_k, \dots, x_l)$. The circuit C' after substitution is still a fair semicircuit computing $f|_{R'}$. ♢

For simplicity, we abuse the notation *circuit* to mean *fair semicircuit* in the rest of the paper.

2.6 Substitution to gate: xor-reconstruction

We now introduce the operation for substituting constant values to gates, which is called *xor-reconstruction*. Let C be a circuit, and G be a topologically minimal $\wedge$-type gate[5] fed by I_1 and I_2, where I_1 is a gate. The subcircuit computing I_1 does not involve $\wedge$-type gates, so that it is a cyclic xor-circuit. We can trivialize G by substituting constant value of I_1 in the cyclic xor-circuit.

Proposition 2.7. Let I be a gate in a fair cyclic xor-circuit computing an affine function that depends on x_i, then there exists a path from x_i to I in the circuit. ♢

Proof. Let $\mathcal{X}$ be the set of gates unreachable from x_i. It is easy to verify that the subcircuit containing the gates in $\mathcal{X}$ is a fair semicircuit, whose inputs are variables except for x_i. In such case, the output of gates in $\mathcal{X}$ are uniquely determined by the assignment of variables except for x_i. Since I depends on x_i, we then conclude that $I \notin \mathcal{X}$. □

Let I be a gate in a cyclic xor-circuit that depends on x_i. By Proposition 2.7, there is a path from x_i to I, say

$$x = I_0 \to I_1 \to I_2 \to \cdots \to I_k = I.$$

Assume that I_j computes a linear function $f_j \in \{\oplus, \equiv\}$ of I_{j-1} and T_j, where $T_1, T_2, \dots, T_k$ are not necessarily distinct. The output of the gates on this path satisfies the following system of linear equations.

$$\begin{cases} I_1 = f_1(I_0, T_1) \\ I_2 = f_2(I_1, T_2) \\ \quad \cdots \\ I_k = f_k(I_{k-1}, T_k) \end{cases} \tag{1}$$

[5]If a circuit does not contain an $\wedge$-type gate, it computes an affine function of variables so that can be trivialized by a single affine substitution. Hence during the gate elimination procedure, we may assume that there exists an $\wedge$-type gate.

Since linear functions are invertible, we can rewrite the equation $I_j = f_j(I_{j-1}, T_j)$ as $I_{j-1} = g_j(I_j, T_j)$ for some linear function $g_j \in \{\oplus, \equiv\}$, so that the system of linear equations is as follow.

$$\begin{cases} I_{k-1} = g_k(I_k, T_k) \\ I_{k-2} = g_{k-1}(I_{k-1}, T_{k-1}) \\ \quad \cdots \\ \quad I_0 = g_1(I_1, T_1) \end{cases} \tag{2}$$

This system of linear equations naturally corresponds to a circuit where $I = I_k$ becomes an 'input' and $x_i = I_0$ becomes a gate. To substitute $I \leftarrow b$, we only need to replace the gate I_k with a constant b, replace the variable x_i with a gate Z, and rewire the circuit according to this system of linear equations (see Figure 7).

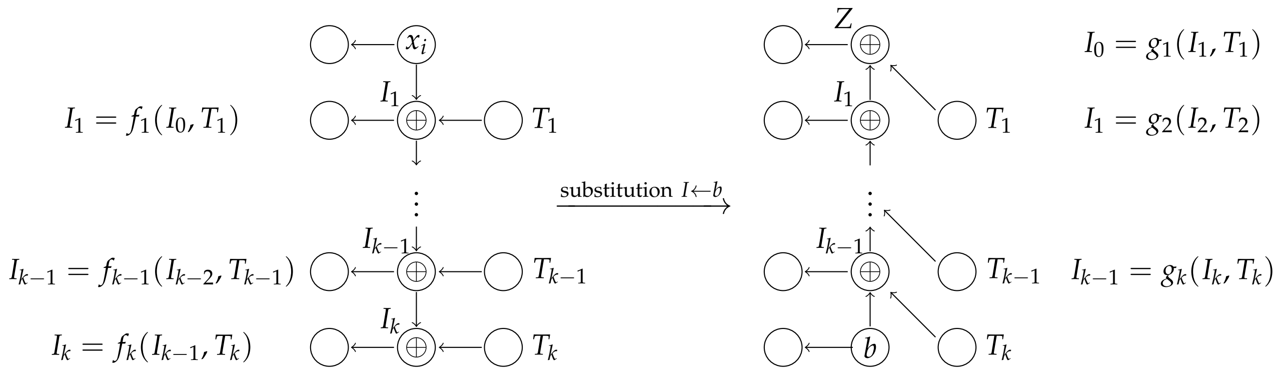

$I_1 = f_1(I_0, T_1)$
$I_{k-1} = f_{k-1}(I_{k-2}, T_{k-1})$
$I_k = f_k(I_{k-1}, T_k)$
x_i
I_1
T_1
I_{k-1}
T_{k-1}
I_k
T_k
substitution $I \leftarrow b$
Z
b
$I_0 = g_1(I_1, T_1)$
$I_1 = g_2(I_2, T_2)$
$I_{k-1} = g_k(I_k, T_k)$

Figure 7: Xor-reconstruction.

Proposition 2.8. Let I be a gate in a fair cyclic xor-circuit C computing f, the xor-reconstruction produce a fair xor-circuit C' computing $f|_{I=b}$. $\diamondsuit$

Proof. It is easy to see that the system of linear equation corresponding to C' is equivalent to that of C, while the only difference is that the output of I is fixed to be b. $\square$

We now formulate the xor-reconstruction for semicircuit. Let C be a circuit computing $f|_R$ for a function f and an rdq-source R. Let I be an $\oplus$-type gate in the cyclic xor-circuit involved in C.

Constant substitution to gate. If I depends on an unprotected variable x_i, we can substitute $I \leftarrow b$ for constant b. More precisely, we further restrict R to R' by making x_i a linear variable with proper affine equation, and perform xor-reconstruction to C to obtain C' computing $f|_{R'}$.

3 Troubled gates and normalization lemma

The main technical ingredient in Find, Golovnev, Hirsch, and Kulikov [FGHK16] allowing us to bypass the $3n$ barrier is to consider the introduction and destruction of the bottleneck structure, which is called *troubled gate* in their paper. Informally speaking, they argue that for each substitution, one may either eliminate $k > 3$ gates while introducing at most $4k$ troubled gates, or eliminate exactly 3 gates while destructing one troubled gate without forming any additional troubled gates. With this intuition they perform an *amortized analysis* to this gate elimination procedure, where the number of troubled gates is considered as the *potential* of the circuit.

We notice that their $4k$ upper bound for the introduction of troubled gates is not tight. We strengthen their analysis in two dimensions, which will be presented in this and the subsequent section.

1. We notice that adjacent troubled gates form a nicer structure, we called *troubled pairs*, whose eliminations are no longer troubled. This tells us that only those troubled gates not contained in pairs are *truly troubled*. From this intuition we may improve the $4k$ bound to $2k$, without diving into the case analysis.
2. The original $4k$ bound is obtained by counting the number of troubled gates introduced during the elimination of each gate *separately*. In their case analysis, however, we actually eliminate the gates from a compact local structure, so that we can perform a more fine-grained analysis. To capture the effect of local structure, we reformulate the normalization procedure (i.e. the procedure that eliminate gates), which will be used in a systematic reexamination of the case analysis in [FGHK16] later.

3.1 Troubled gates, troubled pairs and potential

Definition 3.1 (Troubled gate). A gate G is called *troubled* if it is an $\wedge$-type 1-gate fed by two 2-variables (see Figure 8). Two troubled gates are called *adjacent* if they share at least one common input variable. $\diamondsuit$

Such a structure is troubled since there is no obvious way of eliminating more than three gates per substitution to the variables. For example, by substituting proper constant value to x in Figure 8, we can trivialize G while making the descendants of x and G degenerate, but still eliminating only three gates.

Definition 3.2 (Troubled pair). A *troubled pair* (G, G') is a pair of adjacent troubled gates (see Figure 9). The input variables that feed both G and G' are called *inner variables*, and the variables that feed one of G and G' are called *boundary variables*. $\diamondsuit$

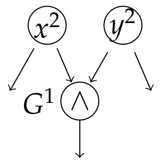


Figure 8: A troubled gate G.

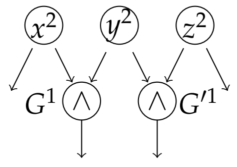


Figure 9: A troubled pair (G, G').

Note that it might be the case that $x = z$ in Figure 9, i.e. a troubled pair (G, G') may contain two inner variables and no boundary variable. Such a troubled pair without boundary variable is said to be *compact*.

Later analysis in Section 3.4 will show that a troubled pair is actually a nice structure for gate elimination. If we maintain a disjoint set of troubled pairs, only troubled gates that are not contained in it will bother us. This motivates us to define the notion of *packing*.

Definition 3.3 (Packing). Let C be a circuit and $\mathcal{T}$ be the set of all troubled pairs. A subset of disjoint troubled pairs $\mathcal{P} \subseteq \mathcal{T}$ is called a *packing* of C, and each troubled pair $(G, G') \in \mathcal{P}$ is called a *pack*. By disjoint, we mean that each gate can be contained in at most one pack. A packing is called *maximal* if there is no packing containing it. $\diamondsuit$

Definition 3.4 (Truly troubled gate). Let C be a circuit and $\mathcal{P}$ be a packing. A gate is said to be *truly troubled* if it is a troubled gate that does not contained in a pack. $\diamondsuit$

Similar to Find, Golovnev, Hirsch, and Kulikov [FGHK16], we will analyze the introduction and destruction of truly troubled gates during gate elimination. However, a truly troubled gate can be introduced either directly or by unpacking a pack, and the latter case usually leads to tedious case study. For technical convenience, we will analyze the change of *potential* defined as follow.

Definition 3.5 (Potential). Let C be a circuit and $\mathcal{P}$ be a packing. We define the potential $\Phi(C, \mathcal{P})$ as the number of truly troubled gates plus $|\mathcal{P}|$, or equivalent, as the number of troubled gates minus $|\mathcal{P}|$. $\diamondsuit$

If G is a truly troubled gate introduced by unpacking $(G, G') \in \mathcal{P}$, its couple G' will not be troubled, unless the inner variables do not feed G' any more. We will show it later that the special case cannot happen in our proof. In such case, the potential increment can only be caused by directly introduced truly troubled gates or packs.

3.2 Circuit complexity measure

Recall that to perform an amortized analysis to the number of gates eliminated, we define the *circuit complexity measure* as a linear combination of the number of gates and bottlenecks.

Definition 3.6 (Circuit complexity measure). Let C be a circuit, $\mathcal{P}$ be a packing and R be an rdq-source. The circuit complexity measure $\mu(C, \mathcal{P}, R)$ is defined as

$$\mu(C, \mathcal{P}, R) \triangleq g + \alpha_I \cdot i + \alpha_Q \cdot q + \alpha_\phi \cdot \Phi(C, \mathcal{P}),$$

where g is the number of gates, i is the number of *influential* inputs and q is the number of quadratic equations. An input is called *influential* if it is either a 1^+-variable or protected. Note that α_I, α_Q and α_ϕ are positive constants that will be chosen later. $\diamondsuit$

We will analyze the complexity measure decrement during gate elimination procedure to obtain a complexity measure lower bound, which will be translated to a circuit size lower bound by an upper bound of the potential. This is possible since we have the following crucial observation.

Lemma 3.7 (Lemma 3 in [FGHK16]). Let C be a circuit computing an affine disperser for dimension d, then the number of troubled gates is at most $\frac{n}{2} + \frac{5d}{2}$. $\diamondsuit$

Corollary 3.8. Let C be a circuit computing an affine disperser for dimension d. For all packing $\mathcal{P}$ of C, $\Phi(C, \mathcal{P}) \le \frac{n}{2} + \frac{5d}{2}$. ♢

Proof. For any packing $\mathcal{P}$ of C, the potential $\Phi(C, \mathcal{P})$ equals to the number of troubled gates minus $|\mathcal{P}|$. Since the number of troubled gates is at most $\frac{n}{2} + \frac{5d}{2}$ by Lemma 3.7, we also have $\Phi(C, \mathcal{P}) \le \frac{n}{2} + \frac{5d}{2}$. □

With Corollary 3.8 and Definition 3.6, we immediately get the following theorem.

Theorem 3.9. Assume that computing a function $f : \mathbb{F}_2^n \to \mathbb{F}_2$ requires complexity $\mu(C, \mathcal{P}, \varnothing) \ge \mu_0$ for any circuit C and packing $\mathcal{P}$, we must have $|C| \ge \mu_0 - \alpha_I \cdot n - \alpha_\phi(\frac{n}{2} + d)$. Furthermore, if f is an affine disperser for sublinear dimension and $\mu_0 = \delta n - o(n)$, we have

$$|C| \ge \left(\delta - \alpha_I - \frac{\alpha_\phi}{2}\right) n - o(n).$$ ♢

3.3 Normalization of circuits

After substitutions are applied to the circuit, some gates may become obviously redundant (for example, a gate fed by a constant becomes trivial or degenerate). These gates will be eliminated by applying the following five normalization rules. A circuit is called *normalized* if no normalization rule can apply to it. Also, we will carefully analyze the complexity measure decrement for each normalization. Based on this analysis, we will refine the case study in the original $(3 + \frac{1}{86})n$ proof, which leads to our new $3.1n$ circuit lower bounds.

Recall that the potential may increase by direct introduction of packs and troubled gates, or by destruction of a pack (G_1, G_2) such that they become non-adjacent troubled gates. We will show in Proposition 3.10 that the latter case is impossible. So in the following analysis we only consider potential increment caused by direct introduction of packs and troubled gates.

