

A Symbolic Approach to the All-Pairs Shortest-Paths Problem

Daniel Sawitzki*

University of Dortmund, Computer Science 2
D-44221 Dortmund, Germany
daniel.sawitzki@cs.uni-dortmund.de
<http://ls2-www.cs.uni-dortmund.de/~sawitzki/>

Abstract. Graphs can be represented symbolically by the Ordered Binary Decision Diagram (OBDD) of their characteristic function. To solve problems in such implicitly given graphs, specialized symbolic algorithms are needed which are restricted to the use of functional operations offered by the OBDD data structure. In this paper, a symbolic algorithm for the all-pairs shortest-paths (APSP) problem in loopless directed graphs with strictly positive integral edge weights is presented. It requires $\Theta(\log^2(NB))$ OBDD-operations to obtain the lengths and edges of all shortest paths in graphs with N nodes and maximum edge weight B . It is proved that runtime and space usage are polylogarithmic w. r. t. N and B on graph sequences with characteristic bounded-width functions. This convenient property is closed under certain graph composition operations. Moreover, an alternative symbolic approach for general integral edge weights is sketched which does not behave efficiently on general graph sequences with bounded-width functions. Finally, two variants of the APSP problem are briefly discussed.

1 Introduction

Algorithms on graphs G with node set V and edge set $E \subseteq V^2$ typically work on adjacency lists of size $\Theta(|V| + |E|)$ or on adjacency matrices of size $\Theta(|V|^2)$. These representations are called *explicit*. However, there are application areas in which problems on graphs of such large size have to be solved that an explicit representation on today's computers is not possible. In the area of logic synthesis and verification, state-transition graphs with for example 10^{27} nodes and 10^{36} edges occur. Other applications produce graphs which are representable in explicit form, but for which even runtimes of efficient polynomial algorithms are not practicable anymore. Modeling of the WWW, street, or social networks are examples of this problem scenario.

Yet, we expect the large graphs occurring in application areas to contain regularities. If we consider graphs as Boolean functions, we can represent them by *Ordered Binary Decision Diagrams* (OBDDs) [3, 4, 24]. This data structure

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

is well established in verification and synthesis of sequential circuits [11, 12, 14, 15, 24] due to its good compression of regular structures. In order to represent a graph $G = (V, E)$ by an OBDD, its edge set E is considered as a *characteristic Boolean function* χ_E , which maps binary encodings of E 's elements to 1 and all others to 0. This representation is called *implicit* or *symbolic*, and is not essentially larger than explicit ones. Nevertheless, we hope that advantageous properties of G lead to small, that is sublinear OBDD-sizes [23, 25].

Having such an OBDD-representation of a graph, we are interested in solving problems on it without extracting too much explicit information from it. Algorithms that are mainly restricted to the use of functional operations are called *implicit* or *symbolic algorithms* [9, 10, 13, 19, 20, 24, 26, 27]. They are considered as heuristics to save time and/or space when large structured input graphs do not fit into the internal memory anymore. Then, we hope that each OBDD-operation processes many edges in parallel. The runtime of such methods depends on the number of executed operations as well as on the efficiency of each single one. The latter in turn depends on the size of the operand OBDDs.

Bahar et al. [1] presented a symbolic shortest-path algorithm for graphs represented by Algebraic Decision Diagrams (ADDs), which are difficult to analyze and useful only for a small number of different weight values. In [19], the algorithms of Dijkstra and Bellman-Ford are transformed into symbolic methods and evaluated in experiments. Although they perform efficiently on a variety of instances, their runtime is always at least linear in the depth of the shortest-paths tree. In this paper, we present a symbolic OBDD-algorithm for the all-pairs shortest-paths problem (called APSP-algorithm) that enables polylogarithmic runtime independent of the input graph's diameter. Given a symbolically represented loopless directed graph $G = (V, E, c)$ with strictly positive integral edge weights, it computes the length of shortest paths from node u to node v (called u - v -path in the following) for every connected pair $(u, v) \in V^2$, as well as the edges of such paths. The algorithm performs $\Theta(\log^2(NB))$ OBDD-operations on graphs with N nodes and maximum edge weight B .

