Electronic Colloquium on Computational Complexity, Report No. 167 (2014)

On the Minimization of (Complete) Ordered Binary Decision Diagrams[*]

Beate Bollig[**]

TU Dortmund, LS2 Informatik, Germany

Abstract. Ordered binary decision diagrams (OBDDs) are a popular data structure for Boolean functions. Some applications work with a restricted variant called complete OBDDs which is strongly related to nonuniform deterministic finite automata. One of its complexity measures is the width which has been investigated in several areas in computer science like machine learning, property testing, and the design and analysis of implicit graph algorithms. For a given function the size and the width of a (complete) OBDD is very sensitive to the choice of the variable ordering but the computation of an optimal variable ordering for the OBDD size is known to be NP-hard. Since optimal variable orderings with respect to the OBDD size are not necessarily optimal for the complete model or the OBDD width and hardly anything about the relation between optimal variable orderings with respect to the size and the width is known, this relationship is investigated. Here, using a new reduction idea it is shown that the size minimization problem for complete OBDDs and the width minimization problem are NP-hard.

1 Introduction

Motivation and results Ordered binary decision diagrams (OBDDs) are very restricted branching programs, a model well-known in complexity theory for space bounded computations. They are a popular data structure for Boolean functions [8]. Among the many areas of applications are verification, model checking, and computer aided design (for a survey see, e.g., [23]). Complete OBDDs are a restricted variant of OBDDs where on all computation paths all variables have to be tested. (For the formal definitions see Section 2.) They are closely related to nonuniform deterministic finite automata for Boolean languages L, where $L \subseteq \{0,1\}^n$, $n \in \mathbb{N}$. Therefore, they are a fundamental model. An early application are parallel algorithms for Boolean operations [16, 17]. A complexity measure of (complete) OBDDs is the width, the maximal number of nodes labeled by the same variable, which has been investigated in several areas in computer science. For example, OBDDs of bounded width have been studied in the machine learning context rather extensively. In particular, the influence of the width on the difficulty of the corresponding learning problem has been analyzed (see, e.g., [11]). It has been shown that OBDDs of width 2 are PAC-learnable while OBDDs of width at least 3 are as hard to learn as DNF formulas. Moreover, also in complexity theory the width of OBDDs has been investigated, e.g., in property testing. Lower and upper bounds have been shown for testing functions for the property of being computable by an ordered binary decision diagram of small width, i.e., given oracle access to a Boolean function f testing whether f can be represented by an OBDD of small width or is in a certain sense far from any such function [7, 14, 18]. Newman has presented a property testing algorithm for any property decidable by an ordered binary decision diagram of

[*] Preliminary versions of parts of this paper have been appeared in [2, 3]

[**] The author is supported by DFG project BO 2755/1-2.

ISSN 1433-8092

constant width [15]. The performance of his algorithm, i.e., the number of queries, depends highly on the width of the input OBDD. Furthermore, the OBDD width has been used in the analysis of implicit graph algorithms [19, 25]

Maybe the most important issue of OBDDs is the possibility to choose the variable ordering The size and the width of (complete) OBDD representing a function f, defined on n Boolean variables and essentially dependent on all of them, heavily depends on the chosen variable ordering and may vary between linear and exponential size and constant and exponential width with respect to n. An example for such a function is the most significant bit of binary addition. Therefore, the choice of good variable orderings is a key problem for the usability of (complete) OBDDs. For the size of OBDDs it is well-known that we cannot expect efficient algorithms for the computation of optimal variable orderings for a function given by an OBDD representation since the corresponding optimization problem is NP-hard [5]. Even more we cannot hope to design efficient approximation algorithms unless NP = P [20]. Optimal variable orderings for OBDDs do not necessarily carry over to optimal variable orderings for the complete model representing the same function. An example is the multiplexer, also called direct storage access function (for a formal definition see Section 3). Hence, it seems to be a natural question to ask for the computational complexity of the variable ordering problem for complete OBDDs. The NP-completeness or nonapproximability proofs for OBDDs do not work for the complete model because one key idea in the reductions is the crucial property that in OBDDs not all variables have to be tested on a path from the source to one of the sinks. Here, we prove that also for complete OBDDs the decision variant of the problem to compute an optimal variable ordering for a function given by a complete OBDD is NP-complete. The problems for OBDDs and complete OBDDs look quite similar but it is unclear how to relate the complexity of the problems directly. Therefore, for the reduction we choose the same NP-complete problem as in the NP-completeness proof for OBDDs but our construction is different and we use a new reduction idea. Although many exponential lower bounds on the size of (complete) OBDDs for Boolean functions are known and the method how to obtain such bounds is simple, there are only few functions where the size of (complete) OBDDs is asymptotically known exactly (see, e.g., [1, 4, 6].) Therefore, the proof may also be interesting on its own right in order to strengthen the ability to prove tight lower bounds. Moreover, knowledge on the relation of good variable orderings and Boolean functions is fundamental for the design of heuristics to compute good orderings. Furthermore, the computational complexity to find an optimal variable ordering such that the corresponding OBDD width for the representation of a given function is minimal is investigated. Using a new reduction it is shown that the computation of the minimal OBDD width is NP-hard. The frame of the two NP-completeness proofs in the paper is the same which underlines the robustness of our construction. The proof are different because of the different objective functions. In addition we present examples which prove that optimal variable orderings with respect to the width are not necessarily optimal with respect to the size of complete OBDDs and vice versa.

Organization of the paper The rest of the paper is organized as follows. In Section 2 we recall the main definitions concerning OBDDs and complete OBDDs. Furthermore, we give a short summary on the known results how the sizes of the two models are related. Section 3 presents some results on the relation between optimal variable orderings with respect to the size and the width of OBDDs. Furthermore, it is shown that optimal variable orderings with respect to the width are not necessarily optimal with respect to the size of complete OBDDs. Section 4 is devoted to the NP-completeness proofs of the variable ordering problems for the width of OBDDs and the size of complete OBDDs. It is described why the NP-completeness proof for OBDDs does not work for the complete variant. The last section contains a counterexample which proves that optimal variable orderings with respect to the size are not necessarily optimal with respect to the width of complete OBDDs. Finally, we finish the paper with some open problems.

2 Preliminaries

In the following we assume that the reader is familiar with fundamental graph theoretical concepts (otherwise see, e.g., [10] for more details). In this section we briefly recall the main notions concerning OBDDs and discuss the relation between OBDDs and complete OBDDs.

On (complete) ordered binary decision diagrams OBDDs are a popular dynamic data structure in areas working with Boolean functions, like circuit verification or model checking.

Definition 1. *Let $X_n = \{x_1, \ldots, x_n\}$ be a set of Boolean variables. A* variable ordering *π on X_n is given by a permutation on $\{1, \ldots, n\}$ leading to the ordered list $x_{\pi(1)}, \ldots, x_{\pi(n)}$ of the variables. A* π-OBDD *on X_n is a directed acyclic graph $G = (V, E)$ whose sinks are labeled by the Boolean constants 0 and 1 and whose non-sink (or* decision*) nodes are labeled by Boolean variables from X_n. Each decision node has two outgoing edges, one labeled by 0 and the other by 1. The edges between decision nodes have to respect the variable ordering π, i.e., if an edge leads from an x_i-node to an x_j-node, then $\pi^{-1}(i) < \pi^{-1}(j)$ (x_i precedes x_j in $x_{\pi(1)}, \ldots, x_{\pi(n)}$). At each node v a Boolean function $f_v \in B_n$, i.e., $\{0,1\}^n \to \{0,1\}$, is represented. A c-sink represents the constant function c. If f_{v_0} and f_{v_1} are the functions at the 0- or 1-successor of v, resp., and v is labeled by x_i, f_v is defined by Shannon's decomposition rule $f_v(a) := \overline{a_i} f_{v_0}(a) \vee a_i f_{v_1}(a)$. In order to evaluate $f_v(b)$, $b \in \{0,1\}^n$, start at v. After reaching an x_i-node choose the outgoing edge with label b_i until a sink is reached. The label of this sink defines $f_v(b)$. The* size *of a π-OBDD G is equal to the number of its decision nodes. A π-OBDD of minimal size for a given function f and a fixed variable ordering π is unique up to isomorphism. A π-OBDD for a function f is called* reduced *if it is the minimal π-OBDD for f. The* π-OBDD size *of a function f, denoted by π-OBDD(f), is the size of the reduced π-OBDD representing f. An* OBDD *is a π-OBDD for an arbitrary variable ordering π. The* OBDD size *of f is the minimum of all π-OBDD(f).*

Since (ordered) binary decision diagrams are a nonuniform model of computation, usually sequences of binary decision diagrams $(G_n)_{n\in\mathbb{N}}$ representing a sequence of Boolean functions $(f_n)_{n\in\mathbb{N}}$ with respect to sequences of variable orderings $(\pi_n)_{n\in\mathbb{N}}$ are considered, where f_n is a Boolean function on n variables. In the following we simplify the notation because the meaning is clear from the context. To distinguish between vertices of a graph and BDD vertices in the rest of the paper, we refer to the latter as nodes. A variable ordering is called a *natural variable ordering* if π is the identity $1, 2, \ldots, n$. Obviously a variable ordering π can be identified with the corresponding ordering $x_{\pi(1)}, \ldots, x_{\pi(n)}$ of the variables if the meaning is clear from the context.

In order to define the width of a π-OBDD complete OBDDs are introduced. Here, there are only edges between nodes labeled by neighboring variables, i.e., if an edge leads from an x_i-node to an x_j-node, then $\pi^{-1}(i) = \pi^{-1}(j) - 1$.