Rule 1. Let G be a 0-gate fed by I_1 and I_2. Clearly, G can be removed, and the out-degree of I_1 and I_2 are decreased by one. Newly introduced troubled gate can only be I_1, I_2, or the gate fed them. Since troubled gates are fed by 2-variables, I_1 (or I_2) can feed at most two troubled gates. If there are two newly introduced troubled gates fed by I_1 (or I_2), we pack them together. In such case, each input of G can produce a troubled gate or a pack, hence $\Delta\Phi \le 2$, which means that $\Delta\mu \ge 1 - 2\alpha_\phi$.

Rule 2. Let G be a trivialized gate fed by a constant b[6], it can be removed and its descendants become fed by a constant c instead (see Figure 10). If the other input I_1 of G is not a constant, its out-degree is decreased by one and a troubled gate may be introduced. Clearly a newly introduced troubled gate is either I_1 or a gate fed by I_1, so that similar to Rule 1, we can see that $\Delta\Phi \le 1$. Hence $\Delta\mu \ge 1 - \alpha_\phi$.

[6]Rigorously speaking, G is not a trivial gate under our definition, since the function on it may still be non-trivial. Here what we actually mean is that G becomes trivial after we substitute b into the function on G. In the rest of the paper, we will intensively use the similar abuse of notation for a cleaner argument.

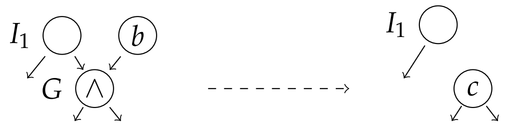


Figure 10: Rule 2.

Rule 3. Let G be a degenerate gate fed by a constant b[7] and another node I_1. Then we can always eliminate G and rewire the circuit in some way, such that the descendants of G becomes directly fed by I_1 (see Figure 11). If G computes the negation of I_1, the functions of the descendants should be modified accordingly. In particular, if G is the output gate, we make I_1 the output gate and change the function computed by I_1 and its descendants properly. Similar to Rule 2, the gates that may become trouble are either I_1 or fed by I_1 after the normalization, hence $\Delta\Phi \leq 1$ and $\Delta\mu \geq 1 - \alpha_\phi$.

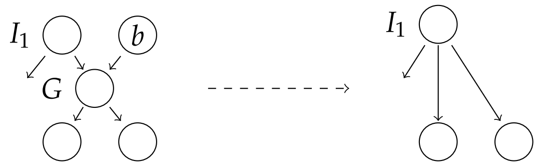


Figure 11: Rule 3.

Rule 4. A gate is called *useless* if it is a 1-gate with only descendant Q, such that another input of Q also feeds G. We may eliminate G and rewire the nodes feeding it to feed Q (see Figure 12). Now we show that $\Delta\Phi \leq 1$. If I_2 is a gate, it is the only gate that may become trouble; otherwise if it is a variable, newly introduced troubled gates must be fed by it, forming a troubled gate or a pack. Hence $\Delta\mu \geq 1 - \alpha_T$.

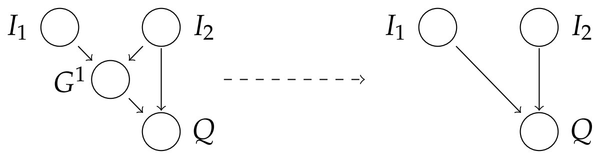


Figure 12: Rule 4.

Rule 5. Let G be a gate whose inputs coincide[8], we make it degenerate (or trivial) by changing one (or both) of its input to a constant, and eliminate it via Rule 2 or Rule 3. By the analysis above, $\Delta\Phi \leq 1$ and $\Delta\mu \geq 1 - \alpha_\phi$.

Proposition 3.10. If a pack (G_1, G_2) is destructed after applying a normalization rule, at least one of them becomes non-troubled, hence potential increment can only be caused by direct introduction of troubled gates or packs. ♢

[7]Note that Rule 2 and Rule 3 only eliminate trivialized or degenerate gates that are fed by a constant. Most trivial and degenerate gates in our proof are eliminated in such fashion, except for Case 0 and some cases using affine or quadratic substitutions.

[8]Note that this rule is necessary, since Rule 3 may introduce such gates.

Proof. Towards a contradiction, we assume that (G_1, G_2) is destructed such that G_1 and G_2 remain troubled. Clearly the inner variable of the pair must change its descendants, since otherwise this pack is not necessarily destructed. The normalization rules, however, only change the descendants of the gates that feeds a gate to be eliminated. Hence at least one of G_1 and G_2 is eliminated, which leads to contradiction. □

We summarize the results above in the following lemma.

Lemma 3.11 (Normalization lemma). Let β be the complexity measure decrement for each normalization rule. For Rule 1 we have $\beta \geq 1 - 2\alpha_\phi$, and for other rules we have $\beta \geq 1 - \alpha_\phi$. Moreover, suppose that the eliminated gate G is fed by I_1 and I_2, the potential increment is caused only by direct introduction of troubled gates or packs, which are either I_1 (or I_2) or fed by them after the normalization. ♢

Note that α_ϕ will be chosen to be smaller than $\frac{1}{2}$, so that $\beta \geq 0$. Since normalization will decrease the complexity measure, it will not bother us to assume that the circuit is normalized during gate elimination procedure.

3.4 Troubled pair elimination

Now we show that troubled pairs are actually not troubled. That is, we can remove them by appropriate substitutions, such that the complexity measure decrement per substitution is sufficiently large. We will call this operation *troubled pair elimination* in the rest of the paper.

Lemma 3.12 (Troubled pair elimination). Let C be a circuit computing $f|_R$ for function f and rdq-source R. Let $\mathcal{P}$ be a non-empty packing of C. Assume that f is not trivial under two substitutions. Then it is possible to perform substitutions such that complexity measure decrement per substitution is at least

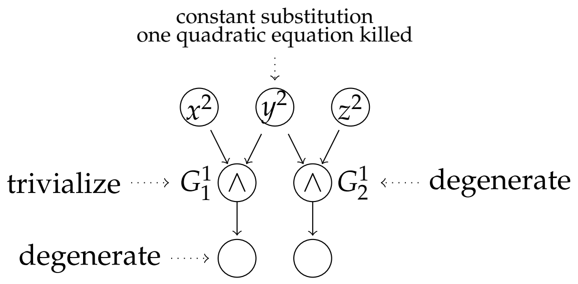
$$\delta_p = \min\left\{\frac{3}{2}\alpha_I, 3 - 3\alpha_\phi + \alpha_I + \alpha_Q\right\}.$$

♢

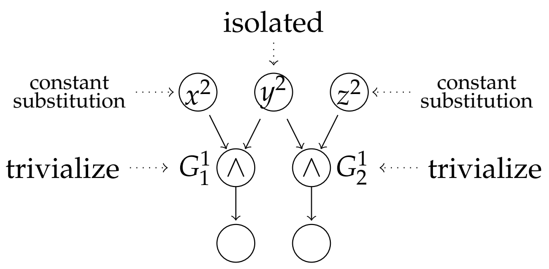
Proof. Let $(G_1, G_2) \in \mathcal{P}$ be a pack. We first consider the case that (G_1, G_2) is not compact (see Figure 13). Let y be the inner variable and $\{x, z\}$ be the boundary variables.

1. If y is protected, we perform appropriate constant substitution to y such that G_1 is trivialized. Also, G_2 and the only descendants of G_1 are degenerate. We then normalize the circuit so that three gates are eliminated and $\Delta\Phi \leq 3$. Since we substitute constant value to a protected variable, a quadratic equation and an influential input are killed, decreasing the total complexity measure by $\Delta\mu \geq 3 - 3\alpha_\phi + \alpha_I + \alpha_Q$.
2. If y is unprotected, we perform appropriate constant substitution to x and z such that both G_1 and G_2 are trivialized (i.e. they have a fixed output independent of the assignments of variables). In such case, all three variables become non-influential, which leads to a total complexity measure decrement $\Delta\mu \geq 3\alpha_I$.

Now we assume that (G_1, G_2) is a compact pack fed by x and y (see Figure 14).



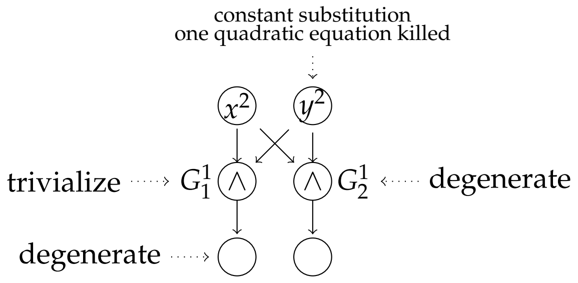
(a) Case when y is protected



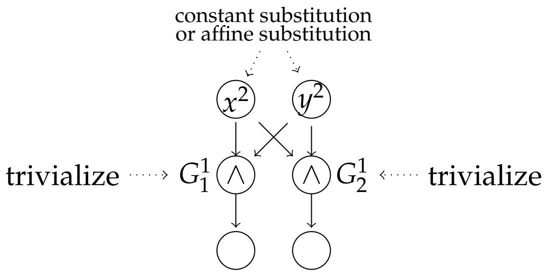
(b) Case when y is unprotected

Figure 13: Troubled pair elimination: incompact case.

1. If one of x and y is protected we may perform constant substitution (like the first case above) such that $\Delta\mu \geq 3 - 3\alpha_\phi + \alpha_I + \alpha_Q$.
2. Otherwise, both x and y are unprotected. By enumerating the functions computed by G_1 and G_2, one can verify that either a constant substitution (to x or y) or an affine substitution $y \leftarrow x \oplus b$ will trivialize both gates. We then replace G_1 and G_2 by constants such that x and y becomes non-influential. Hence we decrease the complexity measure $\Delta\mu \geq 2\alpha_I > \frac{3}{2}\alpha_I$. ☐



(a) Case when y is protected.



(b) Case when both x and y are unprotected.

Figure 14: Troubled pair elimination: compact case.

Note that the bounds in Lemma 3.12 are not tight. However, we will later show that this is not the bottleneck of our proof, hence improving it is not essential.

4 Proof of $3.1n - o(n)$ circuit lower bound

In this section we present the proof of $3.1n - o(n)$ circuit lower bound for affine dispersers. For short, we strengthen the analysis of $(3 + \frac{1}{86})n - o(n)$ lower bound in [FGHK16] by improving the upper bounds of the number of troubled gates introduced, with the help of Lemma 3.11 and Lemma 3.12. We can then prove a $3.1n - o(n)$ circuit lower bound by choosing optimal parameters.

Theorem 4.1. Let C be a circuit computing an affine disperser f for dimension d. For all packing $\mathcal{P}$ of C, we have $\mu(C, \mathcal{P}, \varnothing) \geq \delta(n - 2d - 2)$, where

$$\delta \triangleq \alpha_I + \min\left\{\frac{\alpha_I}{3}, 2 - 2\alpha_\phi + \alpha_Q, 4 - 4\alpha_\phi, 3 + \alpha_\phi, 5 - \alpha_Q, \frac{5 - 2\alpha_\phi + \alpha_Q}{2}\right\}.$$ ♢

The proof of Theorem 4.1 requires tedious case analysis, which will be discussed in Section 4.1. Intuitively, it means that we can perform substitutions to the circuit such that it is simplified, by (1) making variables non-influential (i.e. $\frac{\alpha_I}{3}$), (2) eliminating many gates (i.e. 4β and $5 - \alpha_Q$), or (3) eliminating gates and removing quadratic equations (i.e. $2 - 2\alpha_\phi + \alpha_Q$ and $\frac{1}{2}(5 - 2\alpha_\phi + \alpha_Q)$).

Remainder of Theorem 1.1 (Main Theorem). Assume that C is a B_2 circuit computing an affine disperser for sublinear dimension, then C has size at least $3.1n - o(n)$. ♢

Proof. We strengthen the theorem to prove lower bounds for semicircuits instead of B_2 (acyclic) circuits. Let C be a circuit computing an affine disperser f for sublinear dimension. By Theorem 3.9 and 4.1, we have $|C| \geq \left(\delta - \frac{\alpha_\phi}{2} - \alpha_I\right) n - o(n)$. Now we need to determine $\alpha_\phi, \alpha_Q, \alpha_I \geq 0$. To choose optimal parameters, we solve the following linear program.

$$\begin{aligned}
\max.\ & \delta - \frac{\alpha_\phi}{2} - \alpha_I \\
\text{s.t.}\ & \delta \leq \alpha_I + \min\left\{\frac{\alpha_I}{3}, 2 - 2\alpha_\phi + \alpha_Q, 4 - 4\alpha_\phi, 3 + \alpha_\phi, 5 - \alpha_Q, \frac{5 - 2\alpha_\phi + \alpha_Q}{2}\right\} \\
& \alpha_\phi, \alpha_Q, \alpha_I \geq 0, \\
& \alpha_\phi \leq \frac{1}{2}.
\end{aligned}$$

The optimal solution is $\alpha_\phi = 0.2, \alpha_I = 9.6, \alpha_Q = 1.8$ and $\delta = 12.8$, which gives

$$|C| \geq \left(\delta - \frac{\alpha_\phi}{2} - \alpha_I\right) n - o(n) = 3.1n - o(n).$$

□

4.1 Proof of Theorem 4.1

Remainder of Theorem 4.1. Let C be a circuit computing an affine disperser f for dimension d. For all packing $\mathcal{P}$ of C, we have $\mu(C, \mathcal{P}, \varnothing) \geq \delta(n - 2d - 2)$, where

$$\delta \triangleq \alpha_I + \min\left\{\frac{\alpha_I}{3}, 2 - 2\alpha_\phi + \alpha_Q, 4 - 4\alpha_\phi, 3 + \alpha_\phi, 5 - \alpha_Q, \frac{5 - 2\alpha_\phi + \alpha_Q}{2}\right\}.$$

♢

Proof. By Lemma 2.3, the affine disperser f is not trivial under $n - 2d$ substitutions (out of the four types in Section 2.4 and xor-reconstruction in Section 2.6). Let R be the rdq-source that will be updated by substitutions during gate elimination. To prove the theorem, it is sufficient to show that we can perform at least $n - 2d - 2$ substitutions to the circuit such that the complexity decrement per substitution is at least δ.