The paper is organized as follows: Sections 2 and 3 introduce the principles of symbolic graph representation and preliminaries before presenting the symbolic APSP-algorithm in Sect. 4. Section 5 investigates its runtime and space usage on bounded-width functions as well as graph composition operations preserving the bounded-width property. In Sect. 6, we consider an alternative symbolic approach for general integral edge weights and point to a major disadvantage. Adaptations to two variants of the APSP problem are briefly presented in Sect. 7. Finally, Sect. 8 gives conclusions on the work.

2 Symbolic Graph Representation

We denote the class of Boolean functions $f: \{0, 1\}^n \rightarrow \{0, 1\}$ by B_n . The i th character of a binary number $x \in \{0, 1\}^n$ is denoted by x_i and $|x| := \sum_{i=0}^{n-1} x_i 2^i$ identifies its value.

Consider a directed graph $G = (V, E)$ with node set $V = \{v_0, \dots, v_{N-1}\}$ and edge set $E \subseteq V^2$. G can be represented by a *characteristic* Boolean function $\chi_E \in B_{2n}$ which maps pairs $(x, y) \in \{0, 1\}^{2n}$ of binary node numbers of length $n := \lceil \log N \rceil$ to 1 iff $(v_{|x|}, v_{|y|}) \in E$. We can capture more complex graph properties by adding further arguments to characteristic functions. An additional weight function $c: E \rightarrow \{0, \dots, 2^m - 1\}$ is modeled by $\chi_C \in B_{2n+m}$ which maps triples (x, y, d) to 1 iff $(v_{|x|}, v_{|y|}) \in E$ and $c(v_{|x|}, v_{|y|}) = |d|$.

A Boolean function $f \in B_n$ defined on variables x_0, \dots, x_{n-1} can be represented by an *Ordered Binary Decision Diagram (OBDD)* [3, 4, 24]. An OBDD \mathcal{G} is a directed acyclic graph consisting of *internal nodes* and *sink nodes*. Each internal node is labeled with a Boolean variable x_i , while each sink node is labeled with a Boolean constant. Each internal node is left by two edges one labeled by 0 and the other by 1. A *function pointer* p marks a special node that represents f . Moreover, a permutation $\pi \in \Sigma_n$ called *variable order* must be respected by the internal nodes' labels on every path from p to a sink. For a given variable assignment $a \in \{0, 1\}^n$, we compute the function value $f(a)$ by traversing \mathcal{G} from p to a sink labeled with $f(a)$ while leaving a node x_i via its a_i -edge.

An OBDD \mathcal{G} with variable order π is called π -OBDD. Its size $\text{size}(\mathcal{G})$ is measured by the number of its nodes. The minimal-size π -OBDD for a function $f \in B_n$ is known to be canonical and will be denoted by $\pi\mathcal{G}[f]$ in this paper. We adopt the usual assumption that all OBDDs occurring in symbolic algorithms have minimal size, since all essential OBDD-operations produce minimized diagrams. Figure 1(a) shows the minimal OBDD for an example function. There is an upper bound of $(2 + o(1))2^n/n$ for the OBDD-size of every $f \in B_n$; hence, an edge set $E \subseteq V^2$ has worst-case OBDD-size $\mathcal{O}(V^2/\log|V|)$.

The satisfiability of f can be decided in time $\mathcal{O}(1)$. The negation \bar{f} as well as the replacement of a function variable x_i by a constant a_i (i. e., $f_{|x_i=a_i}$) is obtained in time $\mathcal{O}(\text{size}(\pi\mathcal{G}[f]))$ without enlarging the OBDD. Whether two functions f and g are equivalent (i. e., $f = g$) can be decided in time $\mathcal{O}(\text{size}(\pi\mathcal{G}[f]) + \text{size}(\pi\mathcal{G}[g]))$. These operations are called *cheap*. Further essential operations are the *binary synthesis* $f \otimes g$ for $f, g \in B_n$, $\otimes \in B_2$ (e. g., “ \wedge ” and “ \vee ”), and the *quantification* $(\mathcal{Q}x_i)f$ for a quantifier $\mathcal{Q} \in \{\exists, \forall\}$. In general, the result $\pi\mathcal{G}[f \otimes g]$ has size $\mathcal{O}(\text{size}(\pi\mathcal{G}[f]) \cdot \text{size}(\pi\mathcal{G}[g]))$, which is also the general runtime of this operation. The computation of $\pi\mathcal{G}[(\mathcal{Q}x_i)f]$ can be realized by two cheap operations and one binary synthesis in time and space $\mathcal{O}(\text{size}^2(\pi\mathcal{G}[f]))$.

