An Explicit Lower Bound of $5n - o(n)$ for Boolean Circuits

Kazuo Iwama †, Oded Lachish ‡, Hiroki Morizumi †, and Ran Raz § ⋆

† Graduate School of Informatics, Kyoto University, Kyoto, JAPAN
{iwama, morizumi}@kuis.kyoto-u.ac.jp
‡ Faculty of Computer Science, Haifa University, Haifa, Israel
loded@cs.haifa.ac.il
§ Faculty of mathematics, Weizmann Institute, Rehovot, Israel
ran.raz@weizmann.ac.il

Abstract. We prove a lower bound of $5n - o(n)$ for the circuit complexity of an explicit (constructible in deterministic polynomial time) Boolean function , over the basis U_2. That is, we obtain a lower bound of $5n - o(n)$ for the number of $\{and, or\}$ gates needed to compute a certain Boolean function, over the basis $\{and, or, not\}$ (where the *not* gates are not counted). Our proof is based on a new combinatorial property of Boolean functions, called *Strongly-Two-Dependence*, a notion that may be interesting in its own right. Our lower bound applies to any Strongly-Two-Dependent Boolean function.

1 Introduction

In 1949 Shannon [1] showed that the circuit complexity of almost all Boolean functions is exponential. Shannon's proof is based on a counting argument and hence does not supply an explicit (constructible in deterministic polynomial time) Boolean function which actually has exponential circuit complexity. Finding lower bounds for explicit Boolean functions in the general (non-restricted) model is a central problem in computer science, yet only linear lower bounds have been shown. Lower bounds for explicit Boolean functions were proved for some restricted models of Boolean circuits (e.g., monotone circuits, constant depth circuits, etc').

The following lower bounds were proved for circuits over the base B_2, where B_2 is the base that includes all Boolean functions over two Boolean variables. In 1974 Schnorr [2] proved a lower bound of $2n$. Then Paul [4] proved a $2.5n$-lower bound. Stockmeyer [3] gave the same $2.5n$ bound for a larger family of functions. Blum [5] improved this bound to $2.75n$ and in 1984 [6] proved a lower bound of $3n$. All these results where proved by using the so-called "gate-elimination" approach. The $3n$-bound is still the best result for this model.

In this paper, we consider Boolean circuits over the basis U_2. The basis U_2 is one of the most common basis for Boolean circuits. It contains all the Boolean functions over two variables, except for the *xor* function and its complement.

⋆ research supported by a Israel Science foundation (ISF) grant

Note that any gate over the basis U_2 can be replaced by an *and* gate (or, equivalently, an *or* gate), with the optional addition of *not* gates connected directly to the inputs of the gate, and directly at the output of the gate. Thus, any Boolean circuit over U_2 can be converted into a Boolean circuit over the basis $\{and, or, not\}$, with the exact same number of gates (when the *not* gates are not counted). The circuit complexity of a function over U_2 is equivalent to counting the number of $\{and, or\}$ gates needed to compute the function (when the *not* gates are ignored).

The first lower bound on the size of circuits over the basis U_2, was obtained by Zwick [7] in 1991, he gave a lower bound of $4n - O(1)$ for functions that belong to a specific subset of the symmetric Boolean functions. This lower bound was improved to $4.5n - o(n)$ by Lachish and Raz [8] after a decade. Lachish and Raz [8] proved their lower bound for functions in a new family of Boolean functions they called *Strongly-Two-Dependent*. One year later Iwama and Morizumi [9] showed a lower bound of $5n - o(n)$ for the same family of Boolean functions. As in the case of Boolean circuits over the basis B_2 all these results where proved by using the so-called "gate-elimination" approach. In this paper we combine the works of Lachish and Raz [8] and Iwama and Morizumi [9].

2 Boolean Circuits over U_2

The basis U_2 contains all the Boolean functions over two variables, except for the the *xor* function and its complement. Note that any gate over the basis U_2 can be replaced by an *and* gate (or, equivalently, an *or* gate), with the optional addition of *not* gates connected directly to the inputs to the gate and to the output of the gate. Thus we view the basis U_2 as the set of all Boolean functions $f : \{0,1\}^2 \rightarrow \{0,1\}$ of the sort

$$f(x, y) = ((x \oplus a) \wedge (y \oplus b)) \oplus c,$$

where $a, b, c \in \{0,1\}$. In this paper we deal only with Boolean circuits over the basis U_2. A Boolean circuit over the basis U_2 is a directed acyclic graph with nodes of in-degree 0 or 2. Nodes of in-degree 0 are called *input-nodes*, and each one of them is labeled by a variable in $\{x_1, \cdots, x_n\}$ or a constant 0 or 1. Input-nodes labeled by a constant are called *constant-nodes*. Nodes of in-degree 2 are called *gate-nodes*, and each one of them has two *inputs* and an *output*, and is labeled by a function in U_2. There is a single specific node of out-degree 0 called the *output-node*. If one input of the gate-node is constant then the output is constant or depends on the other input, i.e., the same or its negation. In the former case, the gate-node is called *blocked-gate*. In the latter case, the gate-node is called *through-gate*. For nodes u and v, $u \rightarrow v$ means that the output of the node u is directly connected to one of the v's inputs. $u \xrightarrow{*} v$ means that there is a path from u to v. For a Boolean circuit C, $OUT_C(v)$ denotes the set of gate-nodes, u, such that $v \rightarrow u$. Also $IN_C(v)$ denotes the set of input-nodes, u, such that $u \xrightarrow{*} v$.

Let $X = \{x_1, \cdots, x_n\}$ be the set of input-variables. Given an assignment $\sigma \in \{0,1\}^n$ to the variables in X, we denote by $C(\sigma)$ the value of the output of the circuit C on the assignment $x_i = \sigma_i, 1 \leq i \leq n$. Similarly, for any node

v in the circuit C, we denote by $C_v(\sigma)$ the value of the output of the gate-node v on the assignment $x_i = \sigma_i$. We say that two Boolean circuits C_1 and C_2 are equivalent ($C_1 \equiv C_2$) if they compute the same function. Without loss of generality, we can assume that for every input-variable x_i, there is only one input-node labeled by x_i.

The *size* of a circuit C is the number of gate-nodes in it. We denote this number by $Size(C)$. The circuit complexity of a Boolean function $F : \{0,1\}^n \to \{0,1\}$ is the minimal size of a Boolean circuit that computes F. We denote this number by $Size(F)$. The depth of a node v in a Boolean circuit C is the length of the longest path from v to the output-node, denoted by $Depth_C(v)$. The depth of a circuit C, $Depth(C)$, is the maximal depth of a node v in the circuit. The *degree* of a node v in a Boolean circuit C, denoted by $Degree_C(v)$, is the node's out-degree. We denote by $Degeneracy(C)$ the number of input-variables that have degree one in C. Let x be an input-variable that has degree one in C. Then a node v is called *degenerate* if $x \to v$. Otherwise, v is called *non-degenerate* or *ND*. For our lower bound proof, we use the following measure (see the next section for its purpose):

$$SD(C) = Size(C) - Degeneracy(C).$$

Recall that each gate-node v, having inputs x and y, has the functionality defined by $f(x,y) = ((x \oplus a) \wedge (y \oplus b)) \oplus c$. If we assign value a to x then the value of its output is fixed regardless of the other input y. In this case, we say that fixing $x = a$ *blocks* the gate-node v or simply x *blocks* v. Similarly for y.

A *restriction* θ is a mapping from a set of n variables to $\{0,1,\star\}$. We apply a restriction θ to a Boolean function $F : \{0,1\}^n \to \{0,1\}$ in the following way: for any variable x_i that is mapped by θ to a constant $a_i \in \{0,1\}$, we assign a_i to x_i. We leave all the other variables untouched. We refer to the resulting Boolean function by $F|_\theta$. We use the similar notation, $C|_\theta$, for a Boolean circuit C.

