TECHNICAL REPORT 93-6

The Complexity of the Optimal Variable Ordering Problems of A Shared Binary Decision Diagram

Seiichiro TANI (Univ. of Tokyo)
Kiyoharu HAMAGUCHI (Kyoto Univ.)
Shuzo YAJIMA (Kyoto Univ.)

December, 1993

Department of Information Science
Faculty of Science, University of Tokyo

7–3–1 Hongo, Bunkyo-Ku Tokyo, 113 Japan

TITLE
The Complexity of
the Optimal Variable Ordering Problems
of A Shared Binary Decision Diagram

AUTHORS
Seiichiro TANI (Univ. of Tokyo) Kiyoharu HAMAGUCHI (Kyoto Univ.)
Shuzo YAJIMA (Kyoto Univ.)

KEY WORDS AND PHRASES
binary decision diagram, Boolean function, variable ordering, NP-complete

ABSTRACT
A binary decision diagram (BDD) is a directed acyclic graph for representing a Boolean function. BDD's are widely used in various areas which require Boolean function manipulation, since BDD's can represent efficiently many of practical Boolean functions and have other desirable properties. However the complexity of constructing BDD's has hardly been researched theoretically. In this report, we prove that the optimal variable ordering problem of a shared BDD is *NP*-complete, and touch on the approximation hardness of this problem and related problems of BDD's.

REPORT DATE	**WRITTEN LANGUAGE**
December, 1993	English
TOTAL NO. OF PAGES	**NO. OF REFERENCES**
18	9

ANY OTHER IDENTIFYING INFORMATION OF THIS REPORT
This report is a refined version of the paper which will appear in ISAAC'93.

DISTRIBUTION STATEMENT
First Issue 50 copies

SUPPLEMENTARY NOTES

DEPARTMENT OF INFORMATION SCIENCE
Faculty of Science, University of Tokyo
7-3-1 Hongo, Bunkyo-ku Tokyo, 113 Japan

The Complexity of the Optimal Variable Ordering Problems of A Shared Binary Decision Diagram

Seiichiro TANI (Univ. of Tokyo)
Kiyoharu HAMAGUCHI (Kyoto Univ.)
Shuzo YAJIMA (Kyoto Univ.)

December, 1993

1 Introduction

Many problems in digital logic design and testing, artificial intelligence and combinatorics can be expressed as a sequence of operations on Boolean functions. Since the way of representing Boolean functions has a great influence on computation time and storage requirements for operations on Boolean functions, various ways of representation of Boolean functions have been proposed.

A binary decision diagram (BDD) is a directed acyclic graph representation of a Boolean function proposed by Akers [1] and Bryant [2]. BDD's are used as an efficient way of representing a Boolean function, because BDD's have some desirable properties: the size of BDD's is feasible for many practical functions and any Boolean function can be uniquely represented by a BDD for a given variable ordering. A shared binary decision diagram (SBDD) [7] is a powerful extension of BDD's. An SBDD represents multiple Boolean functions simultaneously.

The methods using an SBDD (or BDD's) in various areas of CAD have been demonstrated to be able to handle large scale problems that cannot be handled by conventional methods. For example, state enumeration methods for finite state machines can be used in a wide range of problems such as implementation verification, design verification, and so on. However traditional enumeration methods cannot handle machines with more than a few million states. Coudert et al. [3] first proposed representing a set of states with an SBDD. Owing to the advantage of this technique, machines with more states came to be handled.

It is important to represent Boolean functions by a BDD (an SBDD) with less nodes, because computation time and storage requirements depend on the number of nodes. In general the number of nodes is highly sensitive to a variable ordering. Therefore it is important to find a variable ordering for a BDD (an SBDD) with as few nodes as possible. Some minimization algorithms for a BDD have been proposed. One of the algorithms was proposed by Friedman et al. [5]. This algorithm manipulates both a given function and intermediate results in the form of truth table. The time complexity of this algorithm

is $O(n^2 3^n)$ for the number of the variables. The optimization algorithm by Ishiura et al. [9] uses BDD's in order to manipulate a given function and intermediate results. This algorithm is based on Friedman's algorithm. However this algorithm minimizes a SBDD with 17 variables owing to the advantage of BDD's. Since optimization algorithms require enormous computation time for functions with many variables, many heuristics and approximation algorithms have been proposed. One of the heuristics are *dynamic weight assignment method* proposed by Minato et al. [7]. This method computes a variable ordering by using topological information of circuits. There are some other heuristics using information of circuits. One of approximation algorithms was *gradual improvement method* proposed by Ishiura et al. [9]. This algorithm is based on exchanges of adjacent m variables (only the case where $m = 2, 3, 4$ is considered) on a variable ordering and has two types, a greedy version and a simulated annealing version. Though there are many practical researches for the problem, theoretical researches can hardly be seen except that it is mentioned without proof by Bryant that the problem of minimizing a BDD for a given Boolean formula is co NP-hard. There is another important problem, the minimization problem of a BDD for a given BDD, whose hardness has not been known yet. It is impossible to show a trivial polynomial transformation from the minimization problem for a given Boolean formula to this problem, because there is the case where, when we construct a BDD for a Boolean formula with the polynomial size for the number of the variables, the size of the BDD is exponential for any variable ordering. It is also important to evaluate the approximation hardness of the problem because of the hardness of constructing a polynomial time exact algorithm. However it has not been shown.

In this report, we consider the optimal variable ordering problem of an SBDD for a given SBDD (OPTIMAL SBDD). We prove that this problem is in NP by constructing a polynomial time nondeterministic algorithm to solve it and prove the NP-hardness of this problem by reducing OPTIMAL LINEAR ARRANGEMENT. The approximation hardness of the minimization problem of a BDD is also touched on.

OPTIMAL SBDD contains as a special case the optimal variable ordering problem of a BDD for a given BDD (OPTIMAL BDD). Thus OPTIMAL BDD is also in NP. However, our result does not carry over for a BDD, and it is left open whether OPTIMAL BDD is NP-hard or not. Since the number of nodes of a BDD (an SBDD) for the Boolean formulae of the polynomial size may be exponential for any variable ordering (for example, multiplier functions) and in many of practical cases an SBDD is given as an input, our result settles more practically important problem concerning a BDD.

2 Preliminaries

For Boolean variables x, y, z_i $(i = 1, 2, \cdots, n)$, we denote AND, OR, NOT, EXOR, a tautology function and an inconsistency function by $x \cdot y$, $x + y$, $\overline{x}$, $x \oplus y$, 1 and 0, respectively. For multiple variables we denote AND, OR and EXOR by $\prod_{i=1}^{n} z_i$, $\sum_{i=1}^{n} z_i$ and $\bigoplus_{i=1}^{n} z_i$, respectively. For a function $f(x_1, x_2, \ldots, x_n)$, the resultig function obtained from f by assigning respectively b_i to x_i $(i = 1, 2, \cdots, m, b_i \in \{0, 1\}, m \leq n)$ is denoted $f|_{x_1=b_1, x_2=b_2, \cdots, x_m=b_m}$. The set of the variables which f depends on is denoted $Supp(f)$, that is, $Supp(f) = \{x \mid f|_{x=0} \neq f|_{x=1}\}$.