Now we prove by case analysis that if $k < n - 2d - 2$ substitutions has been applied to the circuit such that the resulting circuit C computes $f|_R$ with packing $\mathcal{P}$, either it is possible to simplify the circuit (i.e. decrease the complexity measure) without performing substitution, or it is possible to perform $1 \leq t \leq 3$ substitutions with complexity measure decrement $\geq \delta$ per substitution, while the simplified circuit C' remains to compute f restricting to the updated rdq-source. Note the the function is not trivial after performing these substitutions, so the output gate of the circuit will not be trivialized.

Case 0. Now we firstly simplify the circuit such that it does not have obvious redundancy.

Case 0.1. If the circuit is not normalized, we normalize it while $\Delta\mu \geq 0$.

Case 0.2. Assume that there is a gate in the *acyclic part* of the circuit, such that the output gate is not reachable from it. We find a topologically maximal one G. It is easy to see that G is a 0-gate. We can then remove it by normalization Rule 1, with complexity decrement $\Delta\mu \geq 1 - 2\alpha_\phi \geq 0$.

Case 0.3. If there is a gate G outputing a fixed constant c for any assignment to the variables, we directly replace it by c. Similar to the normalization rules, replacing a gate by a constant will increase the potential by at most two (one for each of the inputs of G). Hence a gate is removed while $\Delta\Phi \leq 2$, which means that $\Delta\mu \geq 1 - 2\alpha_\phi \geq 0$.

Similarly, if there is a gate G fed by I_1 and I_2 that computes a function that only depends on I_1, we rewire the circuit such that G is fed by I_1 and a constant 0 (this operation increase the potential by at most one, like the normalization rules). We can then make G degenerate by changing its function properly, so that G can be removed by Rule 3. A gate is removed while $\Delta\Phi \leq 2$, which means that $\Delta\mu \geq 1 - 2\alpha_\phi \geq 0$.

Case 0.4. The circuit has a non-empty packing, then by Lemma 3.12, we can perform substitutions such that $\Delta\mu \geq \delta_p \geq \delta$ per substitution.

From now on, the circuit is normalized with an empty packing, and the output of each gate depends on both of its inputs.

Case 1. There is a protected variable feeding two gates or an $\wedge$-type gate. We substitute an appropriate constant to it, so that a quadratic equation and a free variable is killed, while eliminating at least two gates via Rule 2 and Rule 3. This will decrease the measure by $\Delta\mu \geq 2 - 2\alpha_\phi + \alpha_I + \alpha_Q \geq \delta$, which is enough for our argument.

Case 2. There is a protected 0-variable p. Suppose that its couple in the quadratic equation is q. Then we perform constant substitution to q, making the quadratic equation an affine one. Since both p and q becomes non-influential, we have $\Delta\mu \geq 2\alpha_I \geq \delta$.

From now on, all protected variables are 1*-variables each feeding an* $\oplus$*-type gate.*

Case 3. There is a variable x feeding an $\wedge$-type gate G, such that the total out-degree of x and G is at least 4. We perform appropriate constant substitution to x to trivialize G, while at least 3 descendants of x and G are degenerate. If a gate is fed by both G and x (i.e. double counted), it is then trivialized and its descendants are degenerated. To sum up, at least four gates are eliminated by Rule 2 and Rule 3, so $\Delta\mu \geq 4 - 4\alpha_\phi + \alpha_I \geq \delta$.

After this case, all variables feeding $\wedge$*-type gates have out-degree* 1 *or* 2.

Case 4. There is an $\wedge$-type gate G fed by two variables x and y, and x is a 1-variable. Note that both x and y are unprotected by Case 1. We perform appropriate constant substitution to y such that G is trivialized, making both x and y non-influential. Hence $\Delta\mu \geq 2\alpha_I \geq \delta$.

Case 5. There is an $\wedge$-type gate G fed by two 2-variable x and y. By Case 3, G must be a 1-gate, and hence troubled (see Figure 15). Since the circuit is normalized we know that $B \neq D$ and $C \neq D$. From Case 1, we know that both x and y are unprotected free variables. Clearly, substituting appropriate constant to x (or y) will trivialize G, further removing B (or C), D

and G by normalization. If no troubled gates are introduced during eliminating these three gates, we will obtain $\Delta\mu = 3 + \alpha_\phi + \alpha_I \geq \delta$.

For the rest part of this case, we assume that at least one troubled gate is introduced when we substitute a constant to x (or y) and eliminate the three gates in arbitrary order.

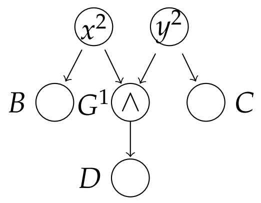


Figure 15: Case 5.

Case 5.1. When $B = C$, we can always apply a constant substitution to x or y, or an affine substitution $x \leftarrow y \oplus c$, such that both G and B outputs a fixed constant regardless of the inputs[9]. We can replace them by constants, making both x and y non-influential. Hence $\Delta\mu \geq 2\alpha_I \geq \delta$.

Case 5.2. When D feeds both B and C, we perform constant substitution to x to trivialize G, and eliminate G, D and B in order. By normalization lemma, the only node that may become or feed a troubled gate is the other input I of D. Furthermore, if I is a gate, then by the observation that its out-degree is not decreased during this process, it cannot *become* troubled. Hence to introduce a troubled gate, I must be a variable u and B must feed another $\wedge$-type gate E (see Figure 16).

Note that $C \neq E$ since E is fed by B but C is not. After the normalization, u should be a 2-variable feeding exactly C and E, hence B should be a 1-gate originally. Instead of substituting to x, we substitute appropriate constant to z and y so that both E and G are trivialized, then eliminate B by Rule 1. These two substitutions kill three influential inputs x, y and z, hence $\Delta\mu \geq \frac{3}{2}\alpha_I \geq \delta$ per substitution.

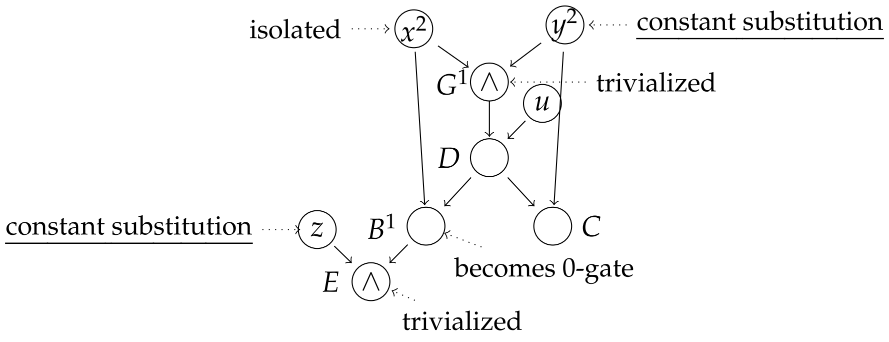


Figure 16: Case 5.2.

After this subcase, we can assume that D does not feed one of B or C.

Case 5.3. If C feeds D and D feeds B (see Figure 17), after substituting constant to x, we can eliminate G, D and B in order. By normalization lemma, the only nodes that may become

[9]This can be checked by enumerating over all possible truth tables.

or feed troubled gates are y and c, while none of them can be like that. Hence no troubled gate is introduce, contradicting with our assumption.

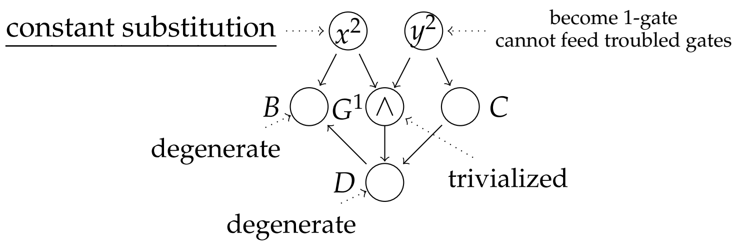


Figure 17: Case 5.3.

After this subcase, we can assume that D is disconnected (in both directions) with one of B and C. By symmetry, we further assume that D is disconnected with B.

Case 5.4. Suppose that B is fed by x and I_1, and D is fed by G and I_2. We now substitute constant value to x to trivialize G. By normalization lemma, a newly introduced troubled gate is either one of I_1 and I_2, or fed by I_1 or I_2 after the normalization. Since B and D are disconnected, the out-degree of I_1 and I_2 are not decreased, hence they cannot be newly introduced troubled gates unless their inputs are involved in this normalization. We then conclude that a newly introduced troubled gate must be fed by I_1 or I_2 after the normalization.

Case 5.4.1. Assume that the troubled gate E is fed by $I_2 = z$. Clearly E must be an $\wedge$-type 1-gate. Since B and D are disconnected, D must feed E before the normalization (see Figure 18).

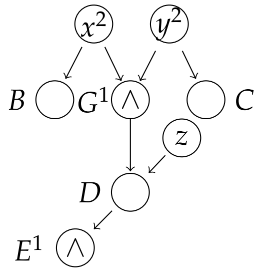


Figure 18: Case 5.4.1.

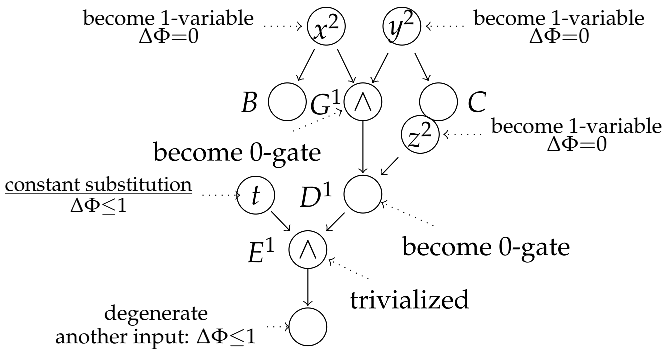
Case 5.4.1.1. If z is a 2-variable, D must be a 1-gate to make E troubled. In this case, the other input of E is either a variable or B, which passes a variable to it after x is substituted by constant.

Case 5.4.1.1.1. If the other input of E is a variable t (see Figure 19), we substitute a constant to it so that E is trivialized, and eliminate E, D, G and the descendant of E in order. By normalization lemma, one may easy to verify that $\Delta\Phi \leq 1$ (which corresponds to the other input of the descendant of E). Hence $\Delta\mu \geq 4 - \alpha_\phi + \alpha_I \geq \delta$.

Now we assume that other input of E is B, which is fed by x and another variable t (see Figure 20). The variable t is passed to E while degenerating B. Notice that x must be unprotected by Case 1.

Case 5.4.1.1.2. If B is an $\wedge$-type gate, then we can substitute constant values to t and y to trivialize B and G, making x a 0-variable. Hence $\Delta\mu \geq \frac{3}{2}\alpha_I \geq \delta$.

Case 5.4.1.1.3. If B is an $\oplus$-type gate, then we can substitute $x \leftarrow t \oplus c$ such that the output of B is fixed regardless of the inputs. This allows us to replace B by a constant. If c is chosen appropriately, we can even trivialize E, further removing D and G by Rule 1, and degenerate the descendent of E by Rule 3. Notice that only t and the other input of the descendent of E can introduce troubled gates (y and z will not since they are 2-variable), hence we have $\Delta\mu \geq \alpha_I + 5 - 2\alpha_\phi \geq \delta$.



become 1-variable $\Delta\Phi$=0
x^2
y^2
become 1-variable $\Delta\Phi$=0
B
G^1
$\wedge$
C
z^2
become 1-variable $\Delta\Phi$=0
become 0-gate
constant substitution $\Delta\Phi\leq 1$
t
D^1
E^1
$\wedge$
become 0-gate
trivialized
degenerate another input: $\Delta\Phi\leq 1$

Figure 19: Case 5.4.1.1.1.

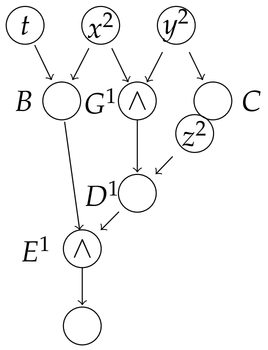


Figure 20: Case 5.4.1.1.2 and 5.4.1.1.3.

In the rest of Case 5.4.1, we can assume that z is a 1-variable. To make E troubled, D must be a 2-gate (see Figure 21).

Case 5.4.1.2. If D is an $\wedge$-type gate, we substitute constant value to z to trivialize D, and eliminate D, G and the two descendants of D in order. Since the elimination of G will not cause potential increment and the other gates are removed by Rule 2 and Rule 3, we can see that $\Delta\Phi \leq 3$, hence $\Delta\mu \geq 4 - 3\alpha_\phi + \alpha_I \geq \delta$.

Case 5.4.1.3. In the case that z is protected, after constant substitution to x, trivializing G and then degenerating B and D, we can further substitute a constant to z such that E is trivialized, which will degenerate the descendant of E and z (which was passed from D). During the whole process, 6 gates are removed while introducing at most 6 troubled gates since all gates are removed via Rule 2 and Rule 3. A quadratic equation involving z is also removed. So in a nutshell,

$$\Delta\mu \geq \frac{6 - 6\alpha_\phi + 2\alpha_I + \alpha_Q}{2} = \frac{2 - 2\alpha_\phi + \alpha_I + \alpha_Q}{2} + \frac{4 - 4\alpha_\phi + \alpha_I}{2} \geq \delta$$

per substitution.

Case 5.4.1.4. Assume that z is unprotected and D is an $\oplus$-type gate (see Figure 22). Since x and y feed an $\wedge$-type gate G, they should also be unprotected (see Case 1). In this case, we can perform a quadratic substitution $z \leftarrow ((x \oplus c_1) \wedge (y \oplus c_2)) \oplus c_3$ for appropriate constants, such that the output of D is a constant b insensitive to the inputs, which trivializes E. Then we normalize the circuit in the following order.

