Electronic Colloquium on Computational Complexity, Revision 1 of Report No. 1 (1998)

The Nonapproximability of OBDD Minimization*

Detlef Sieling
FB Informatik, LS II, Univ. Dortmund
44221 Dortmund, Fed. Rep. of Germany
sieling@ls2.cs.uni-dortmund.de

Abstract The size of Ordered Binary Decision Diagrams (OBDDs) is determined by the chosen variable ordering. A poor choice may cause an OBDD to be too large to fit into the available memory. The decision variant of the variable ordering problem is known to be NP-complete. We strengthen this result by showing that for each constant $c > 1$ there is no polynomial time approximation algorithm with the performance ratio c for the variable ordering problem unless $P = NP$. This result justifies, also from a theoretical point of view, to use heuristics for the variable ordering problem.

1 Introduction

Ordered Binary Decision Diagrams (OBDDs) are the state-of-the-art data structure for Boolean functions in programs for problems like logic synthesis, model checking or circuit verification. The reason is that many functions occurring in such applications can be represented by OBDDs of reasonable size and that for operations on Boolean functions like equivalence test or synthesis with binary operators efficient algorithms on OBDDs are known. Already in the seminal paper of Bryant (1986) asymptotic optimal algorithms for many operations are presented. The most important exception is the variable ordering problem for OBDDs, i.e. the task to compute for a given function a variable ordering minimizing the size of the OBDD for this function. The lack of an efficient algorithm for the variable ordering problem affects the applicability of OBDDs because there are important functions for which OBDDs are of reasonable size only for few variable orderings.

We distinguish two different versions of the variable ordering problem. For the first one the function to represented is given by a circuit. In this case it is NP-hard to compute the size of an OBDD for an optimal variable ordering since the satisfiability problem for OBDDs can be solved in polynomial time while the satisfiability problem for circuits is NP-hard. Heuristics for this version of the variable ordering problem extract information on the connections between the variables from the circuit description. Such heuristics are given, e.g., in the papers of Fujita, Fujisawa and Kawato (1988), Malik, Wang, Brayton and Sangiovanni-Vincentelli (1988) and Butler, Ross, Kapur and Mercer (1991).

In this paper we focus on the second variant of the variable ordering problem, which is defined in the following way:

* An extended abstract of some results of this paper has been accepted for STACS'98 (Sieling (1998)). The author was supported in part by DFG grant We 1066/8.

ISSN 1433-8092

MinOBDD

Instance: An OBDD H for some function f.
Problem: Compute a variable ordering π minimizing the OBDD size for f and π.

We remark that from H and each variable ordering π an OBDD for f with the variable ordering π can be computed in polynomial time (Savický and Wegener (1997), Meinel and Slobodová (1994)). Hence, it suffices to define MinOBDD as the problem to compute an optimal variable ordering instead of a minimum size OBDD.

MinOBDD is the problem that occurs when applying dynamic variable ordering techniques introduced by Rudell (1993): When performing computations on OBDDs, e.g. the computation of an OBDD for a function represented by a circuit, the set of functions that are represented changes during this computation. Hence, also the variable orderings admitting small size OBDDs for the represented functions may change. Therefore, it is reasonable to change the variable ordering during the computation. Bryant (1995) reports that this reordering slows down the computation, but that it is sometimes the only possibility to complete computations without exceeding the available memory.

The argument for the hardness of the first version of the variable ordering problem does not apply to MinOBDD since the function f is already given as an OBDD, for which the satisfiability test can be done in polynomial time. The first step for proving the hardness of MinOBDD was done by Tani, Hamaguchi and Yajima (1993). They proved that the decision variant of the problem MinSBDD is NP-complete. SBDDs (shared binary decision diagrams) are the generalization of OBDDs for the representation of more than one function and MinSBDD is the problem to compute an optimal variable ordering for a set of functions given by an SBDD. Bollig and Wegener (1996) proved that also the decision variant of MinOBDD is NP-complete.

Algorithms for solving the problem MinOBDD exactly are presented in the papers of Ishiura, Sawada and Yajima (1991) and Jeong, Kim and Somenzi (1993). As expected by the NP-completeness results these algorithms have an exponential worst-case run time. Both algorithms are improvements of the algorithm of Friedman and Supowit (1990) for computing minimum size OBDDs. This algorithm obviously has exponential run time since it works on truth tables.

The NP-completeness results justify, also from a theoretical point of view, to apply algorithms that do not necessarily compute optimal solutions. The known heuristics for MinOBDD are based on local search and simulated annealing approaches (see, e.g., Ishiura, Sawada and Yajima (1991), Rudell (1993), Bollig, Löbbing and Wegener (1995, 1996) and Panda and Somenzi (1995)).

Nevertheless, the NP-completeness does not exclude the existence of good approximation algorithms like polynomial time approximation schemes for the variable ordering problem (even if $P \neq NP$). The NP-completeness results for MinOBDD and MinSBDD are proved by reductions from the problem Optimal Linear Arrangement. The reductions seem not to be approximation preserving and we do not know of any nonapproximability result for Optimal Linear Arrangement. The only indication that good approximation algorithms for the variable ordering problem do not exist is the unsuccessful search for such algorithms. The known heuristics do not provide any guarantee for the quality of their results. Usually the heuristics are only tested on some set of benchmark circuits.

In this paper we characterize the complexity of the variable ordering problem more precisely by proving the following theorem.

Theorem 1 If there is some polynomial time approximation algorithm for MinOBDD with the performance ratio c, where c is some constant, then $P = NP$.

In common OBDD packages (see e.g. Brace, Rudell and Bryant (1990)) OBDDs with output inverters are used in order to obtain a smaller representation. It is an easy corollary from Theorem 1 that also for the variable ordering problem for OBDDs with output inverters there is no polynomial time approximation algorithm with a constant performance ratio unless $P = NP$. This follows easily together with the fact that the size of OBDDs may be only halved by introducing output inverters. Since SBDDs are a generalization of OBDDs these results hold for SBDDs as well. We conclude that it does not make sense to search for approximation algorithms with a constant performance ratio for MinOBDD. Our results justify to apply heuristic algorithms for the variable ordering problem.

The paper is organized as follows. In Section 2 we repeat some definitions ands facts concerning OBDDs and approximation algorithms. In Sections 3–5 we prove that the existence of a polynomial time approximation scheme for MinOBDD (and also for a restricted version of MinOBDD which we call MinOBDD*) implies $P = NP$. An overview over this proof is given in Section 3. Finally, we show how to construct a polynomial time approximation scheme for MinOBDD* from a polynomial time approximation algorithm for MinOBDD.

2 Preliminaries

2.1 OBDDs and SBDDs

In this section we shortly repeat some definitions and facts concerning OBDDs and SBDDs. For a more detailed introduction into OBDDs see, e.g., Bryant (1992) or Wegener (1994).

An OBDD H representing some Boolean function $f(x_1, \ldots, x_n)$ is a directed acyclic graph, in which we distinguish sinks and non-sink nodes, also called interior nodes. Sinks are labeled by some Boolean constant 0 or 1. Each interior node v is labeled by some variable x_i and has two outgoing edges, one labeled by 0 and the other one labeled by 1. We say that x_i is tested at v. The ordering condition of OBDDs requires the variables to be tested on each path in the OBDD at most once and according to a prescribed ordering.

With each node v of an OBDD we associate a function f_v, which can be computed in the following way. Let $(a_1, \ldots, a_n)$ be some assignment of the variables. We start the computation at v. If v is labeled by x_i, then we follow that edge leaving v that is labeled by a_i. This is iterated until a sink is reached. The value $f_v(a_1, \ldots, a_n)$ is equal to the label of this sink.

Each OBDD has exactly one source node s and the function represented by the OBDD is the function associated with s. SBDDs are the straightforward generalization of OBDDs for the representation of an arbitrary number of functions (Minato, Ishiura and Yajima (1990)). An SBDD for the functions $f_1, \ldots, f_l$ has l distinguished nodes $s_1, \ldots, s_l$ and f_i is equal to the function associated with s_i.

The size $|H|$ of an OBDD H or SBDD H, resp., is the number of its interior nodes. For the computation of the minimum size of an OBDD for some function f and some fixed variable ordering or the minimum size of an SBDD for some functions $f_1, \ldots, f_l$ and some fixed variable ordering we shall apply the following lemma. This lemma was proved by Sieling and Wegener (1993) for the case of OBDDs. The generalization to SBDDs is straightforward.

Lemma 2 A minimum size SBDD for the functions $f_1, \ldots, f_l$ and the variable ordering $x_1, \ldots, x_n$ contains exactly $|S_i|$ interior nodes labeled by x_i, where

$$S_i = \{f_{j|x_1=c_1,\ldots,x_{i-1}=c_{i-1}} \mid j \in \{1,\ldots,l\}, c_1,\ldots,c_{i-1} \in \{0,1\},$$
$$f_{j|x_1=c_1,\ldots,x_{i-1}=c_{i-1}} \text{ essentially depends on } x_i\}.$$

Furthermore, exactly the functions in S_i are the functions associated with the nodes labeled by x_i.

It is well-known that the minimum size SBDD for functions $f_1, \ldots, f_l$ and some fixed variable ordering π can be obtained from each SBDD for these functions and this variable ordering by applying two reduction rules bottom-up. By the deletion rule each node whose successors are equal can be eliminated. By the merging rule nodes v and w with the same label, the same 0-successor and the same 1-successor can be merged. Altogether, the effect of the reduction rules is that nodes associated with the same function are replaced by a single node. Hence, nodes cannot be merged if they are associated with different functions. The SBDD resulting from the application of the reduction rules is called the reduced SBDD for $f_1, \ldots, f_l$ and π. It is well-known that the reduced SBDD for $f_1, \ldots, f_l$ and π is unique up to isomorphism (Bryant (1986)). Throughout this paper we always assume that OBDDs and SBDDs are reduced.

It is easy to obtain an OBDD for some subfunction $f_{|x_i=c}$ from an OBDD for f. It suffices for each node labeled by x_i to redirect all incoming edges to the c-successor of this node. If the source is labeled by x_i, then the c-successor of the source is defined as the new source. Afterwards the OBDD is reduced. In particular, the OBDD for each subfunction of f and a fixed variable ordering is not larger than the OBDD for f and the same variable ordering.

2.2 The OBDD Size for Partially Symmetric Functions

Lemma 2 only shows how to compute the OBDD size for a fixed variable ordering. Also techniques for proving exponential lower bounds on the OBDD size (for all variable orderings) are well-known (see, e.g., Bryant (1991) or Krause (1991)). In our proof we have a different problem. We determine for a given function the minimum OBDD size for this function and variable orderings leading to this minimum size. Only few such results are known. In order to prove such results we consider properties of a special class of functions, namely partially symmetric functions.

A function f over some set X of variables is called partially symmetric with respect to the partition $X_1, \ldots, X_l$ of X if the variables in each set X_i can be permuted arbitrarily without changing the function. Then also the OBDD size does not change when permuting the variables in each set X_i. The sets $X_1, \ldots, X_l$ are called symmetry sets. (Totally) symmetric functions are the special case of functions with only one symmetry set. Sieling (1996) describes a method for the exact computation of the OBDD size for partially symmetric functions.

We shortly repeat this method only for the case of partially symmetric functions with two symmetry sets X_1 and X_2. Such a function f can be represented by its value matrix W_f. This is an $(|X_1|+1) \times (|X_2|+1)$-matrix where the rows and columns are numbered beginning with 0. The entry (i,j) is equal to the value that f takes on all inputs with i variables of X_1 equal to one and j variables of X_2 equal to one.

Subfunctions of f correspond to submatrices of W_f, which consist of contiguous entries of W_f. Such submatrices are called blocks. If we obtain a subfunction g of f by replacing k variables of X_1 and l

variables of X_2 by constants, then the value matrix W_g of g is a block of W_f of size $(|X_1| + 1 - k) \times (|X_2| + 1 - l)$. Each block of this size corresponds to such a subfunction and vice versa. It is easy to see whether a subfunction described by a block essentially depends on some variable x_i. If $x_i \in X_1$, then the subfunction essentially depends on x_i iff the block contains at least two different rows. If $x_i \in X_2$, then the subfunction essentially depends on x_i iff the block contains at least two different columns.

We apply these facts in order to determine the number of x_i-nodes in a reduced OBDD for some partially symmetric function f with symmetry sets X_1 and X_2. Let the variable ordering be $x_1, \ldots, x_n$. Let the value matrix W_f of f be given. Let $T_1(i,j)$ denote the number of different blocks of size $i \times j$ with at least two different rows and let $T_2(i,j)$ denote the number of different blocks of size $i \times j$ with at least two different columns. Let $k = |X_1 \cap \{x_1, \ldots, x_{i-1}\}|$ be the number of X_1-variables arranged before x_i and $l = |X_2 \cap \{x_1, \ldots, x_{i-1}\}|$ be the number of X_2-variables arranged before x_i. Let $x_i \in X_j$, $j \in \{1,2\}$. By Lemma 2 the number of x_i-nodes of a reduced OBDD for f is $T_j(|X_1| + 1 - k, |X_2| + 1 - l)$.

In order to obtain the total size of a reduced OBDD for f we have to sum up certain values $T_j(\cdot,\cdot)$. We determine these values by means of a grid graph. The node set of the grid graph is $V = \{(i,j) \mid 1 \le i \le |X_1|+1, 1 \le j \le |X_2|+1\}$. The edge set is the union of $E_1 = \{((i,j),(i-1,j)) \mid 2 \le i \le |X_1| + 1, 1 \le j \le |X_2| + 1\}$ and $E_2 = \{((i,j),(i,j-1)) \mid 1 \le i \le |X_1| + 1, 2 \le j \le |X_2| + 1\}$. Each edge $((i,j),(i-1,j)) \in E_1$ is labeled by $T_1(i,j)$ and each edge $((i,j),(i,j-1)) \in E_2$ is labeled by $T_2(i,j)$. Now let some variable ordering π for f be given. We start the computation of the OBDD size for f and π at the source $v = (|X_1|+1, |X_2|+1)$ of the grid graph. If the first variable x_i of the variable ordering is contained in X_1, we follow the E_1-edge leaving v, otherwise we follow the E_2-edge leaving v, and we reach some node v'. The label of the edge (v, v') is equal to the number x_i-nodes. This process is iterated for the second variable of the variable ordering starting at v' and so on. After processing all variables we reach the sink $(1,1)$ of the grid graph. We obtain a path from the source to the sink of the grid graph. The length of this path (with respect to the edge labels) is equal to the OBDD size for f and π. For each variable ordering there is such a path and vice versa. We may obtain an optimal variable ordering by computing a shortest path in the grid graph. We can prove for a given variable ordering that it is optimal by proving that it corresponds to some shortest path from the source to the sink of the grid graph.

2.3 The Nonapproximability of MaxCut and L-Reductions

For the definitions of notions concerning approximation algorithms we follow Garey and Johnson (1979). Let Π be some optimization problem, let D_Π be the set of instances of Π and let A be some algorithm computing legal solutions of Π. For $I \in D_\Pi$ let $A(I)$ be the value of the output of A on instance I and let $OPT(I)$ be the value of an optimal solution for I. The performance ratio of A is defined as $\sup_{I \in D_\Pi}\{A(I)/OPT(I)\}$ if Π is a minimization problem, or $\sup_{I \in D_\Pi}\{OPT(I)/A(I)\}$ if Π is a maximization problem. Hence, the performance ratio is always at least 1. A polynomial time approximation algorithm is a polynomial time algorithm whose performance ratio is bounded by some constant. A polynomial time approximation scheme is a polynomial time algorithm that gets an extra input ε. For each $\varepsilon > 0$ a polynomial time approximation scheme achieves a performance ratio of at most $1 + \varepsilon$.

We prove the nonexistence of polynomial time approximation schemes for MinOBDD under the assumption $P \neq NP$ by an approximation preserving reduction from MaxCut.

MaxCut

Instance: An undirected graph $G = (V, E)$.
Problem: Compute a partition (V_1, V_2) of V maximizing the number of edges in E with one endpoint in V_1 and the other one in V_2. (The number of such edges is called the size of the cut.)

The following nonapproximability result for MaxCut is due to Håstad (1997).

Theorem 3 For each $\gamma > 0$ the existence of a polynomial time approximation algorithm for MaxCut with a performance ratio of at most $1 + {}^1\!/_{16} - \gamma$ implies $P = NP$.

In Sections 3–5 we present an L-reduction from MaxCut to MinOBDD. L-reductions were introduced by Papadimitriou and Yannakakis (1991). Let A and B be optimization problems. Then A reduces to B ($A \leq_L B$) if there are functions φ and ψ that are computable in polynomial time and constants η and β so that for each instance I of A the following holds:

1. $I' = \varphi(I)$ is an instance for B and $OPT(I) \geq \eta OPT(I')$.
2. If s is a solution of I' with cost c', then $\psi(I, s)$ is a solution of I with cost c such that $|c - OPT(I)| \leq \beta |c' - OPT(I')|$.

Let a polynomial time approximation scheme P for some problem B be given and let $A \leq_L B$. It is well-known that a polynomial time approximation scheme for A can be constructed in the following way. For an instance I of the problem A we compute $\varphi(I)$, apply P on $\varphi(I)$ in order to obtain a solution s for the problem B, and compute the solution $\psi(I, s)$ for the problem A. It is easy to compute the performance ratio of the resulting algorithm and to verify that we obtain a polynomial time approximation scheme for A in this way.

3 The Nonexistence of Polynomial Time Approximation Schemes for MinOBDD — An Overview over the Proof

The main result of Sections 3–5 is given by the following theorem.

Theorem 4 The existence of a polynomial time approximation scheme for MinOBDD implies $P = NP$.

We prove Theorem 4 by giving an L-reduction MaxCut$\leq_L$MinOBDD. From the lower bound on the performance ratio of polynomial time approximation algorithms for MaxCut due to Håstad (1997) stated in Theorem 3 and the parameters of the L-reduction we can compute the lower bound $1 + {}^1\!/_{14943} - \gamma$ (for each $\gamma > 0$) on the performance ratio of polynomial time approximation algorithms for MinOBDD (under the assumption $P \neq NP$). We omit this calculation because we prove a much stronger result in Section 6. For the proof in Section 6 we apply that Theorem 4 also holds for a restricted version of MinOBDD, which we call MinOBDD*. The problem MinOBDD* is a promise problem. The definition of such a problem contains an extra condition on the input which is called the promise. A polynomial time approximation algorithm for MinOBDD* has to achieve its performance ratio only for instances fulfilling the promise. The algorithm does not have to check whether an instance fulfills the promise. For instances not fulfilling the promise the algorithm may behave arbitrarily. MinOBDD* is defined in the following way.

MinOBDD*

Instance: An OBDD H with N nodes for some function f.
Promise: The minimum OBDD size for f is at least $N/2$.
Problem: Compute a variable ordering π minimizing the OBDD size for f and π.