3 Preliminaries

The characteristic functions used for symbolic representation are typically defined on a number of k subsets of Boolean variables, each representing a different argument (e. g., $C(x, y, d)$ is defined on nodes x, y and weight d). We assume w. l. o. g. that all arguments consist of the same number of n Boolean variables. If there is no confusion, both a function $\chi_S \in B_{kn}$ defined on $x^{(1)}, \dots, x^{(k)} \in \{0, 1\}^n$ as well as its OBDD-representation $\pi\mathcal{G}[\chi_S]$ will be denoted by $S(x^{(1)}, \dots, x^{(k)})$ in this paper. Quantifications $(\mathcal{Q}x_0^{(i)}, \dots, x_{n-1}^{(i)})$ over all n variables of argument i will be denoted by $(\mathcal{Q}x^{(i)})$.

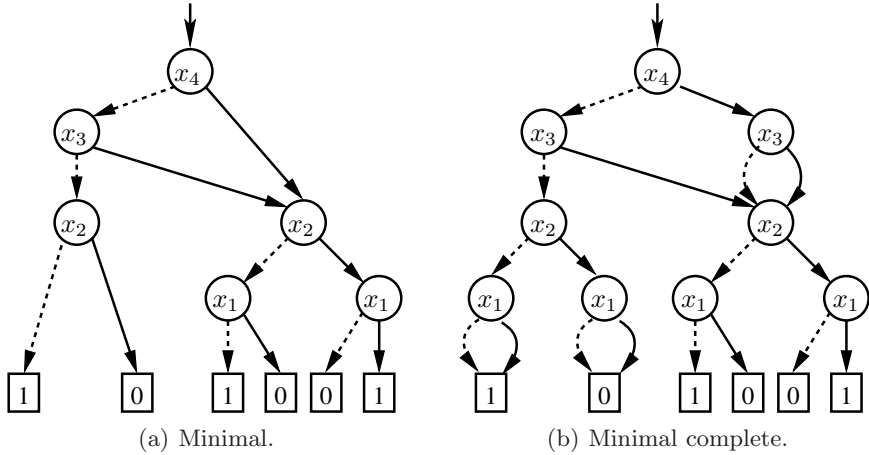


Fig. 1. Minimal (left) and minimal complete (right) π -OBDD for $f(x_1, \dots, x_4) := \bar{x}_1\bar{x}_2 + \bar{x}_2\bar{x}_3\bar{x}_4 + x_1x_2x_3 + x_1x_2x_4$ with $\pi := (4, 3, 2, 1)$

Interleaved Variable Orders. Assume that each of the k function arguments $x^{(1)}, \dots, x^{(k)} \in \{0, 1\}^n$ has its own variable order $\tau_i \in \Sigma_n$. The global order π is called k -interleaved if it respects each τ_i while reading variables $x_j^{(i)}$ with same bit index j en bloc, that is, $\pi := (x_{\tau_1(0)}^{(1)}, x_{\tau_2(0)}^{(2)}, \dots, x_{\tau_k(0)}^{(k)}, x_{\tau_1(1)}^{(1)}, \dots, x_{\tau_k(n-1)}^{(k)})$.

Definition 1. Let $\rho \in \Sigma_k$ and $f \in B_{kn}$ be defined on variables $x^{(1)}, \dots, x^{(k)} \in \{0, 1\}^n$. The argument reordering $\mathcal{R}_\rho: B_{kn} \rightarrow B_{kn}$ is defined by $\mathcal{R}_\rho(f(x^{(1)}, \dots, \dots, x^{(k)})) := f(x^{(\rho(1))}, \dots, x^{(\rho(k))})$.

When using a k -interleaved variable order π , the resulting OBDD $\pi\mathcal{G}[\mathcal{R}_\rho(f)]$ has worst-case size $k^{3k} \cdot \text{size}(\pi\mathcal{G}[f])$ and can be computed in time and space $\mathcal{O}(k^{23k} \cdot \text{size}(\pi\mathcal{G}[f]))$ (see [23]). Because k is independent of f , this is considered as linear in $\text{size}(\pi\mathcal{G}[f])$. For example, argument reordering is used in (2) to replace the original arguments of $C(x, y, d)$ by temporary ones $x^{(1)}$, $x^{(2)}$, and $d^{(2)}$.