A binary decision diagram (BDD) is a directed acyclic graph which represents a

Boolean function [2]. It has node set N containing two types of nodes, terminal nodes and nonterminal nodes. Each nonterminal node v is labeled with a Boolean variable $var(v)$ and has edges directed toward two children: $low(v) \in N$ and $high(v) \in N$. The edges between v and $low(v)$ and that between v and $high(v)$ are called *0-edge* and *1-edge*, respectively. Each terminal node v has a value $value(v) \in \{0, 1\}$.

There is at most one node labeled the same variable on any path from the rootnode to a terminal node. The ordering of variables on each path is consistent with one another. For a BDD representing $f(x_1, x_2, \ldots, x_n)$, we define a variable ordering of a BDD as $\pi = (\pi[1], \pi[2], \cdots, \pi[n])$, where if $i \neq j$, then $\pi[i] \neq \pi[j]$ and $\pi[i], \pi[j] \in \{x_1, x_2, \cdots, x_n\}$, and if $var(v) = \pi[i]$ and $var(low(v)) = \pi[j]$, then $i > j$ (The same is true of $high(v)$).

A BDD with a rootnode v defines a Boolean function $F(v)$ such that if v is a terminal node with $value(v)$ =0 (1), then $F(v)$ =0 (1) and that if v is a nonterminal node with $var(v) = x$, then $F(v) = \overline{x}F(low(v)) + xF(high(v))$. In other words, a path from the rootnode to a terminal node corresponds to an assignment to all the variables, that is, if the edge between v and its child on the path is 0-edge (1-edge), then it corresponds to assigning 0 (1) to $var(v)$.

The size of a BDD is the number of nodes of the BDD. We denote the size of a BDD representing a function f on a variable ordering π by $size(f, \pi)$. If there is no confusion, $size(f, \pi)$ is abbreviated as $size(f)$. The size of a BDD can be reduced by applying repeatedly the following transformation rules

- If there exists a nonterminal node v where $low(v) = high(v)$, then eliminate v and redirect all incoming edges to $low(v)$.

- If there exist equivalent nodes u, v, then eliminate v and redirect all edges into u to v. Two nodes are equivalent if one of the following holds:

  1. If both u and v are terminal nodes, the labels of both nodes are equivalent.
  2. If both u and v are nonterminal nodes, $var(u) = var(v)$, $low(u) = low(v)$ and $high(u) = high(v)$.

For a fixed variable ordering, the maximally reduced BDD's is uniquely determined, which is called a *canonical BDD* for the ordering. In this paper, we call canonical BDD's *BDD's* simply. Examples of BDD's are shown in `Figure1`. In `Figure1`, the variable ordering of (a) is (x_1, x_2, x_3, x_4) and that of (b) is (x_2, x_3, x_1, x_4). `Figure1` shows that the size of a BDD depends on a variable ordering.

Multiple BDD's with the same variable ordering can be joined into a uniquely determined graph by transforming the BDD's under the previously presented rules. This single graph is called a shared binary decision diagram (SBDD) [7]. With the use of SBDD's to represent the set of Boolean functions, not only storage requirement decreases but also the equivalence checking of Boolean functions becomes easy. In practice, SBDD's are used more frequently than BDD's. For example, the SBDD in `Figure2`(b) consists of the four BDD's in `Figure2`(a).

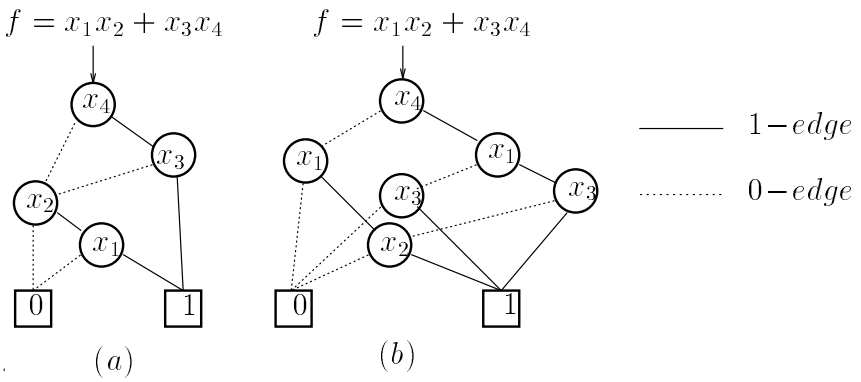

Figure 1: Examples of BDD's

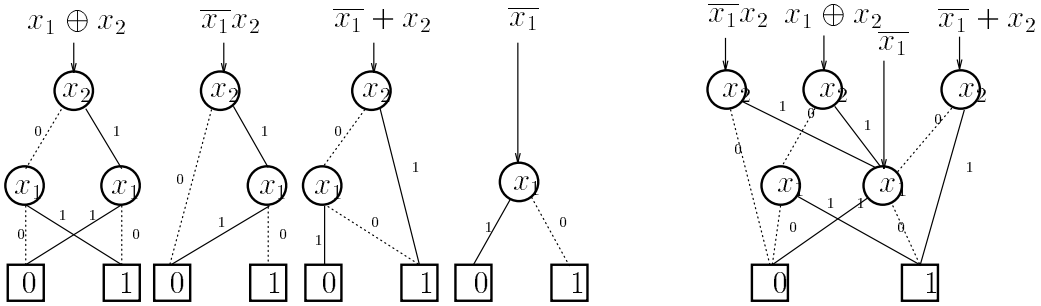

Figure 2: BDD's and an SBDD

3 The Optimal Variable Ordering Problems of an SBDD and Their Complexity

We define the optimal variable ordering problem of an SBDD (OPTIMAL SBDD) and prove its NP-completeness in the following.

[OPTIMAL SBDD]
INSTANCE: An SBDD with n nodes and a positive integer $k(< n)$
QUESTION: Is there a variable ordering for some SBDD with at most k nodes, which represents the same functions as are represented by the given SBDD ?

Theorem 1 *OPTIMAL SBDD is in NP.*

Proof We show a nondeterministic polynomial time algorithm to solve OPTIMAL SBDD. The algorithm chooses a variable ordering π nondeterministically and reconstructs the canonical SBDD with π. If the SBDD with at most k nodes can be reconstructed, the algorithm puts out "yes". However, when the number of the created nodes exceeds k in reconstructing the SBDD, the algorithm stops and puts out "no".