1. Replace D by the constant b, which would not increase the potential.
2. Eliminate G via Rule 1, which would not increase the potential either, since both x and y become 1-variable.

3. Eliminate E via Rule 2, which again introduces no potential increment. This is because, similar to Case 5.4.1.1, the other input of E is either B or a variable t. In the former case, B is fed by 1-variable x, hence cannot be troubled. In the latter case, t itself becomes a 1-variable after eliminating E, introducing no troubled gates.
4. Eliminate the descendent of D via Rule 3. We will argue at the end of this case that it can be ensured that this does not increase the potential.
5. Eliminate the descendent of E via Rule 3, increasing the potential by at most 1.

Moreover, this case will destruct an originally (truly) troubled gate G, which reduces the potential by 1.

So in a nutshell, we can eliminate 5 gates with the net potential change $\Delta\Phi \leq 0$, which guarantees complexity measure decrement $\Delta\mu \geq 5 - \alpha_Q + \alpha_I \geq \delta$.

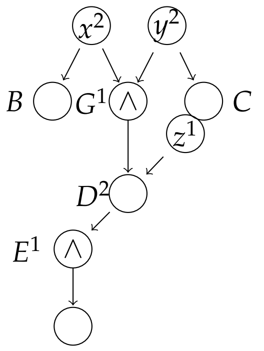


Figure 21: Case 5.4.1.2 and 5.4.1.3.

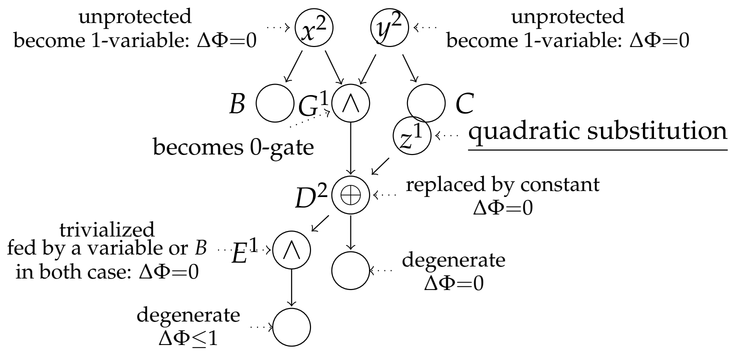
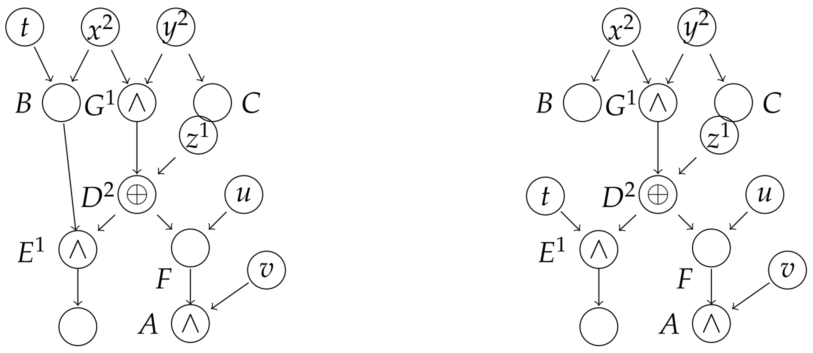


Figure 22: Case 5.4.1.4.



(a) Case 5.4.1.4.1. (b) Case 5.4.1.4.2.

Figure 23: Hard cases of Case 5.4.1.4.

To finish, we now rule out all the cases that the elimination of the descendent of D would introduce a troubled gate.

As shown in Figure 23, we label the descendent of D as F, and the troubled gated introduced in the elimination of F as A. Since replacing D by constant and eliminating G, E will not *pass* any node to another, F must be directly fed by some variable u, and A must be directly fed by D and a 2^+-variable v.

Similarly to Case 5.4.1.1, by our very first assumption that E will become troubled if we perform constant substitution to x, there is a variable t either directly feeds E or passes to E while degenerating B. We now study these cases separately.

Case 5.4.1.4.1. Firstly, we consider the cases when t feeds B and B feeds E (See Figure 23a).

Case 5.4.1.4.1.1. If B is a 1-gate, then we can substitute appropriate constants to y and z such that G, D and E are all trivialized. Then the out-degree of B will be reduced to 0, hence can be removed by Rule 1. All of these will make x a 0-variable, so we make 3 variables non-influential by 2 substitutions, resulting in $\Delta\mu \geq \frac{3}{2}\alpha_I \geq \delta$.

For the rest of Case 5.4.1.4.1, we can assume that B is a 2-gate, so t is a 1-variable (since t becomes troubled by constant substitution to x).

Case 5.4.1.4.1.2. If B is an $\wedge$-type gate, then t is unprotected (see Case 1). In this case, we can trivialize B by constant substitution to x, making t non-influential. This can give us $\Delta\mu \geq 2\alpha_I \geq \delta$.

Case 5.4.1.4.1.3. When B is an $\oplus$-type gate and t is unprotected, we can first trivialize G by constant substitution to y, making x a 1-variable, then trivialize B by affine substitution $x \leftarrow t$. These 2 substitutions make 3 variables (x, y and t) non-influential, hence $\Delta\mu \geq \frac{3}{2}\alpha_I \geq \delta$.

Case 5.4.1.4.1.4. Assume that B is an $\oplus$-type gate and t is protected. Note that the couple of t cannot be x or y since they are unprotected. So we can first perform a constant substitution to its couple to make t unprotected, then apply Case 5.4.1.4.1.3. This will allow us to make 4 varialbes non-influential by 3 substitutions, hence $\Delta\mu \geq \frac{4}{3}\alpha_I \geq \delta$.

Case 5.4.1.4.2. Now we consider the cases when t directly feeds E (See Figure 23b).

After the quadratic substitution of Case 5.4.1.4, A is a troubled gate fed by u and v, but x and y are 1^--variables, hence u, v and x, y are pairwise unequal. This ensures that F is neither B nor C. Moreover, z is a 1-variable feeding D, so u and v will never coincide with z.

Case 5.4.1.4.2.1. Assume that $u = t$ (see Figure 24). Then u is a 2-variable feeding E and F, hence unprotected (see Case 1). In this case, we can first perform the constant substitution to x to trivialize G, then remove B and D by Rule 3. After this, the gates E and F are exactly fed by two 2-variables z and u, hence we can perform a constant substitution to z or u, or an affine substitution $u \leftarrow z \oplus c$ for appropriate c, such that the outputs of both E and F become independent of the inputs. This allows us to make 3 variables (x, z and u) non-influential with 2 substitutions, resulting in $\Delta\mu \geq \frac{3}{2}\alpha_I \geq \delta$.

For the rest of Case 5.4.1.4.2, we can assume that $u \neq t$.

Case 5.4.1.4.2.2. When F is a 1-gate, we can do a constant substitution to v such that we are able to replace A by a constant. This would make F a 0-gate, so we can remove it by Rule 1. Furthermore, a descendent of A can be removed via Rule 3.

Recall that v is a 2^+ variable since A will become troubled under quadratic

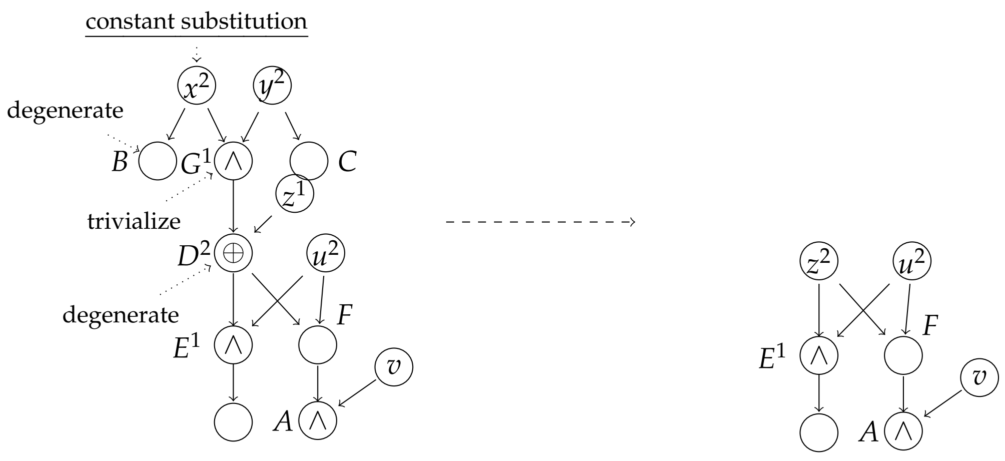


Figure 24: Case 5.4.1.4.2.1.

substitution in Case 5.4.1.4. Therefore, there is another descendent of v that can be removed via Rule 3 (again, if this coincides with the descendent of A, then it would become trivial, further degenerating a descendent).

The trivialization of A does not introduce any potential, since v is a constant by then, and F is fed by a $\oplus$-gate D. The removal of F introduces no potential either, since D is an $\oplus$-type gate, and u becomes a 1-variable. Each of the elimination of the descendants of A and v can introduce 1 potential by Rule 2 and Rule 3. Hence during the process, 4 gates are eliminated with $\Delta\Phi \leq 2$, resulting in $\Delta\mu \geq 4 - 2\alpha_\phi + \alpha_I \geq \delta$.

We can now assume that F is a 2^+-gate originally. Notice that F can feed none of G, D or E, so to make A troubled, F is exactly a 2-gate, and u is a 1-variable initially.

Case 5.4.1.4.2.3. Assume that F is an $\wedge$-type gate. Since $u \neq t$, we can substitutes constants to u and t to trivialize E and F. After this, D will become a 0-gate, and by removing it via Rule 1, we can make z a 0-gate, and hence non-influential (recall that z being unprotected is the assumption of Case 5.4.1.4). This allows us to make 3 variables non-influential by 2 substitutions, giving us $\Delta\mu \geq \frac{3}{2}\alpha_I \geq \delta$.

Case 5.4.1.4.2.4. Assume that F is an $\oplus$-type gate and u is unprotected. Then we can first substitute an appropriate constant to t to trivialize E. This would make D a 1-gate. Now notice that $F = D \oplus u \oplus c_1 = G \oplus z \oplus u \oplus c_2$, so by performing affine substitution $z \leftarrow u \oplus c$ for arbitrary constant c, we can make $F = G \oplus c \oplus c_2$. Therefore, we can rewire the circuit (i.e. make all descendants of F fed by G directly) such that the gates D and F are removed, and z and u become 0-variables. In this way, we make 3 variables non-influential by 2 substitutions, hence $\Delta\mu \geq \frac{3}{2}\alpha_I \geq \delta$.

Case 5.4.1.4.2.5. If F is an $\oplus$-type gate and u is protected, then the couple of u cannot be x, y, z, t or v since none of them is protected. So we can first substitute a constant to the its couple to make u unprotected,

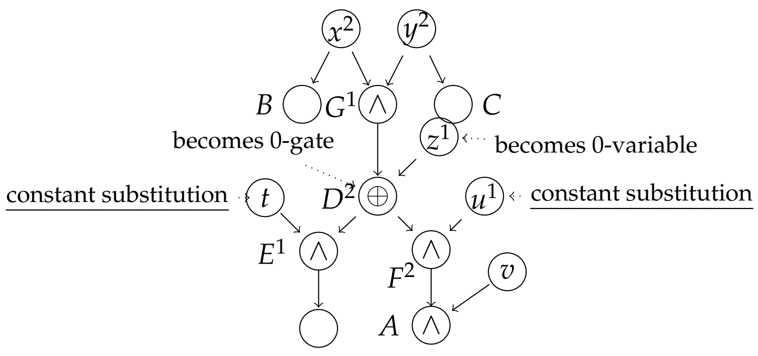


Figure 25: Case 5.4.1.4.2.3.

then apply Case 5.4.1.4.2.4. By this, we use 3 substitutions to make 4 variables non-influential, hence $\Delta\mu \geq \frac{4}{3}\alpha_I \geq \delta$.

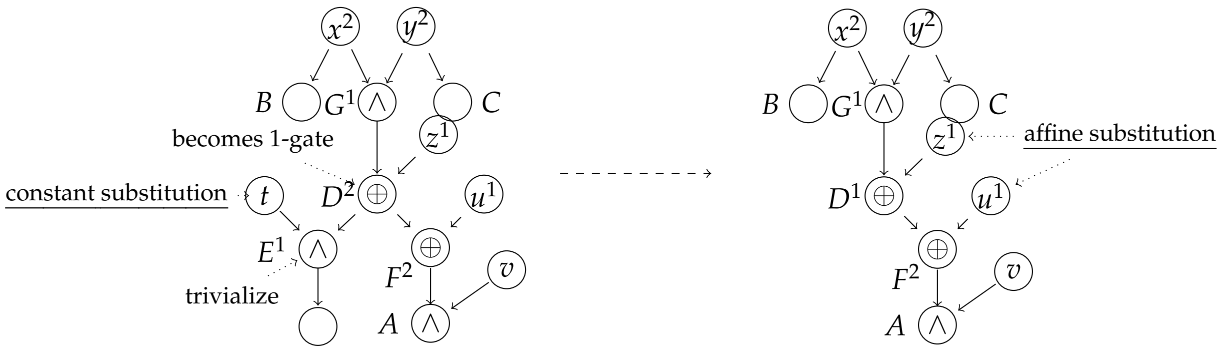


Figure 26: Case 5.4.1.4.2.4.

Case 5.4.2. We now assume that the troubled gate E does not get a variable from D, and hence is fed by $I_1 = t$ from B. In this case, the other input of E must be a variable z (see Figure 27).

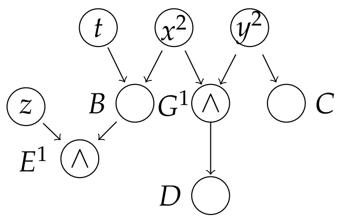


Figure 27: Case 5.4.2.