3 Strongly Two Dependent Boolean Functions

Let $F : \{0,1\}^n \to \{0,1\}$ be a Boolean function and $F[i,j,a,b]$, $1 \le i < j \le n$, $a, b \in \{0,1\}$, be a Boolean function $F|_{\theta[i,j,a,b]}$ where $\theta[i,j,a,b]$ is a restriction that maps x_i and x_j to a and b, respectively. F is called *Two-Dependent* if for any i and j, $1 \le i < j \le n$, $F[i,j,0,0], F[i,j,0,1], F[i,j,1,0]$ and $F[i,j,1,1]$ are all different functions. (For example, if F is a symmetric function then it is not Two-Dependent since $F[i,j,0,1] = F[i,j,1,0]$.) Let $X_m \subseteq \{x_1, \cdots x_n\}$ be a set of m variables, and θ_m be a restriction which maps X_m to {0,1}. Then F is called *(n,k)-Strongly-Two-Dependent* if $F|_{\theta_m}$ is always Two-Dependent for any $0 \le m \le n-k$, any X_m and any θ_m. (If F is (n,k)-Strongly-Two-Dependant then $F|_{\theta_m}$ is obviously $(n-m,k)$-Strongly-Two-Dependent.) It is proved in [8] that an (n,k)-Strongly-Two-Dependent Boolean function for any (sufficiently large) integer n and $k = O(\log n)$ can be constructed explicitly at polynomial time by using a small number of auxiliary variables. We do not present this construction here since as pointed out by Ingo Wegener a $k-mixed$ Boolean function ([10] pages 135–137) is also strongly two dependent and Savickỳ and Žák [11] have shown an explicit construction for such a Boolean Function.

Two-Dependent functions have the following property:

Proposition 3.1. *Let $F : \{0,1\}^n \rightarrow \{0,1\}$ be a Two-Dependent Boolean function over the set of variables $X = \{x_1, \cdots, x_n\}$. Let C be a Boolean circuit that computes F. Then, the following is never satisfied in C: There exist two input variables x_i, x_j such that $OUT_C(x_i) = OUT_C(x_j)$ and $|OUT_C(x_i)| = |OUT_C(x_j)| = 2$ (i.e., x_i, x_j are connected directly to the same two gate-nodes).*

Proof. Let F, C be as in the proposition. Assume for the sake of contradiction that there exist x_i, x_j as in the proposition. Without loss of generality, assume that $i = 1$ and $j = 2$. Let v_1,v_2 be the two different gate nodes, such that, $OUT_C(x_1) = OUT_C(x_2) = \{v_1, v_2\}$. Since v_1, v_2 are labeled by Boolean functions from U_2 there exist two different restrictions σ_1, σ_2 that map x_1, x_2 to $\{0,1\}$ and all other variables to $\star$, such that $C_{v_1}(\sigma_1) = C_{v_1}(\sigma_2)$ and $C_{v_2}(\sigma_1) = C_{v_2}(\sigma_2)$. Note that this is true even if the gates where labeled by Boolean functions from B_2. Thus $C\mid_{\sigma_1} \equiv C\mid_{\sigma_2}$. Hence the Boolean function C computes is not Two-Dependent. Yet F is Two-Dependent. ■

The following two properties are also important in the lower-bound proof. The first one says that a restriction does not "cut" all the paths from a non-restricted input-gate to the final output. The second one says that if a gate v is degenerate, i.e., one of its inputs is connected to x_i such that $|OUT_C(x_i)| = 1$, then the other input of v has paths from many different input-gates.

Proposition 3.2. *Let $F : \{0,1\}^n \rightarrow \{0,1\}$ be an (n, k)-Strongly-Two-Dependent Boolean function over the set of variables $X = \{x_1, \cdots, x_n\}$. Let C be a Boolean circuit that computes F. Then, the following is never satisfied in C: There exist an input-variable x_i, a set X' of at most $n - k$ other input-variables and a restriction θ that maps each input-variable in X' to a constant in $\{0,1\}$, such that, in $C|_\theta$ every path that connects x_i to the output-node contains a gate-node that computes a constant function.*

Proof. Let F, C be as in the proposition. Assume for the sake of contradiction that the case described in the proposition occurs and that x_i is the variable for which the case occurs. Note that for a restriction θ as described in the proposition $C\mid_\theta$ does not depend on the value assigned to x_i. Hence the Boolean function $C\mid_\theta$ computes is not Two-Dependent. Yet $F\mid_\theta$ is Two-Dependent. ■

The following corollary is a special case of the previous proposition

Corollary 3.3. *Let $F : \{0,1\}^n \rightarrow \{0,1\}$ be an (n, k)-Strongly-Two-Dependent Boolean function and let C be a Boolean circuit that computes F. Let v be a gate-node in C and let v' be the node such that $v' \rightarrow v$. Assume that $x_i \rightarrow v$ for an input-variable x_i such that $Degree_C(x_i) = 1$ (i.e., the node v is degenerate.) Then, if the node v' computes a non constant function, then $|IN_C(v')| > n - k$.*

For the gate-elimination, it is convenient if the circuit does not include restricted cases, i.e., those that do not contribute to the computation process of the Boolean circuit. The following propositions gives a method of removing such gates without increasing the SD measure of the circuit.

Proposition 3.4. *Let $F : \{0,1\}^n \to \{0,1\}$ be a Two-Dependent Boolean function and let C be a Boolean circuit that computes F. Assume that C contains one of the following degenerate cases:*

1. *A gate-node v such that a constant node is connected directly to v.*
2. *A gate-node v such that for some constant $a \in \{0,1\}$ and any assignment $\sigma \in \{0,1\}^n$, we have $C_v(\sigma) = a$.*
3. *A gate-node v which is not the output of the circuit such that $Degree_C(v) = 0$.*
4. *A gate-node v such that its two inputs are connected to the same gate.*
5. *An input-variable x_i such that $|OUT_C(x_i)| \geq 2$ and there exists $u, v \in OUT_C(x_i), u \to v$.*

Then, there exists a Boolean circuit $C' \equiv C$ such that $SD(C) \geq SD(C')$ and C' does not contain any of the degenerate cases.

Proof. Let C be as in the proposition. We prove the first case, the third case and the last case. The proof of all the other cases is similar.

Let v be a gate-node such that a constant node labeled by $a \in \{0,1\}$ and a node labeled by u are connected directly to it. Then either v is a through-gate or it is blocked. Thus we can remove the gate v from C and get a new circuit C' that computes the same Boolean function as C computes and $Size(C) > Size(C')$. Observe that if u is a degenerate variable in C and a non degenerate in C' then $Degeneracy(C') = Degeneracy(C) - 1$ and otherwise $Degeneracy(C') \geq Degeneracy(C)$. Hence $SD(C') \leq SD(C)$.

Let v be a non-output gate-node such that $Degree_C(v) = 0$. No input variable of degree one can be connected to v since if there exists such input variable, the output of C does not depend on the input variable, which contradicts the assumption that F is a Two-Dependent. Hence we can remove the gate v from C and get a new circuit C' such that $SD(C') \leq SD(C)$.

Let x_i be such that $|OUT_C(x_i)| \geq 2$ and there exist $u, v \in OUT_C(x_i), u \to v$. Let w be the other node such that $w \to u$. Observe that we can disconnect u from v, connect w to v instead and relabel v in manner such that we get a new circuit C' that computes the same Boolean function as C computes. Since the number of gates in C and C' is the same and $Degeneracy(C') = Degeneracy(C)$ we get that $SD(C') = SD(C)$.