We assume $f_i(i = 1, 2, \ldots, p)$ are functions represented by the given SBDD and *bddtable* is a set of BDD, which is empty initially. Bddtable is used to record the BDD's for the functions defined by the created nodes. Note that the algorithm doesn't create equivalent nodes thanks to referring to bddtable before creating a new node. Firstly the algorithm creates a new node u and adds the BDD for f_i to bddtable. Note that u will be the rootnode of the BDD for f_i and that u may be a terminal node at this point. Secondly the algorithm calls the procedure $RECONST(f, \pi)$. The procedure RECONST reconstructs a canonical BDD in the way of depth first for a given BDD representing f and a variable ordering π. RECONST behaves as follows. If f is tautology or inconsistency, then RECONST terminates. If f is neither tautology nor inconsistency, the children of u are created as follows. We denote a variable $\pi[\max\{t|\pi[t] \in Supp(f)\}] = 0$ by $maxvar(f, \pi)$. Unless the BDD for $f|_{maxvar(f,\pi)=0}$ is in bddtable, RECONST(f, π) creates a new node v and 0-edge from u to v, adds the BDD for the function to bddtable and calls RECONST$(f|_{maxvar(f,\pi)=0}, \pi)$. If the BDD for $f|_{maxvar(f,\pi)=0}$ is in bddtable, RECONST(f, π) creates 0-edge from u to the node defining $f|_{maxvar(f,\pi)=0}$. Note that the nodes defining the function represented by BDD's in bddtable have already been created and that the BDD for the restriction of f is smaller than the input SBDD. Next RECONST(f, π) executes the same for the BDD for $f|_{maxvar(f,\pi)=1}$, but 1-edge instead of 0-edge is created.

The procedure RECONST is called less than k times. Since bddtable is a set of BDD, whether the BDD for some function is in bddtable or not can be checked in polynomial time. Thus the time complexity of the deterministic part of the algorithm is $O(n^{O(1)})$. □

Now we consider the optimal variable ordering problem of a BDD (OPTIMAL BDD). OPTIMAL BDD is obtained by replacing SBDD with BDD in OPTIMAL SBDD. This problem is a special case of OPTIMAL SBDD, that is, a case that the given SBDD in OPTIMAL SBDD represents a single Boolean function. Therefore we obtain the following corollary.

Corollary 1 *OPTIMAL BDD is in NP.*

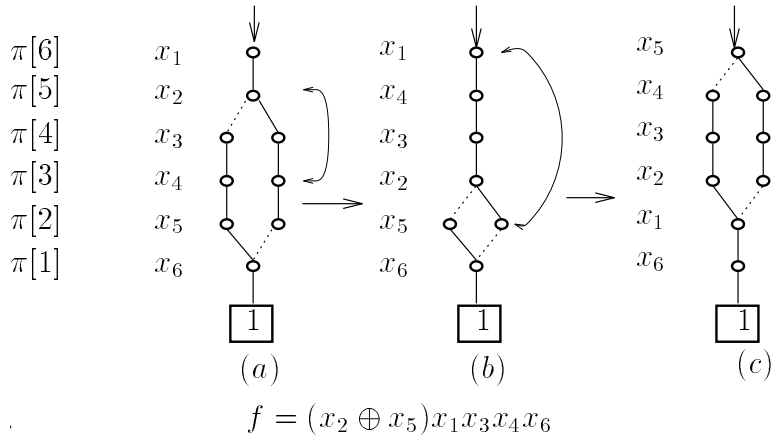

Figure 3: T-Phage Functions

Next we prove the NP hardness of OPTIMAL SBDD by reducing OPTIMAL LINEAR ARRANGEMENT defined below to OPTIMAL SBDD.

[OPTIMAL LINEAR ARRANGEMENT (OLA)] [6]
INSTANCE: A graph $G = (V, E)$ and a positive integer K
QUESTION: Is there a one-to-one function $\psi : V \longrightarrow \{1, 2, 3, \cdots, |V|\}$ such that $\sum_{(u,v)\in E} |\psi(u) - \psi(v)| \leq K$? $|\psi(u) - \psi(v)|$ is called *the cost of edge* (u, v), and $\sum_{(u,v)\in E} |\psi(u) - \psi(v)|$ is called *the cost of graph* G.

The basic idea of the reduction is to reflect the cost of an edge of G by the size of a BDD and to construct an SBDD whose size reflects the cost of G from such BDD's. Now we define a T-phage function as follows. We denote a BDD representing a T-phage function by a BDD_T.

Definition 1 (T-Phage Function) *A Boolean function f is a T-phage function, if it can be expressed by $f(x_1, x_2, \cdots, x_n) = (x_{i_1} \oplus x_{i_2}) \prod_{k\neq i_1,i_2} x_k$, where $(i_1, i_2, \cdots, i_n)$ is a permutation of $(1, 2, \cdots, n)$.* □

Examples of BDD_T's are shown in `Figure3`, where edges into the terminal node labeled 0 are omitted. In the following figures, edges to 0 are abbreviated similarly.

BDD_T's have a convenient property as follows.

Lemma 1 (The Property of BDD_T's) *For a T-phage function $f(x_1, x_2, \cdots, x_n) = (x_{i_1} \oplus x_{i_2}) \prod_{k\neq i_1,i_2} x_k$ such that $\pi[l] = x_{i_1}, \pi[m] = x_{i_2}$, $size(f, \pi)$ is $|l - m| + n + 2$.*

Outline of proof We assume that f is a T-phage function with n variables and that, without loss of generality, $\pi[l] = x_{i_1}$, $\pi[m] = x_{i_2}$ and $l > m$. We define $\mathcal{F}_i$ as the set of the functions obtained by assigning 0 or 1 to each of the variables $\pi[i], \pi[i + 1], \ldots, \pi[n]$ in $Supp(f)$. It can be obtained that $size(f) = |\bigcup_{i=1}^{n} \mathcal{F}_i| + 1$. Note that $|\bigcup_{i=1}^{n} \mathcal{F}_i|$ does not contain the number of rootnode. By enumerating the functions in $\bigcup_{i=1}^{n} \mathcal{F}_i$, we obtain $size(f, \pi)$ is $|l - m| + n + 2$. (Refer to Appendix A.) □

In `Figure3`, the variable ordering of (a) is different from that of (c). However the size of (a) are same with that of (c) since $|l - m| = 3$ in both (a) and (c).

Thus the size of a T-phage function one-to-one corresponds to cost of an edge (u, v) of G with ψ if we assign a T-phage function $f(x_1, x_2, \ldots, x_n) = (x_u \oplus x_v) \prod_{k \neq u,v} x_k$ to the edge and it is satisfied that $\pi[\psi(v)] = x_v$ for any node v of G. However, if each T-phage function corresponding to each edge depends on the same set of variables, the SBDD constructed from the BDD_T's does not reflect cost of G, because many nodes of BDD_T's are shared in constructing an SBDD.