Theorem 5 The existence of a polynomial time approximation scheme for MinOBDD* implies $P = NP$.

In order to prove Theorem 5 we may use the same proof as for Theorem 4 since for all instances G for MaxCut the OBDD $\varphi(G)$ fulfills the promise. Hence, in the same way as outlined at the end of Section 2 a polynomial time approximation scheme for MaxCut can be constructed from a polynomial time approximation scheme for MinOBDD*.

Let $G = (V, E)$ be an instance for MaxCut. W.l.o.g. we assume that all nodes in G have a degree of at least two, that $|E| \geq 250$ and that G is not bipartite. (For bipartite graphs MaxCut is trivial.)

Let $n = |V|$ and let $m = |E|$. Sometimes we identify V with the set $\{1, \ldots, n\}$ and E with the set $\{1, \ldots, m\}$. Let $E(v)$ be the set of edges incident to v and let $d(v) = |E(v)|$ be the degree of v.

By the mapping φ we obtain from G an OBDD $H = \varphi(G)$, which computes some function $\mathcal{F}$. We divide the description of H into two steps. In the first step (given in Section 4) we only consider some subfunctions of $\mathcal{F}$ and represent these subfunctions by an SBDD. The variable ordering of this SBDD encodes a cut of G in such a way that the SBDD size corresponds to the size of the cut. In other words, there is a mapping ψ that describes how to obtain a cut from the variable ordering. However, there are some variable orderings not encoding any cut of G. By enforcing components we will make sure that in such cases the SBDD size is so large that the second condition of the definition of L-reductions is fulfilled by a cut of size 0.

We shall introduce the following functions in the first step:

- Functions $f_1, \ldots, f_m$. These functions connect the given graph G and the OBDD H in the following way: If the i-th edge of G is contained in the cut corresponding to the variable ordering of H, then the function f_i has the OBDD size 6. If the i-th edge is not contained in the cut, then the OBDD size is 7. The functions $f_1, \ldots, f_m$ are combined into a single function f.
- Functions $g, h_1, \ldots, h_5$. The functions $f_1, \ldots, f_m$ have the desired property only under further assumptions on the variable ordering. These assumptions include that the variable ordering encodes a cut of G. The functions $g, h_1, \ldots, h_5$ enforce properties of the variable ordering. This means that the size of an SBDD for $f_1, \ldots, f_m, g, h_1, \ldots, h_5$ for variable orderings that do not encode any cut for G is so large that the second condition of the definition of the L-reductions is fulfilled for a cut of size 0. Finally, the functions $h_1, \ldots, h_5$ are combined and we obtain only two functions h^* and h^{**}.
- Functions h', h''. These functions enforce certain properties of the variable ordering that are helpful when combining all these functions into a single function that is represented by an OBDD.

In the second step, which we describe in Section 5, we combine f, g, h^*, h^{**}, h' and h'' into a single function $\mathcal{F}$ so that it can be represented by an OBDD. Again there is a correspondence between the variable ordering of the OBDD and the cut as well as a correspondence between the OBDD size and the cut size.

4 Construction of an SBDD

4.1 The Functions $f_1, \ldots, f_m$

First we introduce the variables on which the functions $f_1, \ldots, f_m$ are defined. There are the variables x_e and y_e for $e \in \{1, \ldots, 2m\}$ and the variables z_e^v for $v \in V$ and $e \in E(v)$. This means that for each edge $e = \{u, v\} \in E$ there are four variables, namely x_e, y_e, z_e^u and z_e^v. The variables x_e and y_e, where $e \in \{m+1, \ldots, 2m\}$, are not associated with any edge. They are used later on in order to define enforcing components.

For all $e = \{u, v\} \in E$ we define the function $f_e : \{0,1\}^4 \to \{0,1\}$ by

$$f_e(x_e, y_e, z_e^u, z_e^v) = \begin{cases} 0 & \text{if } x_e + z_e^u + z_e^v = 0 \text{ or } x_e + z_e^u + z_e^v = 2, \\ 1 & \text{if } x_e + z_e^u + z_e^v = 1, \\ y_e & \text{if } x_e + z_e^u + z_e^v = 3. \end{cases}$$

Note that the functions f_e and $f_{e'}$ for $e \neq e'$ are defined on disjoint sets of variables. Hence, there are no mergings between OBDDs for f_e and $f_{e'}$. The function f_e is partially symmetric with respect to the sets $\{x_e, z_e^u, z_e^v\}$ and $\{y_e\}$. Hence, the OBDD size for f_e is determined only by the position of y_e among $\{x_e, z_e^u, z_e^v, y_e\}$, but not by the relative ordering of x_e, z_e^u and z_e^v. It is easy to check that the OBDD size is 8, if y_e is the first variable among $\{x_e, z_e^u, z_e^v, y_e\}$, that the OBDD size is 7, if y_e is the second or fourth variable, and that the OBDD size is 6, if y_e is the third variable.

Now we explain the relationship between a cut (V_1, V_2) of G and the ordering of the variables. If the variable z_e^u is arranged before y_e in the variable ordering, then the node u is contained in V_1. If z_e^u is arranged after y_e, then the node u is contained in V_2. By introducing the functions h_1, h_2 and h_3 we shall make sure that x_e is always arranged before y_e. Then y_e cannot be the first variable among $\{x_e, z_e^u, z_e^v, y_e\}$ and, hence, it does not occur that the OBDD for f_e has size 8. Now consider the case that $e = \{u, v\}$ is contained in the cut. This means that $u \in V_1$ and $v \in V_2$ or vice versa. In both cases y_e is at the third position among $\{x_e, z_e^u, z_e^v, y_e\}$ and, hence, the OBDD size is 6. If $e = \{u, v\}$ is not contained in the cut, then either $u \in V_1$ and $v \in V_1$ or $u \in V_2$ and $v \in V_2$. In the former case y_e is at the fourth position among $\{x_e, z_e^u, z_e^v, y_e\}$, in the latter case y_e is at the second position. Hence, the OBDD size for f_e is 7.

This construction only works if the classification of u belonging to V_1 or V_2 is consistent for all variables z_e^u, $e \in E(u)$. If there are edges $e = \{u, v\}$ and $e' = \{u, w\}$, it must not happen that z_e^u is arranged before y_e in the variable ordering and $z_{e'}^u$ after $y_{e'}$. Altogether, we shall represent functions in the SBDD enforcing the following two properties of the variable ordering.

(P1) All x-variables are arranged before all y-variables.

(P2) For each node $u \in V$ the following holds. Either for all $e \in E(u)$ the variable z_e^u is arranged before y_e or for all $e \in E(u)$ the variable z_e^u is arranged after y_e.

These properties are enforced by the functions g and $h_1, \ldots, h_5$, which we introduce in the following sections.

Finally, we combine all the functions $f_1, \ldots, f_m$ into a single function f. For that purpose we introduce $m-1$ new variables $\alpha_1, \ldots, \alpha_{m-1}$ and define

$$f = \bar{\alpha}_1 f_1 \vee (\alpha_1 \bar{\alpha}_2) f_2 \vee (\alpha_1 \alpha_2 \bar{\alpha}_3) f_3 \vee \ldots \vee (\alpha_1 \ldots \alpha_{m-2} \bar{\alpha}_{m-1}) f_{m-1} \vee (\alpha_1 \ldots \alpha_{m-1}) f_m.$$

This construction was already used by Bollig and Wegener (1996). We prove that it is optimal to arrange the α-variables before the x-, y- and z-variables. Then the top of an OBDD for f consists of a switch of $m-1$ nodes labeled by α-variables. Below this switch we have m disjoint OBDDs for the functions $f_1, \ldots, f_m$.

Lemma 6 Let A be an OBDD for f and some variable ordering π. Let π' be the variable ordering that we obtain from π by moving the variables $\alpha_1, \ldots, \alpha_{m-1}$ in this order to the beginning without changing the relative ordering of the other variables. Let A' be the OBDD for f and π'. Then the size of A' is not larger than the size of A.

Proof We start with the variable ordering π and show that the OBDD size does not increase when moving the variable α_1 to the beginning of the variable ordering. Afterwards, it can be shown in the same way that the OBDD size does not increase when moving the variable α_2 to the second position and so on. The OBDD A and the OBDD A'' that we obtain by moving α_1 to the first position are shown in Fig. 1. For the sake of simplicity let $w_1, \ldots, w_l$ denote all variables except α_1 and let $w_1, \ldots, w_l$ be the ordering of those variables in A and A''. We observe that the functions f_i essentially depend on disjoint sets of variables. This implies that the parts of the OBDD A'' that are reached for $\alpha_1 = 0$ and $\alpha_1 = 1$ do not share interior nodes.

(Insert Figure 1 here)

By Lemma 2 the parts of A and A'' below the α_1-layer of A are isomorphic and, therefore, of the same size. Now consider some variable w_i that is arranged before α_1 in A. Let p be the number of w_i-nodes of A''. Either all these nodes are reached for $\alpha_1 = 0$ or all these nodes are reached for $\alpha_1 = 1$. W.l.o.g. assume that all these nodes are reached if $\alpha_1 = 0$. We obtain the subfunctions of f computed at these nodes if we replace α_1 by 0 and $w_1, \ldots, w_{i-1}$ by constants in all possible ways. We obtain at least the same number of subfunctions essentially depending on w_i if we drop the replacement of α_1 by 0. Hence, by Lemma 2 the number of w_i-nodes in A is not smaller than the number of w_i-nodes in A''. □

Now we can describe the relation between the OBDD size for f and the size of the cut in G corresponding to the variable ordering of the OBDD. Here we still need the assumption that (P1) and (P2) hold. Later on we shall make sure by enforcing components that (P1) and (P2) hold.

Lemma 7 The graph G has a cut of size at least c iff f has an OBDD of size $8m - 1 - c$ with a variable ordering fulfilling (P1) and (P2).

4.2 The Function g

The function g will make sure that (P2) is fulfilled. For the definition of g we introduce new variables $a_1, \ldots, a_{2m}$, $d_1, \ldots, d_{2m}$ and $\gamma_1, \ldots, \gamma_{17m-2}$. First we define some components from which g is built up. For each $v \in V$ let $p_v = \bigoplus_{e \in E(v)} z_e^v$ and let $\Gamma = \bigoplus_{i=1}^{17m-2} \gamma_i$. Let

$$p^* = \bigwedge_{v \in V} p_v.$$

Furthermore, let

$$g^* = \bigvee_{i=1}^{2m} a_1 \ldots a_{i-1} \bar{a}_i y_i d_i \ldots d_{2m}.$$

Then the function g is defined by

$$g = g^* \wedge p^* \wedge \Gamma.$$

First, we informally describe how g enforces property (P2). In optimal variable orderings for p^* the z-variables are arranged blockwise, i.e. for each $v \in V$ the variables z_e^v for all $e \in E(v)$ are arranged in adjacent levels. Then an OBDD for $p^* = \bigwedge p_v$ is a concatenation of OBDDs for the functions p_v.

Now consider an OBDD for $g^* \wedge p^*$. First note that g^* and p^* are defined on disjoint sets of variables. In order to obtain an OBDD for $g^* \wedge p^*$ we may concatenate OBDDs for g^* and p^* or we may insert an OBDD for g^* in the concatenation of OBDDs for p^*. If we insert the OBDD for g^* between two blocks of z-variables, then we obtain an OBDD for $g^* \wedge p^*$ whose size is the sum of the sizes of the OBDDs for g^* and p^*. But if we arrange the OBDD for g^* in such a way that some variable z_e^v is tested before the variables that g^* depends on and some other variable $z_{e'}^v$ after the variables that g^* depends on, then we need two copies of the OBDD for g^*. In Lemma 11 we show that we obtain a similar increase of the OBDD size of g also for other variable orderings violating (P2). Here it is important that the OBDD for g^* is of large width. In order to make sure that an OBDD for g^* is of large width we require the following properties of the variable ordering.

(P3) All a-variables are arranged before all y-variables.

(P4) All y-variables are arranged before all d-variables.

If these properties are fulfilled, then an OBDD for g^* has width $2m$ (see Fig. 2). This OBDD has size $6m$ which is optimal since g^* essentially depends on $6m$ variables.

(Insert Figure 2 here)

The function Γ makes it possible to count the number of mergings between the nodes in the OBDD for g and the SBDD for the functions h' and h'', which we introduce later on. These mergings will effect the γ-variables to be arranged after the x-, y-, z-, a- and d-variables in optimal variable orderings. At the moment one may imagine the function Γ as a function p_v for a pseudo-node v with degree $17m - 2$.

In the following lemmas we give estimates on the size of the OBDDs for g and different variable orderings. Already here we take into account that the functions f and g are represented in the same SBDD so that we have to consider mergings between the representations of f and g.

Lemma 8 Consider a variable ordering where the z_e^v-variables are arranged blockwise, where the a-, y- and d-variables are arranged in the order $a_1, \ldots, a_{2m}, y_1, \ldots, y_{2m}, d_1, \ldots, d_{2m}$ between two blocks of the z_e^v-variables and where the γ-variables are the last variables. Then the OBDD size for g is $44m - n - 5$. No interior node of this OBDD can be merged with a node of an OBDD for f.

Proof It is easy to see that we need $6m$ nodes for the representation of g^*, that we need $\sum_{v \in V}(2d(v) - 1) = 4m - n$ nodes for p^* and $34m - 5$ nodes for Γ. All functions computed at these nodes essentially depend on at least one γ-variable. Hence, there are no mergings with nodes of any representation of f. □

Lemma 9 Each OBDD for $p^* \wedge \Gamma$ has at least $38m - n - 9$ interior nodes that cannot be merged with nodes from an OBDD for f. If the last variable in the variable ordering is a γ-variable, then each OBDD for $p^* \wedge \Gamma$ has at least $38m - n - 5$ interior nodes that cannot be merged with nodes from an OBDD for f.

Proof Obviously, each OBDD for $p^* \wedge \Gamma$ contains at least $34m - 5$ nodes labeled by γ-variables. Since f does not essentially depend on any γ-variable, no mergings are possible between those nodes and nodes in the representation of f.

Now we count the nodes labeled by the variables z_e^v. First we note that p^* has the following property: there is an input q for p^* with $p^*(q) = 1$ and $p^*(q') = 0$ for all inputs q' that we obtain from q by negating one z_e^v-variable. (In other words, the critical complexity of p^* is maximal. For more details on the critical complexity see Bublitz, Schürfeld, Voigt and Wegener (1986).) We call the computation paths for such inputs q critical paths. In each OBDD for p^* on each critical path all variables z_e^v are tested. In particular, all nodes on a critical path are associated with functions essentially depending on all remaining variables in the variable ordering.

Let $v \in V$ be fixed. Let $z_{e^*}^v$ be the first variable among the variables z_e^v in the variable ordering. Then in each OBDD for $p^* \wedge \Gamma$ there is at least one node labeled by $z_{e^*}^v$ lying on some critical path. If $z_{e^*}^v$ is not the first variable among z_e^v in the variable ordering, then in each OBDD for $p^* \wedge \Gamma$ there are at least two nodes labeled by $z_{e^*}^v$ lying on critical paths. Altogether, there are at least $\sum_{v \in V}(2d(v) - 1) = 4m - n$ nodes labeled by z_e^v-variables and lying on critical paths.

If a γ-variable is the last variable in the variable ordering, the functions associated with the nodes that we counted essentially depend on this γ-variable and, hence, no mergings with nodes in the representation of f are possible. This implies the second claim of the lemma.

In order to prove the first claim we consider all levels of z_e^v-nodes except the last two levels. Then the number of nodes on critical paths is at least $4m - n - 4$. The functions associated with those nodes essentially depend on all remaining z_e^v-variables, i.e. on at least three z_e^v-variables. Since $p^* \wedge \Gamma$ does not essentially depend on any α-variable, also the functions associated with those nodes do not essentially depend on any α-variable. The claim that no mergings between those nodes and nodes of each OBDD for f are possible follows from the observation that f does not have any subfunction essentially depending on at least three z_e^v-variables but not essentially depending on any α-variable. □

Lemma 10 Each SBDD for f and g has at least $49m - n - 10$ interior nodes.

Proof By Lemma 9 there are at least $38m - n - 9$ nodes in each OBDD for $p^* \wedge \Gamma$ that cannot be merged with nodes from an OBDD for f. Each OBDD for g contains at least $2m$ nodes labeled by a-variables and $2m$-nodes labeled by d-variables since g essentially depends on all a- and d-variables. For the representation of f we need at least $7m - 1$ nodes. The sum of these lower bounds is $49m - n - 10$, which implies the lemma. □

Lemma 11 Each SBDD for f and g with a variable ordering fulfilling (P3) and (P4) but not fulfilling (P2) has at least $53m - n - 14$ nodes.

Proof Since (P2) is not fulfilled, for some $v \in V$ and some $p, q \in E(v)$ it holds that z_p^v is arranged before y_p and z_q^v is arranged after y_q. We already know the following lower bounds: There are at least $7m-1$ nodes in the OBDD for f. In the OBDD for g there are at least $2m$ nodes labeled by a-variables and $2m$ nodes labeled by d-variables. By the proof of Lemma 9 there are at least $(38m - n - 9) - 4$ nodes labeled by γ-variables or z_e^v-variables except z_p^v and z_q^v. In the following we prove new lower bounds on the number of z_p^v-, z_q^v- and y-nodes under the assumption that (P3) and (P4) hold but not (P2). We prove the lower bound on the number of such nodes for the subfunction g' of g that we obtain in the following way. For all $w \in V$, $w \neq v$, we replace the variables z_e^w by constants in such a way that $p_w = 1$. We replace the variables z_e^v except z_p^v and z_q^v by 0. Then g' only depends on the a-, y- and d-variables and on z_p^v and z_q^v.

Furthermore, we assume that z_p^v is arranged before z_q^v. For the other case the same proof works after exchanging p and q.

Because of (P3) and (P4) the a-variables are arranged before the y-variables and the y-variables are arranged before the d-variables. Furthermore, we know that z_p^v is arranged before all d-variables. Otherwise, z_p^v and z_q^v are arranged after all y-variables and, hence, they do not lead to a violation of (P2). Similarly it follows that z_q^v is arranged after all a-variables.

Let $k \in \{1, \dots, 2m\}$. Let g_k'' be the subfunction of g' that we obtain by replacing a_k by 0 and all other a-variables by 1. Then

$$g_k'' = y_k d_k \dots d_{2m} (z_p^v \oplus z_q^v).$$

We distinguish the following three cases.

Case 1 y_k is arranged before z_p^v.
We know that z_q^v is arranged after z_p^v and that z_p^v is arranged before all d-variables. Hence, in each OBDD for g_k'' there is a y_k-node associated with g_k'' and a z_p^v-node associated with $g''_{k|y_k=1}$. Since g_k'' is a subfunction of g, also in each OBDD for g there is at least one node labeled by y_k and one node labeled by z_p^v.

Case 2 y_k is arranged after z_p^v and before z_q^v.
We count the number of y_k-nodes associated with $g''_{k|z_p^v=0}$ and $g''_{k|z_p^v=1}$. Obviously, there are two nodes.