■

Proposition 3.5. *Let $F : \{0,1\}^n \to \{0,1\}$ be an (n,k)-Strongly-Two-Dependent Boolean function and let C be a Boolean circuit that computes F. Then $Degeneracy(C) \leq k$, if C does not contain any one of the degenerate cases of proposition 3.4.*

Proof. Let F, C be as in the proposition. Assume for the sake of contradiction that $Degeneracy(C) > k$. Let v be a degenerate gate-node such that $Depth_C(v) \geq Depth_C(u)$ for every degenerate gate-node u. An input-variable of degree one is connected directly to v, and let w be the other node which is connected directly to v. Since we selected v as above none of the input-variable of degree one is in $IN_C(w)$ and hence $|IN_C(w)| < n-k$. This contradicts Corollary 3.3.

■

4 The lower bound

4.1 The lower bound

In this section we prove following Lemma 4.1, the lower bound Theorem (Theorem 4.2) is a direct result of this Lemma.

Lemma 4.1. *Let $F : \{0,1\}^n \rightarrow \{0,1\}$ be an (n,k)-Strongly-Two-Dependent Boolean function and assume that $n-k \geq k+4$ and $n-k \geq 5$. Let C be a Boolean circuit that computes F. Then, there exists a set of one or two input-variables X' (i.e., $|X'| \leq 2$) and a constant $c_i \in \{0,1\}$ for each $x_i \in X'$ such that for the restriction θ that maps each variable $x_i \in X'$ to c_i, the following is satisfied: There exists a Boolean circuit $C' \equiv C|_\theta$ such that*

$$SD(C) \geq SD(C') + 5 \cdot |X'|.$$

Theorem 4.2. *Let $F : \{0,1\}^n \rightarrow \{0,1\}$ be an (n,k)-Strongly-Two-Dependent Boolean function such that $k = o(n)$. Then,*

$$Size(F) \geq 5n - o(n)$$

Proof. Let C a Boolean circuit that computes F. We generate a sequence of Boolean circuit $C_0, ..C_l$ by iteratively applying Lemma 4.1 to C. (Note that this is possible by the definition of Strongly-Two-Dependent). More formally, we have $C_0 = C$ and C_{i+1} is obtained from C_i by applying Lemma 4.1. We stop when the number of remaining input-variables is smaller than $2k+4$ or $k+5$. By Lemma 4.1, $SD(C) \geq SD(C_l) + 5n - o(n)$. By Proposition 3.5, we can assume that $Degeneracy(C) \leq k$. Therefore, $Size(C) \geq 5n - o(n)$, which immediately implies the theorem. ■

4.2 Preliminaries for the Proof of Lemma 4.1

In this and the next sections (4.2 and 4.3), we always treat Boolean circuits which compute (n,k)-Strongly-Two-Dependent Boolean functions such that $n-k \geq k+4$ and $n-k \geq 5$, which is often omitted to mention. Also, we always assume that the circuits do not include degenerate cases described in Proposition 3.4. Those nodes can be removed without increasing SD as mentioned in its proof. Furthermore we can always assume that the number of degenerate variables is at most k by Proposition 3.5. Our argument in the rest of the paper has the standard structure, which is explained in the proof of our first lemma:

Lemma 4.3. *Suppose that there is an input-variable x_i, such that, (i) $OUT_C(x_i) = \{v_1, v_2, v_3\}$ and (ii) $OUT_C(v_1) \cup OUT_C(v_2) \cup OUT_C(v_3)$ includes at least three ND gate-nodes. Then SD decreases by at least five by fixing x_i appropriately.*

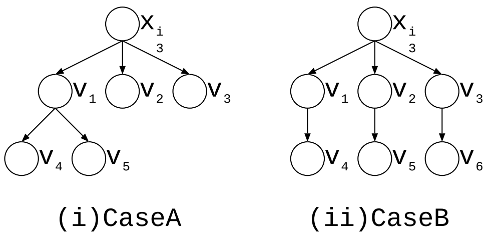

Fig. 1. Lemma 4.3

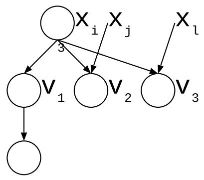

Fig. 2. Lemma 4.4

Proof. Since $OUT_C(v_1) \cup OUT_C(v_2) \cup OUT_C(v_3)$ includes at least three ND gate-nodes, considering the following two cases is enough:

Case A $OUT_C(v_1)$ or $OUT_C(v_2)$ or $OUT_C(v_3)$ includes at least two different ND gate-nodes: without loss of generality, we can assume that $OUT_C(v_1)$ includes such gate-nodes. (see Fig. 1 (i).) One can see that, by fixing x_1 appropriately, we can block v_1, which allows us to remove v_4 and v_5, too. v_2 and v_3 can also be removed. Note that the gate-nodes v_1 to v_5 are all different by Proposition 3.4 and v_4 to v_5 are ND gates by the assumption of the lemma. v_1 to v_3 are also ND by Corollary 3.3. Hence removing v_1 to v_5 does not decrease *Degeneracy*(C). (*Degeneracy*(C) may increase, but that is not important for us since *increasing* Degeneracy forces SD to decrease.) To summarize all these situations, we write as follows (when a gate is removed since its output is fixed (e.g., by being blocked), we say that the gate is "killed"):

Fix x_i s.t. v_1 blocked $\Rightarrow$ Killed: v_1, Removed: v_1, v_2, v_3, v_4, v_5.

Non degenerate: v_1, v_2, v_3 (by Corollary 3.3) v_4, v_5 (by (ii)) $\Rightarrow$ *Degeneracy*: ± 0.

Case B Each of $OUT_C(v_1)$, $OUT_C(v_2)$ and $OUT_C(v_3)$ includes at least one ND gate-node, v_4, v_5 and v_6, respectively, which are all different (see Fig. 1 (ii)). One can see that, by fixing x_1 appropriately, we can block at least two of v_1, v_2 and v_3 regardless of their gate-types. Without loss of generality, we assume that v_1 and v_2 are blocked, which allows us to remove v_4 and v_5, too. v_3 can also be removed. Note that the gate-nodes v_1 to v_5 are all different by Proposition 3.4 and v_4 to v_5 are ND gates by the assumption of the lemma. v_1 to v_3 are also ND by Corollary 3.3. To summarize:

Fix x_i s.t. v_1, v_2 blocked $\Rightarrow$ Killed: v_1, v_2, Removed: v_1, v_2, v_3, v_4, v_5.

Non degenerate: v_1, v_2, v_3 (by Corollary 3.3) v_4, v_5 (by (ii)) $\Rightarrow$ *Degeneracy*: ± 0. ∎

Lemma 4.4. *Suppose that there is an input-variable x_i, such that, (i) $OUT_C(x_i) = \{v_1, v_2, v_3\}$, (ii) $OUT_C(v_1)$ includes at least one ND gate-node and (iii) $IN_C(v_2) = \{x_i, x_j\}$ and $IN_C(v_3) = \{x_i, x_l\}$ where x_j and x_l are both input-variables such that $i \neq j$, $i \neq l$. Then, SD decreases by at least five by fixing x_i appropriately.*

Proof. See Fig. 2. Three main cases, A, B and C exists:

Case A $OUT_C(v_1) \cap OUT_C(v_2) = \emptyset$ and $OUT_C(v_1) \cap OUT_C(v_3) = \emptyset$ and $OUT_C(v_2) \cap OUT_C(v_3) = \emptyset$: See Fig. 3 (i). v_4 is an ND gate-node guaranteed by (ii) above. v_4, v_5 and v_6 are all different gate-nodes by the condition of the case. v_5 and v_6 are also ND by Corollary 3.3. (By fixing x_i and x_j, we can block v_5. Similarly for v_6.) Thus, we can apply Lemma 4.3.