Definition 2. *An* OBDD *on X_n is* complete *if all paths from the source to one of the sinks have length n. The* width *of a complete* OBDD *is the maximal number of nodes labeled by* *the same variable. A complete π-*OBDD *of minimal size for a given function f and a fixed variable ordering π is unique up to isomorphism. A π-*OBDD *for a function f is called* quasi-reduced *if it is the minimal complete π-*OBDD *for f. The complete π-*OBDD size *of a function f, denoted by π-*QOBDD(f)*, is the size of the quasi-reduced π-*OBDD *representing f. The* π-OBDD width *of a function f, denoted by π-*OBDD$_w(f)$*, is the width of the quasi-reduced π-*OBDD *representing f. A complete* OBDD*, or* QOBDD *for short, is a complete π-*OBDD *for an arbitrary variable ordering π. The complete* OBDD size *or* QOBDD size*i of f, denoted by* QOBDD(f)*, is the minimum of all π-*QOBDD(f)*. The* OBDD width *of f is the minimum of all π-*OBDD$_w(f)$*.*

Complete OBDDs are closely related to nonuniform finite automata for Boolean languages L, where $L \subseteq \{0,1\}^n$ (see, e.g., Section 3.2 in [23]). With respect to natural variable orderings they differ from nonuniform deterministic finite automata only in the minor aspect that variables have still to be tested if the corresponding subfunction is the constant function 0.

Definition 3. *A* level *p, $1 \le p \le n$, in a complete π-*OBDD *G contains all nodes in G labeled by the variable at position p in the variable ordering π.*

Let f be a Boolean function on the variables $x_1, \dots, x_n$. The *subfunction* $f_{|x_i=c}$, $1 \le i \le n$ and $c \in \{0,1\}$, is defined as $f(x_1, \dots, x_{i-1}, c, x_{i+1}, \dots, x_n)$. A function f *depends essentially* on a Boolean variable z if $f_{|z=0} \neq f_{|z=1}$. Now, the size of the (quasi-)reduced π-OBDD representing f can be described by the following result.

Proposition 1 ([21]). *The number of $x_{\pi(i)}$-nodes in the quasi-reduced (reduced) π-*OBDD *for f is equal to the number of different subfunctions*

$$f_{|x_{\pi(1)}=a_1,\dots,x_{\pi(i-1)}=a_{i-1}} \text{ (that essentially depend on } x_{\pi(i)}\text{)},$$

where $a_1, \dots, a_{i-1} \in \{0,1\}$.

In a reduced OBDD each node encodes a different function, whereas in a quasi-reduced OBDD each node labeled by the same variable represents a different function.

On the relation between OBDDs and QOBDDs Wegener has compared the size of quasi-reduced OBDDs with the size of reduced OBDDs for functions defined on n Boolean variables. For the natural variable ordering he has proved that the quotient is at most $1 + \mathcal{O}(2^{-n/3} \cdot n)$ for almost all Boolean functions, i.e., all but a fraction of $\mathcal{O}(2^{-n/3})$. This result does not rule out the possibility that for many functions there exists some ordering of the variables where the difference is significantly larger but it has also been shown that the maximal quotient of the quasi-reduced OBDD and the size of the reduced OBDD with respect to the same variable ordering for a function f where the maximum is taken over all variable orderings is at most $1 + \mathcal{O}(2^{-n/3} \cdot n)$ for almost all Boolean functions (Theorem 3 and 4 in [24]). Nevertheless, one may ask for the quotient of the representation sizes not for almost all but for some important Boolean functions. It is obvious that π-QOBDD$(f) \le (n+1)\pi$-OBDD(f) for all Boolean functions $f \in B_n$. Furthermore, it is not difficult to see that π-QOBDD$(f) = \Theta(n \cdot (\pi\text{-OBDD}(f)))$ for some function f that depends essentially on all n variables. It is easy to see that the multiplexer is an example for such a function and a variable ordering where the so-called address variables are tested before the so-called data variables (see Section 3 for a formal definition). In order to compare the size of OBDDs and QOBDDs more closely one may ask whether there exists a Boolean function $f \in B_n$ that depends essentially on all variables and QOBDD$(f_n) = \Theta(n \cdot \text{OBDD}(f_n))$. In [6] this question has been answered in the affirmative. Therefore, we can conclude that the OBDD size of a function may be a size factor of $\Theta(n)$ smaller than its QOBDD size. The multiplexer has been a good candidate for a function with the largest possible gap between the OBDD and the QOBDD size but it has turned out that the multiplexer only leads to a size gap of $\Theta(n/\log n)$ [6] (see also Section 3). Despite all these results we hardly know anything about the relation between the variable orderings that lead to minimal OBDDs and variable orderings for minimal QOBDDs for a given function f. In the next section we will recall that optimal variable orderings for one model are not necessarily optimal for the other one.

3 On Optimal Variable Orderings for (Q)OBDDs

In this section we recall that optimal variable orderings with respect to the size of OBDDs are not necessarily optimal with respect to the size of complete OBDDs and vice versa. Moreover, we show that variable orderings that are optimal with respect to the size are not necessarily optimal for the width of OBDDs. Finally, we prove that optimal variable orderings with respect to the width are not necessarily optimal with respect to the size of (Q)OBDDs. Our example for the first results is the multiplexer or direct storage access function.

Definition 4. *Let $n = 2^k$. The multiplexer MUX_n is defined on $n+k$ variables $a_{k-1}, \ldots, a_0$, $x_0, \ldots, x_{n-1}$. The output of $\mathrm{MUX}_n(a,x)$ is $x_{|a|_2}$, where $|a|_2 := \sum_{i=0}^{k-1} a_i 2^i$. The a-variables are called address variables, the x-variables are the data variables.*

Let π be the variable ordering $a_0, a_1, \ldots, a_k, x_0, x_1, \ldots, x_{n-1}$ and π' be the ordering $a_{k-1}, a_{k-2}, \ldots, a_{k-m}, x_0, x_1, \ldots, x_{n-1}, a_{k-m-1}, \ldots, a_0$. The ordering π is optimal with respect to the OBDD size for the multiplexer and π' is optimal with respect to the QOBDD size [4, 6]. More precisely, π-OBDD(MUX) is $\mathcal{O}(n)$, π-QOBDD(MUX) is $\mathcal{O}(n^2)$, and π'-QOBDD(MUX) is $\mathcal{O}(n^2/\log n)$. Moreover, π-OBDD$_w$(MUX) is $\mathcal{O}(n)$ and π'-OBDD$_w$(MUX) is $\mathcal{O}(n/\log n)$ but π'-OBDD(MUX) is $\mathcal{O}(n^{3/2}/\log n)$. Therefore, we can conclude that π is optimal with respect to the OBDD size but not the OBDD width or the QOBDD size and π' is optimal with respect to the QOBDD size but not regarding the OBDD size.

In the remaining section we show that an optimal variable ordering with respect to the width is not necessarily optimal with respect to the size of complete OBDDs. Later we will see that also an optimal variable ordering with respect to the size of complete OBDDs is not necessarily optimal with respect to the width of OBDDs (see Section 5). Here, the proof idea is simple. We combine two Boolean functions defined on disjoint sets of variables by conjunction, where the first one dominates the width of the OBDD representation. A concatenation of an optimal variable ordering for the first Boolean function together with variable orderings for the second one results in an optimal variable ordering with respect to the width of the OBDD representation for the combination. Here, we use the fact that the width of a Boolean function cannot be smaller than the width of any subfunction. For the second Boolean function we can choose suborderings which are not optimal with respect to the size without changing the width of the representation for the combination of the Boolean functions. Therefore, we are done. Now, we get more precisely. The auxiliary function g_4 is defined on the x-variables $x_1, \ldots, x_4$ and $g_4 := (x_1 \wedge x_2) \vee (x_3 \wedge x_4)$. Let f_n be an arbitrary Boolean function defined on the y-variables $y_1, \ldots, y_n$ whose OBDD width is at least 4. Obviously such a function exists. Let π be an optimal variable ordering with respect to the OBDD width of f_n. A new Boolean function is defined as conjunction of f_n and g_4. The function h_n is defined on the variables $x_1, \ldots, x_4, y_1, \ldots y_n$, and $h_n(x,y) := f_n(y) \wedge g_4(x)$.

Let π' be $y_{\pi(1)}, \dots, y_{\pi(n)}, x_1, x_2, x_3, x_4$ and let π'' be $y_{\pi(1)}, \dots, y_{\pi(n)}, x_1, x_3, x_2, x_4$. It is not difficult to prove that the variable orderings π' and π'' are both optimal with respect to the OBDD width of h_n but π'-QOBDD(h_n) is smaller than π''-QOBDD(h_n). Therefore, the variable ordering π'' is optimal with respect to the OBDD width but not with respect to the size of complete OBDDs for the function h_n. Figure 1 shows the quasi-reduced sub-OBDDs for g_4.

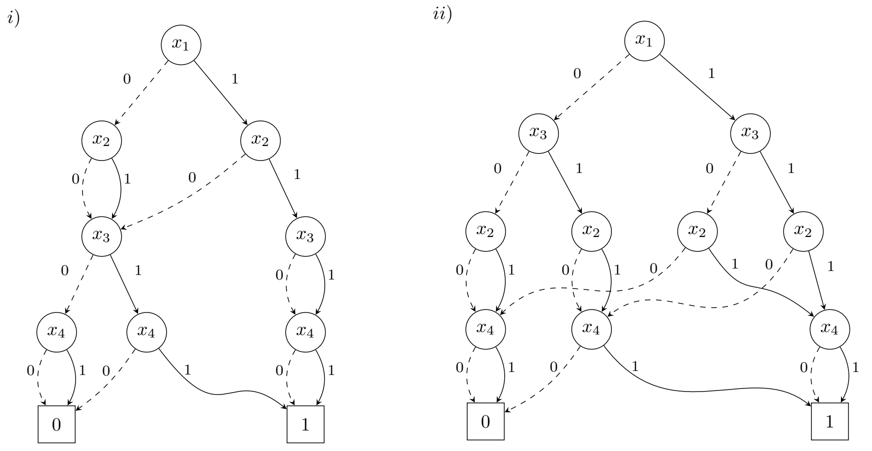

Fig. 1. Quasi-reduced OBDDs with respect to the variable ordering *i)* x_1, x_2, x_3, x_4 and *ii)* x_1, x_3, x_2, x_4 for the function $g_4(x) = (x_1 \wedge x_2) \vee (x_3 \wedge x_4)$.

Note, our counterexample can also be used to show that optimal variable orderings with respect to the width are not necessarily optimal regarding the OBDD size and it can also be used with respect to alternative definitions of the OBDD width (see also the remark at the end of Section 4).