Case 3 y_k is arranged after z_q^v.
We know that z_p^v is arranged before z_q^v and z_q^v is arranged after all a-variables. We count the number of z_q^v-nodes. With these nodes the functions $g''_{k|z_p^v=0}$ and $g''_{k|z_p^v=1}$ are associated. Hence, there are two z_q^v-nodes.

For different k all such subfunctions are different because they contain the conjunction $d_k \dots d_{2m}$. For each $k \in \{1, \dots, 2m\}$ there are at least two nodes labeled by z_p^v, z_q^v or y_k. Altogether, there are $4m$ such nodes.

There are no mergings between these nodes and nodes in the OBDD for f since the functions associated with these nodes essentially depend on d_{2m} which does not hold for f nor any subfunction of f. The derived lower bound $4m$ together with the lower bounds given at the beginning of the proof implies the lemma. □

4.3 The Functions $h_1, \ldots, h_5$

Our aim is to include functions in the OBDD that ensure (P1), (P3) and (P4). Let us consider e.g. (P3). A function enforcing (P3) has its optimal variable ordering if all a-variables are arranged before all y-variables. The problem is that an OBDD for this function and the variable ordering $a_1, \ldots, a_{2m-1}, y_1, a_{2m}, y_2, \ldots, y_{2m}$ is not substantially larger. (Here we have the problem that we only meet the first condition of the definition of the L-reductions if the size of the constructed OBDD is linear in m. Hence, also the function enforcing (P3) must have an OBDD of linear size.) This is the reason why we introduce new variables $b_1, \ldots, b_{2m}$ in order to increase the distance between the a-variables and the y-variables. Similarly we introduce variables $c_1, \ldots, c_{2m}$ in order to increase the distance between the y-variables and the d-variables. The function $h : \{0,1\}^{4m} \to \{0,1\}$ is defined by

$$h(p_1, \ldots, p_{2m}, q_1, \ldots, q_{2m}) = \begin{cases} 1 & \text{if } (p_1 + \ldots + p_{2m}) \equiv 0 \bmod 5, \\ q_1 \oplus \ldots \oplus q_{2m} & \text{if } (p_1 + \ldots + p_{2m}) \equiv 1 \bmod 5, \\ 0 & \text{if } (p_1 + \ldots + p_{2m}) \equiv 2 \bmod 5, \\ q_1 \oplus \ldots \oplus q_{2m} \oplus 1 & \text{if } (p_1 + \ldots + p_{2m}) \equiv 3 \bmod 5, \\ 0 & \text{if } (p_1 + \ldots + p_{2m}) \equiv 4 \bmod 5. \end{cases}$$

We introduce the following five functions $h_1, \ldots, h_5$ in order to ensure (P1), (P3) and (P4) to be fulfilled.

$$\begin{aligned} h_1 &= h(x_1, \ldots, x_{2m}, a_1, \ldots, a_{2m}), \\ h_2 &= h(a_1, \ldots, a_{2m}, b_1, \ldots, b_{2m}), \\ h_3 &= h(b_1, \ldots, b_{2m}, y_1, \ldots, y_{2m}), \\ h_4 &= h(y_1, \ldots, y_{2m}, c_1, \ldots, c_{2m}), \\ h_5 &= h(c_1, \ldots, c_{2m}, d_1, \ldots, d_{2m}). \end{aligned}$$

Note that $h(p_1, \ldots, p_{2m}, q_1, \ldots, q_{2m})$ is partially symmetric with respect to $\{p_1, \ldots, p_{2m}\}$ and $\{q_1, \ldots, q_{2m}\}$. This implies that the functions $h_1, \ldots, h_5$ do not influence the relative ordering of the x-variables (or a-, b-, y-, c- and d-variables, resp.). The value matrix of h is periodic with the periods 5 in the rows and 2 in the columns. Its upper left part is

$$\begin{matrix} 1 & 1 \\ 0 & 1 \\ 0 & 0 \\ 1 & 0 \\ 0 & 0 \end{matrix}$$

In the following lemmas we state the properties that we use to prove that the h-functions enforce (P1), (P3) and (P4).

Lemma 12 The OBDD size for $h(p_1, \ldots, p_{2m}, q_1, \ldots, q_{2m})$ and the variable orderings where all p-variables are arranged before all q-variables is $14m - 10$. Only such variable orderings are optimal.

Lemma 13 The OBDD size for $h(p_1, \ldots, p_{2m}, q_1, \ldots, q_{2m})$ and each variable ordering where after the first q-variable at least m of the p-variables are arranged is at least $19m - 10$.

Lemma 14 The OBDD size for $h(p_1, \ldots, p_{2m}, q_1, \ldots, q_{2m})$ and each variable ordering where before the last p-variable at least m of the q-variables are arranged is at least $18m - 10$.

If, e.g., (P3) is not fulfilled, then there is some a-variable that is tested after at least m of the b-variables or there is some y-variable that is tested before at least m of the b-variables. In the former case the OBDD size for h_2 is at least $18m - 10$ rather than $14m - 10$, in the latter case the OBDD size for h_3 is at least $19m - 10$ rather than $14m - 10$. In this way it is made sure that (P1), (P3) and (P4) are fulfilled. Here we can see why we introduced $2m$ rather than only m of the x- and y-variables. The aim is to make the difference between the OBDD size for h and the variable orderings described in Lemma 13 and Lemma 14 and the OBDD size for h and an optimal variable ordering larger.

Before we prove the lemmas we describe how to replace the five functions $h_1, \ldots, h_5$ by only two functions h^* and h^{**}. We note that h_1, h_3 and h_5 are defined on disjoint sets of variables. Hence, we may introduce two new variables α_m and α_{m+1} and define $h^* = \bar{\alpha}_m h_1 \vee \alpha_m \bar{\alpha}_{m+1} h_3 \vee \alpha_m \alpha_{m+1} h_5$. Similarly we can combine h_2 and h_4 by introducing a new variable α_{m+2} and defining $h^{**} = \bar{\alpha}_{m+2} h_2 \vee \alpha_{m+2} h_4$. In the same way as in Lemma 6 it can be proved that for each variable ordering the OBDD size does not increase when moving the α-variables before the other variables.

Proof of Lemma 12, Lemma 13 and Lemma 14 We apply the technique of Sieling (1996) described in Section 2.2 for the computation of the OBDD size of h. First we compute the grid graph for h. Let $k = 2m$. Then the grid graph has the node set $\{(i, j) | 1 \leq i, j \leq k + 1\}$. The labels $T_l(i, j)$ of the edges of the grid graph can be obtained by counting the number of different $i \times j$-blocks of W_h with at least two different rows, if $l = 1$, or with at least two different columns, if $l = 2$. The result of this tedious but nevertheless simple task is shown in Fig. 3.

(Insert Figure 3 here)

The next step is to compute for each node (i, j) the length $L(i, j)$ of a shortest path from (i, j) to the sink $(1, 1)$ of the grid graph. This can be done by running through the grid graph in some reversed topological order. For the sink we have $L(1, 1) = 0$. Now let (i, j) be some non-sink node. Then (i, j) may have one or two successors. If (i, j) has one successor, then $L(i, j)$ is the sum of the label of the edge from (i, j) to its successor and the value $L(\cdot, \cdot)$ of the successor. In this case we color the edge from (i, j) to its successor green. If (i, j) has two successors $(i - 1, j)$ and $(i, j - 1)$, we set $L(i, j) = \min\{L(i - 1, j) + T_1(i, j), L(i, j - 1) + T_2(i, j)\}$. If $L(i, j) = L(i - 1, j) + T_1(i, j)$, we color the edge from (i, j) to $(i - 1, j)$ green and otherwise red. The color of the other edge leaving (i, j) is determined similarly. The result is that for each node (i, j) we know the length of a shortest path to the sink. We may obtain this path by starting at (i, j) and by running always through green edges.

We do not give the intermediate results of this tedious computation. The results that we need are listed in the tables below. Furthermore, $L(k + 1, k + 1) = 7k - 10 = 14m - 10$ and the only path from the source to the sink only consisting of green edges is the path $(k + 1, k + 1), (k, k + 1), (k -$

$1, k+1), \ldots, (1, k+1), (1, k), \ldots, (1, 1)$, i.e. the path corresponding to all variable orderings where all p-variables are arranged before all q-variables. This implies Lemma 12.

Similar to the computation of $L(i, j)$ we may compute for each node (i, j) the length $M(i, j)$ of a shortest path from the source $(k+1, k+1)$ to (i, j). Now we start at the source and run through the graph in some topological order. Unfortunately, this approach leads to a more complicated case distinction, which we can avoid because we do not need $M(i, j)$ for all nodes (i, j). We simplify the computation of $M(i, j)$ in the following way (see also Fig. 4). First we compute $M(i, j)$ only for those nodes (i, j) where $i \geq k-3$ (upper part of the grid graph) or $j \geq k$ (left part of the grid graph). For the nodes $(2, j)$, where $j \leq k-1$ we compute the length $M^*(2, j)$ of a shortest path from the source through $(2, k)$ to $(2, j)$. Then we prove that this is the length of a shortest path from the source, i.e. that $M(2, j) = M^*(2, j)$.

(Insert Figure 4 here)

It is easy to check in Fig. 3 that for nodes (i', j'), where $i' \leq k-3$, it holds that the length of the path from (i', j') to $(i'-1, j'-1)$ via $(i'-1, j')$ is not larger than the length of the path from (i', j') to $(i'-1, j'-1)$ via $(i', j'-1)$. This implies the following. If a shortest path from the source to $(2, j)$ does not go through $(2, k)$, then there is no shortest path going through any of the nodes $(k-3, k), \ldots, (3, k)$. Then the shortest path goes through at least one of the nodes $(k-3, k-1), \ldots, (k-3, j)$. The path leaves one these nodes through the edge $((k-3, j^*), (k-4, j^*))$, where $j \leq j^* \leq k-1$. By the same argument as above the cost does not increase if we instead run from $(k-3, j^*)$ through the edges directed downwards to the node $(2, j^*)$. The cost of this path from the source to $(2, j)$ is sum of the path length from the source to $(k-3, j^*)$, which is $M(k-3, j^*) = 2k - 2j^* + 21$, the path length from $(k-3, j^*)$ to $(2, j^*)$, which is $10(k-5)$, and the cost from $(2, j^*)$ to $(2, j)$, which is $6(j^* - j)$. This sum is $12k + 4j^* - 6j - 29$. This is minimal for $j^* = j$. Hence, the cost is $12k - 2j - 29$. Since $M^*(2, j) = 11k - 6j - 11$, it follows (together with $k \geq 500$ because of $m \geq 250$) that the path through $(2, k)$ is cheaper. In this way we obtain $M(2, j) = M^*(2, j)$.

If we know the values $L(i, j)$ and $M(i, j)$, we can compute the length of a shortest path from the source $(k+1, k+1)$ to the sink $(1, 1)$ through the node (i, j) as $L(i, j) + M(i, j)$. Now consider the variable orderings described in Lemma 13. The paths corresponding to such variable orderings run through at least one of the nodes $(k+1, k), (k, k), \ldots, (k/2 + 1, k)$. In the following table we give the L- and M-values for these nodes.

(i, j)	$L(i, j)$	$M(i, j)$	$L(i, j) + M(i, j)$
$(k+1, k)$	$12k - 34$	1	$12k - 33$
(k, k)	$12k - 27$	3	$12k - 24$
$(k-1, k)$	$12k - 31$	6	$12k - 25$
$(k-2, k)$	$12k - 37$	10	$12k - 27$
$(k-3, k)$	$12k - 45$	15	$12k - 30$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
(i, k), where $i \leq k-3$	$2k + 10i - 15$	$5k - 5i$	$7k + 5i - 15$

The term $7k + 5i - 15$ takes its minimum value for $i = k/2 + 1$. The minimum value is $19/2 \cdot k - 10$. This implies Lemma 13.

Now consider the variable orderings described in Lemma 14. The paths corresponding to such variable orderings run through at least one of the nodes $(2,1),(2,2),\ldots,(2,{}^{k}\!/_{2}+1)$. Again we give the L- and M-values for these nodes.

(i,j)	$L(i,j)$	$M(i,j)$	$L(i,j)+M(i,j)$
$(2,1)$	2	$11k-17$	$11k-15$
$(2,2)$	8	$11k-23$	$11k-15$
$(2,3)$	11	$11k-29$	$11k-18$
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$(2,j)$, where $j\leq 3$	$2j+5$	$11k-6j-11$	$11k-4j-6$

The term $11k-4j-6$ takes its minimum value for $j={}^{k}\!/_{2}+1$. The minimum value is $9k-10$. This implies Lemma 14. □

4.4 The Functions h' and h''

Up to now we have the following $32m$ variables: x_i, y_i, a_i, b_i, c_i and d_i for $i\in\{1,\ldots,2m\}$, the variables $\alpha_1,\ldots,\alpha_{m+2},\gamma_1,\ldots,\gamma_{17m-2}$ and $2m$ variables z_e^v. In order to simplify the presentation in Section 5 we rename all these variables to $s_1,\ldots,s_{32m}$. The exact correspondence between the old names and the new names is not important. We define

$$h'=s_1\oplus\ldots\oplus s_{32m}\qquad\text{and}\qquad h''=\overline{h'}.$$

Since h' and h'' are parity functions, the following lemma holds.

Lemma 15 For all variable orderings a minimum size SBDD for h' and h'' consists of exactly $64m$ interior nodes.

If $\gamma_1,\ldots,\gamma_{17m-2}$ are the last variables in the variable ordering, exactly $34m-5$ nodes of the SBDD for h' and h'' can be merged with nodes of the OBDD for g.

4.5 The Relationship Between the SBDD Size for f, g, h^*, h^{**}, h' and h'' and the Cut Size for G

First we show how to obtain a variable ordering for an SBDD if a cut is given. In Lemma 17 we show that it is possible in polynomial time to construct a cut from an SBDD. This is used later on when computing the function ψ. For both constructions the relationship between the SBDD size and the cut size is the same.

Lemma 16 If $G=(V,E)$ has a cut of size c, then there is an SBDD for the functions f, g, h^*, h^{**}, h' and h'' with at most $152m-n-54-c$ nodes.

Proof Let (V_1, V_2) be the partition corresponding to the cut of size c. Since G is not bipartite, there is an edge e^* not contained in the cut. W.l.o.g. we may assume that both endpoints of e^* are contained in V_1. (Otherwise we exchange V_1 and V_2.)

We choose the following variable ordering for the SBDD H.

1. $\alpha_1, \dots, \alpha_{m+2}$,
2. all variables z_e^v for all $e \in E(v)$ blockwise for all $v \in V_1$,
3. all x_e-variables in an arbitrary order where x_{e^*} is the last variable,
4. $a_1, \dots, a_{2m}, b_1, \dots, b_{2m}$,
5. all y-variables in an arbitrary order where y_{e^*} is the last variable,
6. $c_1, \dots, c_{2m}, d_1, \dots, d_{2m}$,
7. all variables z_e^v for all $e \in E(v)$ blockwise for all $v \in V_2$,
8. $\gamma_1, \dots, \gamma_{17m-2}$.

This variable ordering has the properties (P1)–(P4). Then $8m - 1 - c$ nodes suffice to represent the function f and $44m - n - 5$ nodes suffice for g. By Lemma 12 there is an SBDD with $5(14m - 10) + 3$ nodes for h^* and h^{**}. Here the term 3 is added for the nodes labeled by α_m, α_{m+1} and α_{m+2}. Finally, an SBDD for h' and h'' contains $64m$ nodes. The sum of these upper bounds is $186m - n - 53 - c$. In the following we show that we save $34m + 1$ nodes by mergings. This implies the lemma.

Since the γ-variables are the last variables in the variable ordering, there are $34m - 5$ nodes in the OBDD for g and the SBDD for h' and h'' that can be merged.

Now we consider the OBDDs for h_1 and h_2. The only variables that both functions depend on are the a-variables. Hence, only a-nodes can be merged. Let us consider the a_{2m}-nodes. In the OBDD for h_1 the subfunctions computed at a_{2m}-nodes can be obtained by replacing all x-variables and all a-variables except a_{2m} by constants. Hence, we have to consider 1×2-blocks of W_{h_1}. There are only two such blocks computing a function that essentially depends on a_{2m}, namely $[0\ 1]$ and $[1\ 0]$, which correspond to the subfunctions a_{2m} and $\overline{a_{2m}}$. On the a_{2m}-level of h_2 those subfunctions are computed that can be obtained from h_2 by replacing $a_1, \dots, a_{2m-1}$ by constants. Hence, we have to look for $2 \times (2m+1)$-blocks of W_{h_2}. There are five such blocks, namely

$$\begin{bmatrix} 1 & 1 & 1 & 1 & \dots \\ 0 & 1 & 0 & 1 & \dots \end{bmatrix}, \quad \begin{bmatrix} 0 & 1 & 0 & 1 & \dots \\ 0 & 0 & 0 & 0 & \dots \end{bmatrix}, \quad \begin{bmatrix} 0 & 0 & 0 & 0 & \dots \\ 1 & 0 & 1 & 0 & \dots \end{bmatrix},$$

$$\begin{bmatrix} 1 & 0 & 1 & 0 & \dots \\ 0 & 0 & 0 & 0 & \dots \end{bmatrix}, \quad \begin{bmatrix} 0 & 0 & 0 & 0 & \dots \\ 1 & 1 & 1 & 1 & \dots \end{bmatrix}.$$

The last one represents the function a_{2m}. Hence, there is a merging of one node labeled by a_{2m}. In the same way it can be shown that there are no mergings between a_{2m-1}-nodes of h_1 and h_2. By the same arguments there is one pair of b_{2m}-nodes of h_2 and h_3, one pair of y_{e^*}-nodes of h_3 and h_4 and one pair of c_{2m}-nodes of h_4 and h_5 that may be merged.

The x_e-variables are the only variables on which both f and h_1 essentially depend. The last x_e-variable in the variable ordering is the variable x_{e^*}. By the choice of e^* it follows that both endpoints v^* and w^* of e^* are contained in V_1. Hence, $z_{e^*}^{v^*}$ and $z_{e^*}^{w^*}$ are arranged before x_{e^*}. This implies that on the third level of the OBDD for f_{e^*} there are three nodes labeled by x_{e^*}, which are associated with the functions x_{e^*}, $\bar{x}_{e^*}$ and $x_{e^*} \wedge y_{e^*}$. For the first of these subfunctions there is also a node in the OBDD for h_1. Hence, there is one merging. Similarly it can be shown that the OBDDs for f_{e^*} and h_3 contain nodes computing the function y_{e^*}. Hence, these nodes can be merged. We do not obtain another merging between the nodes computing y_{e^*} in the OBDDs for f_{e^*} and h_4 because we already merged those nodes for h_3 and h_4. Altogether, $6m + 5$ nodes can be saved by mergings. We remark that there are no more mergings for the given variable ordering. □

Now we show how to construct a cut from an SBDD.