Case 5.4.2.1. If B is a 2^+-gate, then it must be $\oplus$-type (see Case 3). Since x must be unprotected (see Case 1), we can substitute $x \leftarrow t \oplus c$ such that the output of B is fixed regardless of the inputs. We can then replace B by a constant. If c is chosen appropriately, we can even further trivialize E. Note that we rewire the circuit such that G is fed by t instead of x to make x disconnected with the circuit, which will not introduce troubled gates. Since E and two descendants of them are removed by Rule 2 and Rule 3, and we can carefully check that the replacement of B by a constant can introduce only 1 potential increment (caused by t), the overall potential increment is bounded by $\Delta\Phi \leq 4$, hence $\Delta\mu \geq 4 - 4\alpha_\phi + \alpha_I \geq \delta$.

Case 5.4.2.2. If B is a 1-gate, we substitute constant values to z and y to trivialize both E and G. This will make B a 0-gate, so can be further removed by Rule 1. Note that z, y and x become non-influential, which means that $\Delta\mu \geq \frac{3}{2}\alpha_I \geq \delta$ per substitution.

From now on, no $\wedge$-type gate is directly fed by two variables. Let G be a topologically minimal $\wedge$-type gate fed by I_1 and I_2 (G exists since the function can be trivialized by a single affine substitution otherwise). Clearly I_1 and I_2 are computed by a fair cyclic-xor circuit, hence we can perform xor-reconstruction to them.

Case 6. Let x be a protected variable. By Case 1 and Case 2, it is a 1-variable feeding an $\oplus$-type gate P. Now we try to substitute $x \leftarrow d$ and normalize the circuit. At least P will be eliminated by Rule 3. We now rule out all cases when this operation *does not* give $\Delta\mu = 1 + \alpha_I + \alpha_Q$.

Case 6.1. Assume that another gate Q is eliminated during normalization. Furthermore, we assume that it is the second gate (P is the first one) to be eliminated. Now we show that Q is not eliminated by Rule 1.

1. If P is the output gate, after it is removed by Rule 3, Q will be the output gate, so that it should not be removed by Rule 1.
2. If P is not the output gate, after it is removed by Rule 3, the out-degree of Q would never decrease, hence cannot be removed by Rule 1.

The elimination of these two gates produce $\Delta\Phi \leq 2$ by normalization lemma, which means that $\Delta\mu \geq 2 - 2\alpha_\phi + \alpha_Q + \alpha_I \geq \delta$.

Case 6.2. Assume that P is the only gate eliminated, and $\Delta\Phi = 1$ (by normalization lemma, $\Delta\Phi \leq 1$). We now argue that this case can be avoided. Let R be the other input of P apart from x. To produce potential increment, R is either an $\wedge$-type gate or a variable.

Case 6.2.1. Assume that R is an $\wedge$-type gate which is troubled after this normalization. We argue that it is originally troubled, hence it does not produce potential increment. By Rule 3, it does not get new inputs, and the out-degrees of the variables feeding it do not decrease. If it is indeed a newly introduced troubled gate, its out-degree must decrease, which means that P can only be an output gate and R is originally a 2-gate. However, the output gate P is not reachable from the other descendant of R (otherwise there is a cycle containing an $\wedge$-type gate), which is impossible by Case 0.2.

Case 6.2.2. Assume that $R = u$ is a variable. Since its out-degree does not decrease during Rule 3, to feed troubled gates, it must be a 2^--variable before the normalization.

Case 6.2.2.1. Assume that u is an unprotected 1-variable (see Figure 28). Let y be the couple of x. We substitute constant value to y to make x unprotected. Then we perform affine substitution $x \leftarrow u$ such that P is trivialized. These two substitutions make three variables x, y and u non-influential, resulting in $\Delta\mu \geq \frac{3}{2}\alpha_I \geq \delta$ per substitution.

Case 6.2.2.2. Assume that u is a protected 1-variable, such that x and u are coupled with each other (see Figure 28). We perform affine substitution $x \leftarrow u$ to them, trivializing P. Both of x and u become non-influential, resulting in $\Delta\mu \geq 2\alpha_I \geq \delta$.

Case 6.2.2.3. Assume that u is a protected 1-variable with couple $v \neq x$ (see Figure 28). Let y be the couple of x. We perform constant substitution to y and v to make both x and u unprotected. Then we perform affine substitution $x \leftarrow u$ to trivialize P. These three substitutions make four variables, x, y, u and v non-influential, $\Delta\mu \geq \frac{4}{3}\alpha_I \geq \delta$.

Case 6.2.2.4. Assume that u is a 2-variable, which means that P must be a 1-gate (see Figure 29). Let A be the other descendant of u and B be the descendant of P. Note that $A \neq B$ since otherwise P is useless. Similar to Case 6.2.1, A cannot be the newly introduced troubled gate, so B is. In such case, B is an $\wedge$-type gate fed by P and a variable t. We substitute appropriate constant value to t to trivialize B, which allows us to further remove P by Rule 1. We then substitute constant value to the couple of x to make it unprotected. These two substitutions make three variables (x, t and the couple of x) non-influential, resulting in $\Delta\mu \geq \frac{3\alpha_I}{2}$.

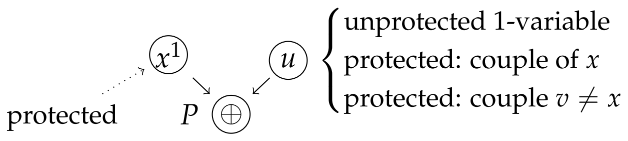


Figure 28: Case 6.2.2.1 ∼ 6.2.2.3

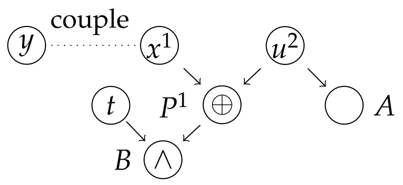


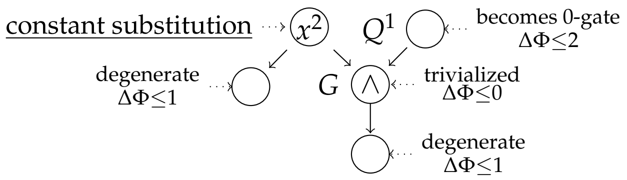
Figure 29: Case 6.2.2.4

From now on, we assume that substituting constant value to a protected variable will eliminate exactly one gate without introducing any troubled ones, hence $\Delta\mu = 1 + \alpha_I + \alpha_Q$.

Case 7. Assume that $I_1 = x$ is a 2-variable and $I_2 = Q$ is a 1-gate (see Figure 30). Note that x is unprotected (Case 1) and does not feed Q (otherwise Q is useless). We perform constant substitution to x to trivialize G and eliminate four gates in total: G, Q and two descendants of G and x. We now show that $\Delta\Phi \leq 4$.

1. Since Q is eliminated by Rule 1, it has $\Delta\Phi \leq 2$.
2. Since G is trivialized and both of its inputs neither become nor feed troubled gates, it has $\Delta\Phi \leq 0$.
3. Since the other two gates are removed by Rule 3, each of them have $\Delta\Phi \leq 1$.

Hence $\Delta\mu \geq 4 - 4\alpha_\phi + \alpha_I \geq \delta$ in total.



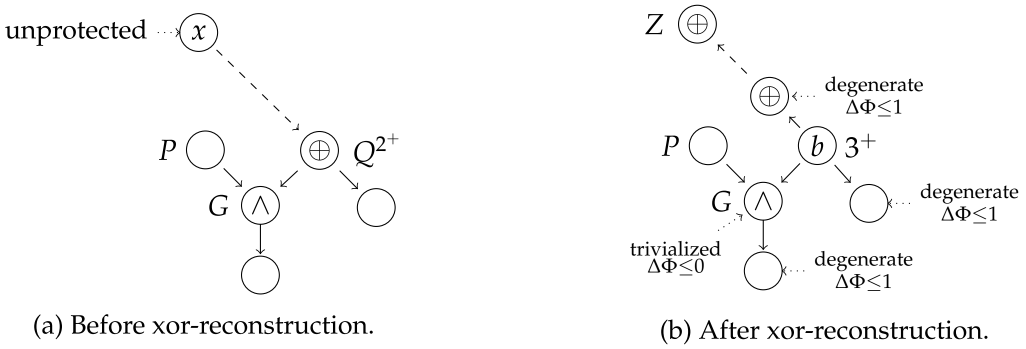
constant substitution
x^2
Q^1
becomes 0-gate $\Delta\Phi \leq 2$
degenerate $\Delta\Phi \leq 1$
G
$\wedge$
trivialized $\Delta\Phi \leq 0$
degenerate $\Delta\Phi \leq 1$

Figure 30: Case 7.

Case 8. Without loss of generality, we assume that $Q = I_2$ is a gate, while $P = I_1$ can be either a gate or a variable. If P is a gate, we further assume that the out-degree of P is not larger than Q. By the minimality of G, Q computes some affine function.

Case 8.1. Assume that Q computes an affine function depending on some *unprotected* variable x. By Proposition 2.7, there is a path from x to Q. We shall substitute an appropriate constant to Q via xor-reconstruction (see Section 2.6), such that G is trivialized. Now we exploit the type of node P to show that this will decrease the complexity measure by at least δ.

Case 8.1.1. Assume that Q is a 2^+-gate. After the xor-construction, it is replaced by a constant b feeding at least three gates (see Figure 31). Since we remove the gate Q while adding a new gate Z (see Figure 7), without introducing any new troubled gates (recall that the circuit has an empty packing originally due to Case 0.4), the potential remains unchanged. We then normalize the circuit such that the three descendants of b and at least one descendant of G are removed by Rule 2 and Rule 3. Hence $\Delta\Phi \leq 4$, so that $\Delta\mu \geq 4 - 4\alpha_\phi + \alpha_I \geq \delta$.


unprotected
x
P
Q^{2^+}
G
$\wedge$
$\oplus$
Z
degenerate $\Delta\Phi\leq 1$
P
b
3^+
G
degenerate $\Delta\Phi\leq 1$
trivialized $\Delta\Phi\leq 0$
degenerate $\Delta\Phi\leq 1$

(a) Before xor-reconstruction. (b) After xor-reconstruction.

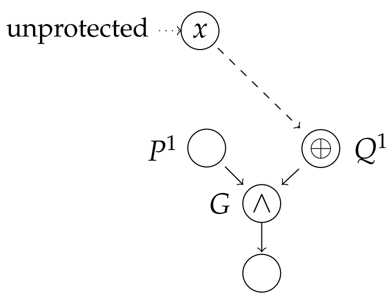
Figure 31: Case 8.1.1.

Case 8.1.2. Assume that Q is a 1-gate and $P = t$ is a variable. By Case 1 and Case 7, we can assume that t is an unprotected 1-variable. Hence the xor-reconstruction will make both t and x non-influential, resulting in $\Delta\Phi \geq 2\alpha_I \geq \delta$.

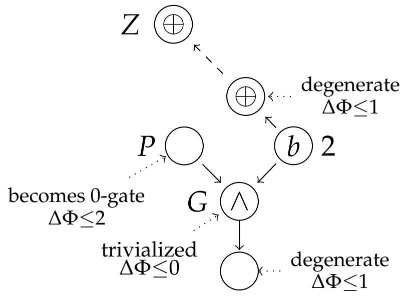
Case 8.1.3. Assume that both P and Q are 1-gates. After the xor-reconstruction, Q is replaced by a constant b of out-degree 2 (see Figure 32). We then normalize the circuit such that G, P, and the other two descendants of b and G are removed. Notice that this will remove 4 gates in total: P and the descendant of G does not coincide or there will be a cycle containing $\wedge$-type gate; P and the descendant of b (apart from G) does not coincide since P has only one descendent which is G before xor-reconstruction, and if P is fed by b, it should feed Q in the original circuit. We now show that $\Delta\Phi \leq 4$.

1. P is eliminated by Rule 1, it contributes $\Delta\Phi \leq 2$.
2. G is trivialized and both of its inputs are eliminated, it has $\Delta\Phi \leq 0$.
3. The other two gates are removed by Rule 2 or Rule 3, they contribute $\Delta\Phi \leq 2$ in total.

Hence $\Delta\mu \geq 4 - 4\alpha_\phi + \alpha_I \geq \delta$.



(a) Before xor-reconstruction.


Z
degenerate $\Delta\Phi\leq 1$
P
b 2
becomes 0-gate $\Delta\Phi\leq 2$
G
trivialized $\Delta\Phi\leq 0$
degenerate $\Delta\Phi\leq 1$

(b) After xor-reconstruction.

Figure 32: Case 8.1.3.

*For the rest of Case 8, we assume that **for any** topologically minimal $\wedge$-type gate G fed by some P and an $\oplus$-type gate Q satisfying the assumption at the start of Case 8 (if P is a gate, then the out-degree of P is not larger than Q), Q computes an affine function depending only on **protected** variables. **Moreover**, if I_1 is an $\oplus$-type gate with out-degree **equal** to Q, it also computes an affine function depending only on **protected** variables.*

Case 8.2. Now we assume that Q computes an affine function depending on a set of protected variables, say I.

Case 8.2.1. Assume that one of P and Q, say Q, is fed by two (protected) variables x_j and x_k which are coupled with each other. By Case 1 and Case 2, both of them are 1-variables. We can substitute $x_j \leftarrow x_k \oplus c$ to trivialize Q and further killing the quadratic equation containing x_j and x_k. This will make both of the two variables non-influential, giving $\Delta\mu \geq 2\alpha_I \geq \delta$.

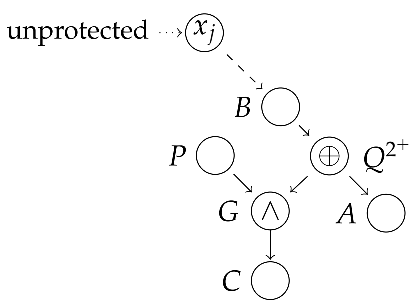
Case 8.2.2. Assume that Q is a 2^+-gate. Let $x_j \in I$ be a protected variable and x_k be its couple. We try to substitute $x_k \leftarrow d$ and normalize the circuit. After Case 6, exactly one gate (the descendant of x_k) is eliminated and $\Delta\mu = 1 + \alpha_I + \alpha_Q$.