Case B $OUT_C(v_1) \cap OUT_C(v_2) \neq \emptyset$ or $OUT_C(v_1) \cap OUT_C(v_3) \neq \emptyset$: without loss of generality, assume that $OUT_C(v_1) \cap OUT_C(v_2) \neq \emptyset$. See Fig. 3 (ii). Two sub cases exist:

Case B.1 Suppose that we can fix x_i such that it blocks v_1, v_2: There is at least one ND gate-node, say v_6, in $OUT_C(v_1) \cup OUT_C(v_2) \cup OUT_C(v_4)$ other than v_4 for the following reason: Suppose that all gate-nodes (except v_4) in $OUT_C(v_1) \cup OUT_C(v_2) \cup OUT_C(v_4)$ are degenerate. Then by setting appropriate values to the input-nodes connected to these degenerate nodes and by setting x_l to block v_3, all the paths from x_i are blocked. Since the number of degenerate gate-nodes is at most k, this fact contradicts Proposition 3.2. Note that v_6 is obviously different from v_1, v_2 or v_4 and it is also different from v_3 whose two inputs are both input-nodes. To summarize:

Fix x_i s.t. v_1, v_2 blocked $\Rightarrow$ Killed: $v_1, v_2 \rightarrow v_4$, Removed: v_1, v_2, v_3, v_4, v_6.

Non degenerate: v_1, v_2, v_3 (by Corollary 3.3) v_4 (obvious) v_6 (mentioned above)

$\Rightarrow$ *Degeneracy*: ± 0.

Remark The above argument breaks if v_4 is the output gate since the paths from x_1 *to the output gate* can no longer be blocked. However, v_4 cannot be the output gate since it is killed only by fixing a few input nodes. In the following we often omit mentioning this fact in similar situations.

Case B.2 We can fix x_i such that it blocks v_1, v_3 or v_2, v_3: Without loss of generality, we assume that v_1 and v_3 are blocked.

Fix x_i s.t. v_1, v_3 blocked $\Rightarrow$ Killed: v_1, v_3, Removed: v_1, v_2, v_3, v_4, v_5.

Non degenerate: v_1, v_2, v_3, v_5 (by Corollary 3.3) v_4 (by (ii))

$\Rightarrow$ *Degeneracy*: ± 0.

Case C $OUT_C(v_2) \cap OUT_C(v_3) \neq \emptyset$: v_4 is an ND gate-node guaranteed by (ii) above. See Fig. 3 (iii).

Case C.1 Suppose that we can fix x_i such that it blocks v_2 and v_3: If $OUT_C(v_5)$ does not includes v_1, $OUT_C(v_5)$ must include a gate-nodes, say v_6, that is different from v_1, v_2, v_3 or v_5 and is ND (by Corollary 3.3 since $IN_C(v_5) = 3$). Such a case is proved like Case B.1. If $OUT_C(v_5)$ includes v_1, then we fix x_i such that it blocks v_2 and v_3, which kills v_5 and then kills v_1 also. To summarize:

Fix x_i s.t. v_2, v_3 blocked $\Rightarrow$ Killed: $v_2, v_3 \rightarrow v_5 \rightarrow v_1$, Removed: v_1, v_2, v_3, v_4, v_5.

Non degenerate: v_1, v_5 (obvious) v_2, v_3 (by Corollary 3.3) v_4 (by (ii))

$\Rightarrow$ *Degeneracy*: ± 0.

Case C.2 We can fix x_i such that it blocks v_1, v_2 or v_1, v_3: Without loss of generality, we assume that v_1 and v_2 are blocked.

Fix x_i s.t. v_1, v_2 blocked $\Rightarrow$ Killed: v_1, v_2, Removed: v_1, v_2, v_3, v_4, v_5.

Non degenerate: v_1, v_2, v_3 (by Corollary 3.3) v_4 (by (ii)) v_5 (obvious)

$\Rightarrow$ *Degeneracy*: ± 0.

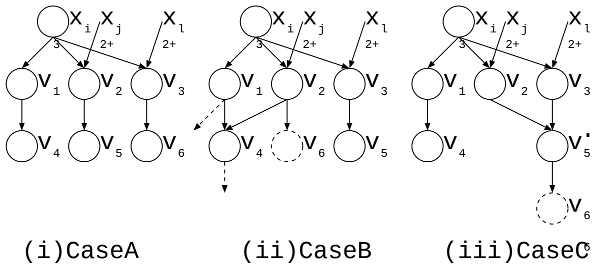

Fig. 3. Main cases of Lemma 4.4

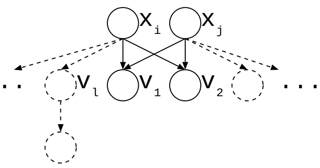

Fig. 4. Lemma 4.5

■

Lemma 4.5. *Suppose that there are two input-variables x_i, x_j, such that $OUT_C(x_i) \supseteq \{v_1, v_2\}$ and $OUT_C(x_j) \supseteq \{v_1, v_2\}$ and $OUT_C(x_i) \cup OUT_C(x_j) \neq \{v_1, v_2\}$. Then, $OUT_C(x_i) \cup OUT_C(x_j)$ includes at least one gate-nodes v_l such that v_l is different from v_1, v_2, and $OUT_C(v_l)$ includes at least one ND gate-node.*

Proof. See Fig. 4. Suppose that there are no such v_l. Then all gate-nodes, say u, except v_1 and v_2 in $OUT_C(x_i) \cup OUT_C(x_j)$ (if any) are connected to degenerate nodes. Those degenerate gate-nodes are blocked by their corresponding inputs, by which we can remove all such u's. Thus, by setting at most k input variable, the circuit is converted to C' such that (i) C' is still Strongly-Two-Dependent by the definition of Strongly-Two-Dependent and (ii) $OUT_C(x_i) = OUT_C(x_j) = \{v_1, v_2\}$. But this contradicts Proposition 3.1. ■

4.3 Proof of Lemma 4.1

Let $F : \{0,1\}^n \rightarrow \{0,1\}$ be an (n,k)-Strongly-Two-Dependent Boolean function and assume that $n-k \geq k+4$ and $n-k \geq 5$. Let C be a Boolean circuit that computes F. Let v_1 be a gate-node such that $Depth(v_1) = Depth(C) - 1$ (we can always find such v_1). The nodes that are connected to v_1 are both input-variables, say x_1 and x_2. By Corollary 3.3, $Degree_C(x_1) \geq 2, Degree_C(x_2) \geq 2$. Four main cases exist:

Case 1 $Degree_C(x_1) \geq 4$ or $Degree_C(x_2) \geq 4$: See Fig. 5. This case is easy.
Fix x_1 s.t. v_1 blocked $\Rightarrow$ Killed: v_1, Removed: v_1, v_2, v_3, v_4, v_5.
Non degenerate: v_1, v_2, v_3, v_4, v_5 (by Corollary 3.3)
$\Rightarrow$ *Degeneracy*: ± 0.

Case 2 $Degree_C(x_1) = 3$ and $Degree_C(x_2) = 3$: Three sub cases exist:

Case 2.1 $|OUT_C(x_1) \cap OUT_C(x_2)| = 3$: See Fig. 6 (i). v_4 is ND by Corollary 3.3. Thus, by Lemma 4.4, SD decreases by at least five by fixing x_1 appropriately.

Case 2.2 $|OUT_C(x_1) \cap OUT_C(x_2)| = 2$: See Fig. 6 (ii). By Lemma 4.5, $OUT_C(v_3)$ or $OUT_C(v_4)$ includes an ND gate-node. Without loss of generality, assume that $OUT_C(v_3)$ includes an ND gate-node, say v_5. Thus, SD decreases by at least five by fixing x_1 by Lemma 4.4.