Multivariate Threshold Functions. The APSP-algorithm contains comparisons like $F(x, y, z) := (|x| + |y| = |z|)$, which can be realized by *multivariate threshold functions*.

Definition 2 (Woelfel [26]). Let $f \in B_{kn}$ be defined on variables $x^{(1)}, \dots, x^{(k)} \in \{0, 1\}^n$. Moreover, let $W, T \in \mathbb{Z}$, and $w_1, \dots, w_k \in \{-W, \dots, W\}$. f is called k -variate threshold function iff

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

W is called the maximum absolute weight of f . The class of k -variate threshold functions $f \in B_{kn}$ with maximum absolute weight W is denoted by $\mathbb{T}_{k,n}^W$.

Obviously, F can be expressed as $(|x| + |y| - |z| \geq 0) \wedge (|z| - |x| - |y| \geq 0)$. Analogue, the relations $>$, \leq , and $<$ can be composed of multivariate threshold functions, too. For constant W and k , such comparisons have π -OBDDs of size $\mathcal{O}(n)$ using a k -interleaved variable order π with increasing bit significance (i. e., $\tau_i = \text{id}$) [26].

4 The Symbolic APSP-Algorithm

We now describe the APSP-algorithm for symbolically represented loopless directed graphs $G = (V, E, c)$ with node set $V = \{v_0, \dots, v_{N-1}\}$, edge set $E \subseteq V^2$, edge weight function $c: E \rightarrow \mathbb{N}_{>0}$, and $B := \max\{c(e) \mid e \in E\}$. The maximum path length in G is $B(N - 1) =: L$. Let $n := \lceil \log(L + 1) \rceil = \Theta(\log N + \log B)$ the number of bits encoding one node number or distance value. The algorithm receives the input graph G as an OBDD for the characteristic function $C(x, y, d)$ with

$$C(x, y, d) = 1 \Leftrightarrow [(v_{|x|}, v_{|y|}) \in E] \wedge [c(v_{|x|}, v_{|y|}) = |d|] .$$

4.1 Computing the Shortest Paths' Lengths

At first, we are interested in the distance function $\text{dist}: V^2 \rightarrow \mathbb{N}_0 \cup \{\infty\}$ which maps node pairs $(u, v) \in V^2$ to the length $\|\bar{p}\|$ of a shortest path $\bar{p} = (u, \dots, v)$ with $\|\bar{p}\| := \sum_{e \in \bar{p}} c(e)$. The algorithm computes dist 's OBDD $S(x, y, d)$ with

$$S(x, y, d) = 1 \Leftrightarrow \text{dist}(v_{|x|}, v_{|y|}) = |d| .$$

We use functions $S_i(x, y, d)$ to represent shortest paths of maximal length $2^i - 1$, i. e., $S_i(x, y, d) = S(x, y, d) \wedge (|d| < 2^i)$. These are computed iteratively for $i \in \{1, \dots, n\}$ until the output of $S(x, y, d) = S_n(x, y, d)$.

We consider $S_1(x, y, d)$. Due to $c(e) \in \mathbb{N}_{>0}$, paths of length $2^1 - 1 = 1$ correspond to edges contained in $C(x, y, d)$, whereas $|d| = 0$ implies $x = y$. Hence, $S_1(x, y, d)$ is defined by

$$S_1(x, y, d) := [(|d| = 1) \wedge C(x, y, d)] \vee [(|d| = 0) \wedge (x = y)] . \tag{1}$$

In general, we compute $S_{i+1}(x, y, d)$ from $C(x, y, d)$ and $S_i(x, y, d)$ using the following lemma.