Lemma 17 If there is an SBDD H for the functions f, g, h^*, h^{**}, h' and h'' of size $s = 152m - n - 54 - c$, then there is a cut of G of size at least c. This cut can be computed in polynomial time.

Proof Let an SBDD H for f, g, h^*, h^{**}, h' and h'' with $s = 152m - n - 54 - c$ nodes be given. First we prove that this OBDD has the properties (P1)–(P4).

Lemma 18 Each SBDD for the functions f, g, h^*, h^{**}, h' and h'' whose variable ordering does not fulfill at least one of the properties (P1)–(P4) has at least $153m - n - 88$ nodes.

Proof of Lemma 18 Let us assume that (P3) is not fulfilled. Then there is some variable a_j that is tested after some variable y_i. Thus at least one of the following statements is true:

1. Before a_j at least half of the b-variables are tested.
2. After y_i at least half of the b-variables are tested.

If the first statement is true, by Lemma 14 the OBDD for h_2 has at least $18m - 10$ nodes. If the second statement is true, by Lemma 13 the OBDD for h_3 has at least $19m - 10$ nodes. For the representation of the other four h-functions we need $4(14m - 10)$ nodes and, hence, for the functions h^* and h^{**} at least $74m - 47$ nodes. By Lemma 10 an SBDD for f and g has at least $49m - n - 10$ nodes. For h' and h'' at least $64m$ nodes are necessary. From the sum of all these lower bounds we have to subtract the number of mergings. In Lemma 19 we show that there are at most $34m + 27$ mergings. Hence, the SBDD for f, g, h^*, h^{**}, h' and h'' has at least $153m - n - 84$ nodes. Similarly we can prove the same lower bound if (P1) or (P4) are not fulfilled.

Now we assume that (P2) does not hold. Furthermore, we assume that (P3) and (P4) hold because we already proved the lower bound if this is not true. By Lemma 11 we need for the representation of f and g at least $53m - n - 14$ nodes. By Lemma 12 we need for h^* and h^{**} at least $5(14m - 10) + 3$ nodes. For h' and h'' we need $64m$ nodes. From the sum $187m - n - 61$ of these lower bounds we again subtract the upper bound $34m + 27$ on the number of mergings and obtain the lower bound $153m - n - 88$ on the SBDD size for f, g, h^*, h^{**}, h' and h''. This completes the proof of Lemma 18. □

Hence, if at least one of the properties (P1)–(P4) is not fulfilled, we get a contradiction because then the SBDD is larger than presumed. If we assume that the last variable in the variable ordering is not a γ-variable, then $34m - 5$ mergings are no longer possible and again we obtain a contradiction to the SBDD size given in the lemma. By Lemma 6 we may move the α-variables to the beginning of the variable ordering without increasing the size of the OBDDs for f, h^* and h^{**}. Also the SBDD size for f, g, h^*, h^{**}, h' and h'' does not become larger since the number of possible mergings does not become smaller when moving the α-variables to the beginning. The reason is that the functions associated with nodes, from which some α_i-node is reachable, essentially depend on α_i and, hence, cannot be merged with nodes of any other OBDD/SBDD.

Now we define the partition (V_1, V_2) that describes the cut. Let V_1 be the set of nodes v of G for which there is some $e \in E(v)$ so that z_e^v is tested before y_e. Let V_2 be the set of nodes v of G for which there is some $e \in E(v)$ so that z_e^v is tested after y_e. From (P2) it follows that (V_1, V_2) is a partition of V. Similar to the proof of Lemma 16 there are at most $34m + 1$ nodes that can be saved by mergings. By Lemma 9 we need for the representation of $p^* \wedge \Gamma$ at least $38m - n - 5$ nodes. For the representation of g^* at least $6m$ nodes are necessary. For h^* and h^{**} at least $70m - 47$ nodes are needed and for h' and h'' at least $64m$ nodes. Hence, there are at most $(152m - n - 54 - c) - (38m - n - 5 + 6m + 70m - 47 + 64m) + 34m + 1 = 8m - 1 - c$ nodes for the representation of f. By Lemma 7 the cut has a size of at least c. □

Lemma 19 Let an SBDD for f and g, an SBDD for h' and h'' and OBDDs for h^* and h^{**} with the same variable ordering be given. If we combine these SBDDs/OBDDs into a single SBDD for f, g, h^*, h^{**}, h' and h'', at most $34m + 27$ nodes may be saved by mergings.

Proof Obviously, there are no mergings of nodes labeled by any α-variable. Hence, we obtain an upper bound on the number of mergings by considering all pairs of OBDDs for $h_1, \ldots, h_5$, the SBDD for f and g and the SBDD for h' and h'' separately. We do not consider the pair of f and g or the pair of h' and h'' because we assume that SBDDs for these pairs are given and all possible mergings were taken into account when computing the SBDD size. This was really done in Lemma 8, Lemma 9, Lemma 10, Lemma 11 and Lemma 15.

We start with the pair of h_1 and h_2. Since h_1 only depends on x- and a-variables and h_2 only depends on a- and b-variables, there may be only mergings of nodes labeled by a-variables and associated with functions that do not essentially depend on any other variable. From the value matrix for h_1 we can see that each block consisting of at least two rows and two columns contains at least two different rows. This means that the subfunction of h_1 corresponding to such a block essentially depends on some x-variable. Hence, we have to consider only those subfunctions of h_1 that correspond to $1 \times (l + 1)$-blocks with at least two different columns and where $l \in \{1, \ldots, 2m\}$. It is easy to see that such blocks describe parity functions of l of the a-variables. Hence, mergings with a-nodes of the OBDD for h_2 are only possible if these nodes are associated with parity functions of a-variables.

In the same way it follows that the subfunctions of h_2 essentially depending only on a-variables correspond to $(l + 1) \times 1$-blocks of the value matrix of h_2. Hence, it suffices to count the number of parity functions corresponding to such blocks. For $l = 1$ we have the 2×1-blocks $\begin{bmatrix} 0 \\ 1 \end{bmatrix}$ and $\begin{bmatrix} 1 \\ 0 \end{bmatrix}$ of the value matrix of h_2. Both blocks describe projections on a single a-variable, which are also parity functions. Hence, at most two mergings are possible for nodes labeled by the last a-variable in the variable ordering. For $l = 2$ we have the following 3×1-blocks.

$$\begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 1 \end{bmatrix}, \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 1 \end{bmatrix}, \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ 0 \end{bmatrix}.$$

The third and the fourth block describe parity functions. Hence, at most two mergings of nodes labeled by the second last a-variable are possible. In the same way it can be shown that at most two mergings are possible for $l = 3$, one merging for $l = 4$ and no merging for $l \geq 5$. Altogether, there are at most 7 mergings between the OBDDs for h_1 and h_2. The same upper bound holds for the number of mergings between the OBDDs for h_2 and h_3, h_3 and h_4, and h_4 and h_5, resp. There are no mergings between other pairs of OBDDs for h_i-functions because for all other pairs the functions essentially depend on disjoint sets of variables. Hence, we have the upper bound 28 for the number of mergings between the OBDDs for the h_i-functions.

Now consider the pair of f and h_1. There may be mergings only of nodes labeled by x-variables. Let x_{e^*} be the last x-variable in the variable ordering. The OBDD for f_{e^*} and, hence, the OBDD for f contains at most two x_{e^*}-variables for which both successors are sinks. Therefore, there are at most two mergings of x_{e^*}-nodes. If x_e is not the last x-variable, then there are no mergings of x_e-nodes because in the OBDD for h_1 there are only x_e-nodes associated with functions that essentially depend on at least two x-variables and on no of the variables $\alpha_1, \ldots, \alpha_{m-1}$. On the other hand, there is no such subfunction of f.

For the pairs of OBDDs for f and h_3 there may be at most one merging, namely a merging of a y-node. By the same arguments as in the last paragraph only mergings of nodes labeled by the last y-variable y_{e^*} in the variable ordering are possible. Furthermore, in each OBDD for f_{e^*} there is at most one y_{e^*}-node for which both successors are sinks. Similarly there is at most one merging between the OBDD for f and h_4. There are no mergings between the OBDD for f and an OBDD for h_2 or h_5 because these functions essentially depend on disjoint sets of variables.

Now we count the number of mergings between the SBDD for h' and h'', the OBDD for g and any other OBDD/SBDD. If the γ-variables are the last variables in the variable ordering, then there may be up to $34m - 5$ mergings between γ-nodes of the SBDD for h' and h'' and the OBDD for g. In this case all functions associated with the nodes of the SBDD for h' and h'' and the OBDD for g essentially depend on some γ-variable. Since this does not hold for the other functions, there are no mergings between the SBDD for h' and h'' or the OBDD for g and any other OBDD/SBDD. Altogether, if a γ-variable is the last variable in the variable ordering, then there are not more than $34m + 27$ nodes that can be saved by mergings.

It remains the case that the last variable in the variable ordering in not a γ-variable. Then the merging of γ-nodes of the SBDD for h' and h'' and the OBDD for g is no longer possible. There are at most n variables that are not γ-variables and whose parity is a subfunction of g. Hence, at most $2n \leq 2m$ mergings between the OBDD for g and the SBDD for h' and h'' are possible. In the same way it follows that there are at most $4m$ mergings between the OBDD for any of the functions $h_1, \ldots, h_5$ and the SBDD for h' and h''. Then the statement of the lemma follows from the upper bound 10 on the number of mergings between the OBDD for g and all the OBDDs for $h_1, \ldots, h_5$.

We start with g and h_1. Mergings are only possible for nodes labeled by a-variables. We already mentioned that the subfunctions of h_1 that do not essentially depend on any x-variable are parity functions of a-variables. Let a_i be the last a-variable in the variable ordering. There may be at most two such subfunctions of h_1, namely a_i and $\overline{a_i}$. Hence, there are at most two mergings. If a_i and a_j are the last two variables in the variable ordering, the subfunctions of h_1 that do not essentially depend on any x-variable are $a_i \oplus a_j$ and $a_i \oplus a_j \oplus 1$. By the definition of g all subfunctions of g that essentially depend only on a-variables are disjunctions of monomials of the form $a_1 \dots a_{k-1}\overline{a_k}$. In particular, the negated literal has the largest index. Parity functions like $a_i \oplus a_j$ cannot be represented as disjunctions of such monomials. Hence, such functions are not subfunctions of g and there are no mergings of a_j-nodes.

A bound on the number of mergings between OBDDs for g and h_2 is obtained in a similar way. Again mergings are only possible for nodes labeled by a-variables and associated with functions that do not essentially depend on any b-variable. As mentioned above such subfunctions of h_2 are described by $(l+1) \times 1$-blocks of W_{h_2}. For $l = 1$ there are two such blocks and, hence, at most two mergings. For $l = 2$, i.e. the second last a-variable, the 3×1-blocks are listed above. Let a_i and a_j be the last two a-variables, where $i < j$. The function corresponding to the first of the blocks listed above is $a_i a_j$, which is a subfunction of g. The second of the blocks listed above corresponds to $a_i \vee a_j$. We prove that this function is not a subfunction of g. The implicants of this function are a_i, a_j, $a_i a_j$, $\overline{a_i} a_j$ and $a_i \overline{a_j}$. The implicant a_j is a subfunction of g only after replacing a_i by 1. But after this replacement we cannot obtain the subfunction $a_i \vee a_j$. Also $\overline{a_i} a_j$ is not of the form of the monomials of g. Hence, only the monomials a_i, $a_i \overline{a_j}$ and $a_i a_j$ are subfunctions of g. The disjunction of these monomials is a_i. Hence, $a_i \vee a_j$ is not a subfunction of g and, hence, there are no mergings for the node of the OBDD for h_2 associated with this function and any node of the OBDD of g. By these arguments it can be shown that there are at most two mergings for $l = 2$, one merging for $l = 3$ and no merging for $l \geq 4$.

Between the OBDDs for g and h_3 there is at most one merging. It suffices to consider nodes labeled by y-variables. There are no subfunctions of g that essentially depend on more than one y-variable and do not essentially depend on any a- or d-variable. Hence, mergings are only possible for nodes associated with subfunctions of g that essentially depend only on one y-variable. Since $\overline{y_i}$ is not a subfunction of g, there is at most one merging.

By similar arguments we obtain that there is at most one merging between the OBDDs for g and h_4 and at most one merging between the OBDDs for g and h_5. □

5 Construction of an OBDD

The function $\mathcal{F}$ that is represented in the OBDD is defined on the variables $s_1, \dots, s_{32m}$, which we already introduced as renamings of the variables on which f, g, h^*, h^{**}, h' and h'' depend, and on $32m + 1$ new variables $t, r_1, \dots, r_{32m}$. The function is defined by

$$\mathcal{F} = \begin{cases} f & \text{if } t = 0 \text{ and } (r_1 + \dots + r_{32m}) \equiv 0 \bmod 4, \\ g & \text{if } t = 0 \text{ and } (r_1 + \dots + r_{32m}) \equiv 1 \bmod 4, \\ h^* & \text{if } t = 0 \text{ and } (r_1 + \dots + r_{32m}) \equiv 2 \bmod 4, \\ h^{**} & \text{if } t = 0 \text{ and } (r_1 + \dots + r_{32m}) \equiv 3 \bmod 4, \\ 1 & \text{if } t = 1 \text{ and } (r_1 + \dots + r_{32m}) \equiv 0 \bmod 5, \\ h' & \text{if } t = 1 \text{ and } (r_1 + \dots + r_{32m}) \equiv 1 \bmod 5, \\ 0 & \text{if } t = 1 \text{ and } (r_1 + \dots + r_{32m}) \equiv 2 \bmod 5, \\ h'' & \text{if } t = 1 \text{ and } (r_1 + \dots + r_{32m}) \equiv 3 \bmod 5, \\ 0 & \text{if } t = 1 \text{ and } (r_1 + \dots + r_{32m}) \equiv 4 \bmod 5. \end{cases}$$

In order to illustrate this definition Fig. 5 shows an OBDD for $\mathcal{F}$ and the (nonoptimal) variable ordering t, r-variables, s-variables. For $t = 0$ the r-variables determine which of the functions f, g, h^* and h^{**} is computed. For $t = 1$ the function $\mathcal{F}_{|t=1} = h(r_1, \ldots, r_{32m}, s_1, \ldots, s_{32m})$ is computed. By Lemma 12 an OBDD for this function has its minimum size if the r-variables are arranged before the s-variables. We are going to prove that also the OBDD for $\mathcal{F}$ takes its minimum size for such variable orderings. We also see that the OBDD contains an SBDD for f, g, h^*, h^{**}, h' and h''. The connection between the OBDD size for $\mathcal{F}$ and the SBDD size for f, g, h^*, h^{**}, h' and h'' is given in Lemma 20.

(Insert Figure 5 here)

Let τ be a variable ordering of the s-variables and π be a variable ordering of t, the r-variables and the s-variables. We call π consistent with τ if the relative ordering of the s-variables is the same for τ and π. The variable ordering π is called τ-optimal if π is consistent with τ and leads to minimum OBDD size for $\mathcal{F}$ among all variables orderings that are consistent with τ.

Lemma 20 Let τ be some ordering of the s-variables and let S_τ be the size of an SBDD for f, g, h^*, h^{**}, h' and h'' and the variable ordering τ. Let π be the following τ-consistent variable ordering: seven r-variables, t, the remaining r-variables, the s-variables according to τ.

1. If $S_\tau < 153m$, then π is τ-optimal and the OBDD size for $\mathcal{F}$ and π is $288m - 27 + S_\tau$.
2. If $S_\tau \geq 153m$, then the OBDD size for $\mathcal{F}$ and each τ-consistent variable ordering is at least $441m - 27$.

The lemma is proved in the following subsections. Here we give an outline of the proof. First the OBDD size for $\mathcal{F}$ and the variable ordering π is computed. Then we consider those variable orderings in which less than $14m$ of the r-variables are arranged before the first s-variable. Such variable orderings lead to an OBDD size for $\mathcal{F}$ of at least $441m - 27$. In particular, such variable orderings are not τ-optimal. In the next step we search for an optimal position of t. If t is arranged after the $14m$ of the r-variables in the top, then again the OBDD size for $\mathcal{F}$ is larger than $441m - 27$. By applying the techniques for calculating the OBDD size for partially symmetric functions (Sieling (1996)) outlined in Section 2.2 we can show that the optimal position of t is after seven (or eight) of the r-variables. Finally, we show that it is optimal to arrange all s-variables after all r-variables. Hence, we constructed a τ-optimal variable ordering. The OBDD for this variable ordering consists of an SBDD for f, g, h^*, h^{**}, h' and h'' and of $288m - 27$ nodes labeled by t and by r-variables. This implies the bounds of both claims of the lemma.

We conclude this section by showing how the results of the last section and Lemma 20 imply Theorem 4 and Theorem 5.

5.1 The OBDD Size of $\mathcal{F}$ and π

We assume that some variable ordering τ on the s-variables is fixed and the π is defined as in Lemma 20. Since the s-variables are the last variables in π, the nodes labeled by s-variables form an SBDD for f, g, h^*, h^{**}, h' and h''. Hence, there are S_τ such nodes.

Now we consider the part of the OBDD for $\mathcal{F}$ and π consisting of nodes labeled by t or by r-variables as an OBDD with eight sinks, which are labeled by f, g, h^*, h^{**}, h', h'', 0 and 1. This OBDD computes a function that depends on $t, r_1, \ldots, r_{32m}$ and takes those eight values. This function is partially symmetric with respect to the symmetry sets $\{t\}$ and $\{r_1, \ldots, r_{32m}\}$. The value matrix is a $2 \times (32m + 1)$-matrix with eight different entries instead of two different entries as in the case of a Boolean function. The value matrix and the corresponding grid graph are shown in Fig. 6. The value matrix is periodic with a period of 4 in the first row and a period of 5 in the second row. It is easy to verify that there are two shortest paths from the source of the grid graph to its sink. These paths are indicated in Fig. 6 by solid edges and one of them is the path corresponding to the variable ordering described in Lemma 20. The length of both paths is $288m - 27$. Hence, for the variable ordering π the OBDD size for $\mathcal{F}$ is exactly $288m - 27 + S_\tau$. If $S_\tau \geq 153m$, then the OBDD size for $\mathcal{F}$ and π is at least $441m - 27$.

(Insert Figure 6 here)

5.2 The Relative Position of the r- and s-Variables

We are going to prove that in τ-optimal variable orderings at least $14m$ of the r-variables are arranged before the first s-variable. We remark that we do not assume anything about the position of t.

Let some variable ordering be given in which less than $14m$ of the r-variables are arranged before the first s-variable s^*. W.l.o.g. let $r_1, \ldots, r_{32m}$ be the relative ordering of the r-variables. We estimate the number of r_i-nodes only for $i \geq 21$ and $i \leq 32m - 4$. We distinguish two cases.

Case 1 The variable r_i is arranged after s^*. Then there are at least 14 nodes labeled by r_i.