Case 8.2.2.1. If Q is not fed by x_k originally, then after the substitution $x_k \leftarrow d$, Q is still a 2^+-gate computing affine function which depends on the unprotected variable x_j. We substitute appropriate constant to Q via xor-reconstruction such that G is trivialized, further degenerating A, B and C (see Figure 33 for the name of the nodes). We now assume that the elimination of B and G does not increase the potential (see Figure 33b), and defer the other cases to Case 8.2.5. Hence the net potential increment is bounded by 2, which gives $\Delta\mu \geq 4 - 2\alpha_\phi + \alpha_I$ for this substitution. Then the two substitutions, in average, produce a complexity measure decrement $\Delta\mu \geq \frac{1}{2}(5 - 2\alpha_\phi + 2\alpha_I + \alpha_Q) \geq \delta$.

Case 8.2.2.2. If Q is fed by x_k, after the substitution $x_k \leftarrow d$ and normalization, A is fed by another input of Q, say Q'. Clearly Q' is also a 2^+-gate depending on an unprotected variable x_j, so we can apply the argument in the previous case with Q replaced by Q'.

After this subcase, we assume that Q is a 1-gate, hence P is either a variable or a 1-gate.

Case 8.2.3. If $P = t$ is a variable, then it is an unprotected 1-variable by Case 1, Case 3 and Case 7. Let $x_j \in I$ be a protected variable with couple x_k (recall that I is the set of variables Q depending on). We first substitute constant to x_k to make x_j unprotected. After we normalize the circuit, G is fed by t and a gate Q' (may



(a) Before xor-reconstruction.

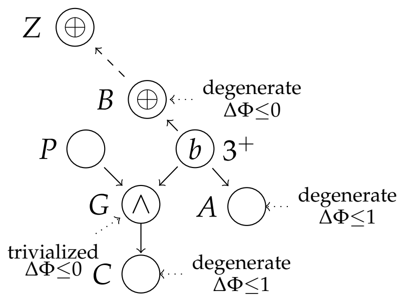

Z
B
degenerate $\Delta\Phi\leq 0$
P
b
3^+
G
A
degenerate $\Delta\Phi\leq 1$
trivialized $\Delta\Phi\leq 0$
C
degenerate $\Delta\Phi\leq 1$
(b) After xor-reconstruction.

Figure 33: Case 8.2.2.1.

be identical to Q) which computes an affine function depending on x_j. Hence by substituting appropriate constant to Q' via xor-reconstruction[10], we can trivialize G, hence make three variables t, x_j and x_k non-influential, which gives $\Delta\mu \geq \frac{3\alpha_I}{2} \geq \delta$.

Case 8.2.4. Assume that P is an $\oplus$-type 1-gate. If P computes an affine function depending on an unprotected variable, we apply Case 8.1.3 by exchanging P and Q. Hence we assume that both P and Q merely depend on protected variables.

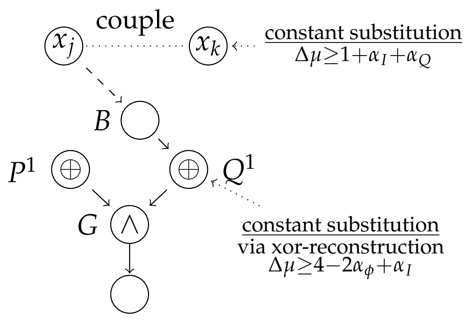

couple
x_j
x_k
constant substitution $\Delta\mu\geq 1+\alpha_I+\alpha_Q$
B
P^1
Q^1
G
constant substitution via xor-reconstruction $\Delta\mu\geq 4-2\alpha_\phi+\alpha_I$

Figure 34: Case 8.2.4.1.

Case 8.2.4.1. Assume that Q (or P) computes an affine function depending on some variable x_j with a couple x_k that does not feed P (or Q). We try to substitute a constant to x_k and normalize the circuit. After Case 6, exactly one gate is eliminated and $\Delta\mu = 1 + \alpha_I + \alpha_Q$. Hence P (or Q) is still an $\oplus$-type 1-gate after the normalization (see Figure 34). Now we have an unprotected variable x_j on which Q (or P) depends, so by substituting appropriate constant to Q (or P) via xor-reconstruction to trivialize G, we have $\Delta\mu \geq 4 - 4\alpha_\phi + \alpha_I$ for the substitution similar to Case 8.1.3. If the potential increment $\Delta\Phi \leq 2$ for this substitution, $\Delta\mu \geq 4 - 2\alpha_\phi + \alpha_I$, then these two substitutions, in average, produce a complexity measure decrement $\Delta\mu \geq \frac{1}{2}(5 - 2\alpha_\phi + 2\alpha_I + \alpha_Q) \geq \delta$. Hence in the rest part of this case we assume that $\Delta\Phi \geq 3$ for the substitution.

1. Similar to Case 8.2.2.1, we now assume that the elimination of G and B (the gate on the reversed path from G to x after xor-reconstruction, see Figure 34

[10]Note that by Case 6, the substitution $x_k \leftarrow d$ and subsequent normalization will not increase the potential, therefore the circuit has an empty packing before the xor-reconstruction. In such case, xor-reconstruction will not increase the potential.

and 38) does not increase the potential, and defer the other cases to Case 8.2.5.

2. The descendant of G is removed by Rule 2 or 3, so that $\Delta\Phi \leq 1$.
3. Now consider the input of P. To produce $\Delta\Phi \geq 3$, both inputs of P need to increase the potential by one.
   (a) Assume that P is fed by some gate T just before its removal. Since there is a path from T to G after xor-reconstruction, one may see that G is reachable from T in the original circuit, therefore T must be an $\oplus$-type gate and cannot become troubled. This is impossible.
   (b) If P is fed by two variables even before the substitution $x_k \leftarrow d$, it depends on them, so both of them are protected variables. By Case 1, they are 1-variables. They can neither feed nor become troubled gates, so that this case is impossible.
   (c) Assume that P is fed by two gates before the substitutions $x_k \leftarrow d$, and two variables t_1 and t_2 are passed to it by the degeneracy of B and the descendant of x_k. This will be considered in the following subcases.
   (d) Finally, to produce $\Delta\Phi \geq 3$, P should be fed by a variable t_1 and an $\oplus$-type gate which passes a variable t_2 to G via the degeneracy of B and the descendant P' of x_k. Just before P is eliminated, t_2 should be a 3-variable feeding an $\wedge$-type gate T, which becomes troubled after the elimination of P. This will be considered in the following subcases.

Case 8.2.4.1.1. If t_1 (or t_2) is a protected variable, after the normalization it becomes a protected variable that either feeds a troubled gate or is the inner variable of a pack. If we further substitute $t_1 \leftarrow c$ (or $t_2 \leftarrow c$) for appropriate c, we can eliminate at least three gates via Rule 2 or Rule 3, while killing a quadratic substitution, which produces $\Delta\mu \geq 3 - 3\alpha_\phi + \alpha_Q + \alpha_I$. In total, we perform three substitutions, and produce

$$\begin{aligned}\Delta\mu &\geq \frac{(1+\alpha_I+\alpha_Q)+(4-4\alpha_\phi+\alpha_I)+(3-3\alpha_\phi+\alpha_Q+\alpha_I)}{3}\\ &\geq \frac{2(2-2\alpha_\phi+\alpha_I+\alpha_Q)}{3}+\frac{4-4\alpha_\phi+\alpha_I}{3}\\ &\geq \delta\end{aligned}$$

per substitution.

Case 8.2.4.1.2. Assume that both t_1 and t_2 are passed to P during the degeneracy of B and the descendent P' of x_k. Without loss of generality, assume that t_2 is passed to P only via B and t_1 is passed to P only via P'. By the general picture of xor-reconstruction (see Figure 7), t_2 feeds Q and B feeds P before the substitution $x_k \leftarrow d$ (see Figure 35). Note that B must be a gate: if $B = x_j$, it is a protected 2-variable, which is impossible by Case 1. Since t_2 is unprotected, Q does not depend on it. Furthermore, $Q = B \oplus t_2 \oplus c_1$, so B must depend on t_2. Since $P = x_k \oplus t_1 \oplus B \oplus c_2$, we can see that P depends on t_2, which is impossible since P depends on no unprotected variables.

Case 8.2.4.1.3. Finally we assume that one of t_1 and t_2, say t_1, directly feeds P before the substitution $x_k \leftarrow d$. Then t_2 either feeds P even before the substitution $x_k \leftarrow d$, or is passed to P via the degenerate gates B and P'.

- If t_2 directly feeds P or is passed to P only via P', P depends on the unprotected variable t_1, which is impossible by Case 8.1.
- Otherwise, t_2 is passed to P by B (alone, or together with P'). Similar to Case 8.2.4.1.2, t_2 feeds Q and B feeds P and Q just after the substitution $x_k \leftarrow d$ (see Figure 36). Since P' is the only degenerate gate at that time, either B feeds P or B feeds Q directly in the original circuit, say B feeds P. Now we show that either P or Q depends on an unprotected variable, so that it is impossible by Case 8.1. If P depends on t_1, we immediately success; otherwise since B feeds P directly in the original circuit, B should depends on t_1, therefore Q depends on t_1, no matter whether B feeds Q directly, or B feeds P' that further feeds Q.

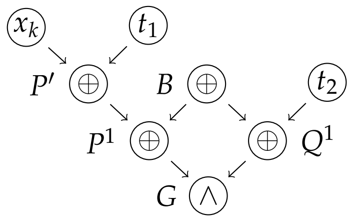


Figure 35: Case 8.2.4.1.2.

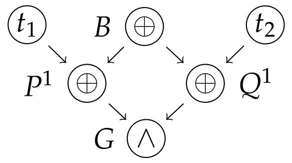


Figure 36: Case 8.2.4.1.3 (just after $x_k \leftarrow d$).

After this case, P only depends on the variables whose couples feed Q, and Q only depends on the variables whose couples feed P.

Case 8.2.4.2. Assume that one of P and Q, say P, is fed by two protected variables x_j and x_k (see Figure 37). By Case 8.2.1, we assume that x_j and x_k are not a couple. By Case 1, we assume can that both x_j and x_k are 1-variables. By substituting constant to both x_j and x_k, we can trivialize P and G, further eliminating Q and at least one descendant of G. During this process, the potential increment is bounded by 3: zero for P and G, two for Q and one for the descendant of A. Hence we eliminate four gates, kill two quadratic equations and two influential variables, which gives $\Delta\mu \geq \frac{1}{2}(4 - 3\alpha_\phi + 2\alpha_I + 2\alpha_Q) \geq \delta$ per substitution.

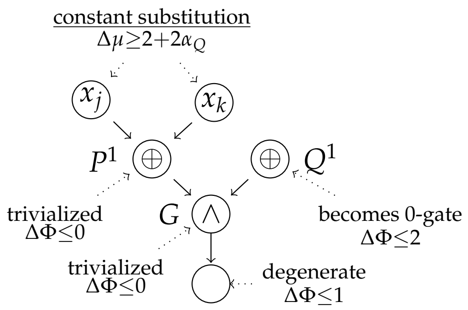


Figure 37: Case 8.2.4.2.

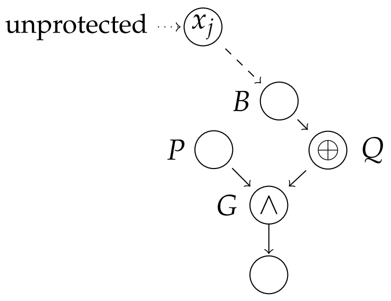
Case 8.2.4.3. Finally we assume that both P and Q are fed by at most one variable. By Case 0.3, both P and Q do not compute a constant value, hence P depends on at least one (protected) variable x_j, whose couple x_k feeds Q. The another input of Q must be a gate R (by Case 8.2.4.2), which do not compute a constant value by Case 0.3. Since Q is a 1-gate feeding an $\wedge$-type gate and x_k is a 1-variable (see Case 1), there is no path from x_k to R, hence R does not depend on x_k. In such case, Q depends on x_k. Let x_i be a variable depended by R, which is also depended by Q. Also, $x_i \neq x_j$, since there is no path from x_j to R, but R depends on x_i. This means that Q depends on two protected variables x_k and x_i that are not couple of each other, hence P must be fed by two protected variables, which is impossible by Case 8.2.4.2.

We now handle the situation when $\Delta\Phi > 2$ is produced in Case 8.2.2 and Case 8.2.4.

Case 8.2.5. Let $x_j \in I$ be a protected variable and x_k be its couple. By Proposition 2.7, there exists a path from x_j to Q. We substitute $x_k \leftarrow d$ and normalize the circuit, such that x_j becomes unprotected. By Case 6, the descendant of x_k is the only gate to be eliminated, with no new troubled gate introduced. After this substitution,

- if x_k does not feed Q in the original circuit, then Q still feeds G, and its out-degree does not decrease;
- otherwise the other input Q' of Q is passed to feed G. Note that Q' is also an $\oplus$-type gate, so we identify it with Q later.

Now we substitute appropriate constant to Q to trivialize G via xor-reconstruction with the variable x_j which is now unprotected. Let B be the preceding node of Q on the path from x_j to Q (see Figure 38). We can eliminate G, B and the descendant of G. We consider the situation when the elimination of B or G produces $\Delta\Phi \geq 1$ (i.e., an input of B or G becomes or feeds a troubled gate after the xor-reconstruction and normalization). Note that we eliminate G first, with B follows, and other gates are eliminated in arbitrary order.



(a) Before xor-reconstruction.