To overcome the difficulty that many nodes of BDD_T's on the same set of variables are shared, we introduce a new set of $|V|$ variables $x_{1j}, x_{2j}, \ldots, x_{|V|j}$ for the jth edge (u_j, v_j) of G, and consider for a T-phage function $f_j(x_{1j}, x_{2j}, \ldots, x_{|V|j}) = (x_{u_j j} \oplus x_{v_j j}) \prod_{i \neq u_j, v_j} x_{ij}$ $(j = 1, 2, \ldots, |E|)$. Note that there are $|V||E|$ variables. If $x_{v1}, x_{v2}, \ldots, x_{v|E|}$ are arranged consecutively on the variable ordering of the SBDD (the ordering among $x_{v1}, x_{v2}, \ldots, x_{v|E|}$ does not matter) for each node v of G (such a variable ordering will be called well-ordered), the size of the SBDD for the variable ordering one-to-one corresponds to the cost of G with the corresponding variable ordering. On the contrary, if the variable ordering is not well ordered, ψ's corresponding to BDD_T's contained in the SBDD are inconsistent.

To force such well-ordered property for the SBDD problem, we add a modified T-phage function $h_i = (x_{i1} \oplus x_{i2} \oplus \ldots \oplus x_{i|E|}) \prod_{j,k \neq i} \overline{x_{kj}}$ for each i. In minimizing the size of the SBDD, this penalty function makes the variables $x_{i1}, x_{i2}, \ldots, x_{i|E|}$ consecutive on the variable ordering.

The above approach almost works fine, but there still exists a small difficulty caused by shared nodes corresponding to $\pi[1]$. To overcome this completely, we introduce another set of $|V|$ variables for each edge, and as a whole consider the following.

Definition 2 (OLAD) *An OLA-Diagram (OLAD) for graph $G = (V, E)$ given in OLA is an SBDD representing the following functions: for $j = 1, 2, \cdots, |E|$,*

$$\begin{cases} f_j = (x_{i_1 j} \oplus x_{i_2 j}) x_{i_3 j} \cdots x_{i_{|V|} j} \\ g_j = (y_{i_1 j} \oplus y_{i_2 j}) y_{i_3 j} \cdots y_{i_{|V|} j} \\ h_i = (x_{i1} \oplus x_{i2} \oplus \cdots \oplus x_{i|E|} \oplus y_{i1} \oplus y_{i2} \oplus \cdots \oplus y_{i|E|}) \prod\limits_{j,k \neq i} \overline{x_{kj}} \cdot \overline{y_{kj}} \end{cases} \tag{1}$$

where $(i_1, i_2, \cdots, i_{|V|})$ is a permutation of $(1, 2, \cdots, |V|)$.

We show the wo-OLAD (`Figure6`) for the graph G (`Figure5`) with one-to-one function ψ such that $\psi(i) = i$ $(i = 1, 2, 3, 4)$.

In an OLAD, the BDD's for f_j's and g_j's share no nodes except the terminal nodes because of $Supp(\delta_1) \cap Supp(\delta_2) = \phi$ where δ_1, δ_2 are two functions of f_j's and g_j's. However the BDD's for h_i's share many nodes. Any BDD for f_j (or g_j) and BDD's for h_i's share the terminal nodes and the nonterminal nodes corresponding to the variable $\pi[1]$ (see `Figure6`). Note that the AND part of the formulae for f_j, g_j is composed of positive literals though the AND part of the formulae for h_i is composed of negative literals.

Now we define several terms for the following proof. We denote the set of variables $\{x_{ij}, y_{ij} | 1 \leq j \leq |E|\}$ by B_i for each i and an element of $B = \{B_1, B_2, \ldots, B_{|V|}\}$ is called a *variable block*. If the variables in each B_i are arranged consecutively on a variable ordering, such a variable ordering and an OLAD with such a variable ordering are called *well-ordered* (*wo*). In the other case, they are called *non-well-ordered* (*nwo*). For a wo-OLAD, we define a *variable block ordering* $\varphi = (\varphi[1], \varphi[2], \cdots, \varphi[|V|])$ such that, for any

variable $\pi[t_1]$ in $\varphi[l_1]$ and any variable $\pi[t_2]$ in $\varphi[l_2]$, it is satisfied that $l_1 < l_2$ if and only if $t_1 < t_2$ where $\varphi[l] \in B$ for each l.

Again we make sure the outline of the proof. We express the cost of G with ψ by a wo-OLAD in the following manner. Firstly, for the jth edge (u_j, v_j) of G, we assign two T-phage functions:

$$f_j(x_{1j}, x_{2j}, ..., x_{|V|j}) = (x_{u_j j} \oplus x_{v_j j}) \prod_{\substack{w \in V \\ w \neq u_j, v_j}} x_{wj}$$

$$g_j(y_{1j}, y_{2j}, ..., y_{|V|j}) = (y_{u_j j} \oplus y_{v_j j}) \prod_{\substack{w \in V \\ w \neq u_j, v_j}} y_{wj}$$

and assign h_i for each i. Secondly we consider a wo-OLAD with $x_{ij}, y_{ij} \in \varphi[\psi(i)]$ for any i, j. There remain two important points to be considered. One is that, as long as the variable block ordering unchanges, the size of the wo-OLAD needs to express the identical cost of G even if the variable ordring changes. The other is that only a wo-OLAD can express the cost of G, that is, an nwo-OLAD cannot express the cost of G. In the following part, we consider influence by exchanging variables in a variable block and relate the cost of G with the size of the corresponding wo-OLAD (Lemma2). Secondly we guarantee that any nwo-OLAD can be transformed into a wo-OLAD with less nodes (Lemma3). Finally we finish the proof by these lemmas.

In the following, we observe the variance of the size of a wo-OLAD when we exchnage variables in a variable block. Since, in an OLAD, the nonterminal nodes of BDD's for f_j's, g_j's and those of BDD's for h_i's are not shared except the nodes v such that $var(v) = \pi[1]$, we can observe BDD's for f_j's, g_j's and BDD's for h_i's separately. The number of nonterminal nodes of each BDD for h_i's does not vary when exchanging variables in a variable block, because, for each i, $Supp(h_i)$ is symmetric in each variable block. In the case of BDD's for f_j and g_j, the situation in exchanging variables in a variable block but $\varphi[1]$ is different from that in exchanging variables in $\varphi[1]$. If we exchange variables in a variable block but $\varphi[1]$, the number of the nodes of each BDD for f_j's, g_j's does not vary because only one variable in $Supp(f_j)$ (or $Supp(g_j)$) is in each variable block and the BDD's for f_j's, g_j's share no nonterminal node. However the number of nodes of the wo-OLAD may vary when we exchange variables in the variable block $\varphi[1]$, since any nonterminal node v in BDD's for f_j's, g_j's such that $var(v) = \pi[1]$ is shared by BDD's for h_i's. The number of such nodes is one or two. A wo-OLAD in the former case is called *type1* and one in the latter case is called *type2*. Note that, for some ψ, there are two wo-OLAD's with the different number of nodes by a node. However we will show that such two different sizes of the wo-OLAD express the identical cost of G and have no possibility to express some other costs of G. We show *type1* and *type2* by `Figure`4, where it is satisfied that $B_{i_1} = \varphi[1]$ and $B_{i_{|V|}} = \varphi[|V|]$.

