

Implicit Flow Maximization on Grid Networks

Daniel Sawitzki*

University of Dortmund, Computer Science 2

D-44221 Dortmund, Germany

`daniel.sawitzki@cs.uni-dortmund.de`

January 7, 2004

Abstract

The implicit flow maximization algorithm of Sawitzki [2] computes a maximum flow on an implicitly given 0–1 network, represented by OBDDs. If the maximum flow value is constant w. r. t. the network size, the algorithm performs $O(\log^2 |V|)$ OBDD-operations. This is the case for $(2^k + 1) \times (2^k + 1)$ -grid networks, whose maximum flow value is 2. It is shown that all OBDDs which are generated during the flow maximization on $(2^k + 1) \times (2^k + 1)$ -grid networks have size $O(k)$. This directly implies an over-all runtime of $O(k^4)$ and a space usage of $O(k^2)$. Further considerations of the OBDD structures improve the runtime result to $O(k^3)$. In experiments, the algorithm beats the implicit method of Hachtel and Somenzi [1] for $k \geq 13$.

1 Introduction

The implicit flow maximization algorithm of Sawitzki [2] computes a maximum flow on an implicitly given 0–1 network $N = (V, E, s, t)$ using the technique of iterative squaring. Hence, it is called IS-algorithm in the following. Its input consists of three characteristic functions $E(x, y)$, $s(x)$, and $t(x)$, which are represented by OBDDs. Therefore, the nodes of V are encoded by binary numbers of length $n := \lceil \log |V| \rceil$.

The IS-algorithm belongs to the class of layered-network methods, and has to construct blocking-flows on layered-networks. It prevents breadth-first searches by using iterative squaring, i.e., it iteratively constructs longer paths by shorter

*Supported by the Deutsche Forschungsgemeinschaft (DFG) as part of the Research Cluster “Algorithms on Large and Complex Networks” (1126).

ones. Because their length is doubled in each iteration, a polylogarithmic number of OBDD-operations may suffice to compute a maximum flow. Hence, the algorithm overcomes the dependence on layered-network depths.

The flow is computed during so-called *sweeps*. For the number \mathcal{S} of performed sweeps, a flow maximization takes $O(n^2\mathcal{S})$ OBDD-operations. On networks with a constant maximum flow value, also a constant number of sweeps suffices. If, moreover, all participating OBDDs have size $O(n)$, each OBDD-operation can be performed in time $O(n^2)$, leading to the over-all runtime of $O(n^4)$. We will prove that this is the case for $(2^k + 1) \times (2^k + 1)$ -grid networks, $k \in \mathbb{N}_0$, if an appropriate node numbering is used. Moreover, in the case of grid networks, the algorithm exclusively generates functions which have complete OBDDs of constant width. This further structural property guarantees a runtime of $O(k)$ for each OBDD-operation, leading to an improved over-all runtime of $O(k^3)$. Additionally, a bound of $O(k^2)$ on the space usage is shown. In contrast, the only other implicit maximum flow algorithm so far of Hachtel and Somenzi [1] needs $\Omega(|V|^{1/2} \log |V|)$ operations. In experiments, the space usage of Hachtel and Somenzi's method is beaten for $k \geq 12$, while its absolute runtime is beaten for $k \geq 15$.

Obviously, there is a maximum flow $F_{\max} \subseteq E$ on grid networks whose OBDD has size $O(n)$ and which can be computed in the same time. Anyhow, it is usual to analyze heuristic methods on simple inputs, although there is an easier way to find optimal solutions for them. One wants to show that there are inputs on which the heuristic is efficient although it does not use special information about it. The grid networks considered in this paper have no direct practical relevance. Nevertheless, grid structures are of interest in application areas like circuit design, for which hopefully similar runtime results hold.

The runtime analysis will use results on multivariate threshold functions and their quantifications of Woelfel [4, 5]. An important part of the analysis will be to show for a variety of characteristic functions that they are composable by $O(1)$ threshold functions, which implies that their OBDDs have size $O(k)$.

The paper is organized as follows: At first, the considered grid networks are formally introduced in Section 2. Their implicit representation is discussed in Section 3. Section 4 introduces the class of multivariate threshold functions as well as results on their OBDD-size. Section 5 is dedicated to the priority function used by the flow maximization. Section 6 gives an overview of the analysis and states its main Theorems 1 and 2. While the latter is proven directly, the proof of Theorem 1 is found at the end of Section 8, because it uses lemmas supplied by Sections 7 and 8. The latter investigate the functions occurring during the layered-network construction respectively during the blocking-flow construction. In addition to these theoretical results, Section 9 presents experimental results of both Sawitzki's algorithm and Hachtel and Somenzi's algorithm [1] on grid networks. Finally, Section 10 gives conclusions, and hints to further results.

This work is an extension to [2], which describes the IS-algorithm and sketches

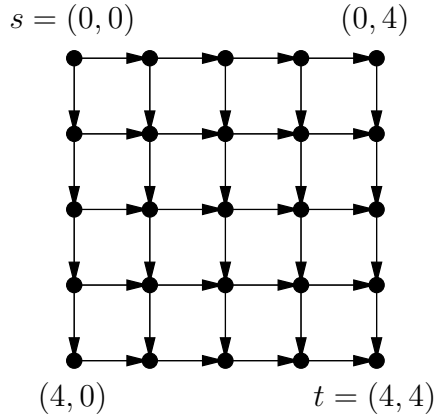


Figure 1: The 5×5 -grid network.

the results for grid networks. Because we have to show bounds on the OBDD-sizes for every step of the algorithm, it is strongly referred to [2] throughout this paper. Therefore, we recommend the reading of this algorithm description in order to understand the proof.

2 Grid Networks

We now formally define $\mathcal{N} \times \mathcal{N}$ -grid networks:

Definition 1 An $\mathcal{N} \times \mathcal{N}$ -grid network, $\mathcal{N} \in \mathbb{N}_{\geq 2}$, is a network $N = (V, E, s, t)$ on the nodes $V = \{(i, j) \mid 0 \leq i, j < \mathcal{N}\}$ that contains the edges

$$E = \left\{ ((i, j), (k, \ell)) \in V^2 \mid k - i + \ell - j = 1, i \leq k, j \leq \ell \right\}.$$

It has a source node $s = (0, 0)$ and a terminal node $t = (\mathcal{N} - 1, \mathcal{N} - 1)$.

The network can be considered as an $\mathcal{N} \times \mathcal{N}$ -matrix of nodes, with node (i, j) placed in row i and column j . Edges are directed from left to right between horizontal neighbors as well as downwards between vertical neighbors. Source s resides in the upper left corner and terminal t in the lower right. The maximum flow value $\text{val}(f_{\max}) = 2$ is independent from \mathcal{N} . Figure 1 shows the 5×5 -grid network.