Case 2 The variable r_i is arranged before s^*. Then there are at least 9 nodes labeled by r_i.

Before we prove these claims, we compute the lower bound on the OBDD size for $\mathcal{F}$ and the given variable ordering using these claims. Let l be the number of r-variables arranged before s^*. Then $l < 14m$. The number of nodes labeled by r-variables is at least

$$\max\{l - 20, 0\} \cdot 9 + (32m - 4 - \max\{l, 20\}) \cdot 14 \geq 378m - 236.$$

The terms $\max\{\cdot\}$ are used because l may be smaller than 20. Furthermore, we know that there are at least $64m$ nodes labeled by s-variables because h' and h'' are subfunctions of $\mathcal{F}$. The sum of the lower bounds is $442m - 236$. Hence, all variable orderings where less than $14m$ of the r-variables are arranged before the first s-variable are not τ-optimal since we assumed $m \geq 250$. For such variable orderings also the lower bound of the second claim of Lemma 20 holds.

Now we prove the lower bounds claimed in Case 1 and Case 2. We distinguish the following subcases.

Case 1a The variables s^* and t are arranged before r_i.

Let R_1 and S_1 be the sets of r-variables and s-variables, resp., that are arranged before r_i. Then $|R_1| \geq 20$, and $|S_1| \geq 1$ because of $s^* \in S_1$. By Lemma 2 the lower bound can be proved by giving 14 assignments to the variables in R_1 and S_1 and to t leading to 14 different subfunctions of $\mathcal{F}$ that essentially depend on r_i.

Let the set R_2 consist of r_i and all variables arranged after r_i in the variable ordering. We prove that the subfunctions obtained by the 14 assignments are different by replacing the variables from R_2

except five r-variables by constants. There are at least five r-variables in R_2 because $i \leq 32m - 4$. The subfunctions that we obtain are symmetric with respect to those five r-variables and, hence, they can be represented by a value vector of length six. (The i-th entry of the value vector, where $i \in \{0, \ldots, 5\}$, gives the value that the function takes if exactly i variables of the input take the value 1.) The functions are different because the value vectors are different. The functions essentially depend on the r-variables because the value vectors are not constant.

In order to simplify the proof we do no longer distinguish between the variables arranged before and after r_i. Instead of this we give assignments to all variables except five r-variables. Since the obtained subfunctions are symmetric, it is not important which r-variables are replaced by constants. The given assignments only differ in the assignments to t, to s^* and to five r-variables. It is clear that these five r-variables can be chosen in such a way that they are arranged before r_i. For s^* and t it is presumed that they are arranged before r_i. Hence, all assignments to variables arranged after r_i are equal and we estimate the number of subfunctions correctly.

First we construct two assignments A_1 and A_2 of the s-variables that only differ in s^*, but not in any other s-variable. It is not difficult to choose these assignments so that

$$\begin{array}{llllll} f(A_1) = 1, & g(A_1) = 0, & h^*(A_1) = 0, & h^{**}(A_1) = 0, & h'(A_1) = 1, & h''(A_1) = 0, \\ f(A_2) = 1, & g(A_2) = 0, & h^*(A_2) = 0, & h^{**}(A_2) = 0, & h'(A_2) = 0, & h''(A_2) = 1. \end{array}$$

In the following table we list the 14 assignments and the corresponding value vectors. In the table $||r|| \equiv k \bmod 4$ means that we choose an assignment to all but five of the r-variables so that the sum of these $32m - 5$ variables is congruent $k \bmod 4$.

Assignment	Value vector
$t = 0,\ A_1,\ \|\|r\|\| \equiv 0 \bmod 4$	1 0 0 0 1 0
$t = 0,\ A_1,\ \|\|r\|\| \equiv 1 \bmod 4$	0 0 0 1 0 0
$t = 0,\ A_1,\ \|\|r\|\| \equiv 2 \bmod 4$	0 0 1 0 0 0
$t = 0,\ A_1,\ \|\|r\|\| \equiv 3 \bmod 4$	0 1 0 0 0 1
$t = 1,\ A_1,\ \|\|r\|\| \equiv 0 \bmod 5$	1 1 0 0 0 1
$t = 1,\ A_1,\ \|\|r\|\| \equiv 1 \bmod 5$	1 0 0 0 1 1
$t = 1,\ A_1,\ \|\|r\|\| \equiv 2 \bmod 5$	0 0 0 1 1 0
$t = 1,\ A_1,\ \|\|r\|\| \equiv 3 \bmod 5$	0 0 1 1 0 0
$t = 1,\ A_1,\ \|\|r\|\| \equiv 4 \bmod 5$	0 1 1 0 0 0
$t = 1,\ A_2,\ \|\|r\|\| \equiv 0 \bmod 5$	1 0 0 1 0 1
$t = 1,\ A_2,\ \|\|r\|\| \equiv 1 \bmod 5$	0 0 1 0 1 0
$t = 1,\ A_2,\ \|\|r\|\| \equiv 2 \bmod 5$	0 1 0 1 0 0
$t = 1,\ A_2,\ \|\|r\|\| \equiv 3 \bmod 5$	1 0 1 0 0 1
$t = 1,\ A_2,\ \|\|r\|\| \equiv 4 \bmod 5$	0 1 0 0 1 0

Case 1b The variable s^* is arranged before r_i, and t is arranged after r_i.

As in Case 1a we choose an assignment A_1 to the s-variables so that

$$f(A_1) = 1, \quad g(A_1) = 0, \quad h^*(A_1) = 0, \quad h^{**}(A_1) = 0, \quad h'(A_1) = 1, \quad h''(A_1) = 0.$$

Then even the OBDD for the subfunction $\mathcal{F}_{|A_1}$ contains 20 nodes labeled by r_i. This subfunction is partially symmetric with respect to $\{t\}$ and to the set of r-variables. The value matrix of $\mathcal{F}_{|A_1}$ is shown in Fig. 7. The number of r_i-nodes is equal to $T_2(2, 32m+2-i)$, i.e. the number of $2 \times (32m+2-i)$-blocks with at least two different columns. Because of $i \geq 21$ and $i \leq 32m - 4$ there are 20 such blocks.

(Insert Figure 7 here)

Case 2a The variable r_i is arranged before s^* and after t.

Again we choose the assignment A_1 as described above and prove that the OBDD for $\mathcal{F}_{|A_1}$ contains 9 nodes labeled by r_i. From the value matrix in Fig. 7 we see that there are 9 different $1 \times (32m+2-i)$-blocks that are not constant. Here we use again $i \geq 21$ and $i \leq 32m - 4$.

Case 2b The variable r_i is arranged before s^* and before t.

By the same arguments as in Case 1b even the lower bound 20 on the number of r_i-nodes follows.

5.3 The Position of *t*

Now we assume that before the first s-variable at least $14m$ of the r-variables are arranged. Otherwise this variable ordering leads to an OBDD size for $\mathcal{F}$ of at least $441m - 27$ and it is not τ-optimal. First we show the following: If t is arranged after the $14m$ r-variables at the beginning of the variable ordering, then the OBDD size is at least $442m - 226$ which is larger than $441m - 27$ because of $m \geq 250$. Hence, for this case the second claim is proved and we know that such variable orderings are not τ-optimal.

Let such a variable ordering be given. We estimate the number of r_i-nodes with $i \leq 14m$ and $i > 14m$ separately. Again we consider the subfunction $\mathcal{F}_{|A_1}$ with the assignment A_1 of the last subsection. The value matrix and the grid graph of this subfunction are shown in Fig. 7. For $i \leq 14m$ the number of r_i-nodes is equal to $T_2(2, 32m + 2 - i)$ because t is arranged after r_i. The number of all such r_i-nodes is

$$\sum_{i=1}^{14m} T_2(2, 32m + 2 - i) = 280m - 190.$$

A lower bound on the number of r_i-nodes with $14m + 1 \leq i \leq 32m - 4$ is $\min\{T_2(1, 32m + 2 - i), T_2(2, 32m + 2 - i)\} = 9$. This means that for each r_i we take the possibilities that t is arranged before and after r_i into account and choose that possibility leading to a smaller number of r_i-nodes. The total number of r_i-nodes with $14m+1 \leq i \leq 32m-4$ at least $(18m-4) \cdot 9 = 162m-36$. The sum of both lower bounds is $442m - 226$ which is larger than $441m - 27$ because of $m \geq 250$.

If we search for an optimal position for t it suffices to consider the first $14m + 1$ levels of the OBDD. In particular, the position of t does not affect the number of s-nodes nor the number of r_i-nodes with $i > 14m$. Hence, it suffices to search in the grid graph of $\mathcal{F}$ in Fig. 6 for a shortest path from the source $(32m + 1, 2)$ to the node $(18m + 2, 1)$ instead of a path to the sink. It is easy to see that we obtain such a shortest path iff t is arranged after seven or eight of the r-variables. Hence, only those variable orderings may be τ-optimal where t is at one of these positions. Also for the proof of the second claim of Lemma 20 it suffices to consider variable orderings where t is at one of these positions.

5.4 The Position of the Remaining r-Variables

In the following we only consider variable orderings that start with seven r-variables, t and then $14m - 7$ of the r-variables. After that the remaining r-variables and the s-variables may be mixed arbitrarily. We prove that the OBDD size for $\mathcal{F}$ does not increase if we move all r-variables before all s-variables. This implies that the variable ordering π described in Lemma 20 is τ-optimal. If $S_\tau \geq 153m$, then either the OBDD size for $\mathcal{F}$ an π is at least $441m - 27$ or all previous steps do not increase the OBDD size for $\mathcal{F}$. But then by the calculation in Section 5.1 the number of nodes labeled by an r-variable or t is at least $288m - 27$ which implies the second claim of Lemma 20. In order to show that we may move all r-variables before all s-variables we proceed in the following steps.

1. We show that for all variable orderings that are still possible that the number of nodes labeled by s-variables is at least S_τ. Hence, the number of such nodes does not increase when moving the r-variables before the s-variables.
2. We move all r-variables except the last four r-variables in the variable ordering before all s-variables and prove that the number of nodes labeled by r-variables does not increase.
3. The number of nodes labeled by the last four r-variables is at least 22. If all r-variables are arranged before all s-variables, then there are exactly 36 nodes labeled by the last four r-variables, namely 9 nodes for each variable.
4. If before the last r-variable at least four s-variables are arranged, then the number of nodes labeled by s-variables is at least $S_\tau + 14$. Hence, in this case we may move all r-variables before all s-variables without increasing the OBDD size.
5. Let r_i be one of the last four r-variables. If before r_i one, two or three s-variables are arranged, then there are at least 9 nodes labeled by r_i. Hence, the OBDD size does not increase when moving r_i before all s-variables.

By the proof of these claims Lemma 20 follows.

Proof of Claim 1 Let $r_1, \ldots, r_{32m}$ be the relative ordering of the r-variables. We replace $r_4, \ldots, r_{32m}$ by the constant 0. Now the remaining r-variables r_1, r_2 and r_3, and t are arranged before the s-variables. It is easy to see from the definition of $\mathcal{F}$ that by choosing appropriate assignments to r_1, r_2, r_3 and t we may obtain the subfunctions f, g, h^*, h^{**}, h' and h''. Hence, the OBDD contains an SBDD for f, g, h^*, h^{**}, h' and h''. Since the replacement of r-variables and t does not increase the number of nodes labeled by s-variables, the OBDD contains at least S_τ nodes labeled by s-variables.

Proof of Claim 2 We consider again the assignment A_1 to the s-variables. The value matrix and the grid graph for the subfunction $\mathcal{F}_{|A_1}$ are shown in Fig. 7. The numbers $T_2(2, 32m + 2 - i)$ for $i \in \{1, \ldots, 7\}$ and $T_2(1, 32m + 2 - i)$ for $i \geq 8$ are exactly the numbers of r_i-nodes in an OBDD for $\mathcal{F}_{|A_1}$ and, therefore, lower bounds on the numbers of r_i-nodes in the OBDD for $\mathcal{F}$. The number of r_i-nodes in an OBDD for $\mathcal{F}$ and the variable ordering π can be seen in Fig. 6. A comparison shows that these numbers are equal for $i \leq 32m - 4$. Hence, the number of r_i-nodes for $i \leq 32m - 4$ does not increase when moving r_i before all s-variables.

Proof of Claim 3 Also the lower bound for the number of r_i-nodes for $i \geq 32m - 3$ follows by considering the subfunction $\mathcal{F}_{|A_1}$. The lower bound is $T_2(1, 5) + \ldots + T_2(1, 2) = 22$. The exact number of r_i-nodes in an OBDD for $\mathcal{F}$ and π can be seen from the value matrix in Fig. 6. Since there are nine different and nonconstant $1 \times (32m + 2 - i)$-blocks, there are exactly nine nodes labeled by r_i. Hence, at most 14 nodes labeled by r-variables may be saved if the last four r-variables are not arranged before all s-variables.

Proof of Claim 4 We consider the case that before the last r_i-variable there are at least four s-variables. We show that there are $S_\tau + 14$ nodes labeled by s-variables. We consider the OBDD for $\mathcal{F}_{|t=1}$ and prove that it contains at least four nodes labeled by the first s-variable and six nodes for each of the second, third and fourth s-variable. At all these nodes subfunctions are computed that are not subfunctions of $\mathcal{F}_{|t=0}$. Hence, none of these nodes can be saved by mergings. On the other hand, the OBDD for $\mathcal{F}_{|t=1}$ and π contains exactly two nodes for each s-variable. Hence, the number of nodes labeled by s-variables in the OBDD exceeds the number of such nodes in the OBDD for the variable ordering π by at least 14.

Since $\mathcal{F}_{|t=1} = h(r_1, \ldots, r_{32m}, s_1, \ldots, s_{32m})$ is partially symmetric with respect the r- and s-variables, we may describe the subfunctions explicitly by giving the corresponding blocks. Let R be the number of r-variables that are arranged after the first s-variable. Since we moved all but the last four r-variables before all s-variables, we have $R \leq 4$. The subfunctions associated with the nodes labeled by the first s-variable correspond to $(R + 1) \times (32m + 1)$-blocks of W_h with at least two different columns. For $R = 1$ these are the blocks listed in the following. For $R > 1$ we may apply the same arguments.

$$\begin{bmatrix} 1 & 1 & 1 & 1 & \cdots \\ 0 & 1 & 0 & 1 & \cdots \end{bmatrix}, \quad \begin{bmatrix} 0 & 1 & 0 & 1 & \cdots \\ 0 & 0 & 0 & 0 & \cdots \end{bmatrix}, \quad \begin{bmatrix} 0 & 0 & 0 & 0 & \cdots \\ 1 & 0 & 1 & 0 & \cdots \end{bmatrix}, \quad \begin{bmatrix} 1 & 0 & 1 & 0 & \cdots \\ 0 & 0 & 0 & 0 & \cdots \end{bmatrix}.$$

The functions described by these blocks have as subfunction the parity of all s-variables. There is no such subfunction of f, g, h^* and h^{**}. Hence, no mergings are possible and the number of nodes labeled by the first s-variable exceeds the number of such nodes in an OBDD for the variable ordering π by at least 2.

Now consider the second s-variable. Let R be the number of r-variables arranged after this s-variable. The subfunctions of $\mathcal{F}_{|t=1}$ associated with the nodes labeled by this s-variable can be described by $(R + 1) \times 32m$-blocks with at least two different columns. For $R = 1$ these blocks are the following ones.

$$\begin{bmatrix} 1 & 1 & 1 & 1 & \cdots \\ 0 & 1 & 0 & 1 & \cdots \end{bmatrix}, \quad \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots \\ 1 & 0 & 1 & 0 & \cdots \end{bmatrix}, \quad \begin{bmatrix} 0 & 1 & 0 & 1 & \cdots \\ 0 & 0 & 0 & 0 & \cdots \end{bmatrix},$$
$$\begin{bmatrix} 0 & 0 & 0 & 0 & \cdots \\ 1 & 0 & 1 & 0 & \cdots \end{bmatrix}, \quad \begin{bmatrix} 0 & 0 & 0 & 0 & \cdots \\ 0 & 1 & 0 & 1 & \cdots \end{bmatrix}, \quad \begin{bmatrix} 1 & 0 & 1 & 0 & \cdots \\ 0 & 0 & 0 & 0 & \cdots \end{bmatrix}.$$

All functions described by these blocks have as subfunction the parity of $32m - 1$ of the s-variables. Again there are no mergings with the OBDD for $\mathcal{F}_{|t=0}$.

The lower bound for the number of nodes labeled by the third and fourth s-variable is proved in the same way.

Proof of Claim 5 Let r_i be one of the last four r-variables. Let j be the number of r-variables arranged after r_i. Then $j \in \{0, 1, 2, 3\}$. Let S be the number of s-variables arranged before r_i. Then $S \in \{1, 2, 3\}$. The subfunctions of $\mathcal{F}_{|t=1}$ that are computed at r_i-nodes can be described by $(j + 2) \times (32m + 1 - S)$-blocks of the value matrix of h with at least two different rows. There are seven such blocks. For $j = 0$ these blocks are listed in the following. For $j > 0$ the same arguments can be applied.

$$\begin{bmatrix} 1 & 1 & 1 & 1 & \cdots \\ 0 & 1 & 0 & 1 & \cdots \end{bmatrix}, \quad \begin{bmatrix} 1 & 1 & 1 & 1 & \cdots \\ 1 & 0 & 1 & 0 & \cdots \end{bmatrix}, \quad \begin{bmatrix} 0 & 1 & 0 & 1 & \cdots \\ 0 & 0 & 0 & 0 & \cdots \end{bmatrix},$$
$$\begin{bmatrix} 0 & 0 & 0 & 0 & \cdots \\ 1 & 0 & 1 & 0 & \cdots \end{bmatrix}, \quad \begin{bmatrix} 0 & 0 & 0 & 0 & \cdots \\ 0 & 1 & 0 & 1 & \cdots \end{bmatrix}, \quad \begin{bmatrix} 1 & 0 & 1 & 0 & \cdots \\ 0 & 0 & 0 & 0 & \cdots \end{bmatrix}, \quad \begin{bmatrix} 0 & 0 & 0 & 0 & \cdots \\ 1 & 1 & 1 & 1 & \cdots \end{bmatrix}.$$

In order to show that there are at least nine nodes labeled by r_i we present two subfunctions of $\mathcal{F}_{|t=0}$. Let s_1, s_2 and s_3 (or a subset of them if $S < 3$) be the s-variables arranged before r_i. If we replace all r-variables except r_i by 0 and s_1, s_2 and s_3 by 1, we obtain

$$\overline{r_i} f_{|s_1=1,s_2=1,s_3=1} \vee r_i g_{|s_1=1,s_2=1,s_3=1}.$$

Similarly we may obtain

$$\overline{r_i} h^*_{|s_1=1,s_2=1,s_3=1} \vee r_i h^{**}_{|s_1=1,s_2=1,s_3=1}.$$

Both functions essentially depend on r_i and are different from the functions described by the blocks given above. The reason is that all except the last one of the blocks listed above describe functions which have as subfunctions the parity of at least $32m - 3$ of the s-variables. The last block describes the functions r_i. Altogether, there are at least nine nodes labeled by r_i. This completes the proof of Lemma 20.