Then for the two types, we consider the relation between cost of G and the size of the corresponding wo-OLAD.

Lemma 2 *For a graph $G = (V, E)$ with cost K, which is given in OLA, construct an OLAD. If this OLAD is well-ordered, the size of* type1 *and* type2, *$k_1(K)$ and $k_2(K)$, respectively, are as follows.*

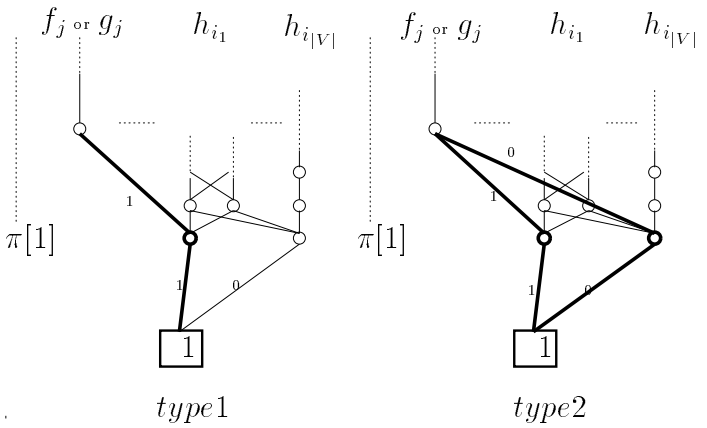

Figure 4: Two types of wo-OLAD's

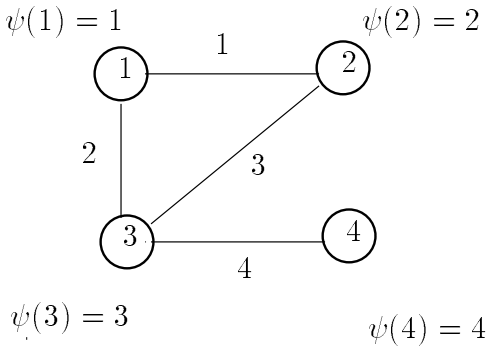

Figure 5: An example of the graph G with ψ given in OLA

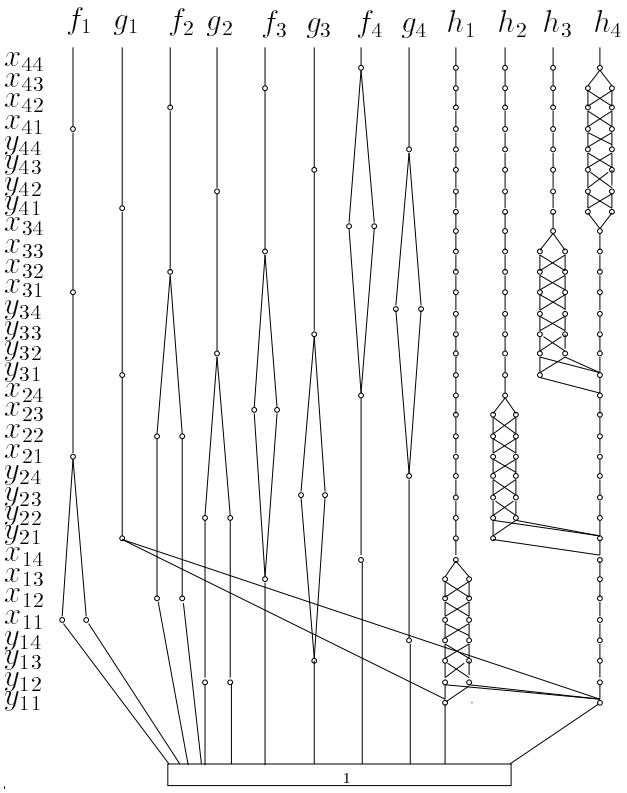

Figure 6: An example of a wo-OLAD

$$k_1(K) = k_2(K) + 1 = 2\left\{K + 1 - |V| + |E||V|(|V| + 2) - |E| \sum_{i=1}^{|V|-2} i\right\}$$

Outline of proof We observe BDD's for f_j's and g_j's, and BDD's for h_i's separately. Firstly we count the nodes of the BDD's for f_j's, g_j's and h_i's, where, for the jth edge (u_j, v_j) of $G = (V, E)$ given in OLA, $size(f_j) = |\psi(u_j) - \psi(v_j)| + |V| + 2$ because Lemma1 can be applied since only one variable in $Supp(f_j)$ is in each variable block. The same is true of $size(g_j)$. For wo-OLAD's for fixed G, $size(h_i)$ is constant. Secondly we count the nodes shared among the BDD's for f_j's, g_j's and h_i's in the wo-OLAD. Then we obtain the number of nodes of the wo-OLAD by subtracting the number of the nodes shared among the BDD's from $\sum_{j=1}^{|E|}(size(f_j) + size(g_j)) + \sum_{i=1}^{|V|} size(h_i)$. (Refer to Appendix B.) □

$|E|$ is at most $\frac{|V|(|V|+1)}{2}$ and K is less than $|E|(|V| - 1)$, where K is the cost of G. Hence $k_1(K)$ and $k_2(K)$ are $O(|V|^4)$.

Corollary 2 *Let k the size of a wo-OLAD and let K the cost of G corresponding to the wo-OLAD. It is satisfied that $K = \lceil k/2 \rceil - \Phi(G)$, where $\Phi(G) = 1 - |V| + |E||V|(|V| + 2) - |E| \sum_{i=1}^{|V|-2} i$*

Corollary 2 shows that the cost of G corresponding to the size of a wo-OLAD can be determined uniquely, since $\Phi(G)$ is constant for fixed G.

We now turn to nwo-OLAD's. The cost of nwo-OLAD does not correspond to the cost of G. Instead, we can transform the given nwo-OLAD to a wo-OLAD with less nodes, whose cost nicely corresponds to that of G

Lemma 3 *An arbitrary nwo-OLAD can be transformed into a functionally equivalent wo-OLAD with less nodes.*

Proof We show an algorithm which transforms an nwo-OLAD into a wo-OLAD with less nodes. This algorithm behaves as follows. Firstly, assuming that $\pi[1]$ is in B_{u_1}, this algorithm makes all the variables in B_{u_1} consecutive without changing the order among them and also without changing the order among variables not in B_{u_1}. After this first stage, we obtain an OLAD with the variable ordering such that $\pi[1], \pi[2], \ldots, \pi[2|E|]$ are all in B_{u_1}. Note that $|B_i| = 2|E|$ for each i. Secondly, assuming that $\pi[2|E| + 1]$ is in B_{u_2}, this algorithm makes all the variables in B_{u_2} consecutive in the same way above. After the second stage, we obtain an OLAD with the variable ordering such that $\pi[1], \pi[2], \ldots, \pi[2|E|]$ are all in B_{u_1} and $\pi[2|E| + 1], \pi[2|E| + 2], \ldots, \pi[4|E|]$ are all in B_{u_2}. The algorithm rearranges all the variables in the same block with $\pi[2|E|(w - 1) + 1]$ in a similar way at the wth stage for $w = 3, 4, \ldots, |V|$. Thus after the wth stage, for $i = 1, 2, ..., w$, $\pi[2|E|(i - 1) + 1], \pi[2|E|(i - 1) + 2], ..., \pi[2|E|i]$ are all in B_{u_i}. In this way, the algorithm transforms an nwo-OLAD into a wo-OLAD with less nodes uniquely. We show the pseud code of this algorithm by `Figure7`, where if condition $\exists s := \min\{t \cdots\}$ in line 7 is true, then $min\{t \cdots\}$ is substituted for s.