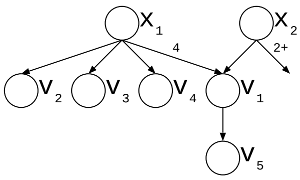

Fig. 5. Case 1

Case 2.3 $|OUT_C(x_1) \cap OUT_C(x_2)| = 1$: See Fig. 6 (iii). Let v_6 be a gate-node in $OUT_C(v_1)$. Without loss of generality, we can assume that $Depth_C(v_6) = Depth(C) - 2$. By the condition of Case 2.3, v_1 through v_5 are all different. By Proposition 3.4, v_6 is different from v_2 through v_5. Thus, v_1 through v_6 are all different. Four sub cases exist: Let w be a node ($\neq v_1$) such that $w \rightarrow v_6$.

Case 2.3.1 $Degree_C(v_1) \geq 2$: SD decreases by at least five by fixing x_1 such that v_1 is blocked.

Case 2.3.2 $Degree_C(v_1) = 1$ and the node w is equal to v_2, v_3, v_4 or v_5: without loss of generality, assume that $w = v_2$. See Fig. 7. Since $Depth_C(v_6) = Depth(C) - 2$, the nodes that are connected to v_2 are both input-variables. By setting x_2 to block v_1 and x_i to block v_2, all the paths from x_1 except the path through v_3 are blocked. By this fact and Proposition 3.2, $OUT_C(v_3)$ includes at least one ND gate-node. Thus, by Lemma 4.4, SD decreases by at least five by fixing x_1.

Case 2.3.3 $Degree_C(v_1) = 1$ and w is not equal to v_2, v_3, v_4 or v_5, and w is not an input-node (Case 2.3.4 is the case that w is an input-node): Let w be v_7 and see Fig. 8. Since v_7 is obviously different from v_1 or v_6, v_1 through v_7 are all different. Note that $Depth_C(v_7) = Depth(C) - 1$ since $Depth_C(v_6) = Depth(C) - 2$, which means the nodes connected to v_7 are both input-variables. By Corollary 3.3, the degree of these two input-variables are two or more. In the following, we only prove the case that $Degree_C(v_7) = 1$. If $Degree_C(v_7) \geq 2$, then we can apply Case 1, 2.1, 2.2, 2.3.1, 3 or 4.

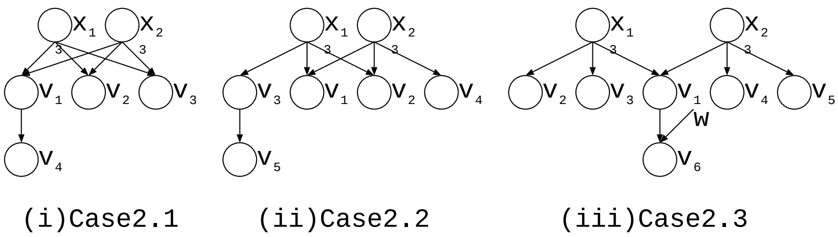

Fig. 6. Sub cases of Case 2

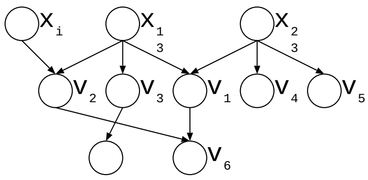

Fig. 7. Case 2.3.2

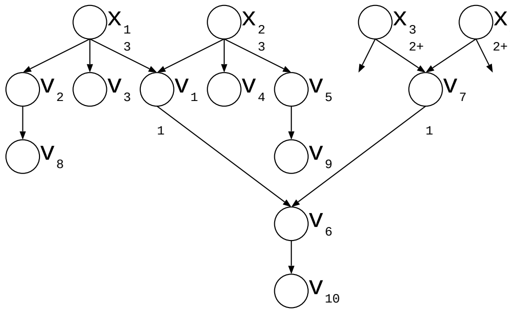

Fig. 8. Case 2.3.3

Suppose that there are two ND gate-nodes in $OUT_C(v_2) \cup OUT_C(v_3)$. Since these gates are obviously different from v_6, $OUT_C(v_1) \cup OUT_C(v_2) \cup OUT_C(v_3)$ includes three ND gate-nodes. Thus we can apply Lemma 4.3. Otherwise, suppose that $OUT_C(v_2) \cup OUT_C(v_3)$ includes only degenerate nodes. Then we can block all the paths from x_1 by setting the input-nodes corresponding to those degenerate nodes and x_2 (to block v_1), which contradicts Proposition 3.2. Thus, from now on we can assume that $OUT_C(v_2) \cup OUT_C(v_3)$ includes exactly one ND gate-node, say v_8. Suppose that $OUT_C(v_8)$ includes no ND gate-nodes. Then the similar contradiction to Proposition 3.2 happens. Thus, $OUT_C(v_8)$ includes one or more ND gate-node. Without loss of generality, we can assume that v_8 is in $OUT_C(v_2)$. Similarly for v_9. Also, let v_{10} be a gate-nodes in $OUT_C(v_6)$. Now all gates are illustrated in Fig. 8.

Since v_8 is different from v_3 by Proposition 3.4 (and others are obvious),

$$v_8 \neq v_1, v_2, v_3 \text{ or } v_6. \qquad (1)$$

Similarly

$$v_9 \neq v_1, v_4, v_5 \text{ or } v_6. \qquad (2)$$

Since two inputs of v_7 are both input-nodes,

$$v_7 \neq v_8, v_9 \text{ or } v_{10}. \qquad (3)$$

Finally, it is obvious that

$$v_{10} \neq v_1 \text{ or } v_6. \qquad (4)$$

See Table 1, where $(1)^*$ in the (v_1, v_8)-entry means that v_1 must be different from v_8 and that was claimed in (1) above. (5) in the (v_4, v_8)-entry means that the case that $v_4 = v_8$ is considered in (5) below. Recall that v_1 through v_7 are all different. Now three sub cases exist:

Table 1. Case 2.3.3

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}
v_8	$(1)^*$	$(1)^*$	$(1)^*$	(5)	(5)	$(1)^*$	$(3)^*$	-	-	-
v_9	$(2)^*$	(5)	(5)	$(2)^*$	$(2)^*$	$(2)^*$	$(3)^*$	(7)	-	-
v_{10}	$(4)^*$	(6)	(6)	(6)	(6)	$(4)^*$	$(3)^*$	(6)	(6)	-

Case 2.3.3.1 We can fix x_1 such that it blocks v_1 and v_2 or we can fix x_2 such that it blocks v_1 and v_5: Without loss of generality, we assume that we can fix x_1 such that it blocks v_1 and v_2. SD decreases by at least five by fixing x_1 such that v_1 and v_2 are blocked.

Case 2.3.3.2 If v_1 is blocked, then its output blocks v_6: If v_6 is killed, then v_7 can be removed since $Degree_C(v_7) = 1$. Therefore:

Fix x_1 s.t. v_1 blocked $\Rightarrow$ Killed: $v_1 \rightarrow v_6$, Removed: v_1, v_2, v_3, v_6, v_7.

Non degenerate: v_1, v_6, v_7 (obvious) v_2, v_3 (by Corollary 3.3)

$\Rightarrow$ *Degeneracy*: ± 0.

Case 2.3.3.3 Neither Case 2.3.3.1 nor Case 2.3.3.2 applies: Further sub cases exist:

Case 2.3.3.3.1 v_8 is equal to v_4 or v_5, or v_9 is equal to v_2 or v_3 (denoted by (5) in Table 1): Assume that v_8 is equal to v_4. We can block v_4 $(= v_8)$ by x_2 and can kill v_6 by x_3 and x_4 (through v_7). Since we are now assuming the case that $OUT_C(v_2) \cup OUT_C(v_3)$ does not include ND gate-nodes other than v_8, this fact contradicts Proposition 3.2 (all the paths from x_1 can be blocked). Similarly for the case that $v_5 = v_8, v_2 = v_9$ and $v_3 = v_9$.