3 Implicit Network Representation

In the analysis of the IS-algorithm [2] on $\mathcal{N} \times \mathcal{N}$ -grid networks we assume $\mathcal{N} = 2^k + 1$, $k \in \mathbb{N}_0$. Hence, they consist of $2^{2k} + 2^{k+1} + 1$ nodes, which we encode by

$n := 2k + 2$ encoding OBDD variables. Although $2k + 1 = \lceil \log |V| \rceil$ variables would suffice, we use an even count of variables in order to represent both node indices symmetrically. We will now investigate the network's node numbering.

Consider a set $S \subseteq V^d$ and its implicit OBDD-representation $S(x^1, \dots, x^d)$ defined on $X_{d,n} := \{x_b^a \mid 1 \leq a \leq d, 0 \leq b < n\}$. In [2], the described maximum flow algorithm is assumed to use an interleaved variable ordering $\pi_{d,n}$ that intermixes the binary numbers of the d node arguments of S . When analyzing the algorithm on grid networks, we also need the row index i and the column index j of each node argument x^a , $1 \leq a \leq d$, to be interleaved. Therefore, $(i, j) \in V$ gets the binary number $i_{m-1}j_{m-1} \dots i_0j_0$ for $m := n/2 = k + 1$.

Note that this node numbering is not chosen purely artificial in order to make the runtime analysis feasible: Both the identification of nodes by the pair (i, j) of their row index i and column index j as well as the interleaved ordering of their bits represents a reasonable and straight way to encode the elements of V .

4 Multivariate Threshold Functions

In [5], Woelfel introduces the class of *multivariate threshold functions*. In order to define them formally, we denote the class of all Boolean functions $f: \{0, 1\}^n \rightarrow \{0, 1\}$ by B_n . Furthermore, $|x|$ denotes the binary value of $x \in \{0, 1\}^n$. We use the set $X_{k,n} := \{x_j^i \mid 1 \leq i \leq k, 0 \leq j < n\}$ of Boolean variables.

Definition 2 *A function $f \in B_{kn}$, $k, n \in \mathbb{N}$, defined on the variable set $X_{k,n}$ is called k -variate threshold function if there are weights $w_1, \dots, w_k \in \mathbb{Z}$ and a threshold $T \in \mathbb{Z}$ such that*

$$f(x^1, \dots, x^k) = \left(\sum_{i=1}^k w_i \cdot |x^i| \geq T \right).$$

The maximum absolute weight of f is defined as $w(f) := \max\{|w_1|, \dots, |w_k|\}$. The class of k -variate threshold functions with maximum absolute weight w defined on $X_{k,n}$ is denoted by $\mathbb{T}_{k,n}^w$.

k -variate threshold functions with maximum weight w have the interesting property that they have OBDDs of width $O(kw)$ and size $O(k^2wn)$.

Furthermore, Woelfel [5] defines decomposable functions.

Definition 3 *Let $f \in B_n$ and \mathcal{C} be a class of functions defined on the variable set $X_{k,n}$. f can be decomposed into m functions of \mathcal{C} if there exists a formula F on m variables and $f_1, \dots, f_m \in \mathcal{C}$ such that $f = F(f_1, \dots, f_m)$. The class of functions decomposable into m functions of \mathcal{C} is denoted by $D[\mathcal{C}, m]$.*

Hence, $D[\mathbb{T}_{k,n}^w, m]$ denotes the class of functions decomposable into m k -variate threshold functions with maximum absolute weight w . From [5, Lemma 3] it

follows that functions $f \in D\left[\mathbb{T}_{O(1),n}^{O(1)}, O(1)\right]$ have OBDDs of width $O(1)$ and size $O(n)$.

Corollary 1 *Both functions $f \in B_{k,n}$, $k, n \in \mathbb{N}$, and \bar{f} defined on the variable set $X_{k,n}$ as*

$$f(x^1, \dots, x^k) = \left(\sum_{i=1}^k w_i \cdot |x^i| \bowtie T \right)$$

for $w_1, \dots, w_k, T \in \mathbb{Z}$, $\bowtie \in \{<, \leq, >, =\}$, and $w := \max\{|w_1|, \dots, |w_k|\}$ are in $D\left[\mathbb{T}_{k,n}^w, O(1)\right]$.

In order to apply these results, we now consider the variable ordering of a function that receives d encodings of grid nodes as arguments. For the s th bit $x_s^{r,\ell}$ ($x_s^{c,\ell}$) of the ℓ th node argument's row (column) index, this node numbering together with the variable ordering $\pi_{d,n}$ results in the permutation

$$(x_0^{r,1}, \dots, x_0^{r,d}, x_0^{c,1}, \dots, x_0^{c,d}, \dots, x_{m-1}^{r,1}, \dots, x_{m-1}^{r,d}, x_{m-1}^{c,1}, \dots, x_{m-1}^{c,d}).$$

In the following, we will alternatively interpret this as the interleaved ordering of $2d$ arguments $\{x^{r,\ell}, x^{c,\ell} \mid 1 \leq \ell \leq d\}$, each of length m . Thereby, it fulfills the ordering condition demanded in [5] from the k multiplicands x^1, \dots, x^k of the multivariate threshold functions.

We will now describe the grid's edge relation $E(x, y)$ as a composition of threshold functions, thereby showing $E \in D\left[\mathbb{T}_{4,m}^1, O(1)\right]$:

$$\begin{aligned} E(x, y) &= (y^r - x^r + y^c - x^c = 1) \\ &\quad \wedge (y^r - x^r \geq 0) \wedge (y^c - x^c \geq 0) \\ &\quad \wedge (x^r \leq \mathcal{N} - 1) \wedge (x^c \leq \mathcal{N} - 1) \wedge (y^r \leq \mathcal{N} - 1) \wedge (y^c \leq \mathcal{N} - 1). \end{aligned}$$

Only those nodes lying in the $\mathcal{N} \times \mathcal{N}$ -square are touched by edges. Moreover, there is an edge between node x and y if and only if they are either horizontal adjacent or vertical adjacent.

5 Priority Function

The algorithm description in [2] does not constitute which priority function to use during the blocking-flow construction. In the following, we assume the use of the datum proximity Π^{dp} introduced in [1]. It is now defined formally in a generalized form, which may receive $d \in 2\mathbb{N}$ node arguments.