In the following, we prove that the number of nodes does not increase at each stage.

Suppose that after the $(w - 1)$th stage, $\pi[2|E|(w - 1) + 1]$ is in B_u and there are q variables $\notin B_u$ among $\pi[2|E|(w - 1) + 1], \pi[2|E|(w - 1) + 2], ..., \pi[\max\{\ t \mid \pi[t] \in B_u\}]$. If $q = 0$, all the variables between $\pi[2|E|(w - 1) + 1]$ and $\pi[\max\{t \mid \pi[t] \in B_u\}]$, that is,

```
1  begin
2     k := 1;
3     while(k < 2|V||E|) do
4        begin
5           substitute u for i such that π[k] ∈ B_i;
6           k := k + 1;
7           while(∃s := min{ t | π[t] ∈ B_u, k ≤ t ≤ 2|V||E|}) do
8              begin
9                 for ct := s downto k + 1 do
10                   exchange π[ct] for π[ct − 1];
11                k := k + 1;
12             end;
13       end;
14 end.
```

Figure 7: The algorithm transforming an nwo-OLAD into a wo-OLAD

$\pi[2|E|w]$ are already all in B_u after $(w-1)$th stage. Thus at the wth stage the variable ordering unchanges. We assume $q > 0$ in the following part of this proof.

To evaluate the variation of the nodes of the OLAD, we observe the BDD's for f_j's and g_j's and the BDD's for h_i's separately. Without loss of generality, we may assume that the nodes of the BDD for h_i with $B_i = \varphi[|V|]$ is not shared by other BDD's in the OLAD, that is, the nodes shared among the BDD's for f_j's, g_j's and h_i's belong to the BDD for h_i with $B_i = \varphi[|V|]$.

A BDD for any h_i can be divided into three parts, that is, subgraphs whose nodes correspond to $(\pi[1] \dots \pi[t_1 - 1])$, $(\pi[t_1], \dots, \pi[t_2 - 1])$ and $(\pi[t_2], \dots, \pi[2mn])$ respectively, where $t_1 = \min\{t \mid \pi[t] \in B_i\}$ and $t_2 = \max\{t \mid \pi[t] \in B_i\}$. These three parts are called *A-part*, *B-part* and *C-part*, respectively. Note that the number of nodes corresponding to each variable in A-part and C-part is only one, and the number of nodes corresponding to each variable in B-part is exactly two. We denote $\min\{t \mid \pi[t] \in B_u\}$ and $\max\{t \mid \pi[t] \in B_u\}$ by t_{min} and t_{max} respectively, and the decrease of the number of the nodes of the BDD's for h_i at the wth stage, by $-\Delta h_i$ $(-\Delta h_i \geq 0)$. We show the BDD for h_u by `Figure8`, where the nodes enclosed by dotted line are shared by the BDD for h_i with $B_i = \varphi[|V|]$.

In the BDD's for h_i's such that $t < t_{min}$ for any $\pi[t] \in B_i$ $(i \neq u)$, only the order of $var(v)$'s for nodes v's in C-part changes at the wth stage. Thus $-\Delta h_i = 0$.

In the BDD's for h_u, at the wth stage, the variables $var(v)$'s$\notin B_u$ for the nodes v's in B-part move into the set of the variables $var(v)$'s for the nodes v's in C-part. The number of such variables is q. Therefore the number of the nodes in B-part decreases by $2q$ and the number of the nodes in C-part increases by q. Thus $-\Delta h_u = q$.

In the BDD's for h_i's such that $t > t_{min}$ for any $\pi[t] \in B_i$ $(i \neq u)$, at the wth stage the variables $var(v)$'s$\in B_u$ for the nodes v's in B-part and C-part move into the set of the variables $var(v)$'s for the nodes v's in A-part. Therefore the sum of the number of the nodes in B-part and C-part decreases, say, by δ. Then the number of the nodes in A-part increases by at most δ. Note that, in A-part and C-part, the number of the

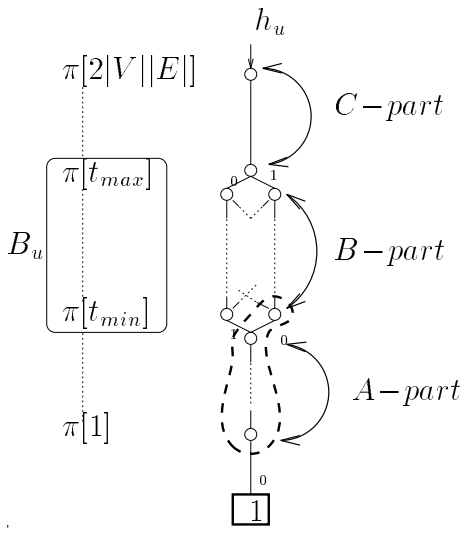

Figure 8: The BDD for h_u

nodes corresponding to each variable is one and, in B-part, the number of the nodes corresponding to each variables is two, and that the nodes in A-part of BDD's for h_i's with $B_i \neq \varphi[|V|]$ is shared by the BDD for h_i with $B_i = \varphi[|V|]$ (see `Figure6`). In the other case, only the order of the variables $var(v)$'s for the nodes v's in A-part changes at the wth stage, and the number of the nodes in each part unchanges. Thus $-\Delta h_i \geq 0$. Note that for at least one of such h_i's, $-\Delta h_i > 0$.

Secondly, we observe the variation of nodes of the BDD's for f_j's and g_j's at the wth stage. We denote the increase of the number of the nodes of each BDD for f_j's and g_j's at the wth stage, by Δf_j and Δg_j respectively. In each BDD for f_j's, we may assume without loss of generality that $x_{uj} = \pi[t]$ after the $(w-1)$th stage and that $x_{uj} = \pi[t-\alpha_j]$ after the wth stage. Then it is satisfied that $\Delta f_j \leq \alpha_j$ by the property of the BDD for T-phage function. In each BDD for g_j's, we may assume that $\Delta g_j \leq \beta_j$ in the same way. Since the intersection of any two of $Supp(f_j)$ and $Supp(g_j)$ $(j = 1, 2, \ldots, |E|)$ is empty, $\sum_{j=1}^{|E|}(\alpha_j + \beta_j) = q$, that is,

$$\sum_{j=1}^{|E|}(\Delta f_j + \Delta g_j) \leq \sum_{j=1}^{|E|}(\alpha_j + \beta_j) = q.$$

Thus we obtain:

$$\sum_{j=1}^{|E|}(\Delta f_j + \Delta g_j) + \sum_{i=1}^{|V|}\Delta h_i < 0.$$

Thus it becomes clear that at each stage, the number of nodes of the OLAD does not vary if $q = 0$, and the number of nodes decreases if $q > 0$.