5.5 The Nonexistence of Polynomial Time Approximation Schemes for MinOBDD and MinOBDD*

We describe how to compute the functions φ and ψ of the L-reduction. For the computation of $\varphi(G)$ we choose the trivial cut $(V, \emptyset)$ of size 0 and the variable ordering τ of the s-variables for this cut as described in the proof of Lemma 16. The SBDD size for f, g, h^*, h^{**}, h' and h'' and this variable ordering is $152m - n - 54$. Hence, by the first claim of Lemma 20 the OBDD size for $\mathcal{F}$ and the τ-consistent variable ordering described in this lemma is $N = 440m - n - 81$. It is easy to see that the OBDD for $\mathcal{F}$ and this variable ordering can be computed in polynomial time. Thus, also φ can be computed in polynomial time.

If c_{max} is the size of a maximum cut of G, by Lemma 16 and Lemma 17 the minimum size of an SBDD for f, g, h^*, h^{**}, h' and h'' is $152m - n - 54 - c_{max}$. By the first claim of Lemma 20 the minimum size of an OBDD for $\mathcal{F}$ is $s_{min} = 440m - n - 81 - c_{max}$. In particular, $s_{min} \geq N/2$ and the promise for MinOBDD* is fulfilled.

Now we consider how to compute ψ. If the input for ψ is a variable ordering with an OBDD size s for $\mathcal{F}$ that is at least $440m - n - 81$, the output is a cut of size $c = 0$. Then the second condition of the definition of the L-reductions is fulfilled because $c_{max} - c = c_{max} = (440m - n - 81) - (440m - n - 81 - c_{max}) \leq s - s_{min}$.

Let an OBDD for $\mathcal{F}$ with a variable ordering π and size $s = 440m - n - 81 - c'$ with $c' > 0$ be given. Let τ be the relative ordering of the s-variables in π and let S_τ be the SBDD size for f, g, h^*, h^{**}, h' and h'' and the variable ordering τ . By the second claim of Lemma 20 it cannot happen that $S_\tau \geq 153m$. We consider the following τ-consistent variable ordering π^*: seven r-variables, t, the remaining r-variables, the s-variables in the ordering τ. By the first claim of Lemma 20 the variable ordering π^* is τ-optimal. This implies that the OBDD size for $\mathcal{F}$ and π^* is not larger than the OBDD size for $\mathcal{F}$ and π, i.e. it is at most $440m - n - 81 - c'$. Then, by the first claim of Lemma 20 we have $S_\tau \leq 152m - n - 54 - c'$. Hence, by Lemma 17 there is a cut of G of size c, where $c \geq c'$. This cut can be computed in polynomial time from π. In particular, ψ can be computed in polynomial time.

Because of $c \geq c' = 440m - n - 81 - s$ and $c_{max} = 440m - n - 81 - s_{min}$ it follows that $c_{max} - c \leq s - s_{min}$, i.e. the second property of the definition of the L-reductions is fulfilled for $\beta = 1$.

In order to prove the first property of the definition of the L-reductions we exploit the well-known fact that for each graph $G = (V, E)$ the size c_{max} of a maximum cut is at least $|E|/2$ (see, e.g., Motwani and Raghavan (1995)). Hence, $m \leq 2c_{max}$. Combining this inequality with $s_{min} = 440m - n - 81 - c_{max}$ we obtain $c_{max} \geq s_{min}/879$. Hence, the first property is fulfilled for $\eta = 1/879$. Altogether, we proved MaxCut $\leq_L$ MinOBDD and MaxCut $\leq_L$ MinOBDD*.

6 The Nonapproximability of MinOBDD

For the proof of Theorem 1 we adopt a technique due to Garey and Johnson (1979) for improving the performance ratio of approximation algorithms. They show for the problem Independent Set that a polynomial time approximation scheme can be constructed from each polynomial time approximation algorithm with a constant performance ratio. The proof is based on multiplying graphs. In the following subsection we define a similar multiplication of Boolean functions. This multiplication can be used to improve approximation algorithms for MinOBDD only for instances with some special properties. We show how to modify an OBDD to match these properties. Afterwards we show how to construct a polynomial time approximation scheme for MinOBDD* from a polynomial time approximation algorithm with a constant performance ratio for MinOBDD. Together with Theorem 5 we obtain Theorem 1.

6.1 Multiplication of Boolean Functions

Let f be a Boolean function on n variables $x_1, \ldots, x_n$ and let g be a Boolean function on l variables. We define $g[f] : \{0,1\}^{nl} \to \{0,1\}$ by

$$g[f] = g(f(x_1^1, \ldots, x_n^1), \ldots, f(x_1^l, \ldots, x_n^l)).$$

We call the set $B^j = \{x_1^j, \ldots, x_n^j\}$ the j-th block of the set of variables. A blockwise variable ordering of the variables that $g[f]$ depends on is an ordering in which the variables of each block B^j are arranged adjacently. Furthermore, we define $f^1 = f$. For $i \geq 2$ we define $f^i = f^{i-1}[f]$.

It is easy to compute an OBDD for $g[f]$ from OBDDs for f and g. In the OBDD for g we replace each node v labeled by x_j by an OBDD G_v for f on the variables B^j. Replacing means that each edge leading to v is redirected to the source of G_v, that the edges of G_v leading to the 0-sink are redirected to the 0-successor of v and that the edges of G_v leading to the 1-sink are redirected to the 1-successor of v.

Now the conjecture is obvious that for each OBDD G for $g[f]$ there is an OBDD G' for $g[f]$ with a blockwise ordering and $|G'| \leq |G|$. For the special case that g is a function depending on two variables this conjecture was proved by Sauerhoff, Wegener and Werchner (1996). In Lemma 25 we prove this conjecture only for the special case that f has additional properties. (We remark that by slight modifications of that proof we obtain for arbitrary f the weaker statement that there is an OBDD G' for $g[f]$ with a blockwise ordering and $|G'| \leq 2|G|$.)

First we define the functions for which we are going to prove that conjecture. Let $h : \{0,1\}^n \to \{0,1\}$ be a function, which is defined on the variables $x_1, \ldots, x_n$. Throughout this section we assume that h essentially depends on all variables. Let a reduced OBDD G_h for h with N interior nodes be given. We define the function $f_h : \{0,1\}^{n+2N} \to \{0,1\}$ by

$$f_h(x_1, \ldots, x_n, y_1, \ldots, y_N, z_1, \ldots, z_N) = y_1 \ldots y_N \overline{h(x_1, \ldots, x_n)} \vee z_1 \ldots z_N h(x_1, \ldots, x_n).$$

In Lemma 25 we prove the above conjecture only for functions $g[f]$, where $f = f_h$ for some function h. Before that proof we prove several technical lemmas.

It is easy to obtain an OBDD G_f for f_h from G_h. In G_h we replace the 0-sink by an OBDD computing $y_1 \ldots y_N$ and the 1-sink by an OBDD computing $z_1 \ldots z_N$. The size of G_f is bounded by $|G_h| + 2N$. We prove that this is optimal. In the following let $OPT(G)$ be the minimum OBDD size for the function computed by G and let $OPT(f)$ be the minimum OBDD size for f.

Lemma 21 Let π be an ordering of the variables $x_1, \ldots, x_n, y_1, \ldots, y_N, z_1, \ldots, z_N$ and let G_π be the OBDD for f_h and π. Let τ be an ordering that we obtain from π by moving $y_1, \ldots, y_N, z_1, \ldots, z_N$ to the end of the variable ordering. Let G_τ be the OBDD for f_h and τ. Then $|G_\tau| \leq |G_\pi|$. Furthermore, $OPT(f_h) = OPT(h) + 2N$.

Proof Let σ be the relative ordering of the x-variables in π and let G_h be the OBDD for h and σ. It is easy to verify that G_τ consists of G_h where the sinks are replaced by OBDDs for the conjunction of the y- and z-variables, resp. On the other hand, we obtain G_h from G_π if we replace all y-variables by 0 and all z-variables by 1. Hence, the number of x-nodes in G_π cannot be smaller than the number of x-nodes in G_τ. Since the number of y- and z-nodes cannot be smaller than $2N$, it holds that G_π cannot be smaller than G_τ. This also implies that we obtain a minimum size OBDD for f_h, if we choose the variable ordering of a minimum size OBDD for h and append $y_1, \ldots, y_N, z_1, \ldots, z_N$ to this variable ordering. □

Hence, it is good to arrange the x-variables at the beginning of the variable ordering. By the following lemma we show that it is really bad to arrange some x-variable at the end of the variable ordering.

Lemma 22 Let G be an OBDD for f_h where the last variable in the variable ordering is an x-variable x^*. Then there are at least $3N$ nodes labeled by y- and z-variables.

Proof Since h essentially depends on all x-variables, there is an assignment c to all x-variables except x^* so that the resulting subfunction $h_{|c}$ is equal to x^* or to $\overline{x^*}$. W.l.o.g. let $h_{|c} = x^*$. By the same assignment we obtain from f_h the subfunction $f_{h|c} = y_1 \ldots y_N \overline{x^*} \vee z_1 \ldots z_N x^*$. W.l.o.g. let the first variable in the variable ordering be a y-variable. Since $f_{h|c}$ essentially depends on all y-variables, there are at least N nodes labeled by y-variables. There are at least two nodes labeled by each z-variable z_i since the OBDD has to distinguish the cases that the conjunction of the y-variables tested before is 0 and 1. Hence, there are at least $2N$ nodes labeled by z-variables. □

In Lemma 24 we show that an OBDD for $g[f_h]$ and a blockwise ordering, where the y- and z-variables are tested at the end of each block, consists of disjoint OBDDs for f_h. The proof of the disjointness is based on the fact that an SBDD for f_h and $\overline{f_h}$ and such a variable ordering consists of OBDDs for f_h and $\overline{f_h}$ that do not share interior nodes.

Lemma 23 Let OBDDs G and $\overline{G}$ for the functions f_h and $\overline{f_h}$ with the same variable ordering be given. We assume that in the variable ordering all y-variables are arranged adjacently and that the same holds for the z-variables. We combine G and $\overline{G}$ to an SBDD and apply the reductions rules. There are no mergings between y-nodes or z-nodes of G and $\overline{G}$. Furthermore, let x^* be an x-variable, which is arranged before the block of y-variables or before the block of z-variables. Then there are no mergings between x^*-nodes of the OBDDs for G and $\overline{G}$.

Proof Let y^* be some y-variable. Let v be a y^*-node of G and let v' be a y^*-node of $\overline{G}$. The function associated with v is a subfunction of f_h, which essentially depends on y^*. Similarly, the function associated with v' is a subfunction of $\overline{f_h}$, which essentially depends on y^*. By the definition of f_h it holds that the subfunctions of f_h that essentially depend on y^* are monotone increasing in y^*. On the other hand the subfunctions of $\overline{f_h}$ that essentially depend on y^* are monotone decreasing in y^*. Hence, the functions associated with v and v' are different and v and v' cannot be merged. The same arguments hold for the nodes labeled by z-variables.

Now assume that after x^* there are all y-variables in the variable ordering. Let v be a node in G labeled by x^*. Since the function associated with v essentially depends on x^*, we can find an assignment c to the x-variables and the z-variables so that in G the computation path for c leads through v and that h takes the value 0 for this assignment of the x-variables. By this assignment we obtain from f_h the subfunction $y_1 \dots y_m$. Hence, the function associated with v essentially depends on all y-variables and is monotone increasing in all y-variables. By the same arguments the function associated with each x^*-node v' of $\overline{G}$ is monotone decreasing in all y-variables. Again there are no mergings. $\Box$

Lemma 24 Let an OBDD G for $g[f_h]$ be given. Let the variable ordering be a blockwise variable ordering where in each block the x-variables are arranged before all y-variables and all z-variables. Let the relative ordering of the blocks be $B_1, \dots, B_l$. Let π^j be the relative ordering of the variables in B^j and let H be an OBDD for f_h and π^j.

Then the following holds. The layer of nodes of G labeled by variables from B^j consists of disjoint copies of H. Each edge leading from some B^i-node, where $i < j$, to some B^j-node leads to the source of such an OBDD. The 0-sink and the 1-sink of each copy are interior nodes of the blocks $B^{j+1}, \dots, B^l$ or sinks of G.

Proof Let v be some node of the B^j-layer with an incoming edge from some previous layer. (If $j = 1$ let v be the source of G.) We prove that v is the source of an OBDD for f_h defined on the B^j-variables. This implies that v is labeled by the first of the variables in π^j. We choose an assignment c to the variables in $B^1, \dots, B^{j-1}$ for which v is reached from the source of G. Then

$$g[f_h]_{|c} = g(p_1, \dots, p_{j-1}, f_h(x_1^j, \dots, z_N^j), \dots, f_h(x_1^l, \dots, z_N^l)),$$

where $p_1, \dots, p_{j-1}$ are constants depending on c. Since v is reached, the value of $g[f_h]_{|c}$ essentially depends on the result of $f_h(x_1^j, \dots, z_N^j)$. Hence, there is an assignment d for the variables in

$B^{j+1}, \ldots, B^l$ so that $g[f_h]_{|c,d}$ is equal to $f_h(x_1^j, \ldots, z_N^j)$ or equal to $\overline{f_h(x_1^j, \ldots, z_N^j)}$. Since the variables in B^j are arranged adjacently, v is the source of an OBDD for $f_h(x_1^j, \ldots, z_N^j)$, where each sink may be an interior node of the blocks $B^{j+1}, \ldots, B^l$ or a sink of G.

It remains to show that for nodes v and v' of the B^j-layer with incoming edges from previous layers the OBDDs G_v and $G_{v'}$ starting at v and v' and consisting of B^j-nodes are disjoint. We show that the functions associated with the nodes of these OBDDs are different. Let the 0-sink and the 1-sink of G_v be the nodes a_0 and a_1, resp., and let the 0-sink and the 1-sink of $G_{v'}$ be b_0 and b_1, resp. (see also Fig. 8). These nodes may be sinks of G or may be interior nodes labeled by variables of $B^{j+1}, \ldots, B^l$.

(Insert Figure 8 here)

Case 1 $a_0 = b_0$ and $a_1 = b_1$. Then the functions associated with v and v' are equal. Since G is reduced, we have $v = v'$.

Case 2 $a_0 \neq b_0$ and $a_0 \neq b_1$. For each node w in G_v there is a path to the node a_0. Hence, from the function associated with w we may obtain the function associated with a_0 by replacing the B^j-variables that are tested at w or after w by appropriate constants. Since this does not hold for any node w' in $G_{v'}$, there are no mergings of nodes of G_v and $G_{v'}$.

Case 3 $a_0 = b_1$ and $a_1 = b_0$. Now we have the same situation as in Lemma 23. By this lemma there are no mergings between nodes of G_v and $G_{v'}$.

For all other cases the same arguments can be applied. □

In the following lemma we prove the above conjecture that OBDDs for $g[f]$ can be reordered to a blockwise ordering without increasing the size for the special case that $f = f_h$ for some h.

Lemma 25 Let the functions $g : \{0,1\}^l \to \{0,1\}$ and $h : \{0,1\}^n \to \{0,1\}$ be given. Let an OBDD G_h consisting of N nodes for h be given and let G be an OBDD for $g[f_h]$. Then an OBDD G^* for $g[f_h]$ with a blockwise variable ordering and $|G^*| \leq |G|$ can be constructed in polynomial time. Furthermore, for the variable ordering of G^* it holds that in each block the x-variables are arranged before all y- and z-variables of this block.

Proof First we reorder G in such a way that the y^j-variables are arranged adjacently. By the procedure that we present in the following the y^j-variables are moved to a position where some of the y^j-variables was arranged before. Hence, if the $y^{j'}$-variables are arranged adjacently, the same holds after reordering the y^j-variables. For this reason, we may successively apply the same procedure for all j in order to rearrange the y^j-variables and afterwards in order to rearrange the z^j-variables.

Now let j be fixed. Let $a(i,j)$ be the number of y_i^j-nodes in G. We choose k in such a way that $a(i,k)$ is equal to the minimum of all $a(i,j)$, $1 \leq i \leq N$. We replace all variables y_i^j except y_k^j by the constant 1. Afterwards, for each node w labeled by y_k^j we delete w and insert a copy of an OBDD computing the conjunction of the y^j-variables. All edges leading to w are redirected to the source of this copy. All edges leading to the c-sink ($c \in \{0,1\}$) of the copy lead to the c-successor of w.

By the first of these steps we obtain an OBDD for the subfunction of $g[f_h]$, where all y_i^j-variables except y_k^j are assigned to 1. By the second step we replace y_k^j by the conjunction of all y^j-variables. Therefore, we get an OBDD for $g[f_h]$. Before the reordering the number of y^j-nodes is $\sum_{i=1}^{N} a(i,j)$. After the reordering the number is $N \cdot a(k,j)$ which is not larger by the definition of k. Altogether, we obtain an OBDD for $g[f_h]$ which is not larger than G and in which for each j the y^j-variables (and z^j-variables, resp.) are arranged adjacently. We call the resulting OBDD again G.

Let π be the variable ordering of G and let π^j be the relative ordering of B^j-variables in π. We compute a reduced OBDD H for f_h and the variable ordering π^j. This can be done by choosing a suitable assignment for the variables in $B^1, \ldots, B^{j-1}, B^{j+1}, \ldots, B^l$ and performing this assignment in G. In particular, H is not larger than G and can be computed in polynomial time.

Let v be a node of G, which is labeled by the variable $t^* \in B^j$. Let $B^j(v)$ denote the set of variables that are contained in B^j and arranged before t^* in π. Let $A^j(v)$ denote the set of variables that are not contained in B^j and arranged before t^* in π.

We define a relation R between the B^j-nodes of G and the nodes of H. Let $(v, w) \in R$ iff

1. v is a node of G labeled by the variable $t^* \in B^j$,
2. w is a node of H labeled by the same variable t^*,
3. there is an assignment a to the variables in $A^j(v)$ and there is an assignment b to the variables in $B^j(v)$ so that in G the computation path for a and b leads from the source to v and that in H the computation path for b leads from the source to w.

We represent the relation R by edges from nodes of G to nodes of H. In order to avoid ambiguities we call such edges R-edges. We remark that R can be computed in polynomial time. This can be done by running simultaneously through G and H as it is done, e.g., by the algorithm for the operation apply (see Bryant (1986)). It holds that $(v, w) \in R$ iff these nodes are labeled by the same variable and reached simultaneously by the apply algorithm.

Lemma 26 Let t^* be a y^j- or z^j-variable, or let t^* be an x^j-variable which is arranged before all y^j- or before all z^j-variables. For all nodes of G that are labeled by t^* the fan-out with respect to the R-edges is at most 1.