Definition 4 *Let $y^1, z^1, \dots, y^{d/2}, z^{d/2} \in \{0, 1\}^n$ be a sequence of grid node numbers, each consisting of a row part $y^{r,\ell} \in \{0, 1\}^m$ resp. $z^{r,\ell} \in \{0, 1\}^m$ and a column part $y^{c,\ell} \in \{0, 1\}^m$ resp. $z^{c,\ell} \in \{0, 1\}^m$. The datum proximity $\Pi^{\text{dp}} \in B_{dn}$ is the binary function with*

$$\Pi^{\text{dp}}(y^1, \dots, y^{d/2}, z^1, \dots, z^{d/2}) := |y^{r,1}y^{c,1} \dots y^{r,d/2}y^{c,d/2}| < |z^{r,1}z^{c,1} \dots z^{r,d/2}z^{c,d/2}|.$$

By rewriting this definition, we see that Π^{dp} can be composed of $O(d^2)$ threshold functions:

$$\begin{aligned}
& \Pi^{\text{dp}}(y^1, \dots, y^{d/2}, z^1, \dots, z^{d/2}) \\
&= (|y^{r,1}| < |z^{r,1}|) \\
&\quad \vee \left[(|y^{r,1}| = |z^{r,1}|) \wedge (|y^{c,1}| < |z^{c,1}|) \right] \\
&\quad \vee \left[(|y^{r,1}| = |z^{r,1}|) \wedge (|y^{c,1}| = |z^{c,1}|) \wedge (|y^{r,2}| < |z^{r,2}|) \right] \\
&\quad \vee \dots \\
&\quad \vee \left[(|y^{r,1}| = |z^{r,1}|) \wedge \dots \wedge (|y^{r,d/2}| = |z^{r,d/2}|) \wedge (|y^{c,d/2}| < |z^{c,d/2}|) \right].
\end{aligned}$$

The blocking-flow construction will only use priority functions with $d = 4$, so it is $\Pi^{\text{dp}} \in D\left[\mathbb{T}_{O(1),m}^1, O(1)\right]$.

6 Analysis

$\mathcal{N} \times \mathcal{N}$ -grid networks have a constant maximum flow value of 2. One phase suffices to compute a maximum flow F_{\max} , whose layered-network $U(x, y) = E(x, y)$ simply contains all edges of $E(x, y)$ as forward-edges. Therefore, the number \mathcal{S} of sweeps is at most 2, why $O(\log^2 |V|) = O(k^2)$ OBDD-operations are executed.

After the only phase, a sophisticated implementation terminates without starting another layered-network construction, because the actual flow value of 2 equals the out-degree of the source node s . This can be tested through counting the satisfying inputs of $E(s, y)$ and $F(s, y)$ in linear time w. r. t. the corresponding OBDD-sizes [3]. Nevertheless, it can also be shown that an unsuccessful layered-network construction is efficient.

In the following, we only focus on the first phase. We state Theorem 1 in order to bound the costs of each of its operations. Its proof is placed at the end of Section 8 because it uses Lemmas 3 and 4, which investigate the functions of the layered-network construction and blocking-flow computation separately.

Theorem 1 *Every function computed during the IS-algorithm on $(2^k + 1) \times (2^k + 1)$ -grid networks, $k \in \mathbb{N}_0$, has a complete OBDD of constant width.*

OBDDs of constant width defined on n variables have size $O(n)$. Hence, each binary operation takes time $O(n^2) = O(k^2)$. The costs besides the expensive OBDD-operations and their use to realize quantifications is of the same magnitude as these and therefore negligible. This directly leads to an over-all runtime of $O(k^4)$.

The following Lemma 1 which is proved in Appendix A involves the completeness property of the OBDDs, and thus enables a further improvement of the runtime result.

Lemma 1 *Let F and G minimal π -OBDDs of functions $f, g \in B_n$, $\pi \in \Sigma_n$, which have complete π -OBDDs \mathcal{F} and \mathcal{G} of constant width k . The binary synthesis [3] $F \otimes G$, $\otimes \in B_2$, generates a minimal π -OBDD H of size $O(n)$ in time and space $O(n)$ which also has a complete π -OBDD \mathcal{H} of constant width k^2 .*

Corollary 2 *Each of the $O(k^2)$ OBDD-operations takes time and space $O(n) = O(k)$ and the maximum flow is computed in over-all runtime $O(k^3)$. \square*

At next, we will consider the space usage on grid networks.

Theorem 2 *The execution of the IS-algorithm on $(2^k + 1) \times (2^k + 1)$ -grid networks, $k \in \mathbb{N}_0$, uses space $O(k^2)$.*

Proof. Due to Corollary 2, each OBDD-operation takes time and space $O(k)$. In [2] it is shown that $O(k^2\mathcal{S})$ OBDDs are computed, for the number \mathcal{S} of sweeps. Due to Theorem 1, each of them has size $O(n) = O(k)$. During the algorithm, only $O(k)$ functions have to be represented in the memory at the same time, because the intermediate OBDDs occurring within quantifications over node numbers may be discarded directly afterwards. For a constant number \mathcal{S} of sweeps in the case of grid networks, this leads to a total space usage of $O(k^2)$. \square

7 Layered-Network Construction

Before investigating the properties of computed functions in order to bound their OBDD-size, we state some basic facts on the layered-network structure:

Fact 1 *The $(2^k + 1) \times (2^k + 1)$ -grid network N corresponds both to the residual network A and the layered-network U of the only necessary phase of the flow maximization.*

Proof. The maximum flow value is 2. Every path from s to t is a shortest augmenting path, consisting only of forward-edges. Independent from the structure of a firstly constructed augmenting path in U , there is a second one. Therefore, every blocking-flow on U consists of two augmenting paths, which implies that one phase suffices. \square

Fact 2 *There is a path from node x to y (x - y -path) of length ℓ in N if and only if the differences of the row and the column indices of y and x are both not negative and their sum is exactly ℓ , formally:*

$$(y^r - x^r \geq 0) \wedge (y^c - x^c \geq 0) \wedge (y^r - x^r + y^c - x^c = \ell).$$

\square

Fact 3 *Because every s - x -path for $x \in V$ has length $x^r + x^c$, node $x \in V$ lies in layer $x^r + x^c$. \square*

Obviously, functions $f \in B_{dn}$ that represent only a constant number of node tuples of V^d are composable of $O(1)$ threshold functions of $\mathbb{T}_{2d,m}^1$. Before the first phase, it is $F(x, y) = 0$ and $A(x, y) = E(x, y)$, where we already know that $E \in D[\mathbb{T}_{4,m}^1, O(1)]$. The analysis of the further functions will be separated like in [2] by preprocessing, partitioning, and edge computation.

7.1 Preprocessing

During the preprocessing of the layered-network construction, the functions $PATH_k^{\leq}$, $PATH_k^=$, $PATH_k^{<}$, $SPATH_k$, and A_k are computed for $k \in \{0, \dots, k^*\}$, $k^* \leq \lceil \log |V| \rceil$. We will now show for each of them that it is in $D[\mathbb{T}_{O(1),m}^1, O(1)]$ for $k > 0$, while this is obvious for $k = 0$.