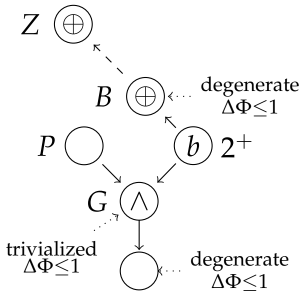

Z
$\oplus$
B
$\oplus$
degenerate $\Delta\Phi\leq 1$
P
b
2^+
G
$\wedge$
trivialized $\Delta\Phi\leq 1$
degenerate $\Delta\Phi\leq 1$
(b) After xor-reconstruction.

Figure 38: Case 8.2.5.

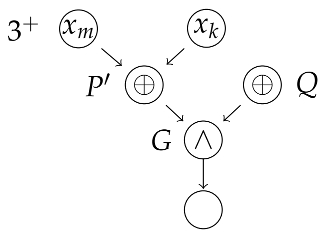


Figure 39: Case 8.2.5.1.1.

Case 8.2.5.1. Assume that the elimination of G increases the potential by one (see Figure 38b). By the minimality, in order to produce a potential increment, P must be a variable x_m whose out-degree decreased from 3 to 2. So it should be a 3-variable after the substitution $x_k \leftarrow d$.

Case 8.2.5.1.1. Assume that x_m is an 3-variable even before the substitution $x_k \leftarrow d$. By Case 3, it cannot feed G before the substitution $x_k \leftarrow d$, hence x_m is passed to G via the degeneracy at substitution $x_k \leftarrow d$ (see Figure 39). For simplicity, we label the degenerate gate as P'.
To increase the potential, x_m must feed an $\wedge$-type gate T just before G is eliminated. By the general picture of xor-reconstruction (see Figure 7), it

should feed T even before the xor-reconstruction, i.e. just after the substitution $x_k \leftarrow d$ and subsequent normalization. Hence, either x_m feeds T even before the substitution $x_k \leftarrow d$, or it is passed to T via the degenerate gate P'. The former case is impossible by Case 3, since otherwise x_m is a 3-variable feeding an $\wedge$-type gate T. In the latter case, P' needs to be a 2^+-gate. This is also impossible, since in order to make T troubled while eliminating G, x_m should be a 2 variable after the elimination, hence P' can only be 1-gate.

Case 8.2.5.1.2. Assume that x_m is a 1-variable before the substitution $x_k \leftarrow d$. In such case, 1-variables x_m and x_k feed a common $\oplus$-type gate P'. Similar to Case 6.2.2.1 $\sim$ Case 6.2.2.3, we can perform affine substitutions to decrease the measure by at least $\Delta\mu \geq \frac{4\alpha_I}{3} \geq \delta$ per substitution.

Case 8.2.5.1.3. Assume that x_m is a 2-variable originally, whose out-degree is increased to 3 after the substitution $x_k \leftarrow d$. Then P' must be a $\oplus$-type (fed by protected variable x_k) 2-gate (to increase the out-degree of x_m) fed by both x_m and x_k. Note that x_m either directly feeds G originally, or is passed to G via the degenerate of P'.

Case 8.2.5.1.3.1. Assume that x_m initially feeds G. To form a troubled gate, P' should feed an $\wedge$-type gate T even before the substitution $x_k \leftarrow d$ (see Figure 40). We can substitute constant values to x_m and x_k to trivialize P', T and G, then degenerate their descendents, so that six gates are removed by Rule 2 and Rule 3. The complexity measure decrement per substitution is at least

$$\Delta\mu \geq \frac{6 - 6\alpha_\phi + \alpha_Q + 2\alpha_I}{2} = \frac{4 - 4\alpha_\phi + \alpha_I}{2} + \frac{2 - 2\alpha_\phi + \alpha_I + \alpha_Q}{2} \geq \delta.$$

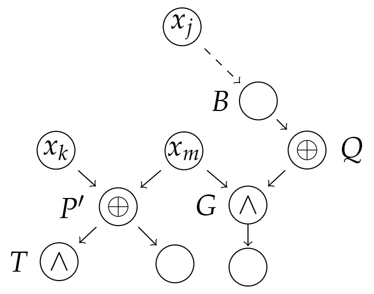


Figure 40: Case 8.2.5.1.3.1: before substitution $x_k \leftarrow d$.

For the rest of Case 8.2.5.1.3, we assume that x_m is passed to G via the degeneracy of P'

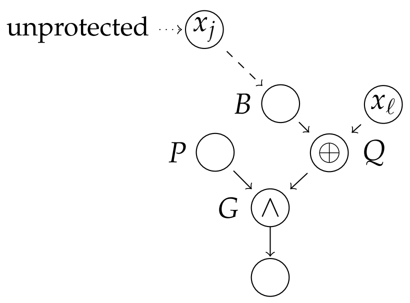
Case 8.2.5.1.3.2. Assume that Q is a 2-gate. Since P' is a 2^+-gate depending on an unprotected variable x_m (see Case 1), we can apply Case 8.1 with P' and Q exchanged (i.e. let Q be P and P' be Q in Case 8.1). This obtain a complexity measure decrement $\Delta\mu \geq \delta$.

Case 8.2.5.1.3.3. Now we assume that Q is a 3^+-gate. We firstly perform the substitution $x_k \leftarrow d$ and normalize the circuit, such that P' is eliminated and Q becomes an $\oplus$-type gate depending on an unprotected variable x_j. By Case 6, the potential decrement is $\Delta\mu \geq 1 + \alpha_Q + \alpha_I$. Then we substitute appropriate constant to Q via xor-reconstruction to trivialize G, such that five gates are eliminated by Rule 2 and Rule 3: three descendants of Q, the first gate on the reversed path from Q to x_j, and the descendant of G. These two substitutions, in average, decrease the potential by

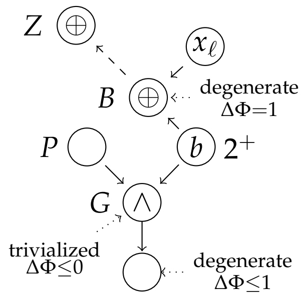
$$\begin{aligned}\Delta\mu &\geq \frac{(1+\alpha_Q+\alpha_I)+(5-5\alpha_\phi+\alpha_I)}{2}\\ &\geq \frac{4-4\alpha_\phi+\alpha_I}{2}+\frac{2-2\alpha_\phi+\alpha_I+\alpha_Q}{2}\\ &\geq \delta.\end{aligned}$$

We now assume the elimination of B increases the potential by one.

Case 8.2.5.2. Assume that B feeds Q even before the constant substitution $x_k \leftarrow d$. In such case, after the xor-reconstruction, the other input of B apart from b must be a node D who becomes or feeds a troubled gate. By the general picture of xor-reconstruction (see Figure 7), D feeds Q before the xor-reconstruction. This means that in the very beginning, D either directly feeds Q, or feeds the descendent P' of x_k, and is passed to B at the normalization following $x_k \leftarrow d$. By normalization lemma, it either becomes or feeds a troubled gate T after the normalization. It cannot be an $\wedge$-type gate by the minimality of G, so D can only be a variable, say x_ℓ (see Figure 41 for the circuit before and after xor-reconstruction).



(a) Before xor-reconstruction.



(b) After xor-reconstruction.

Figure 41: Case 8.2.5.2.

Case 8.2.5.2.1. If x_ℓ is unprotected, then Q cannot depend on x_ℓ after Case 8.1. Since Q computes $B \oplus x_\ell \oplus c$ (or $B \oplus x_\ell \oplus x_k \oplus c$) for some constant c in the original circuit, B should depend on x_ℓ. Now we exploit the local topology of B and T (the troubled gate introduced) to show that this would not be an obstacle.

Case 8.2.5.2.1.1. Assume that B is a 0-gate just before it is eliminated, so the out-degree of x_ℓ is decreased. In such case, $B = Z$, otherwise it at least feeds the next node on the path. This means that $B = x_j$ just before the xor-reconstruction. So at that time, Q is fed by the two variables x_j and x_ℓ, hence depends on them. However, this means that B cannot depend on x_ℓ, which leads to contradiction.

For the rest of Case 8.2.5.2.1, we can assume that just before eliminated, B is a 1^+ gate. This means that the elimination of it will not decrease the out-degree of x_ℓ. Hence, x_ℓ should be passed to the newly introduced troubled gate T during the elimination of B (if x_ℓ feeds T originally, then T should be troubled even before the substitutions). In order for T to become a troubled gate, the other input of it apart from B should be a variable, say t.

Case 8.2.5.2.1.2. If both B and t feeds T even before the substitution $x_k \leftarrow d$, then it is easy to see that T is a topologically minimal $\wedge$-type gate fed by an $\oplus$-type gate B depending on the unprotected variable x_ℓ, which is impossible.

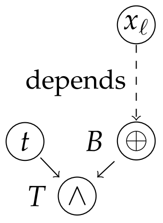


Figure 42: Case 8.2.5.2.1.2 (B feeds T).

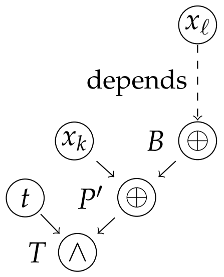


Figure 43: Case 8.2.5.2.1.3 (B to T via $\oplus$-type gate).

Case 8.2.5.2.1.3. Assume that B is passed to T when the descendant P' of x_k degenerates, then t should feed T originally. Since x_k is protected, P' can only be $\oplus$-type (see Figure 43), then T is a topologically minimal $\wedge$-type gate fed by an $\oplus$-type gate P' depending on the unprotected variable x_ℓ, which is also impossible.

For the rest of Case 8.2.5.2.1, we assume that t is passed to T when the descendent P' of x_k degenerates. Then B should feed T originally.

Case 8.2.5.2.1.4. If t is a 1-variable (see Figure 44), then we can observe that P' is an $\oplus$-type gate fed by two 1-variables. This allows us to perform affine substitutions similarly to 6.2.2.1 $\sim$ Case 6.2.2.3, decreasing the measure by at least $\Delta\mu \geq \frac{4\alpha_I}{3} \geq \delta$ per substitution.

Case 8.2.5.2.1.5. If t is a 2-variable (see Figure 45), then P' should be a 1-gate in order to make T troubled in Case 8.2.5.2.1. Also observe that B depends on the unprotected variable x_ℓ, hence is impossible after Case 8.1.

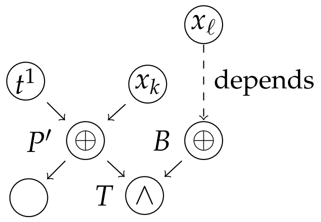


Figure 44: Case 8.2.5.2.1.4 (t has out-degree 1).

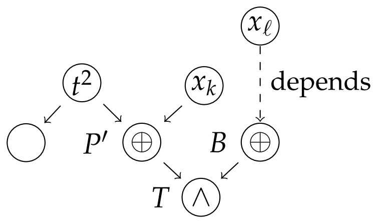


Figure 45: Case 8.2.5.2.1.5 (t has out-degree 2).

Case 8.2.5.2.2. Assume that x_ℓ is protected. Note that the elimination of B increases the potential by 1, which must be related to x_ℓ. This means that after the normalization, x_ℓ either feeds a truly troubled gate, or is an inner variable of some pack. In either case, x_ℓ is a protected 2-variable feeding an $\wedge$-type gate. This allows us to substitute appropriate constant to x_ℓ to remove three gates. In total, three substitutions are applied.

1. A constant substitution $x_k \leftarrow d$, which gives $\Delta\mu \geq 1 + \alpha_I + \alpha_Q$.
2. A xor-reconstruction (see Figure 38b), which removes three gates (B, G and the descendant of G). The elimination of G will not increase the potential, hence $\Delta\Phi \leq 2$, which gives $\Delta\mu \geq 3 - 2\alpha_\phi + \alpha_I$.
3. A constant substitution $x_\ell \leftarrow c$. Three gates are removed by Rule 2 and Rule 3. Now we argue that $\Delta\Phi \leq 1$, which gives $\Delta\mu \geq 3 - \alpha_\phi + \alpha_I + \alpha_Q$.
   (a) One of the three eliminated gates is originally a troubled gate fed by x_ℓ; we eliminate it first. Since both of its inputs are 2-variables, they cannot feed troubled gates after the normalization, hence this normalization causes no potential increment.
   (b) The other two eliminated gates will increase the potential by at most 2, since they are eliminated by Rule 2 and Rule 3.
   (c) Note that x_ℓ either feeds a troubled gate or is the inner variable of a pack. The substitution x_ℓ fully destructs the troubled gate or pack, which decreases the potential by 1.

These three substitutions produce

$$\begin{aligned}\Delta\mu &\geq \frac{7 - 3\alpha_\phi + 3\alpha_I + 2\alpha_Q}{3} \\ &\geq \frac{2}{3}\left(\frac{5 - 2\alpha_\phi + 2\alpha_I + \alpha_Q}{2}\right) + \frac{2 - 2\alpha_\phi + \alpha_I + \alpha_Q}{3} \\ &\geq \delta.\end{aligned}$$

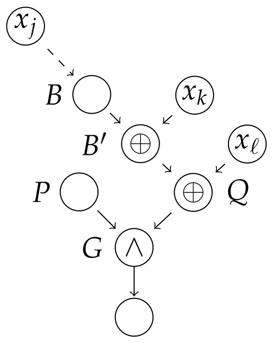


Figure 46: Case 8.2.5.3.

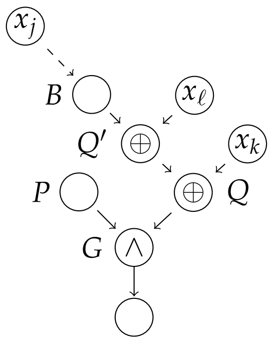


Figure 47: Case 8.2.5.4.

Case 8.2.5.3. Assume that B feeds B' and B' feeds Q in the original circuit, where B' is the gate eliminated in substitution $x_k \leftarrow d$ (see Figure 46). One may carefully check that the analysis in Case 8.2.5.2 still works. We point out some essential facts for this case.