4 On Variable Ordering Problems for (Q)OBDDs

In this section we show that improving the QOBDD size or the OBDD width of a Boolean function is NP-complete. First, we define our BDD optimization problems. We start our NP-completeness proofs by the presentation of two well-known NP-complete layout problems. Next, we show how graphs can be transformed into Boolean functions. Afterwards we look at the corresponding BDD representations and a special class of variable orderings called sandwich variable orderings. We will see how the values for the objective functions for inputs to the considered layout problems are related to the size and the width of BDDs with respect to sandwich variable orderings for the corresponding Boolean functions. Finally, we prove that each variable ordering can be modified into a sandwich variable ordering without increasing the size or the width of the BDD representation, respectively. Remarkably, the two NP-completeness proofs have almost the same frame.

The problems Optimal QOBDD and Optimal OBDD Width In the rest of this section the complexity of the problems Optimal QOBDD and Optimal OBDD Width is investigated.

Definition 5 (Optimal QOBDD). *Given a* QOBDD *G and a size bound B, the answer to the problem* Optimal QOBDD *is yes iff the function represented by G can be represented by a* QOBDD (*respecting an arbitrary variable ordering*) *with at most B nodes.*

The problem Optimal QOBDD is in the complexity class NP since a QOBDD can be guessed. The equivalence of QOBDDs with respect to different variable orderings can be verified similarly as for OBDDs in deterministic polynomial time [12]. The same holds for the next problem.

Definition 6 (Optimal OBDD Width). *Given a* QOBDD *G and a bound W, the answer to the problem* Optimal OBDD Width *is yes iff the function represented by G can be represented by a* QOBDD (*respecting an arbitrary variable ordering*) *whose width is at most W.*

The main result of the paper is the following one.

Theorem 1. *The problem* Optimal QOBDD *and the problem* Optimal OBDD Width *are* NP*-complete.*

In the following it does not matter whether our input for the investigated optimization problems are OBDDs or QOBDDs because one model can be transformed into the other one in linear size for a fixed variable ordering (and functions that essentially depend on all Boolean variables). Furthermore, we sometimes talk about the QOBDD width of Boolean functions which is by definition the same as the OBDD width. Our proof that Optimal OBDD Width is NP-complete also holds for other definitions of the OBDD width (see also the remark at the end of this section). Since we have already seen that it is not difficult to show that both problems are in NP, we finish the proof of Theorem 1 in the rest of this section by presenting polynomial time reductions from two well-known NP-complete problems introduced in the next paragraph.

Two NP-complete layout problems As for Optimal OBDD in [5, 22] our NP-hardness proof for Optimal QOBDD uses a polynomial time reduction from the well-known NP-complete problem Optimal Linear Arrangement. Also for the proof that Optimal OBDD Width is NP-complete we use a so-called layout problem named CUTWIDTH which is known to be NP-hard [13].

A linear arrangement or a layout of an undirected graph with n vertices can be seen as a bijective function from the vertex set into the set $\{1, 2, \ldots, n\}$. Assuming that the vertices are identified by the integers from 1 to n, a layout is a permutation. Graph layout problems are a class of combinatorial optimization problems where the task is to find a permutation of the vertices of an input graph that optimizes a problem specific objective function. A large number of important problems from different applications can be formulated as graph layout problems (see, e.g., [9] for a nice survey on graph layout problems from an algorithmic point of view). Next, we introduce Optimal Linear Arrangement and Cutwidth. The first problem asks for a layout such that the sum of the length of all edges in minimal. For the decision variant the goal is to decide whether there is a linear arrangement such that the sum of the edge lengths is not greater than a bound given as part of the input.

Definition 7 (Optimal Linear Arrangement). *Given an undirected graph $H = (V = \{1, 2, \ldots, n\}, E)$ and a bound S, the answer to the problem* Optimal Linear Arrangement (OLA) *is yes iff there is a permutation τ on $\{1, 2, \ldots, n\}$ such that*

$$cost(\tau) := \sum_{\{u,v\} \in E} |\tau^{-1}(u) - \tau^{-1}(v)| \leq S.$$

Obviously, the cost of τ measures the length of all edges if the vertices of H are arranged in linear order with respect to τ.

For Cutwidth the maximal number of edges cut by any line inserted between two consecutive vertices has to be minimized over all possible layouts. Given a graph H and a positive integer C the decision variant asks whether there is a linear arrangement of its vertices such that any line inserted between two neighboring vertices in the ordering cuts at most C edges of the input graph.

Definition 8 (CUTWIDTH). *Given an undirected graph $H = (V = \{1, 2, \ldots, n\}, E)$ and a permutation τ on $\{1, 2, \ldots, n\}$, the cutwidth of a vertex $v \in V$ with respect to τ, denote $\mathrm{CW}_\tau(v)$ is the number of edges $(u, w) \in E$ satisfying $\tau^{-1}(u) \leq \tau^{-1}(v) < \tau^{-1}(w)$. The cutwidth of H with respect to τ is the maximum cutwidth of its vertices: $\mathrm{CW}_\tau(H) := \max_{v \in V} \mathrm{CW}_\tau(v)$. The answer to the problem* CUTWIDTH *is yes iff for an input graph H and a bound C there is a permutation τ such that $\mathrm{CW}_\tau(H)$ is at most C.*

It is not difficult to see that for the definition of $\mathrm{CW}_\tau(H)$ it does not matter whether $\mathrm{CW}_\tau(v)$ is the number of edges $(u, w) \in E$ satisfying $\tau^{-1}(u) \leq \tau^{-1}(v) < \tau^{-1}(w)$ or $\tau^{-1}(u) < \tau^{-1}(v) \leq \tau^{-1}(w)$.

The polynomial time reductions For the proof of Theorem 1 we use polynomial time reductions from OLA and Cutwidth. Since the frame of the NP-hardness proofs is almost the same, we describe the two polynomial time reductions together. For both layout problems one part of the input is a given graph $H = (V, E)$. In the rest of the proof let $m := |E|$. Furthermore, w.l.o.g. we assume that the degree of each vertex in V, i.e., the number of its neighbors, is at least 2. OLA and CUTWIDTH have a second parameter B and C, respectively, as input. For the reduction from OLA to Optimal QOBDD we have to transform the input (H, B) for OLA into an input (G, S) for Optimal QOBDD in polynomial time such that the QOBDD size of the function represented by G is at most S iff the cost of an optimal linear arrangement for H is at most B. For the reduction from CUTWIDTH to Optimal OBDD Width we have to transform the input (H, C) for CUTWIDTH into an input (G, W) for Optimal OBDD Width in polynomial time such that the OBDD width of the function represented by G is at most W iff the cost of an optimal cut for H is at most C. The transformation from an undirected graph H into a Boolean function represented by an QOBDD of polynomial size is the same for both polynomial time reductions.

For the kth edge $\{i, j\} \in E$, $1 \leq k \leq m$ and $i, j \in \{1, 2, \ldots, n\}$, we introduce an edge function $f_k(v_1, v_2, \ldots, v_n) = (v_i \vee v_j)$. It is easy to see that the size of a quasi-reduced OBDD representing f_k with respect to a variable ordering $v_{\pi(1)}, v_{\pi(2)}, \ldots, v_{\pi(n)}$ is $n + |\pi^{-1}(i) - \pi^{-1}(j)|$ plus additional nodes representing the constant function 0. Figure 2 shows an example of a quasi-reduced OBDD representing an edge function. Another intuition for the definition of the edge function is the following. If we ignore the nodes representing the constant function 0 in a quasi-reduced π-OBDD for the edge function f_k for now, there are two nodes labeled by a variable v_{i_1} iff $\pi^{-1}(i) < \pi^{-1}(i_1) \leq \pi^{-1}(j)$. For all other v-variables the number of nodes is 1. Now, if we consider the quasi-reduced π-OBDDs for all edge functions of an input graph H without mergings between the different representations for the edge functions and ignoring the nodes representing the constant function 0, the maximal number of nodes labeled by the same variable is equal to $m + \mathrm{CW}_\pi(H)$. Here, we use the observation above that for the definition of $\mathrm{CW}_\pi(H)$ it does not matter whether $\mathrm{CW}_\pi(v)$ is the number of edges (u, w) in the input graph H satisfying $\pi^{-1}(u) \leq \pi^{-1}(v) < \pi^{-1}(w)$ or $\pi^{-1}(u) < \pi^{-1}(v) \leq \pi^{-1}(w)$.

We have still several problems to solve. First, in order to obtain a single Boolean function the edge functions have to be combined to one function. Second, we have to make sure that

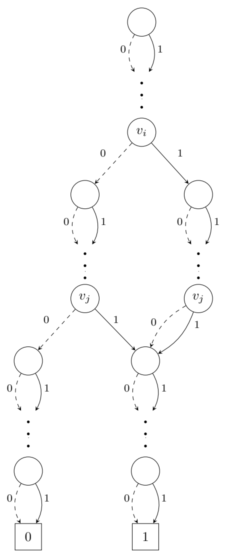

Fig. 2. A quasi-reduced OBDD for the edge function $f_k(v) = v_i \vee v_j$ with respect to a variable ordering π where $\pi^{-1}(i) < \pi^{-1}(j)$.

representations for different edge functions do share the nodes representing the constant function 0 but no other nodes. Finally, the part of the QOBDD for the representation of the edge functions should define the width of the ordered binary decision diagram.

In the following we briefly discuss why the polynomial time reduction in the NP-hardness proof for Optimal OBDD in [5] does not work for our optimization problems. Already in [5] for each edge of the input graph for OLA a corresponding Boolean function has been defined (which is different from our edge function). In order to avoid the sharing of nodes for different edge functions, in case of OBDDs different edge functions have been defined on disjoint sets of variables. To ensure that the orderings for the variables of the different edge functions are not completely independent but correspond to an ordering of the vertices of the input graph for OLA, another function has been added. Its representation size is very large and and its OBDD size is optimal if all variables that correspond to the same vertex in the input graph for OLA are tested one after another. Unfortunately, this idea does not work for QOBDDs. Moreover, the construction does not work for Optimal OBDD Width. Hence, we use another construction to combine the edge function and to prevent the unwanted sharing of BDD nodes. The idea is to use two simple functions to frame the edge functions. For the framing we use counting functions. Intuitively, we will show that there exists an optimal variable ordering where the variables that represent the vertices of the input graph for the layout problems are tested in the middle. Therefore, in the corresponding QOBDD the representations for the edge functions cannot share BDD nodes. Furthermore, we guarantee that a level that corresponds to one of these variables defines the width of the representation.