Functions $PATH_k^{\leq}(x, y)$: From Facts 1 and 2 we know that the length of an residual x - y -path is $y^r - x^r + y^c - x^c$ (if existing). Therefore, it holds

$$PATH_k^{\leq}(x, y) = (y^r - x^r \geq 0) \wedge (y^c - x^c \geq 0) \wedge (y^r - x^r + y^c - x^c \leq 2^k), \quad (1)$$

why it is $PATH_k^{\leq} \in D[\mathbb{T}_{4,m}^1, O(1)]$.

Functions $PATH_k^=(x, y)$: A slight modification of (1) yields the desired property $PATH_k^= \in D[\mathbb{T}_{4,m}^1, O(1)]$:

$$PATH_k^=(x, y) = (y^r - x^r \geq 0) \wedge (y^c - x^c \geq 0) \wedge (y^r - x^r + y^c - x^c = 2^k).$$

Functions $PATH_k^{<}(x, y)$: Again, it is a slight modification of (1) that yields the desired property $PATH_k^{<} \in D[\mathbb{T}_{4,m}^1, O(1)]$:

$$PATH_k^{<}(x, y) = (y^r - x^r \geq 0) \wedge (y^c - x^c \geq 0) \wedge (y^r - x^r + y^c - x^c < 2^k).$$

Functions $SPATH_k(x, y)$: Being composed of $PATH_{k-1}^=(x, y)$ and $PATH_{k-1}^{<}(x, y)$, it is also $SPATH_k \in D[\mathbb{T}_{4,m}^1, O(1)]$.

Functions $A_k(v, w, x, y)$: Edge (x, y) lies on a residual v - w -path of length 2^k if and only if $(x, y) \in E$, $PATH_k^=(v, w) = 1$, and both x and y are lying in the rectangle spanned by v and w . From the following expression of these conditions, it follows $A_k \in D[\mathbb{T}_{4,m}^1, O(1)]$:

$$\begin{aligned} A_k(v, w, x, y) &= E(x, y) \wedge PATH_k^=(v, w) \\ &\wedge (x^r \geq v^r) \wedge (x^r \leq w^r) \wedge (x^c \geq v^c) \wedge (x^c \leq w^c) \\ &\wedge (y^r \geq v^r) \wedge (y^r \leq w^r) \wedge (y^c \geq v^c) \wedge (y^c \leq w^c). \end{aligned}$$

7.2 Partitioning the layered-network

For the layered-network's partitioning it has just to be shown that in Case 3 of all recursion instances of the subalgorithm `findTerminal` the functions $R(x)$ and $Z(x)$ are in $D[\mathbb{T}_{2,m}^1, O(1)]$. Together with the base of induction $Z_0 \in D[\mathbb{T}_{2,m}^1, O(1)]$ for the first call to `findTerminal`, this guarantees also that all argument functions $R(x)$ and $Z(x)$ have the desired threshold property.

Function $Z'(x)$: The first i instances of `findTerminal` partitioned the node layers $0, \dots, \nu := \sum_{j=0}^{i-1} 2^{k_j}$ of U . Because U contains the complete residual network A (Fact 1), argument $Z(x)$ of the current instance `findTerminal`($R(x), Z(x), a$) corresponds to node layer ν . Through computing $Z'(x)$, `findTerminal` proceeds from ν by the next 2^{a-1} layers. Therefore, $Z'(x)$ corresponds to node layer $\nu + 2^{a-1} =: \nu'$ and due to Fact 3 it is $Z'(x) = \text{PATH}_{\nu'}^{\bar{=}}(s, x)$, which is in $D[\mathbb{T}_{2,m}^1, O(1)]$.

Function $R'(x)$: Analogously to $Z'(x)$, the argument $R(x)$ corresponds to all nodes of layers $0, \dots, \nu$ of U . Through computing $R'(x)$, the next 2^{a-1} node layers are added and due to Fact 3 it is $R'(x) = \text{PATH}_{\nu'}^{\leq}(s, x)$, which is in $D[\mathbb{T}_{2,m}^1, O(1)]$.

7.3 Computing the layered-network edges

In the final computation of the layered-network edges, we have to consider the functions $C_i(x, y)$, $C(x, y)$ as well as the reachability analysis on $C^R(x, y)$.

Functions $C_i(x, y)$: $C_i(x, y)$ represents the edges of partition-component i . Due to Fact 1, this corresponds to the edge layers $\sum_{j=0}^{i-1} 2^{k_j} := \mu, \dots, \mu + 2^{k_i} =: \mu'$ of U . It can be expressed as a threshold function that checks if $(x, y) \in E$ and if x lies in a node layer $\nu \in \{\mu, \dots, \mu' - 1\}$:

$$C_i(x, y) = E(x, y) \wedge (x^r + x^c \geq \mu) \wedge (x^r + x^c < \mu'). \quad (2)$$

Function $C(x, y)$: The intermediate result $C^{(i)}(x, y) := \bigvee_{j=0}^i C_j(x, y)$ of the i th disjunction operation during the computation of $C(x, y)$ accordingly represents the edge layers $0, \dots, \sum_{j=0}^i 2^{k_j} =: \mu'$, which can be expressed by leaving one condition from (2):

$$C^{(i)}(x, y) = E(x, y) \wedge (x^r + x^c < \mu').$$

Hence, it is $C_i, C^{(i)}, C \in D[\mathbb{T}_{4,m}^1, O(1)]$.

Reachability analysis: What remains is the reachability analysis on $C^R(x, y)$. It uses functions $RPATH_k^{\leq}(x, y)$ being true for pairs (x, y) with an x - y -path in C^R not longer than 2^k . Due to Fact 1, it holds $RPATH_k^{\leq}(x, y) = PATH_k^{\leq}(y, x)$, for which the threshold property already has been shown. The result $R(x)$ simply contains all grid nodes and it is $U(x, y) = C(x, y)$.

Now, we have considered all functions that occur during the layered-network construction and which are named in the algorithm description in [2]. Nevertheless, their OBDDs are not the only ones appearing during the computation of U : There are intermediate OBDDs within the quantifications over the single bits of node numbers. In order to bound their size, too, we introduce the term of complete OBDDs and state the following lemma of Woelfel [4], whose proof is placed in Appendix A.

Definition 5 *An OBDD with variable set $\{x_0, \dots, x_{n-1}\}$ is called complete if all paths from a source to a sink have length n .*

Lemma 2 *Let $f \in B_n$ be a Boolean function defined on the variables $\{x_0, \dots, x_{n-1}\}$ that has a complete π -OBDD \mathcal{F} , $\pi \in \Sigma_n$, of constant width k . Then, for every sequence x_{i_1}, \dots, x_{i_c} of c variables and a quantifier $Q \in \{\exists, \forall\}$, the function $g := (Qx_{i_r} \dots Qx_{i_1})f$, $r \leq c$, has a complete π -OBDD \mathcal{G} of width $2^k = O(1)$.*

That is, also the existential quantification of the minimized OBDD F of f over an arbitrary variable subset produces intermediate OBDDs of the same asymptotical size $O(|\mathcal{F}|) = O(n)$.