Lemma 1. *For every path $\bar{p} = (p_1, \dots, p_K)$ in G with $K \geq 1$, $\|\bar{p}\| < 2^{i+1}$, $i \in \mathbb{N}_0$, there is an edge $e := (p_j, p_{j+1}) \in \bar{p}$ such that $\bar{p}_1 := (p_1, \dots, p_j)$ and $\bar{p}_2 := (p_{j+1}, \dots, p_K)$ have length $\|\bar{p}_1\|, \|\bar{p}_2\| < 2^i$.*

Proof. We choose edge $e = (p_j, p_{j+1})$ with the smallest index j such that $\|\bar{p}_2\| < 2^i$. If $j = 1$ then $\|\bar{p}_1\| = \|(p_1, p_1)\| = 0$. If $j > 1$ we conclude $\|\bar{p}_1\| = \|(p_1, \dots, p_j)\| < 2^i$ from $\|(p_j, \dots, p_K)\| \geq 2^i$ and $\|\bar{p}\| < 2^{i+1}$. \square

In order to obtain a superset of all paths \bar{p} with $2^i \leq \|\bar{p}\| < 2^{i+1}$, we compute the OBDD $H_{i+1}(x, y, d)$ of all connected pairs $(v_{|x|}, v_{|y|}) \in V^2$ having $v_{|x|}$ - $v_{|y|}$ -paths of length $|d|$ that can be partitioned into parts \bar{p}_1 and \bar{p}_2 by means of Lemma 1.

$$H_{i+1}(x, y, d) := (\exists x^{(1)}, x^{(2)}, d^{(1)}, d^{(2)}, d^{(3)}) [(|d^{(1)}| + |d^{(2)}| + |d^{(3)}| = |d|) \wedge S_i(x, x^{(1)}, d^{(1)}) \wedge C(x^{(1)}, x^{(2)}, d^{(2)}) \wedge S_i(x^{(2)}, y, d^{(3)})] \quad (2)$$

Then, we restrict $H_{i+1}(x, y, d)$ to those triples (x, y, d) with $|d| < 2^{i+1}$ (3) and $\text{dist}(v_{|x|}, v_{|y|}) = |d|$ (4); i. e., there is no shorter $|d^{(1)}|$ fulfilling $H_{i+1}(x, y, d^{(1)})$.

$$H'_{i+1}(x, y, d) := H_{i+1}(x, y, d) \wedge (|d| < 2^{i+1}) \quad (3)$$

$$\wedge \overline{(\exists d^{(1)}) [(|d^{(1)}| < |d|) \wedge H_{i+1}(x, y, d^{(1)})]} \quad (4)$$

Finally, we cover paths shorter than 2^i by adding the previously computed $S_i(x, y, d)$ and obtain $S_{i+1}(x, y, d)$.

$$S_{i+1}(x, y, d) := S_i(x, y, d) \vee H'_{i+1}(x, y, d) \quad (5)$$

The output $S_n(x, y, d) = S(x, y, d)$ represents the OBDD for the all-pairs shortest-paths function dist .

4.2 Computing the Shortest Paths' Edges

Having computed $S(x, y, d)$, we are interested in the edges being part of shortest paths. We represent these by the OBDD $P(w, x, y, z)$ with

$$P(w, x, y, z) = 1 :\Leftrightarrow (v_{|w|}, v_{|x|}) \in E \text{ is part of a shortest } v_{|y|}\text{-}v_{|z|}\text{-path} .$$

Edge $(v_{|w|}, v_{|x|})$ lies on a shortest $v_{|y|}$ - $v_{|z|}$ -path iff $\text{dist}(v_{|y|}, v_{|w|}) + c(v_{|w|}, v_{|x|}) + \text{dist}(v_{|x|}, v_{|z|}) = \text{dist}(v_{|y|}, v_{|z|})$. This is expressed as

$$P(w, x, y, z) := (\exists d, d^{(1)}, d^{(2)}, d^{(3)}) [(|d^{(1)}| + |d^{(2)}| + |d^{(3)}| = |d|) \wedge S(y, w, d^{(1)}) \wedge C(w, x, d^{(2)}) \wedge S(x, z, d^{(3)}) \wedge S(y, z, d)] \quad (6)$$

We now consider the number of OBDD-operations the APSP-algorithm requires.