There are $|V|$ stages until the algorithm stops. At at least one of the $|V|$ stages it is satisfied that $q > 0$, since the OLAD given as an input of the algorithm is non-well-ordered. □

We give a wo-OLAD for some cost of G and $2(K+\Phi(G))$ as an instance of OPTIMAL SBDD. We assume that there exists an OLAD with at most $2(K+\Phi(G))$ nodes, which represents the same functions as the given wo-OLAD. If this OLAD is well-ordered, there exists function ψ defined in OLA for the cost at most K for an arbitrary positive integer K by `Corollary2`. If the OLAD is non-well-ordered, it is guaranteed that there exists a wo-OLAD with less nodes by `Lemma3`. Thus in this case, there also exists one to one function ψ for the cost at most K. Therefore OLA is polynomially reducible to OPTIMAL SBDD, since the number of nodes of a wo-OLAD is $O(|V|^4)$. Then we obtain the following theorem.

Theorem 2 *OPTIMAL SBDD is NP-complete.*

4 Related Problems

4.1 The Complexity of the Optimal Variable Ordering Problem of A BDD for a given formula

The optimal variable ordering problem of a BDD for a given Boolean formula (OPTIMAL BDD FOR FORMULA) is also an important problem.
[OPTIMAL BDD FOR FORMULA]
INSTANCE: A Boolean formula and a positive integer k
QUESTION: Is there a variable ordering for some BDD with at most k nodes, which represents the same function as is represented by the given formula ?

The co NP-hardness of this problem is shown by reducing the tautology checking problem (TAUTOLOGY), which has been shown co NP-complete. TAUTOLOGY is defined as follows.
[TAUTOLOGY]
INSTANCE: A Boolean formula F
QUESTION: Is F a tautology ?

Theorem 3 *OPTIMAL BDD FOR FORMULA is co NP-hard.*

Proof We give as an instance of OPTIMAL BDD FOR FORMULA a Boolean formula $x \cdot \overline{F}$, where F is the Boolean Formula given in TAUTOLOGY and the variable x is not in $Supp(F)$, and a positive integer $k = 1$. The answer is "yes", that is, there exists a BDD for the formula with one node, if and only if F is a tautology. Thus TAUTOLOGY is reduced to the problem. □

4.2 The Approximation Hardness

The approximation hardness of the minimization problem of a BDD for a given BDD, defined as follows has not been shown.
[MINIMUM BDD]
INSTANCE: A BDD
REQUEST : Find a variable ordering which minimizes the given BDD.

The minimization problem of an SBDD (MINIMUM SBDD) can be defined in the same way. It can be shown that MINIMUM SBDD is NP-hard by reducing OPTIMAL SBDD. As for the approximation hardness of MINIMUM SBDD, the following theorem is easy to show.

Theorem 4 *If $P \neq NP$, then no polynomial time approximation algorithm A for MINIMUM SBDD can guarantee that, for a some fixed constant J, $|A(I) - OPT(I)| \leq J$, where I is an instance of MINIMUM SBDD and $A(I)$ and $OPT(I)$ are the costs of the solutions by A and an optimization algorithm, respectively. We denote the number of the nonterminal nodes of the SBDD determined by a solution, a variable ordering, by the cost of the solution.*

Proof Suppose that A is such an approximation algorithm. We can assume that, without loss of generality, I is an SBDD representing a set of functions $\Theta(x_1, x_2, ..., x_n)$. From I, we can construct a new instance I' which is an SBDD representing a set of functions $\bigcup_{i=1}^{J+1} \Theta(y_{i1}, y_{i2}, ..., y_{in})$ in polynomial time. Now we apply A to I'. From the obtained solution with cost $A(I')$, we can construct a solution for I with cost $\lfloor A(I')/(J+1) \rfloor$ in polynomial time. It is easy to see that $OPT(I') = (J+1)OPT(I)$. Then:

$$|A(I') - OPT(I')| \leq J$$

$$(J+1)\left|\left\lfloor \frac{A(I')}{J+1} \right\rfloor - OPT(I)\right| \leq J$$

We obtain $\lfloor A(I')/(J+1) \rfloor - OPT(I) = 0$ from the last formula. Thus we can construct an optimization algorithm for MINIMUM SBDD, and it is polynomial time algorithm, contradicting the assumption that $P \neq NP$. □

5 Concluding Remarks

It is an unavoidable and important problem to find a variable ordering which minimizes the size of a BDD or an SBDD, since variable ordering has a great influence on the size which computation time and storage requirements for Boolean function manipulation depend on.

In this report, we consider OPTIMAL SBDD and prove that it is NP-complete. However the NP-hardness of OPTIMAL BDD is still an open problem. The NP-hardness of OPTIMAL BDD is a stronger conjecture than the NP-hardness of OPTIMAL SBDD, because there is a trivial polynomial transformation from OPTIMAL BDD to OPTIMAL SBDD.

Concerning approximation algorithm, many heuristics for the minimization problem of a BDD have been proposed, but there is no approximation algorithm which guarantees some approximation ratio. Hence it is also important to discuss the approximability of the problem. To show the approximation hardness of the minimization problem of BDD's is also an important problem because it is hard to solve the problem exactly. However this kind of researches has not be seen. Therefore this is one of important future works.

Acknowledgment

Thanks are due to all the members of Yajima's Laboratory for their valuable discussions and comments. Authors would like to thank Prof. Hiroshi Imai of The University of Tokyo for reading this manuscript.

References

[1] S. B. Akers: Binary Decision Diagrams, IEEE Trans. on Computers, C-27(1978), 509-516.

[2] R. E. Bryant: Graph Based Algorithms for Boolean Function Manipulation, IEEE Trans. on Computers, C-35(1986), 677-691.

[3] O. Coudert, C. Berthet, and J. C. Madre: Verification of Sequential Machines Based on Symbolic Execution, Proceedings of the Workshop on Automatic Verification Methods for Finite State Systems, Grenoble, France(1989).

[4] M. Fujita, H. Fujisawa and N. Kawato: Evaluation and Improvements of Boolean Comparison Method Based on Binary Decision Diagrams, IEEE ICCAD-88(1988), 2-5.

[5] S. J. Friedman and K. J. Supowit: Finding the Optimal Variable Ordering for Binary Decision Diagrams, IEEE Trans. on computers, C-39(1986), 710-713.

[6] M. R. Garey, D. S. Johnson and L. Stockmeyer: Some Simplified NP-complete Graph Problems, Theoretical Computer Science 1(1976), 237-267.