Now, we make the ideas more precise. For a variable vector $z = (z_1, \ldots, z_n)$, $n \in \mathbb{N}$, let $\| z \|$ be $\sum_{i=1}^{n} z_i$. The function $F \in B^{2m+n}$ is defined on the variable vectors $u = (u_1, u_2, \ldots, u_m)$, $v = (v_1, v_2, \ldots, v_n)$, and $w = (w_1, w_2, \ldots, w_m)$, and

$$F(u, v, w) := \bigvee_{i=1}^{m} (\| u \| = i) \wedge f_i(v) \wedge (\| w \| = i).$$

The u- and the w-variables are called weight variables and the v-variables are called vertex variables since they represent the vertices of the input graph H. We call a vertex variable v_j *essential for an edge function* f_i iff j is incident to the ith edge in H. F is symmetric on the u-variables and on the w-variables, respectively. Here, a function is *symmetric* on two variables x_i and x_j if the function does not change when exchanging the variables x_i and x_j. If $P = \{i_1, \ldots, i_m\}$ is the set of positions of the u-variables (or w-variables, respectively) in a variable ordering π, it does not matter which u-variable is tested on which position in P for the corresponding size or width of a QOBDD for F. Moreover, the roles of the u-and the w-variables are exchangeable, therefore, in the remaining part of the section we assume w.l.o.g. that the u-variables are tested in the ordering $u_1, u_2, \ldots, u_m$ and the w-variables in the ordering $w_1, w_2, \ldots, w_m$ and u_1 is the first variable of all u- and w-variables. Our transformation computes the (quasi-reduced) OBDD representing F with respect to the ordering $u_1, u_2, \ldots, u_m, v_1, v_2, \ldots, v_n, w_1, w_2, \ldots, w_m$ in polynomial time.

A *sandwich variable ordering* is a variable ordering where the v-variables are tested between the u- and the w-variables, and all u-variables as well as all w-variables are tested consecutively. Figure 3 shows the simplified shape of a quasi-reduced OBDD for F with respect to sandwich orderings.

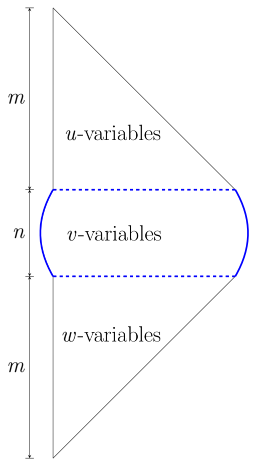

Fig. 3. The (simplified) shape of a quasi-reduced OBDD for F with respect to a sandwich ordering, where the vertex variables are tested between the u- and the w-variables. The width of the quasi-reduced OBDD is at least $m + 1$ (and at most $2m + 1$) and the width is defined by a level that contains nodes labeled by a v-variable.

The following lemma is not difficult to prove.

Lemma 1. *Let π be a sandwich variable ordering and let π' be the subordering of π on the v-variables. Then the π-QOBDD size of $F(u, v, w)$ is*

$$m \cdot (m + 1)/2 + n \cdot m + cost(\pi') + m \cdot (m + 1)/2 + (m - 1) + (n + m).$$

Moreover, π-OBDD_w of $F(u, v, w)$ is $m + 1 + \text{CW}_{\pi'}(H)$.

We are now able to define the bounds S and W in our reductions. Let $S := m \cdot (m + 1)/2 + n \cdot m + B + m \cdot (m + 1)/2 + (m - 1) + (n + m)$ and $W := m + 1 + C$.

In order to prove the correctness of our reductions we have to show that the input graph H has a linear arrangement whose cost is bounded by B iff F can be represented by a QOBDD with at most S nodes and it has a minimum cut whose cost is bounded by C iff F can be represented by a QOBDD whose width is at most W. Using Lemma 1 the only-if-parts are easy. The if-parts of the correctness proofs are the tricky ones. By our considerations above it remains to prove that some variable ordering for $F = (u, v, w)$ which is optimal with respect to the QOBDD size or OBDD width, respectively, is a sandwich variable ordering.

Our proof structure is to modify a given (optimal) variable ordering π in three phases until it is a sandwich variable ordering. We will see that each phase does not increase the size and the width of the BDD representation, therefore we will be done.

In all phases we do not change the ordering among the u-variables, among the v-variables, and among the w-variables, respectively. Remember we assume w.l.o.g. that the first weight variable is a u-variable. First, we ensure that all u-variables are tested before all w-variables. We do this by exchanging the positions of the first w-variable in the variable ordering and the following u-variable without increasing the size and the width of the corresponding QOBDD. Since the procedure can be iterated, we are done. Figure 4 illustrates the modification in the variable ordering after one step.

Lemma 2. *Let π be a variable ordering on the u-, v-, and w-variables and let i_k be the position of the variable u_k and let j_k be the position of the variable w_k, $1 \le j \le m$. Furthermore, let j_1 be between i_l and i_{l+1}, $l \in \{1, \ldots, m-1\}$. Let π' be the variable ordering where the variable w_1 is at position i_{l+1} and u_{l+1} is at position j_1 and all other variables are ordered according to π. Then π'-OBDD$_w(F)$ is not larger than π-OBDD$_w(F)$ and π'-QOBDD(F) is not larger than π-QOBDD(F).*

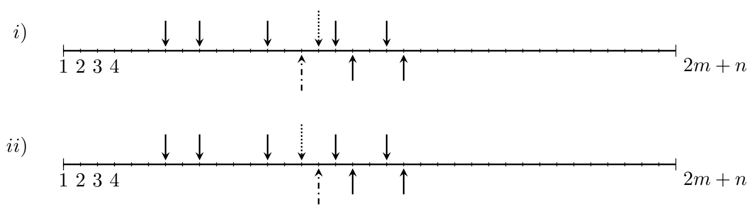

Fig. 4. A simplified description how the variable orderings π and π' differ. The arrows pointing down indicate the positions of the u-variables and the arrows pointing up the positions of the w-variables. Figure *i)* symbolizes the variable ordering π, Figure *ii)* the ordering π'.

Next, we change the variable ordering in such a way that the u-variables are tested in the beginning. Figure 5 visualizes how the variable orderings differ.

Lemma 3. *Let π be a variable ordering on the u-, v-, and w-variables where all u-variables are before the w-variable. Let π' be the variable ordering that starts with the u-variables followed by the remaining variables in the same order as in π. Then π'-OBDD$_w(F)$ is not larger than π-OBDD$_w(F)$ and π'-QOBDD(F) is not larger than π-QOBDD(F).*

Finally, we modify the variable ordering such that the w-variables are tested in the end. Figure 6 illustrates the difference between variable orderings.

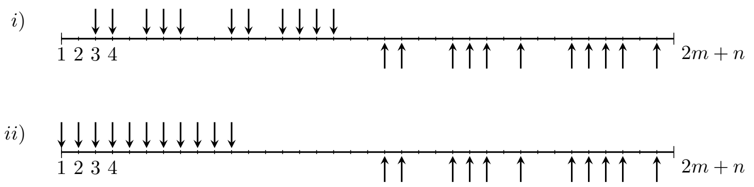

Fig. 5. A simplified description how the variable orderings π and π' differ. Figure *i)* symbolizes the variable ordering π and Figure *ii)* illustrates the ordering π'. In both orderings the u-variables are before the w-variables, furthermore in π' all u-variables are in the beginning of the ordering.

Lemma 4. *Let π be a variable ordering on the u-, v-, and w-variables that starts with the u-variables. Let π' be the variable ordering where the u-variables are in the beginning of the ordering, the v-variables are ordered in the same suborder as in π, and the w-variables are the last variables in the ordering. Then π'-$\mathrm{OBDD}_w(F)$ is not larger than π-$\mathrm{OBDD}_w(F)$ and π'-$\mathrm{QOBDD}(F)$ is not larger than π-$\mathrm{QOBDD}(F)$.*

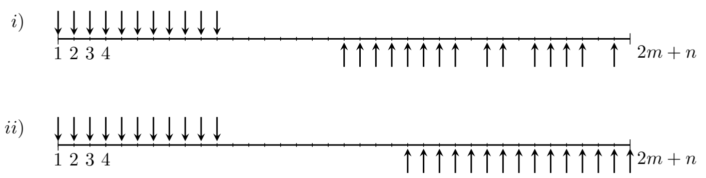

Fig. 6. A simplified description how the variable orderings π and π' differ. Figure *i)* symbolizes the variable ordering π and Figure *ii)* illustrates the ordering π'. In both orderings the u-variables are in the beginning. Moreover, in π' the w-variables are in the end of the ordering.

Since each phase does not increase the size and the width of the BDD representation, we are done. By Lemma 2-4 we have shown the following results.