Lemma 3 *Every function computed during the layered-network construction of the IS-algorithm on $(2^k + 1) \times (2^k + 1)$ -grid networks, $k \in \mathbb{N}_0$, has a complete OBDD of constant width.*

Proof. In Section 7, it is shown that every named function computed during the layered-network construction in the IS-algorithm on $(2^k + 1) \times (2^k + 1)$ -grid networks, $k \in \mathbb{N}_0$, is in $D\left[\mathbb{T}_{O(1),m}^1, O(1)\right]$. Due to [5], such functions have complete OBDDs of constant width. Other OBDDs only occur during the computation of named functions, which are obtained by $O(1)$ OBDD-operations each.

Due to Lemmas 1, intermediate results after $O(1)$ binary syntheses have still complete OBDDs of constant width. Due to Lemma 2, the same holds for intermediate results within each quantification as well as after $O(1)$ quantifications. All occurring OBDDs have been considered. \square

8 Blocking-Flow Construction

In this section, the fact that the layered-network depth 2^{k+1} of $(2^k + 1) \times (2^k + 1)$ -grid networks is a power of 2 will be used for the first time. A sophisticated

implementation of the IS-algorithm considers only the path lengths $\ell_m^1 = 2^{k-m+1}$ for $m \in \{0, \dots, k+1\}$, and omits the unnecessary lengths $\ell_m^2 = \ell_m^1 - 1$. One sweep using the multi-path construction with one selection iteration per path length 2^m suffices to construct both augmenting paths. We will show for the functions $D_{\ell_m^1}$ and $S_{\ell_m^1}$ that they are in $D\left[\mathbb{T}_{O(1),m}^1, O(1)\right]$.

D_1 and S_1 are composed of threshold functions that are already known. We now investigate the constructed paths of length 2^m with $m \geq 1$. The node selection uses the priority function Π^{dp} , which privileges small row/column indices. It receives eight indices, interpreted as row and column of start- and end-node of two edges (v, w) and (y, z) . It is $\Pi^{\text{dp}}(v^r, v^c, w^r, w^c, y^r, y^c, z^r, z^c) = 1$ if and only if the binary value $|v^r v^c w^r w^c|$ is smaller than $|y^r y^c z^r z^c|$.

We now consider the seven different cases occurring in the first selection, i.e. the computation of D_2 :

1. Figure 2(a): Because x has a lower row index than y , both edges (u, w) and (v, w) choose (w, x) . Because u has a lower row index than v , the selected edge (w, x) in turn chooses (u, w) — D_2 gains (u, w, w, x) .
2. Figure 2(b): Both edges (u, w) and (v, w) are only able to choose (w, x) . As in Case 1, (u, w) wins the selection and D_2 gains (u, w, w, x) .
3. Figure 2(c): The only possible path is (u, w, w, x) .
4. Figure 2(d): As in Case 1, (w, x) is preferred by (u, w) . The latter has no competitor, so D_2 gains (u, w, w, x) .
5. Figure 2(e): As in Case 1, (w, x) is preferred by (v, w) . Because it has no competitor, D_2 gains (v, w, w, x) .
6. Figure 2(f): The only possible path is (v, w, w, y) .
7. Figure 2(g): Both edges (u, w) and (v, w) are only able to choose (w, y) . As in Cases 1 and 2, (u, w) wins the selection and D_2 gains (u, w, w, y) .

We have considered all paths of D_2 , and partition them into two sets: We denote the paths of Cases 1 to 4 by D_2^1 , while we denote paths of Cases 5 to 7 by D_2^2 . Analogously, it can be seen that further paths of length 2^m , $m > 1$, can also be partitioned into sets $D_{2^m}^1$ and $D_{2^m}^2$, where $D_{2^m}^a$ exclusively contains paths composed of parts of $D_{2^{m-1}}^a$, for $a \in \{1, 2\}$.

Paths (v, w, y, z) of $D_{2^m}^1$ are generated for any pair of a vertical edge (v, w) and a horizontal edge (y, z) , such that v and z lie on the same diagonal and have a distance of 2^m (see Figure 3(a)). Hence, $D_{2^m}^1$ may be expressed as a composition

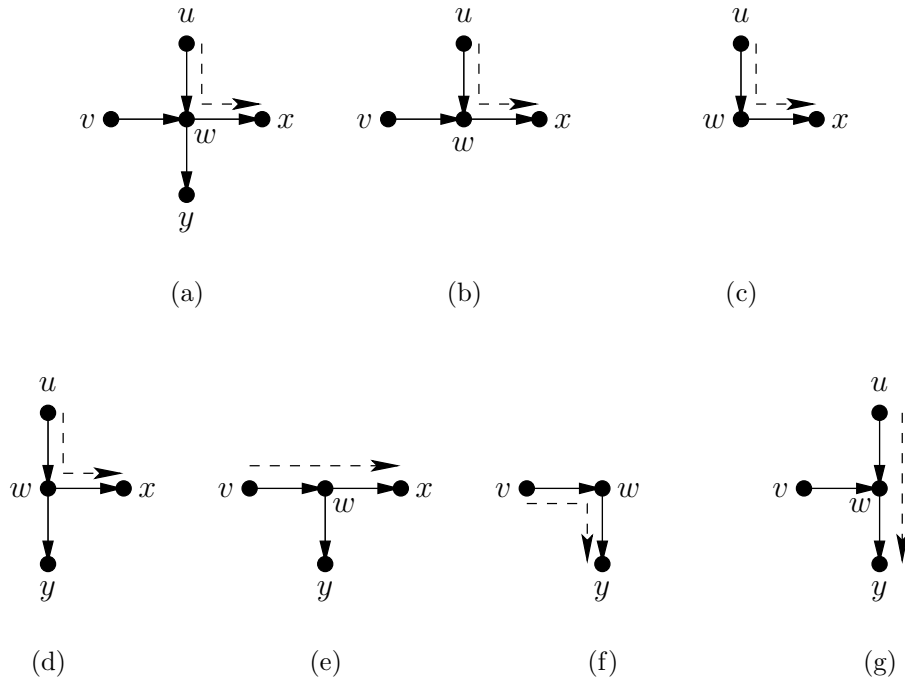


Figure 2: The seven cases occurring during the selection of D_2 . Solid arrows represent grid edges, while dashed arrows indicate constructed paths.