Case 2.3.3.3.2 v_{10} is equal to v_2, v_3, v_4, v_5, v_8 or v_9 (denoted by (6) in Table 1): Assume that v_{10} is equal to v_2. We can block v_1 by x_2, and we can kill v_2 $(= v_{10})$ by x_3 and x_4 since we are now assuming that if v_1 is blocked v_6 becomes a through-gate. Hence, $OUT_C(v_3)$ must include an ND gate, say u, by Proposition 3.2. Recall that we are now assuming that $OUT_C(v_2) \cup OUT_C(v_3)$ has only one ND gate-node (the other cases were already discussed). Hence u must be v_8, namely, both v_2 and v_3 are connected to v_8.

On the other hand, when we assume that v_{10} is equal to v_3, $|IN_C(v_{10})| = 4$ and hence v_{10} $(= v_3)$ is not connected to a degenerate gate by Corollary 3.3. Since we are now assuming that $OUT_C(v_2) \cup OUT_C(v_3)$ has only one ND gate-node, both v_2 and v_3 are connected to v_8.

Let u_1 be an ND gate-node in $OUT_C(v_8)$ which must exist by Proposition 3.2. Now, if we can fix x_1 such that it blocks v_1 and v_3, then:

Fix x_1 s.t. v_1, v_3 blocked $\Rightarrow$ Killed: v_1, v_3, Removed: v_1, v_2, v_3, v_6, v_8.

Non degenerate: v_1, v_6, v_8 (obvious) v_2, v_3 (by Corollary 3.3)

$\Rightarrow$ *Degeneracy*: ± 0.

if we can fix x_1 such that it blocks v_2 and v_3, then:

Fix x_1 s.t. v_2, v_3 blocked $\Rightarrow$ Killed: $v_2, v_3 \rightarrow v_8$, Removed: v_1, v_2, v_3, v_8, u_1.

Non degenerate: v_1, v_8 (obvious) v_2, v_3 (by Corollary 3.3) u_1 (above)

$\Rightarrow$ *Degeneracy*: ± 0.

Similarly for the case that $v_4 = v_{10}$ and $v_5 = v_{10}$.

Assume that v_{10} is equal to v_8. We can block v_1 by x_2, and we can kill v_8 $(= v_{10})$ by x_3, x_4 since we are now assuming that if v_1 is blocked v_6 becomes a through-gate. Since we are now assuming the case that $OUT_C(v_2) \cup OUT_C(v_3)$ includes only one ND gate-node $(= v_8)$, this fact contradicts Proposition 3.2 (all the paths from x_1 can be blocked). Similarly for the case that $v_9 = v_{10}$.

Case 2.3.3.3.3 Now one can see that what remains to be considered is the case that $v_8 = v_9$ and the case that all the gates are different. Suppose that v_1

through v_{10} are all different: We block v_2 by x_1 and v_5 by x_2. This assignment kills v_6 (Reason: Recall that we cannot block v_1 and v_2 or v_1 and v_5 at the same time. Hence the current value of neither x_1 nor x_2 blocks v_1. Since we are now assuming that if v_1 is blocked, then its output, say z, does not block v_6, the current output of v_1 must be $\overline{z}$ (otherwise v_1's output would be constant), which does block v_6). To summarize:

Fix x_1 s.t. v_2 blocked and Fix x_2 s.t. v_5 blocked $\Rightarrow$

Killed: $v_1, v_2, v_5 \rightarrow v_6$, Removed: $v_1, v_2, v_3, v_4, v_5, v_6, v_7, v_8, v_9, v_{10}$.

Non degenerate: v_1, v_6, v_7 (obvious) $v_2, v_3, v_4, v_5, v_{10}$ (by Corollary 3.3)

v_8, v_9 (above) $\Rightarrow$ *Degeneracy*: ± 0.

Case 2.3.3.3.4 v_8 is equal to v_9 (denoted by (7) in Table 1): We can assume that all the other gate-nodes are different. Recall that $OUT_C(v_8)$ includes at least one ND gate-node, say, u_2. One can easily see that we can remove this new gate by the same assignment as Case 2.3.3.3.3.(The output values of its two parent nodes are both fixed.) If u_2 is only such ND gate node and is equal to v_3, then we can again claim that $OUT_C(v_3)$ includes a new ND gate-node, say, u_3, which is removed by the same assignment. We can continue this argument for the cases that $u_2 = v_4, u_2 = v_{10}, u_3 = v_4$ and so on.

Case 2.3.4 $Degree_C(v_1) = 1$ and w is an input-node: See Fig. 9. By Corollary 3.3, $Degree_C(x_5) \geq 2$. As before we first show that we can assume that v_1 through v_7 are all different. Suppose that $v_2 = v_7$. Then, since we can block v_1 by x_2 and block v_2 ($= v_7$) by x_5, $OUT_C(v_3)$ includes an ND gate-node by Proposition 3.2. Thus, SD decreases by at least five by fixing x_1 appropriately by Lemma 4.4. Similarly for the case that $v_3 = v_7, v_4 = v_7$ and $v_5 = v_7$. Thus, we can assume that v_7 is different from v_2, v_3, v_4 or v_5. Since v_7 is obviously different from v_1 or v_6, v_1 through v_7 are all different.

Case 2.3.4.1. $Degree_C(x_5) \geq 3$: Let u_6 be a gate-node in $OUT_C(x_5)$ other than v_6 and v_7, and v_{10} be a gate-node in $OUT_C(v_6)$. v_1, v_6, v_7, v_{10} and u_6 are all different by Proposition 3.4. By fixing x_5 such that it blocks v_6, v_1 is removed since v_6 is killed and $Degree_C(v_1) = 1$. To summarize:

Fix x_5 s.t. v_6 blocked $\Rightarrow$ Killed: v_6, Removed: $v_1, v_6, v_7, v_{10}, u_6$.

Non degenerate: v_1, v_6 (obvious) v_7, v_{10}, u_6 (by Corollary 3.3)

$\Rightarrow$ *Degeneracy*: ± 0.

Case 2.3.4.2 $Degree_C(x_5) = 2$: Exactly as before (Case 2.3.3), we can assume, without loss of generality, that $OUT_C(v_2) \cup OUT_C(v_3)$ includes exactly one ND gate-node, say v_8 in $OUT_C(v_2)$. Similarly for v_9. Suppose that v_7 is equal to v_8. Then, by fixing x_5 and x_2 such that they block v_7 and v_1, respectively, we can imply a contradiction to Proposition 3.2. Thus, v_7 is different from v_8. v_7 is different from v_9 similarly and from v_{10} by Proposition 3.4. Thus v_7 is different from all the other gate-nodes. Now all gates are illustrated in Fig. 9.

Now, we can make exactly the same argument as in Case 2.3.3 excepting: (i) When v_6 is killed, v_7 is also killed previously. This time, it is not killed but the degree of x_5 becomes one, which increases $Degeneracy(C)$ by one and decreases SD by one. (ii) Instead of blocking gate-nodes using x_3 and x_4, we can now use x_5.