Corollary 1. *There exists a sandwich variable ordering that is optimal with respect to the size of a* QOBDD *for $F(u, v, w)$.*

Corollary 2. *There exists a sandwich variable ordering that is optimal with respect to the width of an* OBDD *for $F(u, v, w)$.*

Remark Note that throughout the paper the width of an OBDD is defined via the complete or leveled model as is often the case in the literature (see, e.g., [7, 11, 14, 15, 18]). One might argue that in applications often reduced OBDDs are used and therefore another definition of the width like the maximal number of nodes labeled by the same variable or the maximal number of nodes with the same distance from the source would be more appropriate. However, in this case our NP-completeness proof can be adapted by a modification of the edge functions from $f(v) = v_i \vee v_j$ into

$$f(v) = (v_i \oplus v_j) \wedge \bigwedge_{\substack{1 \le \ell \le n \\ \ell \notin \{i,j\}}} v_\ell.$$

5 On Optimal Variable Orderings for (Q)OBDDs

In this section we demonstrate that an ordering optimal with respect to the QOBDD size is not necessarily optimal with respect to the OBDD width. For clarity we present our counterexample not as a Boolean function but as a graph. First, we justify why we can do that. We have already shown that the problems Optimal QOBDD and Optimal OBDD Width are NP-complete (see Section 4). For these results we have used two polynomial time reductions from two well-known NP-complete graph problems: Optimal Linear Arrangement and Cutwidth. The first part of the reductions in the NP-completeness proofs is the polynomial time transformation from an input graph $G = (V, E)$ into a corresponding Boolean function F. For each edge a single Boolean function is created and in order to combine all Boolean functions for the edges into one Boolean function and to avoid mergings between subfunctions for different edges, the so-called edge functions are framed by counting functions. The NP-completeness proofs have shown that there exist optimal variable orderings called sandwich variable orderings where the v-variables are tested between the u- and the w-variables (see Figure 3 for the structure of an optimal QOBDD with respect to such an ordering). Now, let S be the sum of the length of all edges for a linear ordering of the vertices in G and C be the cutwidth of the ordering. Then there are $((m+1)n + S)$ v-nodes in a reduced QOBDD for F with respect to a sandwich ordering where the v-variables are ordered with respect to the given linear ordering. Moreover, it can be shown that the width is $m + 1 + C$. Therefore, it is sufficient to present an input graph G, for which there exist two linear orderings of the vertices such that the sum of the length of all edges is minimal but the corresponding cutwidth of G with respect to the two orderings are different. It is not surprising that an optimal linear ordering for Optimal Linear Arrangement is not necessarily optimal for Cutwidth. Nevertheless, despite all our efforts we could not find an example in the literature for which an optimal solution for Optimal Linear Arrangement is not optimal for Cutwidth. Therefore, in the following we present such an example. Figure 7 shows the graph we are looking for.

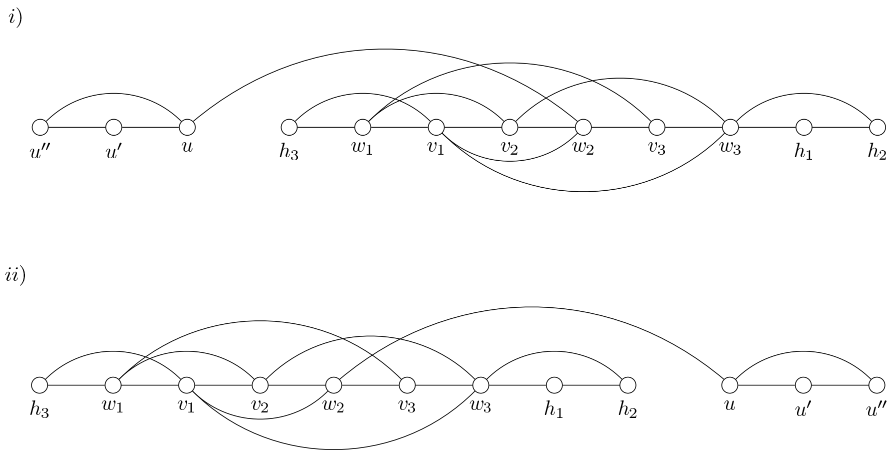

Fig. 7. The graph in our counterexample with respect to two different linear orderings *i)* $u'', u', u, h_3, w_1, v_1, v_2, w_2, v_3, w_3, h_1, h_2$ and *ii)* $h_3, w_1, v_1, v_2, w_2, v_3, w_3, h_1, h_2, u, u', u''$.

For the orderings

$$u'', u', u, h_3, w_1, v_1, v_2, w_2, v_3, w_3, h_1, h_2$$

and

$$h_3, w_1, v_1, v_2, w_2, v_3, w_3, h_1, h_2, u, u', u''$$

the sum of the lengths of all edges is 36. By case inspection it can be shown that the orderings are minimal with respect to the sum of the lengths of all edges. The cutwidth of the first ordering is 6, the cutwidth of the second one 5. Therefore, we are done. To generalize our result for sequences of Boolean functions, we combine the Boolean function F of our graph by conjunction with the parity function on the variables $z_1, z_2, \ldots, z_n$. The variable orderings to be considered start with the z-variables in arbitrary order followed by the two orderings for F mentioned above. We finish the presentation of our counterexample by some remarks on the intuition how the graph has been constructed. Our counterexample consists of four subgraphs. For technical reasons in one of our NP-completeness proofs, each vertex has a degree of at least two which we have to take into consideration by the construction of our counterexample. The core of the graph is a complete bipartite graph on the variables in $V = \{v_1, v_2, v_3\}$ and $W = \{w_1, w_2, w_3\}$ together with an additional edge between v_1 and v_2. The additional edge should destroy the symmetry between V and W. Moreover, there are two auxiliary subgraphs. The first one ensures that the vertices v_1 and w_1 have to be at the left or right border in a linear subordering of the V- and W-variables in order to minimize the sum of the lengths of all edges. The task of the second one is similar with respect to the vertices v_3 and w_3 but in addition another vertex is introduced to guarantee that the vertex w_2 has to be just in the middle of a linear (sub-)ordering. The important part of the fourth subgraph is the vertex u which is connected to the vertex w_2. As each vertex should have degree at least two the vertices u' and u'' are defined which build together with the vertex u a triangle. The cutwidth of the graph varies if u is tested before or after the vertex w_2, respectively.

Concluding Remarks

We finish the paper with some open problems. An open question is how large the difference could be between the minimal size of a complete OBDD for a function f and the size of a complete minimal OBDD for f. We know that the difference cannot asymptotically be larger than a factor of n and a lower bound of $\log n$ is known for the multiplexer [6]. Moreover, given two variable orderings π and π' such that π-OBDD(f) and π'-OBDD(f) are (almost) of the same order, e.g., $\pi\text{-OBDD}(f) = \Theta(\pi'\text{-OBDD}(f))$, are the quasi-reduced OBDDs for f with respect to π and π' also of the same order? Furthermore, we do not know whether there exists a function defined on n Boolean variables such that the difference between QOBDD(f) and $\text{OBDD}_w(f)$ is only $o(n)$. Otherwise, if there are any approximation algorithms for one minimization problem they could be used for the other one (and vice versa) leading approximately to the same performance ratio.

References

1. Bollig, B.: On the size of (generalized) OBDDs for threshold functions. Inf. Process. Lett. 109(10), 499–503 (2009)
2. Bollig, B.: On the complexity of some ordering problems. In: Proc. of MFCS. pp. 118–129. LNCS 8635, Springer (2014)
3. Bollig, B.: On the width of ordered binary decision diagrams. In: Proc. of COCOA. LNCS 8881, Springer (2014)

4. Bollig, B., Range, N., Wegener, I.: Exact OBDD bounds for some fundamental functions. Theory of Computing Systems 47(2), 593–609 (2010)
5. Bollig, B., Wegener, I.: Improving the variable ordering of OBDDs is NP-complete. IEEE Trans. Computers 45(9), 993–1002 (1996)
6. Bollig, B., Wegener, I.: Asymptotically optimal bounds for OBDDs and the solution of some basic OBDD problems. Journal of Computing and System Science 61(3), 558–579 (2000)
7. Brody, J., Matulef, K., Wu, C.: Lower bounds for testing computability by small width OBDDs. In: TAMC. pp. 320–331. LNCS 6648, Springer (2011)
8. Bryant, R.: Graph-based algorithms for boolean function manipulation. IEEE Trans. Computers 35(8), 677–691 (1986)
9. Díaz, J., Petit, J., Serna, M.J.: A survey of graph layout problems. ACM Comput. Surv. 34(3), 313–356 (2002)
10. Diestel, R.: Graph Theory, 4th Edition, Graduate texts in mathematics, vol. 173. Springer (2012)
11. Ergün, F., Kumar, R., Rubinfeld, R.: On learning bounded-width branching programs. In: COLT. pp. 361–368 (1995)
12. Fortune, F., Hopcroft, J., Schmidt, E.: The complexity of equivalence and containment for free single variable program schemes. In: Proc. of ICALP. pp. 227–240. LNCS 62, Springer (1978)
13. Gavril, F.: Some NP-complete problems on graphs. In: 11th Conference on Information Science and Systems. pp. 91–95 (1977)
14. Goldreich, O.: On testing computability by small width OBDDs. In: APPROX-RANDOM. pp. 574–587 (2010)
15. Newman, I.: Testing membership in languages that have small width branching programs. SIAM J. Comput. 31(5), 1557–1570 (2002)
16. Ochi, H., Ishiura, N., Yajima, S.: Breadth-first manipulation of SBDD of boolean functions for vector processing. In: DAC. pp. 413–416 (1991)
17. Ochi, H., Yasuoka, K., Yajima, S.: Breadth-first manipulation of very large binary-decision diagrams. In: ICCAD. pp. 48–55 (1993)
18. Ron, D., Tsur, G.: Testing computability by width-two OBDDs. Theor. Comput. Sci. 420, 64–79 (2012)
19. Sawitzki, D.: The complexity of problems on implicitly represented inputs. In: Proc. of SOFSEM. pp. 471–482. LNCS 3831, Springer (2006)
20. Sieling, D.: The nonapproximability of OBDD minimization. Information and Computation 172(2), 103–138 (2002)
21. Sieling, D., Wegener, I.: NC-algorithms for operations on binary decision diagrams. Parallel Processing Letters 3, 3–12 (1993)
22. Tani, S., Hamagushi, K., Yajima, S.: The complexity of the optimal variable ordering problems of a shared binary decision diagram. In: Proc. of ISAAC. pp. 389–396. LNCS 762, Springer (1993)
23. Wegener, I.: Branching programs and binary decision diagrams: theory and applications. SIAM (2000)
24. Wegener, I.: The size of reduced OBDDs and optimal read-once branching programs for almost all boolean functions. IEEE Trans. Computers 43(11), 1262–1269 (1994)
25. Woelfel, P.: Symbolic topological sorting with OBDDs. J. Discrete Algorithms 4(1), 51–71 (2006)

Appendix A: Proof of Lemma 2

Let G be the quasi-reduced π-QOBDD representing the function F on the u-, v-, and w-variables, and let G' be the quasi-reduced π'-QOBDD for F.

Proposition 1 implies directly that only the size of a level between $j_1 + 1$ to i_{l+1} in the variable ordering π and π', respectively, could be different in G and G'. Therefore, we compare the number of subfunctions represented by a node labeled by a v-variable at a position between j_1+1 and $i_{l+1}-1$ in G and G', and additionally we compare the number of u_{l+1}-nodes in G with the number of w_1-nodes in G'. Our aim is to prove that any level p in G is at least as large as any level p in G' for $j_1+1 \leq p \leq i_{l+1}$. We start with descriptions of the subfunctions in G and G'. In the following we assume that f_0 is the constant function 0.