Proof Assume that there is some node v in G, which is labeled by t^* and for which there are two different nodes w and w' in H, where $(v, w) \in R$ and $(v, w') \in R$. We shall obtain a contradiction by a cut-and-paste argument.

Let a and b be the assignments to the variables in $A^j(v)$ and $B^j(v)$, resp., which exist by the definition of R because of $(v, w) \in R$. Let a' and b' be defined similarly because of $(v, w') \in R$. Now consider the subfunction $g[f_h]_{|a,b}$. By the definition of a and b this function is equal to the function which is associated with v. We call this function F_v. Obviously F_v essentially depends on t^*. Hence, we can construct an assignment c to the variables that do not belong to B^j and that are arranged after t^* in π so that $F_{v|c}$ essentially depends on t^*. Since $F_{v|c} = g[f_h]_{|a,b,c}$, also $g[f_h]_{|a,c}$ essentially depends on t^*. By the assignments a and c all variables not contained in B^j are replaced by constants. Hence, $g[f_h]_{|a,c}$ is of the form $g(p_1, \ldots, p_{j-1}, f_h(x_1^j, \ldots, z_N^j), p_{j+1}, \ldots, p_l)$, where the constants p_i depend on a and c. Such a subfunction of $g[f_h]$ is equal to some constant function or to $f_h(x_1^j, \ldots, z_N^j)$ or $\overline{f_h(x_1^j, \ldots, z_N^j)}$. Since the subfunction $g[f_h]_{|a,c}$ essentially depends on t^*, it cannot be a constant function. Hence $g[f_h]_{|a,c} = f_h(x_1^j, \ldots, z_N^j)$ or $g[f_h]_{|a,c} = \overline{f_h(x_1^j, \ldots, z_N^j)}$.

Since $F_{v|c} = g[f_h]_{|a',b',c}$, we obtain by the same arguments that $g[f_h]_{|a',c} = f_h(x_1^j, \ldots, z_N^j)$ or $g[f_h]_{|a',c} = \overline{f_h(x_1^j, \ldots, z_N^j)}$. Hence, either $g[f_h]_{|a,c} = g[f_h]_{|a',c}$ or $g[f_h]_{|a,c} = \overline{g[f_h]_{|a',c}}$

Case A $g[f_h]_{|a,c} = g[f_h]_{|a',c}$.

From G we obtain two OBDDs for $g[f_h]_{|a,c}$ by performing the assignments a, c and a', c, resp. After reduction these OBDDs are isomorphic. Since in the bottom of the OBDDs the same assignment c is performed, the same function is associated with the copies of v in both OBDDs. This means that we reach the same node v for the assignments b and b'. On the other hand in the (reduced) OBDD H we reach different nodes for b and b'. Since H computes $g[f_h]_{|a,c}$ or $\overline{g[f_h]_{|a,c}}$, this is a contradiction.

Case B $g[f_h]_{|a,c} = \overline{g[f_h]_{|a',c}}$.

Again we compute OBDDs for $g[f_h]_{|a,c}$ and $g[f_h]_{|a',c}$ by performing the assignments a, c and a', c, resp. We obtain OBDDs for the functions f_h and $\overline{f_h}$. Since in the bottom the same assignment c was performed, the copies of v are associated with the same function, i.e. they can be merged. Remember that v is labeled by a y^j- or z^j-variable or by an x^j-variable after which all y^j- or all z^j-variables are arranged. Hence, we obtain a contradiction because by Lemma 23 there are no mergings between such interior nodes of OBDDs for f_h and $\overline{f_h}$. This completes the proof of Lemma 26. □

Now we describe how to perform the reordering of G so that the variables of B^j are arranged adjacently. For all nodes of H we compute the fan-in with respect to the R-edges. Let w be a node of H with the minimum fan-in r. Let w be labeled by t. We choose an assignment b to the variables in $B^j - \{t\}$. The assignment for the variables arranged before t is chosen so that in H the node w is reached. Then the resulting subfunction of f_h essentially depends on t. The variables arranged after t are assigned in such a way that also the subfunction $f_{h|b}$ essentially depends on t. Then this subfunction is equal to t or to $\overline{t}$.

Now we perform in G the assignment b of the variables in $B^j - \{t\}$ and apply the reduction rules. We obtain an OBDD G', which contains at most r nodes labeled by t, because only those nodes v labeled by t may survive these operations for which $(v, w) \in R$. The function represented by G' is $g[f_h]_{|b} = g(\ldots, f_h(x_1^{j-1}, \ldots, z_N^{j-1}), f_{h|b}, f_h(x_1^{j+1}, \ldots, z_N^{j+1}), \ldots)$. We distinguish two cases.

Case 1 The last N variables in π^j are $y_1^j, \ldots, y_N^j$ or $z_1^j, \ldots, z_N^j$.

We reorder H so that all x-variables are arranged before all y- and all z-variables. By Lemma 21 the resulting OBDD H' is not larger than H. We replace each node v in G' that is labeled by t by a copy H'_v of H'. This means that each edge leading to v is redirected to the source of H'_v. If $f_{h|b} = t$ then each edge leading to the c-sink of H'_v is redirected to the c-successor of v ($c \in \{0, 1\}$). If $f_{h|b} = \overline{t}$ then each edge leading to the c-sink of H'_v is redirected to the $\overline{c}$-successor of v. Finally, the resulting OBDD is reduced and we obtain an OBDD G''.

Obviously, G'' represents the function $g[f_h]$, because in $g[f_h]_{|b}$ we replaced $f_{h|b}$ by f_h. It remains to show that G'' is not larger than G. First we note that the number of nodes labeled by variables not contained in B^j does not increase by any of the performed operations. Now consider the number of nodes labeled by B^j-variables. Since the fan-out (with respect to the R-edges) of each B^j-node in G is at most 1, the number of R-edges is a lower bound on the number of B^j-nodes in G. Since the fan-in of each node of H with respect to the R-edges is at least r, there are at least $r|H|$ nodes labeled by B^j-variables in G. On the other hand, r copies of H' are inserted in G. These copies have $r|H'| \leq r|H|$ nodes. Also by the reduction the OBDD cannot become larger.

Case 2 The last variable in π^j is an x^j-variable.

We choose for H' the OBDD for f_h that we obtain from the given OBDD G_h for h by replacing the sinks by OBDDs for the conjunctions of the y^j- and z^j-variables, resp., as described after the definition of f_h. Remember that $|H'| = 3N$. We perform the same operations as in Case 1 for this choice of H'. By the same arguments as in Case 1 we obtain an OBDD G'' for $g[f_h]$. Again the number of nodes that are not labeled by B^j-variables does not become larger. Hence, we only have to show that the number of B^j-nodes does not increase.

First we prove a lower bound on the number of nodes labeled by y^j- and z^j-variables in G. Since for such nodes the fan-out is bounded by 1, the number of R-edges leading to y^j- and z^j-nodes of H is a lower bound for the number of y^j- and z^j-nodes in G. By Lemma 22 there are at least $3N$ nodes labeled by y^j- and z^j-variables in H. Hence, there are at least $3Nr$ such nodes in G. In G'' at most r copies of H' are inserted. Hence, at most $|H'|r = 3Nr$ nodes labeled by B^j-variables are inserted and the number of B^j-nodes in G'' is not larger than the number of B^j-nodes in G. □

6.2 Construction of an Approximation Scheme for MinOBDD*

In this section we prove the following lemma.

Lemma 27 Let $c > 1$ be some constant. If there is a polynomial time approximation algorithm for MinOBDD with the performance ratio c, then there is a polynomial time approximation scheme for MinOBDD*.

Obviously, Theorem 1 follows directly from Lemma 27 and Theorem 5.

Proof of Lemma 27 Let A be a polynomial time approximation algorithm for MinOBDD with the performance ratio c. An instance of the polynomial time approximation scheme for MinOBDD* consists of a function h and a parameter $\varepsilon > 0$. Hence, the algorithm has to achieve the performance ratio $1 + \varepsilon$. We may assume that h is given by a reduced OBDD G_h, which consists of N nodes. Let h be defined on n variables $x_1, \ldots, x_n$. If h does not essentially depend on all these variables, we consider h to be defined on a smaller set of variables. Since the polynomial time approximation scheme may behave arbitrarily on instances not fulfilling the promise of MinOBDD*, we may assume that $OPT(h) \geq N/2$. The algorithm works in the following way.

1. We choose $k = \left\lceil \frac{5 \log c}{\log(1+\varepsilon)} \right\rceil$.
2. Let $y_1, \ldots, y_N, z_1, \ldots, z_N$ be new variables. We construct an OBDD H for f_h and the variable ordering that we obtain by concatenating the variable ordering of G_h and $y_1, \ldots, y_N, z_1, \ldots, z_N$. Then H consists of $3N$ interior nodes.
3. We compute an OBDD $H^{(k)}$ for the function f_h^k. We start with the OBDD H and replace each node v by a copy H_v of H. For nodes v and w labeled by the same variable the copies H_v and H_w have the same set of variables. On the other hand, if v and w are labeled by different variables, the copies have disjoint sets of variables. This process is iterated for $k - 1$ times in order to obtain $H^{(k)}$. Then $|H^{(k)}| = |H|^k$.
4. We apply A on $H^{(k)}$ and obtain an OBDD H_A for f_h^k.

5. Since $f_h^k = f_h^{k-1}[f_h]$, we may choose $g = f_h^{k-1}$ and apply the polynomial time algorithm from Lemma 25 on H_A. We obtain an OBDD H_1 which by Lemma 24 consists of disjoint copies of OBDDs for f_h. There are $(2N+n)^{k-1}$ blocks. In H_1 we search in the variable orderings of all these blocks for the best variable ordering π^*. If we reorder all blocks to this variable ordering, the size of H_1 does not become larger.

   In H_1 we replace each of the disjoint copies of the OBDDs for f_h by an OBDD-node. In this way we may obtain an OBDD for f_h^{k-1}. Now we can apply the algorithm of Lemma 25 on this OBDD, search for the best variable ordering and so on. This can be iterated until we obtain an OBDD for f_h^1. Let π^* be the best variable ordering found in all these iterations. We obtain π from π^* by deleting the y- and z-variables in π^*. The output of the algorithm is π.

The run time of this algorithm is bounded by some polynomial since k does not depend on the length of the input. It remains to show that the performance ratio is bounded by $1+\varepsilon$. Let G_π be the OBDD for h and π. Then $|G_\pi|$ is the value of the output. Let H^* be an OBDD for f_h and π^*. Then $|G_\pi| \leq |H^*| - 2N$. It holds that $|H^*|^k \leq |H_A|$ because we get a smaller OBDD for f_h^k if we replace the variable ordering of some block by a better variable ordering. Hence, $|G_\pi| \leq |H_A|^{1/k} - 2N$. Since c is the performance ratio of A, we know $|H_A| \leq c \cdot OPT(H^{(k)})$. Hence, $|G_\pi| \leq c^{1/k} \cdot OPT(H^{(k)})^{1/k} - 2N$.

Now we prove $OPT(H^{(k)}) \leq OPT(H)^k$. We start with an OBDD G for f_h and an optimal variable ordering. This OBDD has size $OPT(H)$. By the procedure outlined in Step 3 we may compute an OBDD for f_h^k of size $OPT(H)^k$. This implies the claimed inequality. Hence, $|G_\pi| \leq c^{1/k} \cdot OPT(H) - 2N$. By Lemma 21 we have $OPT(H) = OPT(h) + 2N$. Hence, $|G_\pi| \leq (c^{1/k} - 1) \cdot 2N + c^{1/k} \cdot OPT(h)$. Since the algorithm has to work correctly only on instances fulfilling the promise, we may assume $N \leq 2OPT(h)$. Hence, $|G_\pi| \leq (5c^{1/k} - 4)OPT(h) \leq c^{5/k}OPT(h) \leq (1+\varepsilon)OPT(h)$. The last inequality follows from the definition of k. Hence, we have proved that the algorithm achieves a performance ratio of $1+\varepsilon$ on all instances fulfilling the promise, i.e. it is a polynomial time approximation scheme for MinOBDD*. □

Acknowledgment

This work was inspired by the presentations and discussions at the GI Research Seminar on proof verification and approximation algorithms in Dagstuhl in April 1997. I thank Clemens Gröpl and Martin Sauerhoff for fruitful discussions on the results of Section 6.

References

Bollig, B., Löbbing, M. and Wegener, I. (1995), Simulated annealing to improve variable orderings for OBDDs, *in* "Proc. of International Workshop on Logic Synthesis IWLS," pp. 5.1–5.10.

Bollig, B., Löbbing, M. and Wegener, I. (1996), On the effect of local changes in the variable ordering of ordered decision diagrams, *Information Processing Letters* 59, 233–239.

Bollig, B. and Wegener, I. (1996), Improving the variable ordering of OBDDs is NP-complete, *IEEE Transactions on Computers* 45, 993–1002.

Brace, K.S., Rudell, R.L. and Bryant, R.E. (1990), Efficient implementation of a BDD package, *in* "Proc. of 27th Design Automation Conference DAC," pp. 40–45.

Bryant, R.E. (1986), Graph-based algorithms for Boolean function manipulation, *IEEE Transactions on Computers* 35, 677–691.

Bryant, R.E. (1991), On the complexity of VLSI implementations and graph representations of Boolean functions with application to integer multiplication, *IEEE Transactions on Computers* 40, 205–213.

Bryant, R.E. (1992), Symbolic Boolean manipulation with ordered binary-decision diagrams, *ACM Computing Surveys* 24, 293–318.

Bryant, R.E. (1995), Binary decision diagrams and beyond: enabling technologies for formal verification, *in* "Proc. of International Conference on Computer-Aided Design ICCAD," pp. 236–243.

Bublitz, S., Schürfeld, U., Voigt, B. and Wegener, I. (1986), Properties of complexity measures for PRAMs and WRAMs, *Theoretical Computer Science* 48, 53–73.

Butler, K.M., Ross, D.E., Kapur, R. and Mercer, M.R. (1991), Heuristics to compute variable orderings for efficient manipulation of ordered binary decision diagrams, *in* "Proc. of 28th Design Automation Conference DAC," pp. 417–420.

Friedman, S.J. and Supowit, K.J. (1990), Finding the optimal variable ordering for binary decision diagrams, *IEEE Transactions on Computers* 39, 710–713.

Fujita, M., Fujisawa, H. and Kawato, N. (1988), Evaluation and improvements of Boolean comparison method based on binary decision diagrams, *in* "Proc. of International Conference on Computer-Aided Design ICCAD," pp. 2–5.

Garey, M.R. and Johnson, D.S. (1979), "Computers and Intractability: A Guide to the Theory of NP-Completeness," W.H. Freeman.

Håstad, J. (1997), Some optimal inapproximability results, *in* "Proc. of 29th Symposium on Theory of Computing STOC," pp. 1–10.

Ishiura, N., Sawada, H. and Yajima, S. (1991), Minimization of binary decision diagrams based on exchanges of variables, *in* "Proc. of International Conference on Computer-Aided Design ICCAD," pp. 472–475.

Jeong, S.-W., Kim, T.-S. and Somenzi, F. (1993), An efficient method for optimal BDD ordering computation, *in* "Proc. of International Conference on VLSI and CAD (ICVC)," pp. 252–256.

Krause, M. (1991), Lower bounds for depth-restricted branching programs, *Information and Computation* 91, 1–14.

Malik, S., Wang, A.R., Brayton, R.K. and Sangiovanni-Vincentelli, A. (1988), Logic verification using binary decision diagrams in a logic synthesis environment, *in* "Proc. of International Conference on Computer-Aided Design ICCAD," pp. 6–9.

Meinel, C. and Slobodová, A. (1994), On the complexity of constructing optimal ordered binary decision diagrams, *in* "Proc. of International Symposium on Mathematical Foundations of Computer Science MFCS," pp. 515–524.

Minato, S., Ishiura, N. and Yajima, S. (1990), Shared binary decision diagram with attributed edges for efficient Boolean function manipulation, *in* "Proc. of 27th Design Automation Conference DAC," pp. 52–57.

Motwani, R. and Raghavan, P. (1995), “Randomized Algorithms,” Cambridge University Press.

Panda, S. and Somenzi, F. (1995), Who are the variables in your neighborhood, *in* “Proc. of International Conference on Computer-Aided Design ICCAD,” pp. 74–77.

Papadimitriou, C.H. and Yannakakis, M. (1991), Optimization, approximation, and complexity classes, *Journal of Computer and System Sciences* 43, 425–440.

Rudell, R. (1993), Dynamic variable ordering for ordered binary decision diagrams, *in* “Proc. of International Conference on Computer-Aided Design ICCAD,” pp. 42–47.

Sauerhoff, M., Wegener, I. and Werchner, R. (1996), Optimal ordered binary decision diagrams for fanout-free circuits, *in* “Proc. of the Synthesis and System Integration of Mixed Technologies SASIMI,” pp. 197–204.

Savický, P. and Wegener, I. (1997), Efficient algorithms for the transformation between different types of binary decision diagrams, *Acta Informatica* 34, 245–256.

Sieling, D. (1996), Variable orderings and the size of OBDDs for partially symmetric Boolean functions, *in* “Proc. of the Synthesis and System Integration of Mixed Technologies SASIMI,” pp. 189–196, submitted to *Random Structures & Algorithms*.

Sieling, D. (1998), On the existence of polynomial time approximation schemes for OBDD minimization, *to appear in* “Proc. of 15th Symposium on Theoretical Aspects of Computer Science STACS.”

Sieling, D. and Wegener, I. (1993), NC-algorithms for operations on binary decision diagrams, *Parallel Processing Letters* 3, 3–12.

Tani, S., Hamaguchi, K. and Yajima, S. (1993), The complexity of the optimal variable ordering problems of shared binary decision diagrams, *in* “Proc. of 4th International Symposium on Algorithms and Computation ISAAC,” pp. 389–398.

Wegener, I. (1994), Efficient data structures for Boolean functions, *Discrete Mathematics* 136, 347–372.