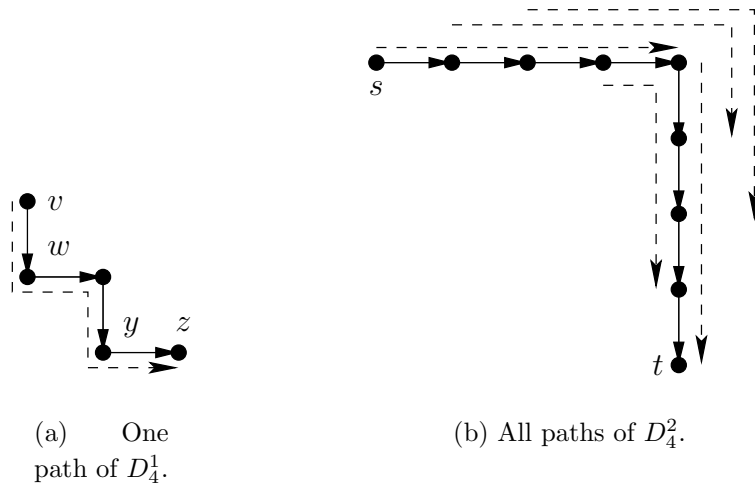


Figure 3: The different path structures of D_4^1 and D_4^2 . Solid arrows represent grid edges, while dashed arrows indicate constructed paths.

of threshold functions, why it is in $D\left[\mathbb{T}_{O(1),m}^1, O(1)\right]$:

$$\begin{aligned} D_{2^m}^1(v, w, y, z) &= E(v, w) \wedge E(y, z) \\ &\wedge ((v^c = w^c) \wedge (y^r = z^r)) \\ &\wedge (z^r - v^r = 2^{m-1}) \wedge (z^c - v^c = 2^{m-1}). \end{aligned}$$

At next, we have to show the threshold property for $S_{2^m}^1$, which manages the edges of paths (w, x, \dots, y, z) . The following expression checks if (u, v) is an edge and if the path from (w, x) to (y, z) exists in $D_{2^m}^1$. Moreover, (u, v) must lie in the rectangle between w and z in order to be part of this path. Finally, the end-node v of horizontal edges (u, v) must lie on the same diagonal as w , while the start-node u of vertical edges (u, v) has to lie on the same diagonal as w .

$$\begin{aligned} S_{2^m}^1(u, v, w, x, y, z) &= E(u, v) \wedge D_{2^m}^1(w, x, y, z) \\ &\wedge (w^r \leq u^r) \wedge (w^c \leq u^c) \\ &\wedge (v^r \leq z^r) \wedge (v^c \leq z^c) \\ &\wedge \left[((u^r = v^r) \wedge (v^r - w^r = v^c - w^c)) \right. \\ &\quad \left. \vee ((u^c = v^c) \wedge (u^r - w^r = u^c - w^c)) \right] \end{aligned}$$

We now consider the paths of $D_{2^m}^2$ (see Figure 3(b)). These are simply all paths of length 2^m on the upper right boundary of the grid network. This is expressed by the following composition of threshold functions:

$$\begin{aligned} D_{2^m}^2(v, w, y, z) &= E(v, w) \wedge E(y, z) \\ &\wedge ((w^r = 0) \vee (v^c = \mathcal{N} - 1)) \\ &\wedge ((z^r = 0) \vee (y^c = \mathcal{N} - 1)) \\ &\wedge (z^r - v^r \geq 0) \wedge (z^c - v^c \geq 0) \\ &\wedge (z^r - v^r + z^c - v^c = 2^m). \end{aligned}$$

The following composition of threshold functions represents $S_{2^m}^2$. It checks, if (u, v) is a proper edge, and if there is a path from (w, x) to (y, z) . Furthermore, (u, v) has to lie on the upper right boundary between w and z .

$$\begin{aligned} S_{2^m}^2(u, v, w, x, y, z) &= E(u, v) \wedge D_{2^m}^2(w, x, y, z) \\ &\wedge ((v^r = 0) \vee (u^c = \mathcal{N} - 1)) \\ &\wedge (w^r \leq u^r) \wedge (w^c \leq u^c) \\ &\wedge (v^r \leq z^r) \wedge (v^c \leq z^c) \end{aligned}$$

Finally, $S_{2^{k+1}}^1$ contains one augmenting path that is structured like in Figure 3(a), while the path of $S_{2^{k+1}}^2$ simply contains all edges of the upper right

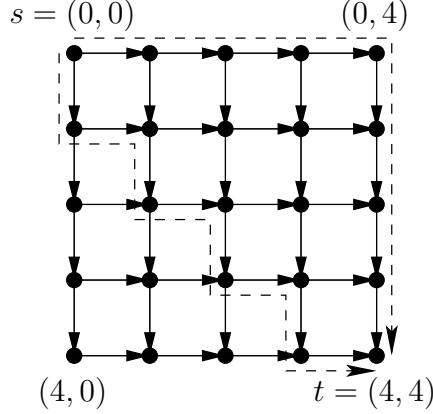


Figure 4: The constructed maximum flow in the 5×5 -grid network. Solid arrows represent grid edges, while dashed arrows indicate constructed paths.

boundary of the grid. Figure 4 shows the constructed maximum flow in the 5×5 -grid network.

Due to $D_{2^m} = D_{2^m}^1 \vee D_{2^m}^2$ and $S_{2^m} = S_{2^m}^1 \vee S_{2^m}^2$, both D_{2^m} and S_{2^m} are in $D[\mathbb{T}_{O(1),m}^1, O(1)]$. Now, we can state the central lemma of this section.

Lemma 4 *Every function computed during the only sweep of the blocking-flow construction of the IS-algorithm on $(2^k + 1) \times (2^k + 1)$ -grid networks, $k \in \mathbb{N}_0$, has a complete OBDD of constant width.*

Proof. In Section 8 it is shown that the functions D_{2^m} and S_{2^m} , $m \in \{0, \dots, k + 1\}$, computed during the layered-network construction in the IS-algorithm on $(2^k + 1) \times (2^k + 1)$ -grid networks, $k \in \mathbb{N}_0$, are in $D[\mathbb{T}_{O(1),m}^1, O(1)]$. Due to [5], such functions have complete OBDDs of constant width. Each of the functions $P_{2^m}^S$, $P_{2^m}^E$, P_{2^m} , $P_{2^m}^*$, Q_{2^m} , and B^* results from $O(1)$ quantifications and binary syntheses of functions in $D[\mathbb{T}_{O(1),m}^1, O(1)]$. Analogously to the proof of Lemma 3, we may apply Lemmas 1 and 2 successively in order to show that all occurring functions have complete OBDDs of constant width. \square

We are now able to prove Theorem 1.