[7] S. Minato, N. Ishiura and S. Yajima: Shared Binary Decision Diagram with Attributed Edges for Efficient Boolean Function Manipulation, Proc. 27th ACM/IEEE DAC(1990), 52-57.

[8] M. R. Mercer, R. Kapur and D. E. Ross: Functional Approaches to Generating Ordering for Efficient Symbolic Representations, Proc. 29th ACM/IEEE DAC(1992), 624-629.

[9] N. Ishiura, H. Sawada and S. Yajima: Minimization of Binary Decision Diagrams Based on Exchanges of Variables, IEEE ICCAD-91(1991), 472-475.

Appendix

A Proof of Lemma1

Without loss of generality, we can assume $\pi[l] = x_{i_1}$, $\pi[m] = x_{i_2}$ and $l > m$. We denote $\left\{ f|_{\bigwedge_{t=i}^{n} \pi[l]=b_t} \mid b_t \in \{0,1\}, t = i, i+1, \cdots, n \right\}$ by $\mathcal{F}_i$. Then it is satisfied that $size(f,\pi) = |\bigcup_{i=1}^{n} \mathcal{F}_i| + 1$. $\mathcal{F}_i$ is expressed as:
for $i > l$

$$\mathcal{F}_i = \left\{ 0,\ (\pi[l] \oplus \pi[m]) \prod_{\substack{t=1 \\ t \neq l,m}}^{i-1} \pi[t] \right\},$$

for $m < i \leq l$

$$\mathcal{F}_i = \left\{ 0,\ \pi[m] \prod_{\substack{t=1 \\ t \neq m}}^{i-1} \pi[t],\ \overline{\pi[m]} \prod_{\substack{t=1 \\ t \neq m}}^{i-1} \pi[t] \right\},$$

for $1 < i \leq m$

$$\mathcal{F}_i = \left\{ 0,\ \prod_{t=1}^{i-1} \pi[t] \right\},$$

for $i = 1$

$$\mathcal{F}_i = \{0, 1\},$$

Thus $|\bigcup_{i=1}^{n} \mathcal{F}_i| + 1 = |l - m| + n + 2$. □

B Proof of Lemma2

We observe BDD's for f_j's and g_j's, and BDD's for h_i's separately. Firstly we count the nodes of the BDD's for f_j's, g_j's and h_i's. For the jth edge (u_j, v_j) of $G = (V, E)$ given in OLA, $size(f_j) = |\psi(u_j) - \psi(v_j)| + |V| + 2$ because Lemma1 can be applied since only one variable in $Supp(f_j)$ is in each variable block. $\sum_{j=1}^{|E|} size(f_j)$ equals $\sum_{j=1}^{|E|} (|\psi(u_j) - \psi(v_j)| + |V| + 2)$, that is, $K + (|V| + 2)|E|$. Since the same is true of $size(g_j)$, $\sum_{j=1}^{|E|} size(f_j) + \sum_{j=1}^{|E|} size(g_j) = 2K + 2(|V| + 2)|E|$. Next we consider the number of nodes for h_i in a wo-TSB.
We denote $\max\{\ t \mid \pi[t] \in B_i\}$, $\min\{\ t \mid \pi[t] \in B_i\}$ by t_{max}, t_{min}, respectively, and we define

$$\mathcal{H}_{ij} = \left\{ h_i|_{\bigwedge_{t=j}^{2|V||E|} \pi[t]=b_t} \mid b_t \in \{0,1\}, t = j, j+1, \cdots, 2|V||E| \right\}.$$

Then $size(h_i)$ can be expressed as $\left|\bigcup_{j=1}^{2|V||E|} \mathcal{H}_{ij}\right| + 1$. Observe that the following hold.
For $t_{max} < j \leq 2|V||E|$

$$\mathcal{H}_{ij} = \left\{ 0,\ \left(\prod_{t=t_{max}+1}^{j-1} \overline{\pi[t]} \right) \cdot \left(\bigoplus_{t=t_{min}}^{t_{max}} \pi[t] \right) \cdot \prod_{t=1}^{t_{min}-1} \overline{\pi[t]} \right\},$$

for $t_{min} < j \leq t_{max}$

$$\mathcal{H}_{ij} = \left\{ 0,\ \left(\bigoplus_{t=t_{min}}^{j-1} \pi[t] \right) \cdot \prod_{t=1}^{t_{min}-1} \overline{\pi[t]},\ \overline{\left(\bigoplus_{t=t_{min}}^{j-1} \pi[t] \right)} \cdot \prod_{t=1}^{t_{min}-1} \overline{\pi[t]} \right\},$$

for $1 < j \leq t_{min}$

$$\mathcal{H}_{ij} = \left\{ 0,\ \prod_{t=1}^{j-1} \overline{\pi[t]} \right\},$$

for $j = 1$

$$\mathcal{H}_{ij} = \{0,\ 1\}.$$

Thus $\sum_{i=1}^{|V|} size(h_i) = \sum_{i=1}^{|V|} \left(\left| \bigcup_{j=1}^{2|V||E|} \mathcal{H}_{ij} \right| + 1 \right) = |V|\,(2|V||E| + 2|E| + 1)$.

Secondly we count the nodes shared among the BDD's for f_j's, g_j's and h_i's. Without loss of generality, we can assume that all the nodes shared among the BDD's for f_j's, g_j's and h_i's belong to the BDD for h_i with $B_i = \varphi[|V|]$, that is, the nodes shared among the BDD's are all shared by the BDD for h_i with $B_i = \varphi[|V|]$, denoted by BDD_h.

The number of the nonterminal nodes, which are shared by BDD_h, in BDD's for f_j's and g_j's is, one in the case of type1, two in the case of type2.

The nonterminal nodes v, which are shared by BDD_h, in the BDD's for h_i's with $B_i \neq \varphi[|V|]$ satisfies $F(v) = \overline{\pi[t_{min}]} \prod_{t=1}^{t_{min}-1} \overline{\pi[t]}$ or $F(v) = \prod_{t=1}^{j-1} \overline{\pi[t]}$ $(1 < j \leq t_{min})$. Thus the number of such nonterminal nodes is $2(l-1)|E| + 3$ with $\varphi[l] = B_i$.

The number of terminal nodes, which are shared by BDD_h, in the BDD's for f_j's, g_j's and h_i's is $2(2|E| + |V| - 1)$. We denote the number of the nodes shared among f_j's, g_j's and h_i's by Δ. In the case of type1,

$$\Delta = \sum_{l=1}^{|V|-1} \{2(l-1)|E| + 3\} + 2(2|E| + |V| - 1) + 1.$$

Thus the number of the nodes in type1-wo-OLAD can be expressed as follows.

$$\sum_{j=1}^{|E|} size(f_j) + \sum_{j=1}^{|E|} size(g_j) + \sum_{i=1}^{|V|} size(h_i) - \Delta$$

We obtain the lemma by evaluating the above formula. The same is true of type2 except that nonterminal nodes for $\pi[1]$ is shared more than the case of type1 by a node. □