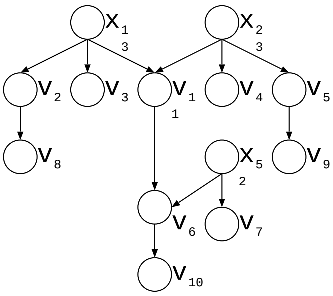

Fig. 9. Case 2.3.4

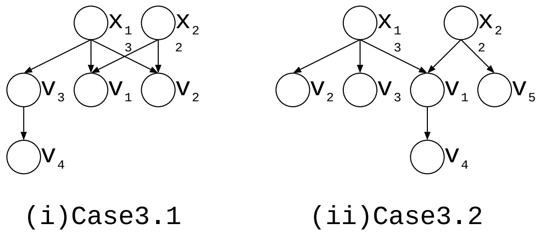

Fig. 10. Sub cases of Case 3

Case 3 $Degree_C(x_1) = 3$ and $Degree_C(x_2) = 2$ or $Degree_C(x_1) = 2$ and $Degree_C(x_2) = 3$: without loss of generality, assume that $Degree_C(x_1) = 3$ and $Degree_C(x_2) = 2$. Two sub cases exist:

Case 3.1 $|OUT_C(x_1) \cap OUT_C(x_2)| = 2$: See Fig. 10 (i). By Lemma 4.5, $OUT_C(v_3)$ includes at least one ND gate-node, say v_4. By Lemma 4.4, SD decreases by at least five by fixing x_1 appropriately.

Case 3.2 $|OUT_C(x_1) \cap OUT_C(x_2)| = 1$: See Fig. 10 (ii). By Proposition 3.4, v_1, v_2, v_3, v_4 and v_5 are all different. As shown below, we can remove only four gate-nodes but at the same time, we can increases $Degeneracy(C)$ by one:

Fix x_1 s.t. v_1 blocked ⇒ Killed: v_1, Removed: v_1, v_2, v_3, v_4.

Non degenerate: v_1 (obvious) v_2, v_3, v_4 (by Corollary 3.3), $Degree_{C'}(x_2) = 1$ ⇒ $Degeneracy$: +1.

Case 4 $Degree_C(x_1) = 2$ and $Degree_C(x_2) = 2$: See Fig. 11. By Proposition 3.1, $OUT_C(x_1) \neq OUT_C(x_2)$. Let v_4 be a gate-node in $OUT_C(v_1)$. Without loss of generality, we can assume that $Depth_C(v_4) = Depth(C) - 2$. By Proposition 3.4, v_1 through v_4 are all different. Four sub cases exist: Let w be a node ($\neq v_1$) such that $w \to v_4$.

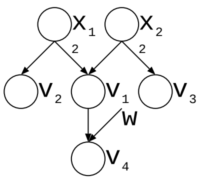

Fig. 11. Case 4

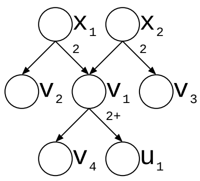

Fig. 12. Case 4.1

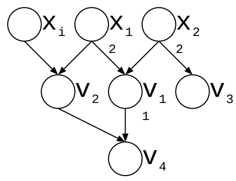

Fig. 13. Case 4.2

Case 4.1 $Degree_C(v_1) \geq 2$: See Fig. 12. By Proposition 3.4, v_1, v_2, v_3, v_4 and u_1 are all different. To summarize:

Fix x_1 s.t. v_1 blocked ⇒ Killed: v_1, Removed: v_1, v_2, v_4, u_1.

Non degenerate: v_1 (obvious) v_2, v_4, u_1 (by Corollary 3.3), $Degree_{C'}(x_2) = 1$ ⇒ $Degeneracy$: +1.

Case 4.2 $Degree_C(v_1) = 1$ and the node w is equal to v_2 or v_3: See Fig. 13. Without loss of generality, we can assume that $w = v_2$. Since $Depth_C(v_4) = Depth(C) - 2$, the nodes that are connected to v_2 are both input-variables. By setting x_2 to block v_1 and x_i to block v_2, all the paths from x_i are blocked, contradicting Proposition 3.2. Thus, this sub case cannot happen.

Case 4.3 $Degree_C(v_1) = 1$ and w is an input-node: See Fig. 14. By Corollary 3.3, $Degree_C(x_3) \geq 2$. Let u_2 be a gate-node in $OUT_C(x_3)$ that is different from v_4. Suppose that u_2 is equal to v_2. By setting x_2 to block v_1 and setting x_3 to block u_2 $(= v_2)$, all paths from x_1 are blocked, contradicting Proposition 3.2. Thus, u_2 is different from v_2, and from v_3 similarly. Thus, v_1 through v_4 and u_2 are all different. By fixing x_3 to block v_4, v_1 is removed since v_4 is killed. Thus:

Fix x_3 s.t. v_4 blocked $\Rightarrow$ Killed: v_4, Removed: v_1, v_4, u_2.

Non degenerate: v_1, v_4 (obvious) u_2 (by Corollary 3.3),

$Degree_{C'}(x_1) = 1$ and $Degree_{C'}(x_2) = 1$ $\Rightarrow$ *Degeneracy*: +2.

Note that v_2 and/or v_3 may also be removed by, e.g., the removed v_4, which only replaces the increase of *Degeneracy*.

Case 4.4 $Degree_C(v_1) = 1$ and w is not equal to v_2 or v_3, and w is not an input-node: Let w be v_5 and see Fig. 15. Since v_5 is obviously different from v_1 or v_4, v_1 through v_5 are all different. Since $Depth_C(v_5) = Depth(C) - 1$, the nodes connected to v_5 are both input-variables of degree two or more. In the following, we only prove the case that $Degree_C(x_4) = Degree_C(x_5) = 2$ and $Degree_C(v_5) = 1$. For the other cases, we can apply the previous cases. Also note that $OUT_C(x_4) \neq OUT_C(x_5)$ by Proposition 3.1.

Assume that v_2 is equal to v_6. By setting x_2 to block v_1 and setting x_4 to block v_2 $(= v_6)$, all paths from x_1 are blocked, contradicting Proposition 3.2. Thus, v_2 is different from v_6. Similarly for $v_2 = v_7, v_3 = v_6$ and $v_3 = v_7$. By Proposition 3.1, v_6 is different from v_7, and hence v_1 through v_7 are all different.

Suppose that $OUT_C(v_2)$ includes only degenerate nodes. Then we can block all the paths from x_1 by setting the input-nodes corresponding to those degenerate nodes and x_2 (to block v_1), which contradicts Proposition 3.2. Thus, we can assume that $OUT_C(v_2)$ includes one or more ND gate-node. Let v_8 be one of such ND gate-nodes. Suppose that $OUT_C(v_2) \cup OUT_C(v_8)$ includes no ND gate-nodes except v_8. Then this again contradicts Proposition 3.2. Thus, we can assume that $OUT_C(v_2) \cup OUT_C(v_8)$ includes one or more ND gate-node except v_8. Similarly for v_9. Also, let v_{10} be a gate-nodes in $OUT_C(v_4)$. Now all gates are illustrated in Fig. 15. See Table 2 for the distinctions of gate-nodes.

It is obvious that

$$v_8 \neq v_1, v_2 \text{ or } v_4,\ v_9 \neq v_1, v_3 \text{ or } v_4,\ v_{10} \neq v_1 \text{ or } v_4. \tag{1}$$

Since two inputs of v_5 are both input-nodes,

$$v_5 \neq v_8, v_9 \text{ or } v_{10}. \tag{2}$$

Recall that v_1 through v_7 are all different. Now three sub cases exist:

Case 4.4.1 We can fix x_1 so as to block v_1 and v_2, or x_2 so as to block v_1 and v_3, or x_4 so as to block v_5 and v_6, or x_5 so as to block v_5 and v_7: without loss of generality, assume that we can fix x_1 such that it blocks v_1 and v_2. It is easy to see that:

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}
v_8	$(1)^*$	$(1)^*$	(3)	$(1)^*$	$(2)^*$	(3)	(3)	-	-	-
v_9	$(1)^*$	(3)	$(1)^*$	$(1)^*$	$(2)^*$	(3)	(3)	(5)	-	-
v_{10}	$(1)^*$	(4)	(4)	$(1)^*$	$(2)^*$	(4)	(4)	(6)	(6)	-

Table 2. Case 4.4.3

Fix x_1 s.t. v_1, v_2 blocked $\Rightarrow$ Killed: v_1, v_2, Removed: v_1, v_2, v_4, v_8.
Non degenerate: v_1, v_4 (obvious) v_2 (by Corollary 3.3) v_8 (above),
$Degree_{C'}(x_2) = 1$ $\Rightarrow$ *Degeneracy*: +1.