Proof of Theorem 1. Due to Lemmas 3 and 4, both the layered-network construction and the blocking-flow computation exclusively generate functions that have complete OBDDs of constant width. Only such functions take part within the updates of $B(x, y)$ and $U(x, y)$ after the application of a path construction. This holds also for the flow augmentation at the end of the algorithm's flow augmentation phase. \square

9 Experimental Results

Both the IS-algorithm and Hachtel and Somenzi's maximum flow method [1] have been implemented¹ in C++. In order to confirm the practical relevance of Theorems 1 and 2, both algorithms have been applied on $(2^k + 1) \times (2^k + 1)$ -grids for $0 \leq k \leq 16$ on a PC with Pentium 4 2GHz processor and 512 MB of main memory. For $k > 16$, the system's memory did not suffice to apply Hachtel and Somenzi's method. In contrast, the IS-algorithm was executed up to $k = 19$, and did not reach the memory limit in the experiments. Figures 5(a) and 5(b) show the experimental results by means of runtime and space usage. Space is measured in the maximum number of nodes contained in all OBDDs at any time. It can be seen that the IS-algorithm beats the space usage of Hachtel and Somenzi's method for $k \geq 13$, while the runtime is beaten for $k \geq 16$.

10 Conclusions

It has been proven that the IS-algorithm computes a maximum flow on $(2^k + 1) \times (2^k + 1)$ -grid networks in time $O(k^3)$ and space $O(k^2)$. This is one of the few results [5] so far on the over-all runtime and space usage of an implicit graph algorithm, while most analyses only consider the number of executed OBDD-operations. The practical relevance of this result is confirmed by experiments, in which Sawitzki's algorithm beats a previous method of Hachtel and Somenzi [1] w. r. t. time and space for $k \geq 16$.

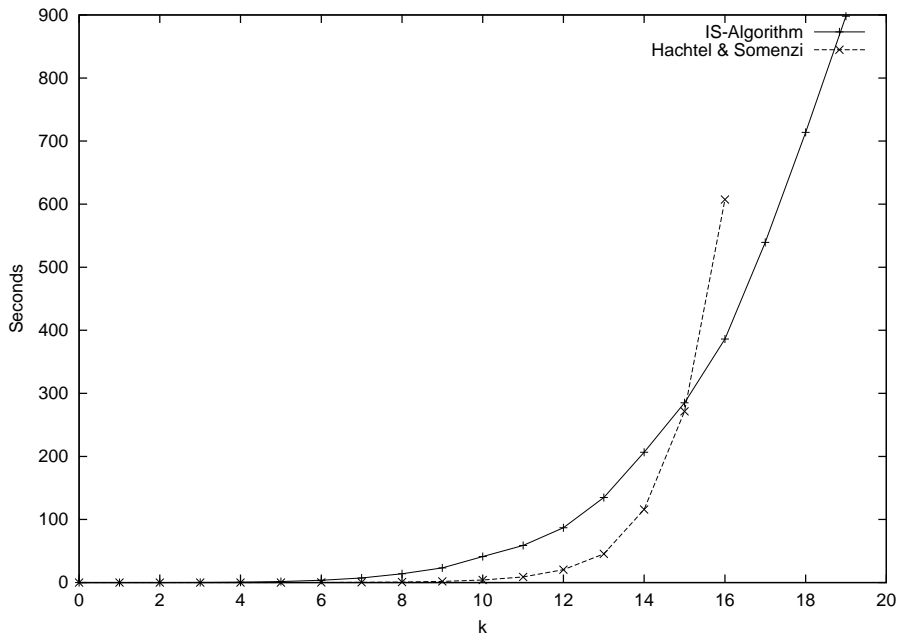
It is easy to see that the layered-network remains the same if the square between s and t is embedded within a larger $\mathcal{N} \times \mathcal{M}$ -grid, $\mathcal{N}, \mathcal{M} \in \mathbb{N}_{\geq 2}$. Because the OBDD-sizes that occur during the layered-network construction stay logarithmic w. r. t. $\mathcal{N} \cdot \mathcal{M}$, the runtime and space usage stay polylogarithmic in this scenario. Moreover, we assume similar results for $\mathcal{N} \times \mathcal{N}$ -grids with arbitrary \mathcal{N} as well as for $\mathcal{N} \times \mathcal{M}$ -grids with $(\mathcal{N} - 1) \cdot (\mathcal{M} - 1) = 2^k$ for some $k \in \mathbb{N}$.

Grid structures are of interest in application areas like circuit design. Therefore, we hope that the flow maximization is also efficient on less regular networks with more practical relevance.

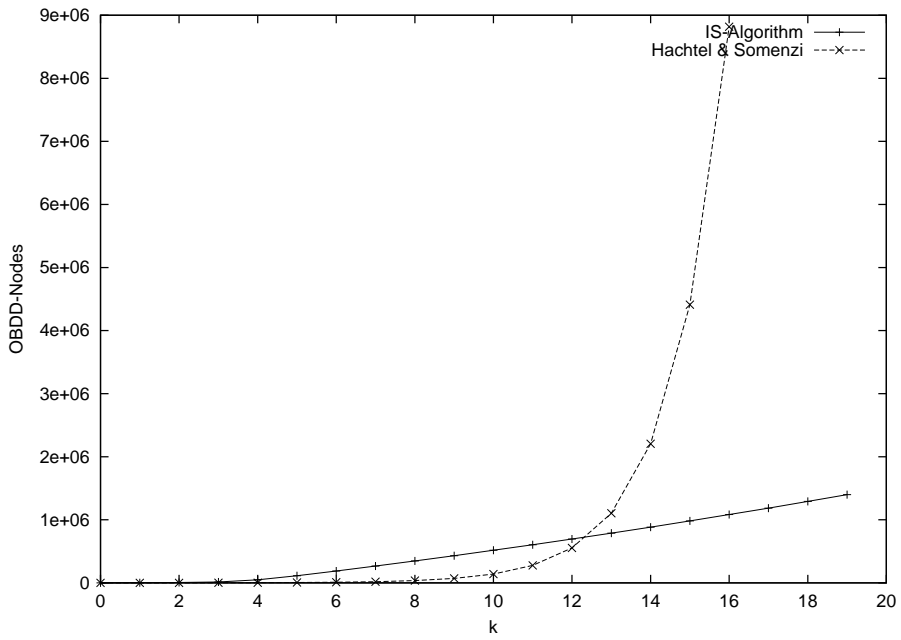
Acknowledgments

Thanks to Ingo Wegener and Philipp Woelfel for helpful discussions.

¹Implementation available at <http://thefigaro.sourceforge.net/>.



(a) Runtime comparison.



(b) Space usage comparison.

Figure 5: Experimental results on $(2^k + 1) \times (2^k + 1)$ -grid networks.