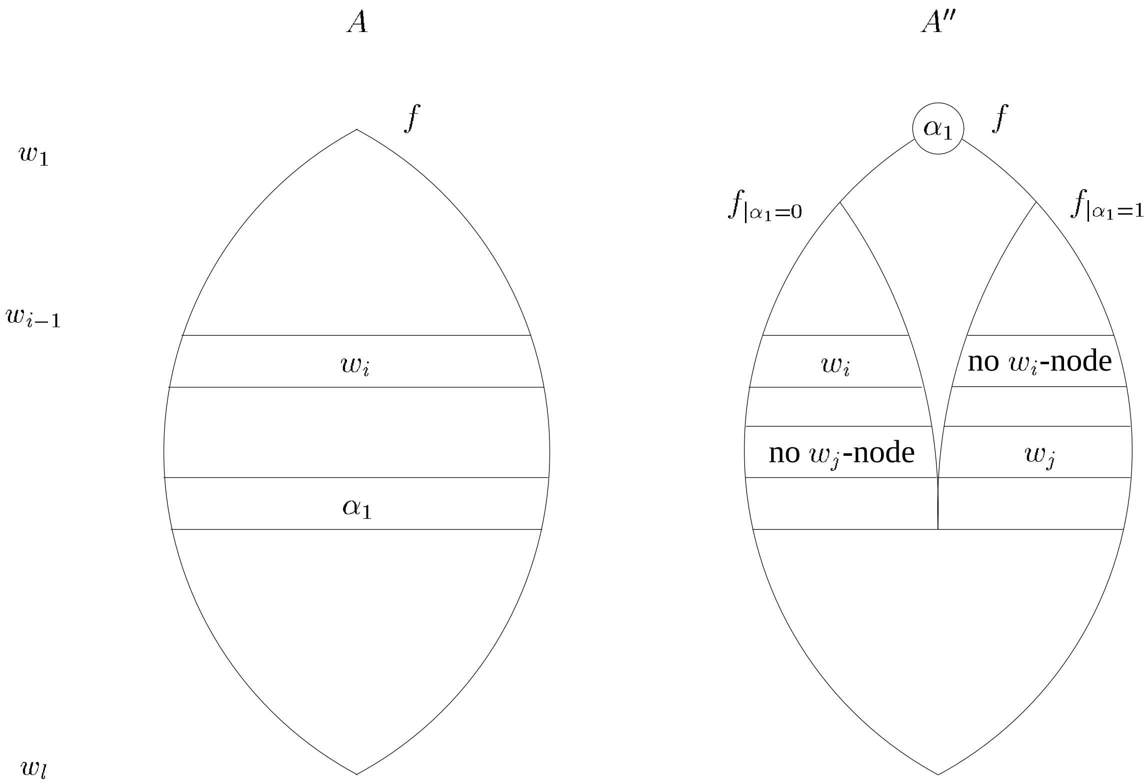

Figure 1: The situation before and after moving α_1 to the top.

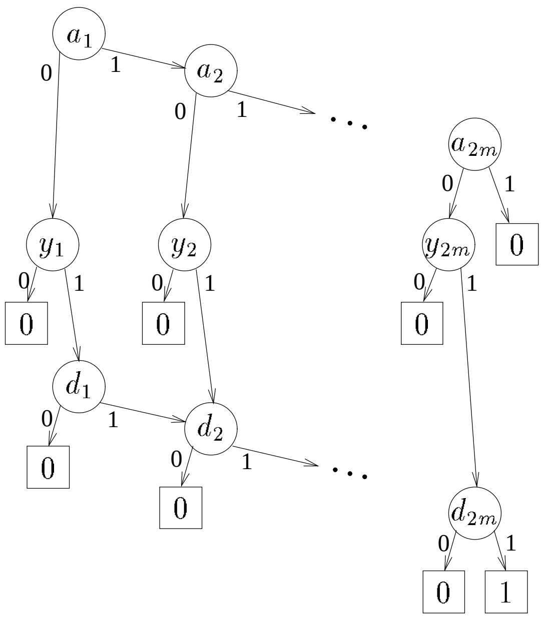

Figure 2: An OBDD for g^* and the variable ordering $a_1, \ldots, a_{2m}, y_1, \ldots, y_{2m}, d_1, \ldots, d_{2m}$.

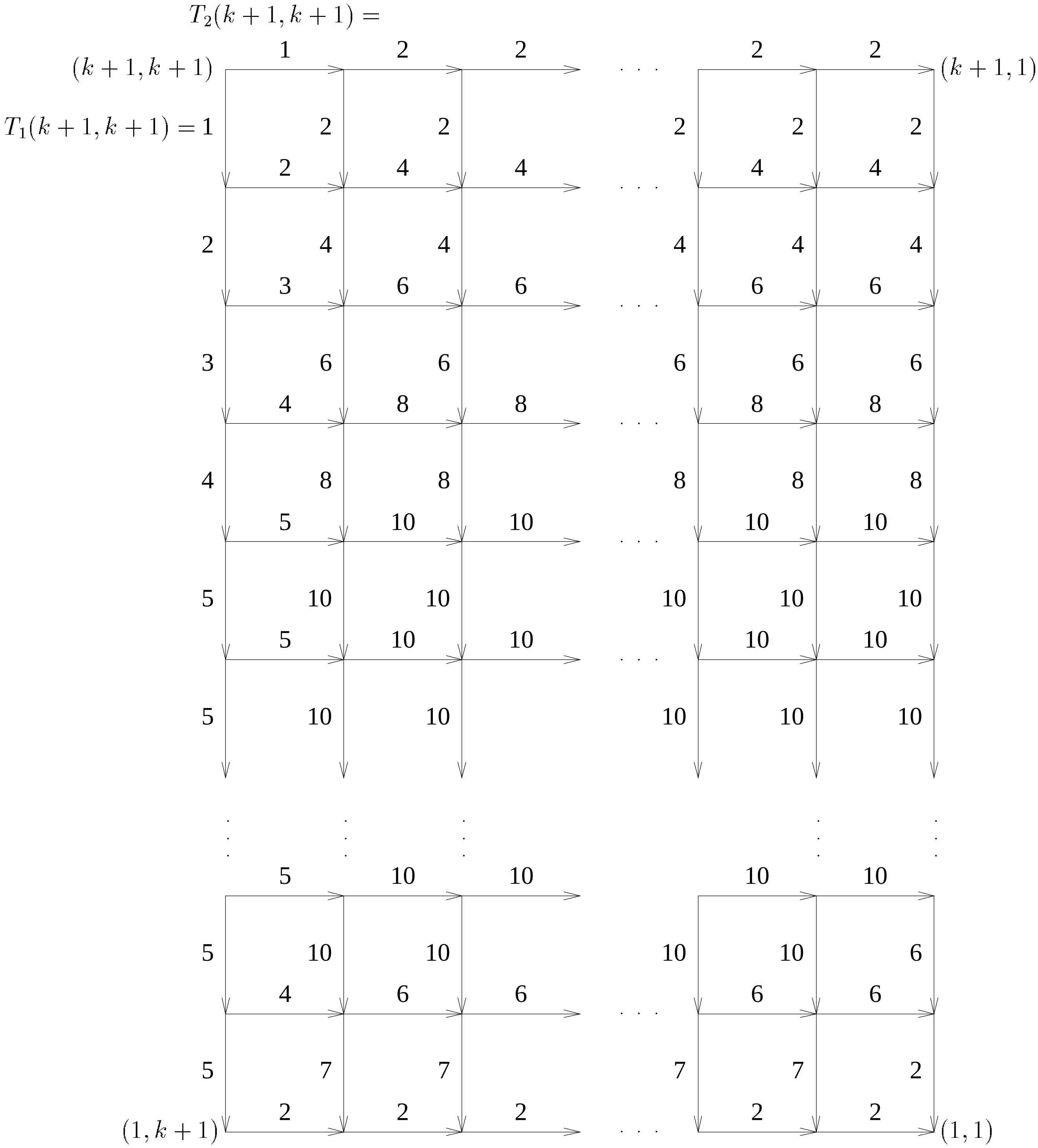

Figure 3: The grid graph for $h(p_1,\ldots,p_{2m},q_1,\ldots,q_{2m})$.

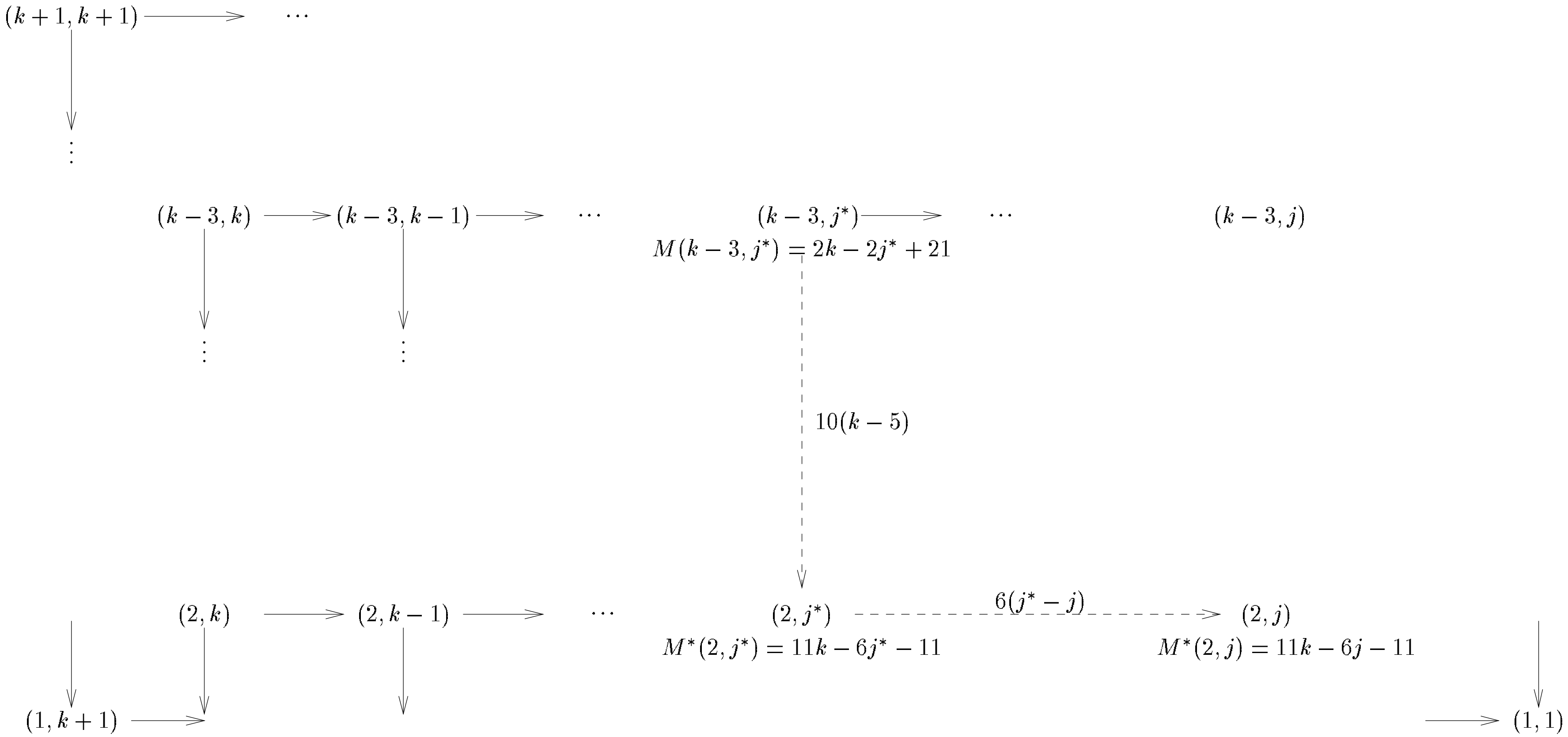

Figure 4: The grid graph of the function h. The Figure shows the situation when computing the length $M(2, j)$ of a shortest path from the source $(k+1, k+1)$ to the node $(2, j)$.

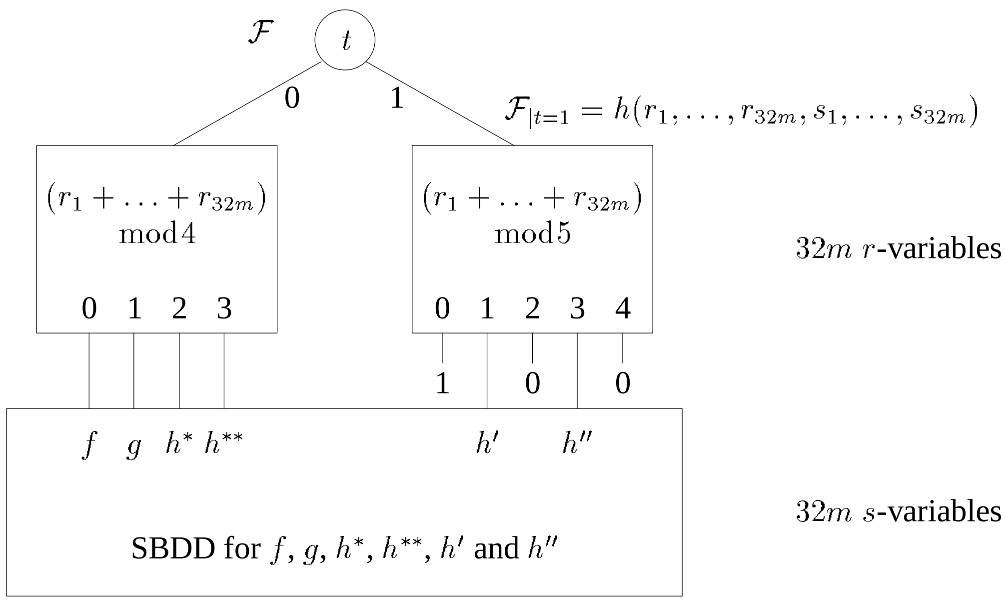

Figure 5: An OBDD for $\mathcal{F}$ and the (nonoptimal) variable ordering t, r-variables, s-variables.

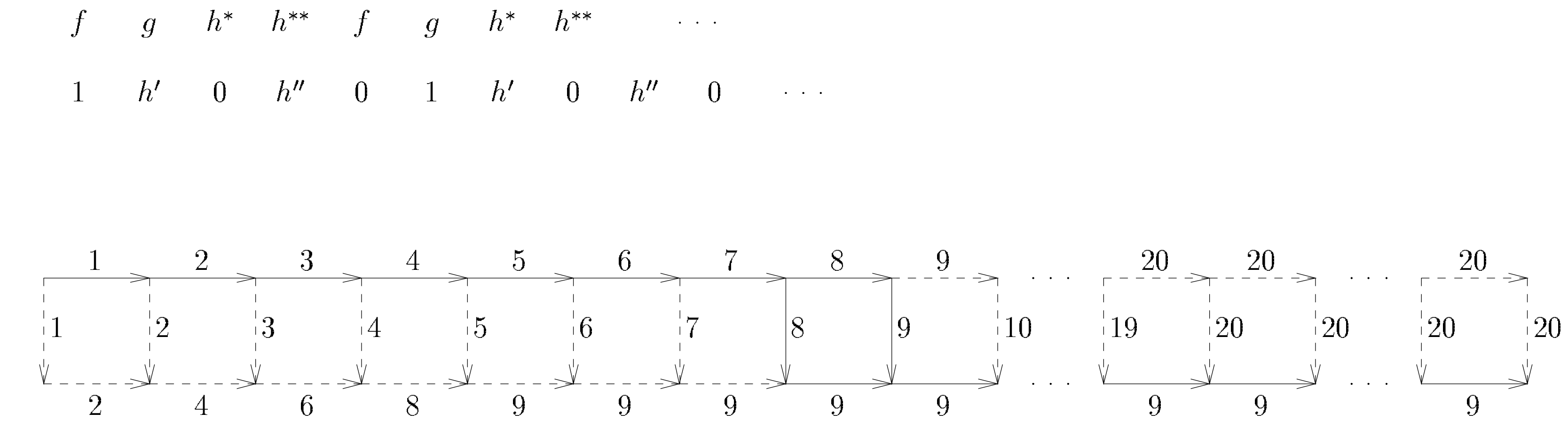

Figure 6: The value matrix and the grid graph for the variant of the function $\mathcal{F}$ that maps assignments of t and the r-variables to the set $\{f, g, h^*, h^{**}, h', h'', 0, 1\}$. The symmetry sets of this function are $\{t\}$ and $\{r_1, \ldots, r_{32m}\}$.

1	0	0	0	1	0	0	0	$\cdots$		
1	1	0	0	0	1	1	0	0	0	$\cdots$

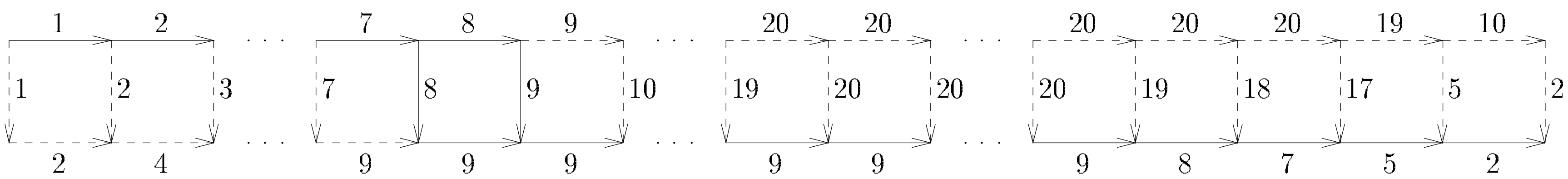

Figure 7: The value matrix and the grid graph of $\mathcal{F}_{|A_1}$. The symmetry sets of this function are $\{t\}$ and $\{r_1, \ldots, r_{32m}\}$.

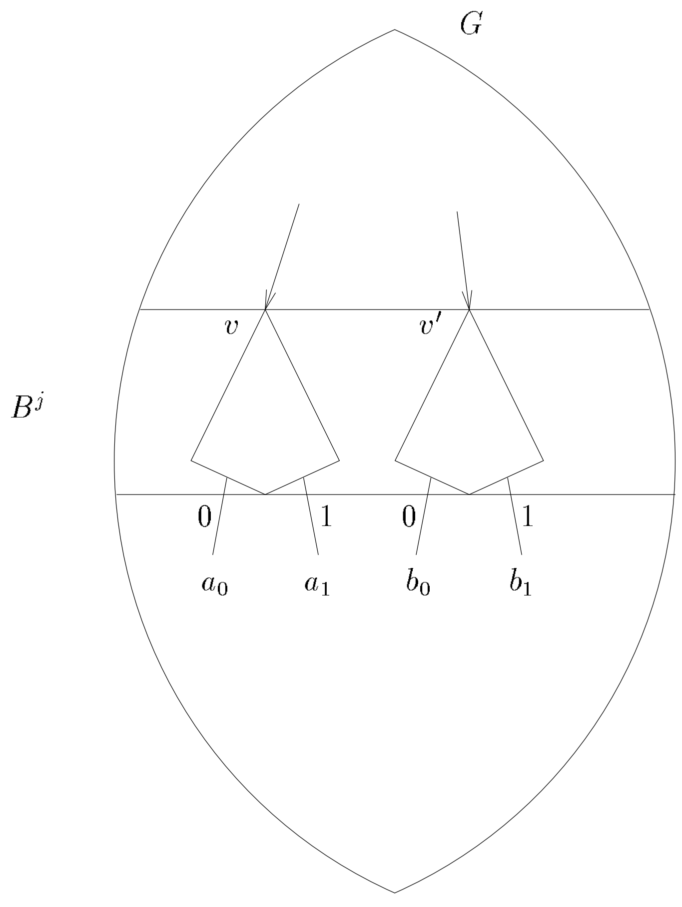

Figure 8: OBDDs for f_h in the B^j-layer.

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'