Definition 9. *An edge function f_i* can still be addressed *for a subfunction represented on a level between j_1+1 and i_{l+1} iff*
- *the number of u-variables set to 1, or u_{one} for short, is k, $k \leq i$,*
- *the number of w-variables set to 1, or w_{one} for short, is k', $k' \in \{0,1\}$,*
- *$m-l \geq i-k$ and $m-1 \geq i-k'$.*

Now, a subfunction of F can be described by u_{one}, w_{one}, and the assignments to the v-variables that are essential for the edge functions that can still be addressed. If u_{one} is k, $0 \leq k < l$ or $w_{one}=1$, these edge functions are $f_k,\dots,f_{k+m-l}$. For $k=l$ and $w_{one}=0$ the still addressable edge functions are $f_l,\dots,f_{m-1}$. For an edge function that can still be addressed we only have to know whether one of its essential vertex variables has already been set to 1 or which of its essential vertex variables could still be set to 1. Let f'_i denote the corresponding subfunction of f_i. Then the considered subfunctions represented in G can be described by triples $\left(u_{one}, w_{one}, \left[f'_{u_{one}},\dots,f'_{u_{one}+m-l}\right]\right)$, $u_{one} \leq l-1$ or $w_{one}=1$, and by $\left(l,0,\left[f'_l,\dots,f'_{m-1}\right]\right)$. In a similar way the subfunctions in G' can be described by triples, where w_{one} is always 0 since the first w-variable is tested not until position i_{l+1} in π'.

The following two facts are not difficult to see.

Fact 1 *Two non-constant subfunctions represented in G and obtained by replacing the same subset of variables by constants could not be equal if only the number of u-variables set to 1 but not the number of w-variables set to 1 are equal.*

Fact 2 *Two non-constant subfunctions represented in G obtained by replacing the same subset of variables by constants and for which the number of u-variables set to 1 are different can only be equal if they can be described by $(u_{one}, w_{one}, [*])$ and $(u'_{one}, w'_{one}, [*])$, where $u'_{one} = u_{one}-1$, $w_{one}=1$, and $w'_{one}=0$.*

The reason is simple that because of $F's$ definition on all computation paths to the 1-sink the number of u-variables set to 1 has to be equal to the number of w-variables set to 1. Now, what is the difference between the investigated subfunctions represented in G and G'? Roughly speaking, in G' there are new subfunctions where the number of u-variables set to 1 is $l+1$. On the other hand, w_{one} is in all cases 0, and the number of edge functions that could still be addressed is $m-(l+1)+1 = m-l$, which means 1 less than before.

Together with the two observations above, the following facts are sufficient.

- The number of subfunctions represented in G described by $(k,1,[*])$ is at least as large as the number of subfunctions represented in G' by $(k,0,[*])$ for $k<l$.
- The number of subfunctions represented in G described by $(l,0,[*])$ is at least as large as the number of subfunctions represented in G' by $(l,0,[*])$.
- The number of subfunctions represented in G described by $(l,1,[*])$ is at least as large as the number of subfunctions represented in G' by $(l+1,0,[*])$.

Since the three classes of subfunctions are disjoint, we are done. Altogether, by proving that every level in G is at least as large as in G' we have shown that π'-OBDD$_w(F)$ is not larger than π-OBDD$_w(F)$ and π'-QOBDD(F) is not larger than π-QOBDD(F). Therefore, we are done.

Appendix B: Proof of Lemma 3

Let G be the quasi-reduced π-QOBDD representing the function F on the u-, v-, and w-variables, and let G' be the quasi-reduced π'-QOBDD for F. First, we prove that the width of G' is not larger than G. We start with a classification of the edge functions on a given level p with respect to a variable ordering.

Definition 10. *We call an edge function, or by abuse of notation edge for short,* fresh *at a level p with respect to a variable ordering τ iff none of its essential vertex variables is at a position p' where $p' < p$ in τ. The edge function is* incomplete *at a level p iff exactly one of its essential vertex variables is at a position p', $p' < p$. The edge function is* complete *at a level p iff both its essential vertex variables are at positions p', $p' < p$. Let $e_f^p(\tau)$ be the number of fresh edges, let $e_i^p(\tau)$ be the number of incomplete edges, and let $e_c^p(\tau)$ be the number of complete edges at a position p in the variable ordering τ.*

In the following let ℓ be the position of the variable u_m in π. Proposition 1 implies that only the number of nodes on a level ℓ', $\ell' \leq \ell$, in the variable ordering π and π', respectively, could be different in G and G'. Let $e_i^{\max}$ be the maximal number of incomplete edges at a level ℓ', $\ell' \leq \ell$, in π. Then the maximal width on a level ℓ', $\ell' \leq \ell$ in G' is $m + e_i^{\max} + 1$. Therefore, our aim is to prove that there exists a level that contains at least $m + e_i^{\max} + 1$ nodes in G.

We use the same notation as in the proof of Lemma 2. Subfunctions represented on level ℓ in G can be described as tuples $\left(u_{one}, \left[f'_{u_{one}}, f'_{u_{one}+1}\right]\right)$, where $u_{one} \in \{0, 1, \ldots, m-1\}$ and $f'_{u_{one}}$ and $f'_{u_{one}+1}$ are subfunctions of $f_{u_{one}}$ and $f_{u_{one}+1}$, respectively. By definition of $e_i^{\max}$ we know that there are at least $e_i^{\max}$ edges that are not fresh at level ℓ. Therefore, there are at least $e_i^{\max}$ different values for u_{one} such that the edge function $f_{u_{one}+1}$ is not fresh. If $f_{u_{one}+1}$ is not fresh, there are at least two different non-constant subfunctions $(u_{one}, [*, *])$ on level ℓ in G. Together with the observation that two non-constant subfunctions obtained from F by replacing only u- and v-variables cannot be equal if the number of u-variables set to 1 are different, we can conclude that there are at least $m + e_i^{\max}$ u_m-nodes in G. Now, we make a case inspection.

- If there are two edge functions f_j and f_{j+1}, $j \in \{1, \ldots, m-1\}$, where f_j is not fresh and f_{j+1} is fresh at level ℓ, there exists another subfunction $(j, [*, f_{j+1}])$ not counted before which is represented at a node labeled by u_m.
- If there are two edge functions f_j and f_{j+1} one incomplete and the other one complete, both complete, or both incomplete with different first essential vertex variables (with respect to the variable ordering π), there are at least three subfunctions $(j, [*, *])$. Therefore there exists at least one additional u_m-node not counted before.
- Otherwise, there is only one vertex variable on a position $\ell' \leq \ell$. In this case there are already $m + 1 + e_i^{\ell}(\pi) = m + 1 + e_i^{\max}$ different subfunctions represented on level $\ell + 1$ in G and we are done.

Until now we have seen that the width of G' is not larger than the width of G. In the rest of the proof we show that G' is not larger than G. We say that a variable z has already been tested on a level p in a ρ-QOBDD if z is at a position p', $p' < p$, in ρ. Our proof idea is to use the accounting method in the following way. First, we show that for the first v-variable there is at most one additional node in G' for every u-variable tested after this v-variable in π. For all other v-variables v_j there are at most two more v_j-nodes in G' than in G for every u-variable at a position after v_j in π. On the other hand, for every u-variable u_j at least two u_j-nodes can be saved in G' for every v-variable but the first one at a position before u_j in π. For the first v-variable in π only one u_j-node can be saved. Altogether, G' cannot be larger than G. Now, we make the ideas more precise. We start with the following fact.

Fact 3 *Two non-constant subfunctions obtained from F by replacing only u- and v-variables cannot be equal if the number of u-variables set to 1 are different.*

Iff z is the variable at a position p in a variable ordering, we call the level p also z-level. Next, we look at the number of nodes in G labeled by a vertex variable v_j at a position p in π where the number of u-variables at a position p' in π is g_j, $p' < p$ and $g_j < m$. (Note that by Proposition 1 for $g_j = m$ the number of nodes labeled by such a variable cannot be different in G and G'.) Remember $e_i^p(\tau)$ is the number of incomplete edges at position p in a variable ordering τ. Note that the number of incomplete edges is the same on the v_j-levels in G and G' since the suborderings of the v-variables are the same in π and π'.

Claim 1 *There are at most $2(m-g_j)$ more v_j-nodes in G' than in G. For the first v-variable v_{j_1} in the variable orderings π and π' the difference can even be at most $(m - g_{j_1})$.*

Proof. We start to estimate the number of v_j-nodes in G'. It is not difficult to see that for each fresh or complete edge on the v_j-level only one node labeled by v_j and for an incomplete edge on the v_j-level two v_j-nodes are necessary. One further v_j-node represents the constant function 0. Next, we prove that the number of v_j-nodes in G is at least $(g_j + 1) + e_i^p(\pi) - (m - g_j)$. Using the notation from the proof of Lemma 2 we can describe the considered subfunctions represented at a v_j-node in G by tuples $\left(u_{one}, \left[f'_{u_{one}}, \dots, f'_{u_{one}+m-g_j}\right]\right)$. If an edge function $v' \vee v''$ is incomplete and v' is the variable already tested, there are two subfunctions of this edge function: the constant function 1 and the function v''. Note that subfunctions of different incomplete edge functions on a level p may not be independent because their first essential vertex variable can be the same. Therefore, we have to be very careful not to count QOBDD nodes more than once. To give an example of subfunctions which are not independent we consider two edge functions $v_i \vee v_k$ and $v_i \vee v_\ell$, where v_i has already been tested. There are only two possibilities: both subfunctions are 1 or one is v_k and the other one is v_ℓ. Therefore, we define an injective mapping from the incomplete edge functions f_i, $i \le g_j$, into the set $\{1, \dots, g_j\}$ in a straightforward way: f_i is mapped to i. As a result we can count two v_j-nodes for each incomplete edge in $\{f_1, \dots, f_{g_j}\}$. The reason is that there are at least two subfunctions $(i, [*])$, $1 \le i \le g_j$, represented at a v_j-node in G if f_i is incomplete. At least $e_i^p(\pi) - (m - g_j)$ edge functions in $\{f_1, \dots, f_{g_j}\}$ are incomplete. Altogether, using Fact 3 we have shown that there are at least $(g_j + 1) + e_i^p(\pi) - (m - g_j)$ v_j-nodes in G and the difference of the number of v_j-nodes in G and G' is at most $(m + e_i^p(\pi) + 1) - ((g_j + 1) + e_i^p(\pi) - (m - g_j)) = 2(m - g_j)$.