References

- [1] G.D. Hachtel and F. Somenzi. A symbolic algorithm for maximum flow in 0–1 networks. *Formal Methods in System Design*, 10:207–219, 1997.
- [2] D. Sawitzki. Implicit flow maximization by iterative squaring. In *SOFSEM 2004: Theory and Practice of Computer Science*, volume 2932 of *Lecture Notes in Computer Science*, pages 301–313. Springer, 2004.
- [3] I. Wegener. *Branching Programs and Binary Decision Diagrams*. SIAM, Philadelphia, 2000.
- [4] P. Woelfel. Private communication, 2003.
- [5] P. Woelfel. Symbolic topological sorting with OBDDs. In *Mathematical Foundations of Computer Science*, volume 2747 of *Lecture Notes in Computer Science*, pages 671–680. Springer, 2003.

A Proofs

Because there is a variety of possible implementations of the binary synthesis, we adopt the usual assumption that the algorithm recursively constructs the reachable part of F 's and G 's cross product graph before it minimizes the result in linear time.

Proof of Lemma 1. We assume w.l.o.g. $\pi = id$ for the variable ordering π . At first, we convince ourselves that the binary synthesis $\mathcal{F} \otimes \mathcal{G}$ on the complete OBDDs \mathcal{F} and \mathcal{G} takes runtime $O(n)$. Let v and w be two OBDD nodes from \mathcal{F} resp. \mathcal{G} which represent two subfunctions f' resp. g' . Due to the completeness of \mathcal{F} and \mathcal{G} , f' and g' participate in a recursive computation $f' \otimes g'$ if and only if v and w are labeled with the same variable. Hence, \mathcal{F} 's and \mathcal{G} 's cross product graph contains at most k^2 nodes in each layer and the OBDD $\mathcal{H} := \mathcal{F} \otimes \mathcal{G}$ is computed in time and space $O(n)$.

We now investigate the synthesis of the minimal OBDDs F and G . These are obtained from \mathcal{F} resp. \mathcal{G} through successive applications of two minimization operators [3]:

- Eliminate nodes v whose 0- and 1-edge point to the same node.
- Merge nodes v and w which are labeled with the same variable, and which have the same a -successors $v_a = w_a$, $a \in \{0, 1\}$.

For general OBDDs A and B , the result $A \otimes B$ may have size $O(\text{size}(A) \cdot \text{size}(B))$, because recursive computations may combine nodes on different layers. This problem does not occur in $F \otimes G$: a -edges (v, v_a) in F or G , $a \in \{0, 1\}$, which skip layers are the result of elimination operators that have been applied on \mathcal{F} resp. \mathcal{G} . Let v of F lie in layer i , while w of G lies in $j > i$. When a recursive computation $f' \otimes g'$ combines v and w , this corresponds to the combination of two nodes v' and w' in the computation of $\mathcal{F} \otimes \mathcal{G}$ that lie in the same layer $\ell \leq i < j$, and whose successive elimination redirected their ingoing edges to v and w (see Figure 6). Therefore, every recursive binary synthesis within $F \otimes G$ can be associated with one within $\mathcal{F} \otimes \mathcal{G}$. Hence, the OBDD $H := F \otimes G$ contains not more nodes than \mathcal{H} , and can be computed in time and space $O(n)$. \square

Proof of Lemma 2. We assume w.l.o.g. that \mathcal{F} contains exactly k nodes in each layer $\ell \in \{0, \dots, n-1\}$ as well as $\pi = id$ for the variable ordering π . At first, we construct an OBDD \mathcal{P} of width 2^k that contains 2^k nodes V_1, \dots, V_{2^k} in layer ℓ corresponding to the 2^k subsets of \mathcal{F} 's nodes $\{v_1, \dots, v_k\}$ in layer ℓ . Moreover, we define the a -successor V_a , $a \in \{0, 1\}$, of a node V in \mathcal{P} as the union of the a -successors of all original nodes $v \in V$, i. e., $V_a := \bigcup_{v \in V} \{w_a\}$.

We now successively perform the quantifications on \mathcal{P} in order to construct \mathcal{G} , where each constructed OBDD \mathcal{G}_r for $g_r := (Qx_{i_r} \dots Qx_{i_1})f$ will not be larger than \mathcal{P} .

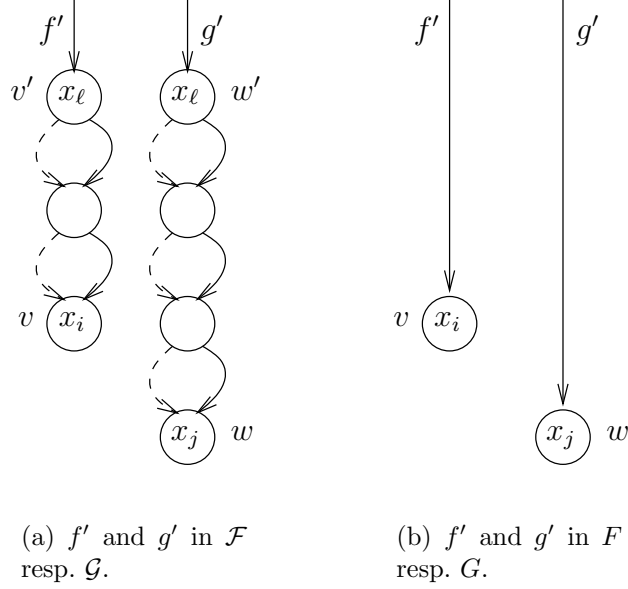


Figure 6: Examples of OBDD-representations of subfunctions f' and g' .

Let V_0 and V_1 be the 0- resp. 1-successor of V . We perform the quantification over variable x_ℓ by replacing all edges (U, V) (which point to nodes V in layer ℓ) by $(U, V_0 \cup V_1)$.

If the evaluation traversal on \mathcal{G}_r for a given input assignment x visits $V \subseteq \{v_1, \dots, v_k\}$, this node represents the union of nodes reachable in the original OBDD \mathcal{F} for any assignment of the quantified variables x_{i_1}, \dots, x_{i_r} .

In generating \mathcal{P} and the OBDDs $\mathcal{G}_1, \dots, \mathcal{G}_c := \mathcal{G}$, we also create terminal subsets of $\{0, 1\}$. When applying a sequence of existential quantifiers ($Q = \exists$), the sets $\{1\}$ and $\{0, 1\}$ are replaced by the terminal 1 and all others by 0—it suffices if one assignment of the quantified variables leads to 1. When applying universal quantifiers ($Q = \forall$), only the set $\{1\}$ is replaced by 1—all assignments must lead to 1.

We have constructed a complete OBDD \mathcal{G} for g of width $2^k = O(1)$. \square