1. Since the normalization of B increases the potential, it is fed by a variable x_ℓ just after the xor-reconstruction. It is easy to see that x_ℓ feeds Q in the original circuit.
2. If x_ℓ is unprotected, Q does not depend on it. Since $Q = B \oplus x_k \oplus x_\ell \oplus c$ for some constant c, B must depend on it. We can also see that B is not a 0-gate before its elimination, otherwise $B = x_j$ and cannot depend on x_ℓ.
3. If x_ℓ is protected, we can substitute x_ℓ to constant after the whole process to produce sufficient complexity measure decrement.

Case 8.2.5.4. Assume that B feeds Q' and Q' feeds Q, where Q is eliminated in substitution $x_k \leftarrow d$, and Q' takes place of Q (see Figure 47). Similar to the previous case, one may carefully check that the analysis in Case 8.2.5.2 still works. □

Acknowledgement. We are grateful to Andrew C. Yao for his support throughout this project. We also thank Yiding Zhang for proofreading earlier versions of this paper and giving lots of useful comments; in particular, he points out the existence of compact troubled pairs and addresses several issues in the case analysis.

References

[AB87] Noga Alon and Ravi Boppana. "The monotone circuit complexity of Boolean functions". In: *Combinatorica* 7 (Mar. 1987), pp. 1–22. DOI: 10.1007/BF02579196 (cit. on p. 2).

[And87] A. E. Andreev. "On a method for obtaining more than quadratic effective lower bounds for the complexity of π-schemes". In: *Vestnik Moskov. Univ. Ser. 1. Mat. Mekh.* (1 1987), pp. 70–73 (cit. on p. 2).

[AB09] Sanjeev Arora and Boaz Barak. *Computational Complexity - A Modern Approach*. Cambridge University Press, 2009. ISBN: 978-0-521-42426-4. URL: http://www.cambridge.org/catalogue/catalogue.asp?isbn=9780521424264 (cit. on p. 6).

[BK12] Eli Ben-Sasson and Swastik Kopparty. "Affine Dispersers from Subspace Polynomials". In: *SIAM J. Comput.* 41.4 (2012), pp. 880–914. DOI: 10.1137/110826254. URL: https://doi.org/10.1137/110826254 (cit. on p. 7).

[Blu84] Norbert Blum. "A Boolean Function Requiring 3n Network Size". In: *Theor. Comput. Sci.* 28 (1984), pp. 337–345. DOI: 10.1016/0304-3975(83)90029-4. URL: https://doi.org/10.1016/0304-3975(83)90029-4 (cit. on p. 1).

[DK11] Evgeny Demenkov and Alexander S. Kulikov. "An Elementary Proof of a 3n - o(n) Lower Bound on the Circuit Complexity of Affine Dispersers". In: *Mathematical Foundations of Computer Science 2011 - 36th International Symposium, MFCS 2011, Warsaw, Poland, August 22-26, 2011. Proceedings*. Ed. by Filip Murlak and Piotr Sankowski. Vol. 6907. Lecture Notes in Computer Science. Springer, 2011, pp. 256–265. DOI: 10.1007/978-3-642-22993-0_25. URL: https://doi.org/10.1007/978-3-642-22993-0_25 (cit. on pp. 2, 3).

[FGHK16] Magnus Gausdal Find, Alexander Golovnev, Edward A. Hirsch, and Alexander S. Kulikov. “A Better-Than-3n Lower Bound for the Circuit Complexity of an Explicit Function”. In: *IEEE 57th Annual Symposium on Foundations of Computer Science, FOCS 2016, 9-11 October 2016, Hyatt Regency, New Brunswick, New Jersey, USA*. Ed. by Irit Dinur. IEEE Computer Society, 2016, pp. 89–98. DOI: 10.1109/FOCS.2016.19. URL: https://doi.org/10.1109/FOCS.2016.19 (cit. on pp. 2, 4, 5, 6, 8, 10, 12, 13, 14, 18).

[GHKK18] Alexander Golovnev, Edward A. Hirsch, Alexander Knop, and Alexander S. Kulikov. “On the limits of gate elimination”. In: *J. Comput. Syst. Sci.* 96 (2018), pp. 107–119. DOI: 10.1016/j.jcss.2018.04.005. URL: https://doi.org/10.1016/j.jcss.2018.04.005 (cit. on p. 4).

[GK16] Alexander Golovnev and Alexander S. Kulikov. “Weighted Gate Elimination: Boolean Dispersers for Quadratic Varieties Imply Improved Circuit Lower Bounds”. In: *Proceedings of the 2016 ACM Conference on Innovations in Theoretical Computer Science*. ITCS ’16. Cambridge, Massachusetts, USA: Association for Computing Machinery, 2016, pp. 405–411. ISBN: 9781450340571. DOI: 10.1145/2840728.2840755. URL: https://doi.org/10.1145/2840728.2840755 (cit. on p. 2).

[GKW21] Alexander Golovnev, Alexander S. Kulikov, and R. Ryan Williams. “Circuit Depth Reductions”. In: *12th Innovations in Theoretical Computer Science Conference, ITCS 2021, January 6-8, 2021, Virtual Conference*. Ed. by James R. Lee. Vol. 185. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2021, 24:1–24:20. DOI: 10.4230/LIPIcs.ITCS.2021.24. URL: https://doi.org/10.4230/LIPIcs.ITCS.2021.24 (cit. on p. 4).

[Hås86] Johan Håstad. “Almost Optimal Lower Bounds for Small Depth Circuits”. In: *Proceedings of the 18th Annual ACM Symposium on Theory of Computing, May 28-30, 1986, Berkeley, California, USA*. Ed. by Juris Hartmanis. ACM, 1986, pp. 6–20. DOI: 10.1145/12130.12132. URL: https://doi.org/10.1145/12130.12132 (cit. on p. 3).

[IN93] Russell Impagliazzo and Noam Nisan. “The Effect of Random Restrictions on Formula Size”. In: *Random Struct. Algorithms* 4.2 (1993), pp. 121–134. DOI: 10.1002/rsa.3240040202. URL: https://doi.org/10.1002/rsa.3240040202 (cit. on p. 2).

[IM02] Kazuo Iwama and Hiroki Morizumi. “An Explicit Lower Bound of 5n - o(n) for Boolean Circuits”. In: *Mathematical Foundations of Computer Science 2002, 27th International Symposium, MFCS 2002, Warsaw, Poland, August 26-30, 2002, Proceedings*. Ed. by Krzysztof Diks and Wojciech Rytter. Vol. 2420. Lecture Notes in Computer Science. Springer, 2002, pp. 353–364. DOI: 10.1007/3-540-45687-2_29. URL: https://doi.org/10.1007/3-540-45687-2%5C_29 (cit. on p. 2).

[Khr71a] V. M. Khrapchenko. “Method of determining lower bounds for the complexity of P-schemes”. In: *Mathematical notes of the Academy of Sciences of the USSR* (1 1971), pp. 474–479 (cit. on p. 2).

[Khr71b] VM Khrapchenko. “Complexity of the realization of a linear function in the class of π-circuits”. In: *Mathematical Notes of the Academy of Sciences of the USSR* 9.1 (1971), pp. 21–23 (cit. on p. 2).

[KM65] Boris M. Kloss and Vadim A. Malyshev. “Estimates of the complexity of certain classes of functions”. In: *Vestn. Moskov. Univ. Ser. 1* 4 (1965), pp. 44–51 (cit. on p. 1).

[LR01] Oded Lachish and Ran Raz. "Explicit lower bound of *4.5n - o(n)* for boolean circuits". In: *Proceedings on 33rd Annual ACM Symposium on Theory of Computing, July 6-8, 2001, Heraklion, Crete, Greece*. Ed. by Jeffrey Scott Vitter, Paul G. Spirakis, and Mihalis Yannakakis. ACM, 2001, pp. 399–408. DOI: 10.1145/380752.380832. URL: https://doi.org/10.1145/380752.380832 (cit. on p. 2).

[Li11] Xin Li. "A New Approach to Affine Extractors and Dispersers". In: *Proceedings of the 26th Annual IEEE Conference on Computational Complexity, CCC 2011, San Jose, California, USA, June 8-10, 2011*. IEEE Computer Society, 2011, pp. 137–147. DOI: 10.1109/CCC.2011.27. URL: https://doi.org/10.1109/CCC.2011.27 (cit. on p. 7).

[Li15] Xin Li. "Extractors for Affine Sources with Polylogarithmic Entropy". In: *Electron. Colloquium Comput. Complex.* 22 (2015), p. 121. URL: http://eccc.hpi-web.de/report/2015/121 (cit. on p. 7).

[MW20] Cody D Murray and R Ryan Williams. "Circuit Lower Bounds for Nondeterministic Quasi-polytime from a New Easy Witness Lemma". In: *SIAM Journal on Computing* 49.5 (2020), STOC18-300–STOC18-322 (cit. on p. 3).

[PZ93] Mike Paterson and Uri Zwick. "Shrinkage of de Morgan Formulae under Restriction". In: *Random Struct. Algorithms* 4.2 (1993), pp. 135–150. DOI: 10.1002/rsa.3240040203. URL: https://doi.org/10.1002/rsa.3240040203 (cit. on p. 2).

[Pau77] Wolfgang J. Paul. "A 2.5 n-Lower Bound on the Combinational Complexity of Boolean Functions". In: *SIAM J. Comput.* 6.3 (1977), pp. 427–443. DOI: 10.1137/0206030. URL: https://doi.org/10.1137/0206030 (cit. on p. 1).

[Raz85] A. A. Razborov. "Lower bounds on the monotone complexity of some Boolean functions". In: *Dokl. Akad. Nauk SSSR* (4 1985), pp. 798–801 (cit. on p. 2).

[Raz87] Alexander A Razborov. "Lower bounds on the size of bounded depth circuits over a complete basis with logical addition". In: *Mathematical Notes of the Academy of Sciences of the USSR* 41.4 (1987), pp. 333–338 (cit. on p. 3).

[Sch74] Claus-Peter Schnorr. "Zwei lineare untere Schranken für die Komplexität Boolescher Funktionen". In: *Computing* 13.2 (1974), pp. 155–171. DOI: 10.1007/BF02246615. URL: https://doi.org/10.1007/BF02246615 (cit. on p. 1).

[Sch76] Claus-Peter Schnorr. "The Combinational Complexity of Equivalence". In: *Theor. Comput. Sci.* 1.4 (1976), pp. 289–295. DOI: 10.1016/0304-3975(76)90073-6. URL: https://doi.org/10.1016/0304-3975(76)90073-6 (cit. on p. 2).

[Sha49] Claude E. Shannon. "The synthesis of two-terminal switching circuits". In: *Bell Syst. Tech. J.* 28.1 (1949), pp. 59–98. DOI: 10.1002/j.1538-7305.1949.tb03624.x. URL: https://doi.org/10.1002/j.1538-7305.1949.tb03624.x (cit. on p. 1).

[Smo87] Roman Smolensky. "Algebraic Methods in the Theory of Lower Bounds for Boolean Circuit Complexity". In: *Proceedings of the 19th Annual ACM Symposium on Theory of Computing, 1987, New York, New York, USA*. Ed. by Alfred V. Aho. ACM, 1987, pp. 77–82. DOI: 10.1145/28395.28404. URL: https://doi.org/10.1145/28395.28404 (cit. on p. 3).

[Sto77] Larry J. Stockmeyer. "On the Combinational Complexity of Certain Symmetric Boolean Functions". In: *Math. Syst. Theory* 10 (1977), pp. 323–336. DOI: 10.1007/BF01683282. URL: https://doi.org/10.1007/BF01683282 (cit. on p. 1).

[Sub61] B. A. Subbotovskaya. “Realization of linear functions by formulas using $\vee$, &, $^{-}$”. In: *Dokl. Akad. Nauk SSSR* (3 1961), pp. 553–555 (cit. on p. 2).

[Tal14] Avishay Tal. “Shrinkage of De Morgan Formulae by Spectral Techniques”. In: *55th IEEE Annual Symposium on Foundations of Computer Science, FOCS 2014, Philadelphia, PA, USA, October 18-21, 2014*. IEEE Computer Society, 2014, pp. 551–560. DOI: 10.1109/FOCS.2014.65. URL: https://doi.org/10.1109/FOCS.2014.65 (cit. on p. 2).

[Val77] Leslie G. Valiant. “Graph-Theoretic Arguments in Low-Level Complexity”. In: *Mathematical Foundations of Computer Science 1977, 6th Symposium, Tatranska Lomnica, Czechoslovakia, September 5-9, 1977, Proceedings*. Ed. by Jozef Gruska. Vol. 53. Lecture Notes in Computer Science. Springer, 1977, pp. 162–176. DOI: 10.1007/3-540-08353-7_135. URL: https://doi.org/10.1007/3-540-08353-7%5C_135 (cit. on p. 4).

[Wil13] Ryan Williams. “Improving Exhaustive Search Implies Superpolynomial Lower Bounds”. In: *SIAM J. Comput.* 42.3 (2013), pp. 1218–1244. DOI: 10.1137/10080703X. URL: https://doi.org/10.1137/10080703X (cit. on p. 3).

[Wil14] Ryan Williams. “Nonuniform ACC Circuit Lower Bounds”. In: *J. ACM* 61.1 (2014), 2:1–2:32. DOI: 10.1145/2559903. URL: https://doi.org/10.1145/2559903 (cit. on p. 3).

[Yao85] A. C. Yao. “Separating the polynomial-time hierarchy by oracles”. In: *26th Annual Symposium on Foundations of Computer Science (sfcs 1985)*. 1985, pp. 1–10. DOI: 10.1109/SFCS.1985.49 (cit. on p. 3).

[Zwi91] Uri Zwick. “A 4n Lower Bound on the Combinational Complexity of Certain Symmetric Boolean Functions over the Basis of Unate Dyadic Boolean Functions”. In: *SIAM J. Comput.* 20.3 (1991), pp. 499–505. DOI: 10.1137/0220032. URL: https://doi.org/10.1137/0220032 (cit. on p. 2).