Since there cannot be incomplete edges if no vertex variable has yet been tested, the difference between the number of v_{j_1}-nodes in G and G' is at most $m - g_{j_1}$. $\square$

In the rest of the proof we compare the number of nodes labeled by a u-variable in G and G'. Let k_j be the number of v-variables that are before the variable u_j in π. To finish the proof it is sufficient to show that in G' there are at least $2k_j - 1$ nodes labeled by u_j less than in G.

Claim 2 *There are at least $2k_j - 1$ less u_j-nodes in G' than in G, where k_j is the number of v-variables that are before u_j in π.*

Proof. There are j u_j-nodes in G'. We have to prove that there are at least $j + (2k_j - 1)$ u_j-nodes in G. As before we describe subfunctions represented at a u_j-node in G as tuples $\left(u_{one}, \left[f'_{u_{one}}, \ldots, f'_{u_{one}+m-(j-1)}\right]\right)$, where $0 \leq u_{one} \leq j - 1$. If all vertex variables before u_j in π are set to 1, the resulting subfunctions in the second part of the tuples can only be the constant subfunction 1 and fresh edge functions. We have j different u_j-nodes for the j different values for u_{one} and replacing all vertex variables before u_j by 1. It remains to prove that there are $2k_j - 1$ further u_j-nodes in G. The problem is not to count subfunctions more than once. In the following an edge function f_i belongs to the tuples $(u_{one}, [*])$, or u_{one}-tuples for short, iff $u_{one} \leq i \leq u_{one} + m - (j - 1)$. Inversely, we say that u_{one}-tuples contain an edge function f_i iff $u_{one} \leq i \leq u_{one} + m - (j - 1)$. By abuse of notation, we also say that an edge belongs to a tuple or a tuple contains an edge. Since by assumption each vertex is incident to at least two edges, we can define an injective mapping from the k_j vertex variables that are before u_j in π to edge functions for which the corresponding vertex variable is essential. Let $VV = \{v_{i_1}, \ldots, v_{i_{k_j}}\}$ be the set of the vertex variables before u_j in π and let f_{j_ℓ} be the chosen edge function for v_{i_ℓ}, $1 \leq \ell \leq k_j$. Our aim is to prove that for every chosen edge function (except possibly f_m) there are at least two further subfunctions. We consider two cases.

If $j = 1$, we consider for each chosen edge function f_{j_ℓ} the following partial assignments:

- all vertex variables in VV without the vertex variables which are essential for f_{j_ℓ} are set to 1, the latter to 0,
- all vertex variables in $VV \setminus \{v_{i_\ell}\}$ are set to 0, v_{i_ℓ} is set to 1.

It is not difficult to see that for each chosen edge the assignments lead to two different subfunctions. It remains to prove that we do not count any subfunction more than once for different chosen edge functions. Two vertex variables cannot be both essential for two different edge functions because multiple edges are not allowed. Moreover, each vertex has degree at least two and each edge function is chosen for at most one vertex variable in VV. Therefore, we can conclude that there are altogether at least $2k_j - 1$ further nodes labeled by u_j and we are done. (For $k_j > 1$, there are even at least $2k_j$ further u_j-nodes.)

If $j > 1$, each edge function f_i, $i < m$, belongs to at least two different kind of tuples, f_m belongs only to $(j - 1)$-tuples. Now, we consider for each chosen edge function (except possibly f_m) two different kind of tuples which contain the investigated edge function. Let f_{j_ℓ} belong to u_{one}-tuples. We consider partial assignments to the variables where the variables $u_1, \ldots, u_{j-1}$ are set in such a way that the number of u-variables set to 1 is u_{one} and all vertex variables in VV without the vertex variables which are essential for f_{j_ℓ} are set to 1, the latter to 0. Similarly to the case that j is equal to 1, we can conclude that there are altogether at least $2k_j - 1$ further nodes labeled by u_j and we are done. □

Appendix C: Proof of Lemma 4

Let G be the quasi-reduced π-QOBDD representing the function F on the u-, v-, and w-variables, and let G' be the quasi-reduced π'-QOBDD for F. First, we prove that the width of G' is not larger than G. The proof follows the line of the proof of Lemma 3 and we use the same notation for the description of subfunctions as in the proofs of Lemma 2 and Lemma 3. Let ℓ be the position of the variable w_1 in π. Proposition 1 implies directly that only the number of nodes on a level ℓ', $\ell' > \ell$, in the variable ordering π and π', respectively, could be different in G and G'. Let $e_i^{\max}$ be the maximal number of incomplete edges at a level ℓ',

$\ell' > \ell$, in π. The maximal width on a level ℓ', $\ell' > \ell$ in G' is $m + e_i^{\max} + 1$. Therefore, our aim is to prove that level $\ell + 1$ contains at least $m + e_i^{\max} + 1$ nodes in G. If the number of u-variables set to 1 is 0, the corresponding subfunctions of F are the constant function 0. Hence, it is sufficient to prove that level $\ell + 1$ in G contains at least $m + e_i^{\max}$ different non-constant subfunctions. Obviously, an edge function that is incomplete at a level ℓ', $\ell' > \ell$, cannot be complete at a level ℓ'', where $\ell'' \leq \ell'$. Thus there are at least $e_i^{\max}$ edges at level $\ell + 1$ that are not complete, therefore $e_i^{\ell+1}(\pi) + e_f^{\ell+1}(\pi) \geq e_i^{max}$. In the following we make a case inspection depending whether the edge f_m is still fresh or not at level $\ell + 1$ in π.

- If the edge function f_m is fresh at level $\ell+1$ in G, there are at least $e_i^{\ell+1}(\pi)+e_c^{\ell+1}(\pi)+1$ subfunctions $(*, [1])$ on level $\ell + 1$ in G. Furthermore, for the incomplete edge functions at level $\ell + 1$ there are at least $e_i^{\ell+1}(\pi) + 1$ subfunctions $(*, [v_i])$, where v_i is the second essential vertex variable of an incomplete edge function at level $\ell + 1$ in π. Finally, there are $2e_f^{\ell+1}(\pi) - 1$ subfunctions $(j, [f_j])$, $j < m$, and $(j - 1, [f_j])$, $j \leq m$, on level $\ell + 1$ in G, where f_j is an arbitrary fresh edge function at level $\ell + 1$. Therefore, altogether there are at least $m + e_i^{\ell+1}(\pi) + e_f^{\ell+1}(\pi) + 1 > m + e_i^{\max}$ different non-constant subfunctions represented on level $\ell + 1$ in G and we are done.
- If the edge function f_m is not fresh at level $\ell+1$ in G, there are at least $e_i^{\ell+1}(\pi)+e_c^{\ell+1}(\pi)$ subfunctions $(*, [1])$ on level $\ell + 1$. Moreover, there are altogether at least $e_i^{\ell+1}(\pi)$ subfunctions $(*, [v_i])$, where v_i is the second essential vertex variable of an incomplete edge function at level $\ell + 1$. Finally, there are $2e_f^{\ell+1}(\pi)$ subfunctions $(j, [f_j])$ and $(j - 1, [f_j])$, on level $\ell+1$ in G, where f_j is an arbitrary fresh edge function at level $\ell+1$. Altogether, there are at least $m + e_i^{\ell+1}(\pi) + e_f^{\ell+1}(\pi) \geq m + e_i^{\max}$ different non-constant subfunctions represented on level $\ell + 1$ in G and we are done.

Altogether, we have seen that π'-OBDD$_w(F)$ is not larger than π-OBDD$_w(F)$.

In the rest of the proof we show that G' is not larger than G. We start with a description of the subfunctions of F where at least all u-variables have been replaced by constants. Remember that for any edge function that can still be addressed we only have to know whether at least one of its essential vertex variables has been set to 1 or which of its essential vertex variables could still be set to 1. Let f'_q denote the corresponding subfunction of f_q. If u_{one} is the number of u-variables set to 1 and w_{one} is the number of w-variables set to 1, the tuple $\left(u_{one} - w_{one}, \left[f'_{u_{one}}\right]\right)$ describes a subfunction of F, where $u_{one} \geq w_{one}$. We can identify the tuples with the corresponding subfunctions and vice versa. The first component of a tuple is called the *distance value*. Obviously, two non-constant subfunctions represented on the same level cannot be the same if the distance values are different. Furthermore, they cannot be the same if the corresponding subfunctions in the second components of the tuples are different. Therefore, two non-constant subfunctions described as $(r, [f_{j_1}])$ and $(r, [f_{j_2}])$, $j_1 \neq j_2$, cannot be equal because different edges are incident to different sets of vertices.

First, we remark that the number of w-nodes in G' representing F is minimal with respect to all variable orderings with the u-variables in the beginning. Remember, ℓ is the first position of a w-variable in π. By Proposition 1 only the number of nodes labeled by a v-variable at a position p, where $p > \ell$, can be different in G and G'. Let v_i be a variable at position p_i in π, $p_i > \ell$, and let g_i be the number of w-variables at a position less than p_i in π. Next, we consider two cases for these v-variables.