Case 4.4.2 If v_1 is blocked then its output blocks v_4, or if v_5 is blocked then its output blocks v_4: without loss of generality, assume the former. Our argument is very similar to Case 2.3.3.2. Instead of removing v_3 in Case 2.3.3.2, $Degeneracy(C)$ increase by one since $Degree_{C'}(x_2)$ is one.

Case 4.4.3 Neither Case 4.4.1 or Case 4.4.2 applies: Further sub cases exist:

Case 4.4.3.1 v_8 is equal to v_3, v_6 or v_7, or v_9 is equal to v_2, v_6 or v_7 (denoted by (3) in Table 2): We only discuss the case that $v_8 = v_3, v_6$ or v_7 (the other case is similar). If $OUT_C(v_2)$ includes an ND gate-node which is different from v_3, v_6 or v_7, then we can select it as v_8 and can apply the other cases. Otherwise, we can show that each of $\{v_3, v_6, v_7\}$ must be in $OUT_C(v_2)$ as follows: Suppose, for example, that $OUT_C(v_2) \supseteq \{v_3, v_6\}$ but $v_7 \notin OUT_C(v_2)$. Then we can set x_2 to block v_3 and x_4 to block v_6. Also we can set x_5 to block v_4 since we are now assuming that we cannot fix x_4 such that it blocks both v_5 and v_6 (i.e., if we block v_6, then v_5 becomes a through-gate). Thus all paths from x_1 are blocked (with the help of all other degenerate nodes in $OUT_C(v_2)$), which contradicts to Proposition 3.2. If $OUT_C(v_2) \supseteq \{v_6\}$ but $v_3, v_7 \notin OUT_C(v_2)$, then we can select x_2 to block v_1 and x_4 to block v_6, which implies the same conclusion as above. All the other cases are similar. Thus, without loss of generality, we can assume $OUT_C(v_2) \supseteq \{v_3, v_6, v_7\}$ and therefore:

Fix x_1 s.t. v_2 blocked $\Rightarrow$ Killed: v_2, Removed: v_1, v_2, v_3, v_6, v_7.
Non degenerate: v_1 (obvious) v_2, v_3, v_6, v_7 (by Corollary 3.3)
$\Rightarrow$ *Degeneracy*: ± 0.

Case 4.4.3.2 v_{10} is equal to v_2, v_3, v_6 or v_7 (denoted by (4) in Table 2): Assume that v_2 is equal to v_{10}. We can set x_2 to block v_1 and we can set x_4, x_5 to block v_2 ($= v_{10}$) (through v_5 and v_4) since we are now assuming that if v_1 is blocked v_4 becomes a through-gate. This fact contradicts Proposition 3.2. Thus, v_2 is different from v_{10}. Similarly for the case that $v_3 = v_{10}, v_6 = v_{10}$ and $v_7 = v_{10}$.

Case 4.4.3.3 v_1 through v_{10} are all different: We can prove by the similar argument as Case 2.3.3.3.3. Namely, v_1, v_2, v_3 and v_4 are killed by proper assignments of x_1 and x_2. Instead of removing v_3, v_4 of Case 2.3.3.3.3, $Degeneracy(C)$ increases by two since $Degree_{C'}(x_4)$ and $Degree_{C'}(x_5)$ are both one.

Case 4.4.3.4 v_8 is equal to v_9 (denoted by (5) in Table 2): We selected v_8 such that $OUT_C(v_2) \cup OUT_C(v_8)$ includes one or more ND gate-node except v_8.

Let u_3 be such an ND gate-node. This new u_3 is removed by setting x_1 and x_2 to the same values as Case 4.4.3.3, since the killed v_2 and v_3 also kill v_8 $(= v_9)$. Thus the decrease of SD does not change. u_3 may be equal to v_6, v_7 or v_{10}. If we cannot select u_3 that is different from v_6 or v_7, we can set appropriately all the input-nodes connected to the degenerate nodes in $OUT_C(v_2) \cup OUT_C(v_8)$ (if any) and also set x_2 to block v_1 and x_4 to block v_6 and x_5 to block v_7, which blocks all paths from x_1, a contradiction to Proposition 3.2. If $u_3 = v_{10}$, then we can find a further new ND gate-node in $OUT_C(v_2) \cup OUT_C(v_8) \cup OUT_C(u_3(= v_{10}))$ which is different from v_6 or v_7 by Proposition 3.2. One can see that this new gate-node is removed by the same assignment as before.

Case 4.4.3.5 v_{10} is equal to v_8 or v_9 (denoted by (6) in Table 2): One can see our circuit is symmetry between the left-side from x_1 and x_2 and the right-side from x_4 and x_5. Therefore we can repeat exactly the same argument from Case 4.1 to Case 4.4.3.4 for the right-side instead of the left-side. Since v_{10} is now assumed to be equal to v_8 or v_9, we do not have to consider the case that v_{10} is equal to gate-nodes below v_6 or v_7. That concludes the proof of Lemma 4.1. ■

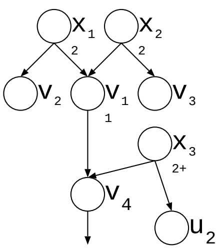

Fig. 14. Case 4.3

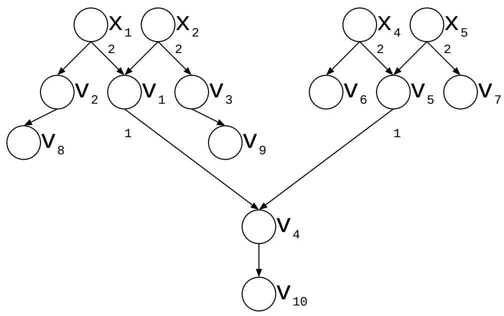

Fig. 15. Case 4.4

References

1. C.E. Shannon. The synthesis of two-terminal switching circuits, *Bell Systems Tech. J.*, vol 28, pages 59–98, 1949.
2. C. Schnorr. Zwei lineare untere Schranken für die Komplexität Boolescher Funktionen. *Computing* 13, pp. 155-171, 1974.
3. L. Stockmeyer. On the combinational complexity of certain symmetric Boolean functions. *Math. System Theory* 10, pp. 323-336, 1977.
4. W. Paul. A 2.5n-lower bound on the combinational complexity of boolean functions. *SIAM J. Comput.* 6, pp. 427-443, 1977.
5. N. Blum. A 2.75n-lower bound on the network complexity of boolean functions. *Tech. Rept.* A 81/05, *Universität* des Saarlandes, 1981.
6. N. Blum. A Boolean function requiring 3n network size. *Theoret. Comput. Sci.*, 28, pp. 337-345, 1984.

7. U. Zwick. A 4n lower bound on the combinatorial complexity of certain symmetric Boolean functions over the basis of unate dyadic Boolean functions. *SIAM J. Comput.* 20, pp. 499-505, 1991.
8. O. Lachish and R. Raz. Explicit lower bound of $4.5n - o(n)$ for Boolean circuits. *Proc. STOC'01*, pp. 399-408, 2001.
9. K. Iwama and H. Morizumi. An explicit lower bound of $5n - o(n)$ for Boolean circuits. *MFCS 2002*, pp. 353-364, 2002.
10. I. Wegener. Branching programs and binary decision diagrams. *SIAM Monographs on Discrete Mathematics and Applications*, 1999.
11. P. Savický. and S. Žáck A large lower bound for 1-branching programs. *ECCC Rep. No. 96-030*, 1996.