Claim 3 *If the number of incomplete and complete edges $e_i^{p_i}(\pi) + e_c^{p_i}(\pi)$ at position p_i in π is at most $m - g_i$, then the number of v_i-nodes in G' is at most as large as in G.*

Proof. Since the suborderings of the v-variables are not different in π and π', the number of incomplete edges is the same on the v_j-levels in G and G'. Let p'_i be the position of v_i in

π'. We can conclude that the number of v_i-nodes in G' is $m + 1 + e_i^{p'_i}(\pi') = m + 1 + e_i^{p_i}(\pi)$. In the following we classify the subfunctions represented at a v_i-node in G in four classes.

i) There is one v_i-node representing the constant function 0 in G, where all u-variables are set to 0.
ii) For each fresh edge f_j at position p_i there exists at least one v_i-node in G representing a subfunction $(*, [f_j])$.
iii) Since by assumption the number $e_i^{p_i}(\pi) + e_c^{p_i}(\pi)$ of incomplete and complete edges at position p_i is at most $m - g_i$, we know that there are at least $e_i^{p_i}(\pi) + e_c^{p_i}(\pi)$ different subfunctions $(*, [1])$.
iv) Let i_k be the number of incomplete edges at position p_i whose second essential vertex variable in the variable ordering is v_k. Since i_k is at most $e_i^{p_i}(\pi)$ and therefore at most $m - g_i$, there exist at least i_k nodes labeled by v_i in G that represent a subfunction $(*, [v_k])$. Moreover, non-constant subfunctions $(*, [v_{k'}])$ and $(*, [v_{k''}])$ for different vertices $v_{k'}$ and $v_{k''}$ cannot be the same.

Obviously, subfunctions from different classes cannot be the same. Therefore, we can conclude that the number of v_i-nodes in G is at least $1 + e_f^{p_i}(\pi) + (e_i^{p_i}(\pi) + e_c^{p_i}(\pi)) + e_i^{p_i}(\pi) = m + 1 + e_i^{p_i}(\pi) = m + 1 + e_i^{p'_i}(\pi')$. □

Now, let v_{i_1} be the first v-variable in π for which the number of incomplete and complete edges $e_i^{p_{i_1}}(\pi) + e_c^{p_{i_1}}(\pi)$ at position p_{i_1} in π is larger than the number $m - g_{i_1}$ of w-variables after v_{i_1} in π. Since the number of incomplete and complete edges and the number of already tested w-variables are increasing, we know that this requirement is also satisfied for all v-variables at a position larger than p_{i_1} in π. Let $v_{i_2}, \dots, v_{i_k}$ be these v-variables.

As we have already mentioned before the number of v_j-nodes in G' is $m + 1 + e_i^{p'_j}(\pi') = m + 1 + e_i^{p_j}(\pi)$ since the subordering of the v-variables does not change from π to π'. Now, we look at the v_j-nodes in G. We may assume that g_j, $j \in \{i_1, \dots, i_k\}$, is larger than 2 because otherwise it can easily be shown that there are at least as many v_j-nodes in G as in G'. As before we classify the subfunctions represented at a v_j-node in four classes.

i) There is one v_j-node representing the constant function 0 in G.
ii) For each fresh edge f_ℓ at position p_j there exists at least one v_j-node in G representing a subfunction $(*, [f_\ell])$.
iii) Since by assumption the number $e_i^{p_j}(\pi) + e_c^{p_j}(\pi)$ of complete and incomplete edges at position p_j is larger than $m - g_j$, there are at least $m - g_j + 1$ v_j-nodes representing different subfunctions $(*, [1])$.
iv) Let $n_{j'}$ be the number of incomplete edges at position p_j whose second essential vertex variable in the variable ordering „pi is $v_{j'}$. (We call these edges incomplete $v_{j'}$-edges for short.) There exist at least $\min(n_{j'}, m - g_j + 1)$ nodes labeled by v_j in G that represent a subfunction $(*, [v_{j'}])$.

If for all incomplete $v_{j'}$-edges, there exist at least $n_{j'}$ nodes labeled by v_j representing a subfunction $(*, [v_{j'}])$ in G, the difference between the number of v_j-nodes in G and G' is at most $g_j - 1$. If not, the difference is at most $(g_j - 1) + (g_j - 2)$. (Note, that we have used the fact that there cannot be more than $m - 1$ incomplete $v_{j'}$-edges.) Our aim is to show that there are at least as many additional w-nodes in G as additional v-nodes necessary in G'. We are faced with two problems. We have to guarantee that we count only additional w-nodes. Moreover, we have to be very careful not to count the additional w-nodes more than once. In order to do so, we make use of an advanced accounting method.

We start with the solution to the first problem. We need $m-\ell'+1$ nodes labeled by $w_1,\dots,w_{m-\ell'+1}$ that represent a subfunction $(\ell',[1])$ in G', $\ell'\in\{1,\dots,m\}$. Furthermore, there are $m-1$ w-nodes representing a subfunction $(0,[1])$ in G'. In G there are $m-\ell'+1$ w-nodes that represent a subfunction $(\ell',[f_{\ell'}])$ or $(\ell',[1])$ in G (depending on the position of the first essential vertex variable for the edge function $f_{\ell'}$). Moreover, there are $m-1$ w-nodes that represent one of the subfunctions $(0,[f_1])$ or $(0,[1])$ (again depending on the position of the first essential vertex variable for f_1). In the following in order to compensate for the additional v-nodes necessary in G', we count none of these w-nodes.

Next, we attack the second problem. Since for each vertex there are at least two incident edges and the input graph for OLA is connected, we can define an injective mapping from the vertices $v_{i_1},\dots,v_{i_k}$ to the set of edges incident to these vertices without considering the edge function f_1. To be more precise, a vertex is mapped to one of its incident edges. Let $f_{c_1},\dots,f_{c_k}$ be the set E' of the chosen edges. A chosen edge f_{c_j} is fresh at position p_{i_j} if v_{i_j} is the first essential vertex variable for f_{i_j} in π. Otherwise, f_{c_j} is incomplete and v_{i_j} is the second essential vertex variable. Now, we prove that there are at least $g_{i_j}-1$ further w-nodes in G. There exist one w_r-node, $1<r\le c_j$, representing one of the subfunctions $\left(c_j-r+1,\left[f_{c_j}\right]\right)$ and $\left(c_j-r+1,\left[v_{i_j}\right]\right)$ and $g_{i_j}-c_j$ w-nodes representing $\left(0,\left[f_{c_j}\right]\right)$ or $\left(0,\left[v_{i_j}\right]\right)$. (Here, we assume that c_j is not larger than g_{i_j} but the bounds can easily be adapted.) The following observation is crucial to guarantee that the w-nodes are not counted more than once. For different edge functions $f_{c_{j'}}$ and $f_{c_{j''}}$ subfunctions $\left(*,\left[f_{c_{j'}}\right]\right)$ and $\left(*,\left[f_{c_{j''}}\right]\right)$ are different. Moreover, nodes labeled by a w-variable that represent a subfunction $\left(*,\left[v_{i_j}\right]\right)$ are only counted for the vertex variable v_{i_j}. Furthermore, we do not count again nodes representing subfunctions $\left(c_j,\left[f_{c_j}\right]\right)$. Therefore, we have $g_{i_j}-1$ additional w-nodes for each vertex v_{i_j}. Next, we are looking for $g_{i_j}-2$ further w-nodes for each vertex v_{i_j}. If $c_j\neq m$, then there exists one w_r-node, $2<r\le c_j$, representing one of the subfunctions $\left(c_j-r+2,\left[f_{c_j}\right]\right)$ and $\left(c_j-r+2,\left[v_{i_j}\right]\right)$ and $g_{i_j}-c_j$ w-nodes representing $\left(1,\left[f_{c_j}\right]\right)$ or $\left(1,\left[v_{i_j}\right]\right)$. Altogether, we have shown that there are $g_{i_j}-2$ further w-nodes for v_{i_j} if $c_j\neq m$. Finally, we investigate the special case that c_j is equal to m. Here, the problem is that we cannot find $g_{i_j}-2$ further w-nodes in a similar way as in the other cases. Hence, we use another idea in order to prove that there are at least as many nodes in G as in G'. Until now we have already proved that there are $m-(g_{i_j}-1)+1$ v_{i_j}-nodes representing subfunctions $(*,[1])$ and the constant function 0 and $g_{i_j}-1$ further w-nodes. It is sufficient to prove that there are at least $e_i^{p_{i_j}}(\pi)$ further nodes. Our aim is to show that for each edge incomplete at level p_{i_j} there is a further node in G that has not been counted before. For this reason we classify the edges which are incomplete at position p_{i_j} into two classes, edges which are already incomplete at position ℓ and edges which are still fresh at position ℓ. (Remember that ℓ is the position of the first w-variable w_1 in π.) Until now we have only counted w_1-nodes representing subfunctions $(c_{j'},[f_{c_{j'}}])$ or $(c_{j'},[1])$, $1\le j'\le k$. Therefore, for each edge in the first class we have one further node labeled by w_1. Obviously, a fresh edge remains fresh until its first essential vertex variable is tested and for each edge fresh at position ℓ but incomplete at position p_{i_j} in π, its first essential vertex variable is at a position between ℓ and p_{i_j}. For the vertex variables in $\{v_{i_1},\dots,v_{i_k}\}$ we have not yet taken into account nodes that represent subfunctions $(*,[f_q])$ for fresh edges f_q up to now. As a result, if there is no v-variable besides $\{v_{i_1},\dots,v_{i_k}\}$ at a position between ℓ and p_{i_j}, we can conclude that for each edge in the second class, incomplete at the v_{i_j}-level but fresh at position ℓ, there is at least one further node labeled by a vertex variable and we are done. Otherwise, let v_i be the first v-variable at a position larger ℓ in π. It is easy to see that

edges fresh at position ℓ are also fresh at position p_i. We know that $e_i^{p_i}(\pi) + e_c^{p_i}(\pi)$ is at most $m - g_i$ because $v_i \neq v_{i_1}$ and we can conclude that g_i is less than m. The reason is that for $g_i = m$ the number of fresh edges at position p_i would be m which means that all w-variables would be before the vertex variables in π and the size of G would be too large anyway. Therefore, for all fresh edges f_q at position p_i except possibly f_m there are at least two different subfunctions $(*, [f_q])$ with different distance values represented at a v_i-node. Furthermore, there is at least one further v_{i_j}-node representing a subfunction $(m - g_{i_j}, [f_m])$ or $(m - g_{i_j}, [v_{i_j}])$ depending whether v_{i_j} is the first or second essential vertex variable for f_m. We have not counted such a v_{i_j}-node before because we have only counted v_{i_j}-nodes representing subfunctions that belong to class *i)* or *iii)* up to now. Altogether, we have at least $e_i^{p_{i_j}}(\pi)$ further nodes which we have not counted before and we are done.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de