Theorem 1. *The symbolic APSP-algorithm computes the functions $S(x, y, d)$ and $P(w, x, y, z)$ by $\Theta(n^2) = \Theta(\log^2(NB))$ OBDD-operations.*

Proof. The computation of each OBDD $S_i(x, y, d)$, $i \in \{1, \dots, n\}$, as well as $P(w, x, y, z)$ consists of a constant number of cheap operations, argument reorderings, binary syntheses, and quantifications over node numbers or distance values. Each such quantification involves $\Theta(n)$ cheap operations and binary syntheses. Due to $n = \Theta(\log(NB))$, a number of $\Theta(n^2) = \Theta(\log^2(NB))$ OBDD-operations is executed. \square

Remark 1. Heuristic methods may behave much worse than the best known general methods in the worst case. Most papers on symbolic methods do not contain any worst-case bounds, because these are not considered as representative. Taking into account that the OBDD-size of any function $f \in B_n$ is bounded by $(2 + o(1))2^n/n$, the pseudopolynomial bounds of $\mathcal{O}(N^{16}B^8 \log^2(NB))$ on runtime and $\mathcal{O}(N^{16}B^8)$ on space are obtained for the symbolic APSP-algorithm.

4.3 Computing Concrete Shortest Paths

In order to obtain the edges of a concrete shortest $v_{|y^*|}-v_{|z^*|}$ -path for fixed y^* and z^* , different methods can be used. A straight-forward method to construct the path nodes $v_{|y^*|} = p_1, \dots, p_K = v_{|z^*|}$ in time $\mathcal{O}(K \cdot \text{size}(P))$ is to replace the y - and z -variables in $P(w, x, y, z)$ by the corresponding Boolean constants y^* and z^* . Then, we maintain a current node number w^* (starting with $w^* := y^*$) which replaces the argument w . The resulting OBDD $P(x)$ depends only on the target node of an edge $(v_{|w^*|}, \cdot)$ being part of a shortest path $(p_1, \dots, v_{|w^*|}, \dots, p_K)$. In time $\mathcal{O}(\text{size}(P))$ we obtain an arbitrary satisfying assignment of x , which becomes the new actual w^* . This is repeated until $w^* = z^*$.

Alternatively, the shortest $v_{|y^*|}-v_{|z^*|}$ -path can be computed by the symbolic blocking-flow construction method presented in [20]. This performs only $\mathcal{O}(\log^2 N)$ OBDD-operations independent from $K = \mathcal{O}(N)$, while it may cause an exponential blow-up of the OBDD-sizes.

5 Bounded-Width Functions

Symbolic algorithms are well established in logic synthesis because they often behave better than explicit methods on interesting instances [11–15, 24]. To be efficient w. r. t. the size of an input graph (i. e., the number of nodes and edges), this graph must have a compact OBDD-representation. The latter in turn is a property of the input and does not depend on the algorithm itself. Therefore, it is reasonable to investigate the behavior of symbolic methods w. r. t. the input's and output's OBDD-size.

Unfortunately, a number of $\Theta(n)$ quantification operations applied on a characteristic function $f \in B_n$ may suffice to cause an exponential blow-up of its OBDD-size, which makes it difficult to analyze symbolic algorithms. Moreover, Feigenbaum et al. [8] proved that even the basic problem of reachability analysis on OBDD-represented graphs is PSPACE-complete. So in in most papers the usability of symbolic algorithms is just proved by experiments on benchmark inputs from special application areas [13–15, 17, 27]. In other works considering more general graph problems, mostly the number of OBDD-operations (often referred to as “symbolic steps”) is bounded as a hint on the actual runtime [2, 9, 10, 18].

Therefore, we propose to consider a class of characteristic functions that enables statements on the over-all runtime and space usage of the symbolic APSP-algorithm, and which has also been successfully used in the analysis of symbolic topological sorting [26] and maximum flow algorithms [20–22].

Definition 3. A π -OBDD for a function $f \in B_n$ is called complete if every path from its function pointer to a sink has length n .

That is, complete OBDDs are not allowed to skip variable tests. The minimal-size complete π -OBDD for $f \in B_n$ is also known to be canonical [24] and will be denoted by $\pi\mathcal{G}_c[f]$ in the following. Figure 1(b) shows the minimal complete OBDD for an example function.

Definition 4. Let $F := (f_n)_{n \in \mathbb{N}}$ be a sequence of functions $f_n \in B_{\mathcal{N}(n)}$, $\mathcal{N}: \mathbb{N} \rightarrow \mathbb{N}$, defined on variables $x_0, \dots, x_{\mathcal{N}(n)-1}$. Moreover, let $\Pi := (\pi_n)_{n \in \mathbb{N}}$ be a sequence of variable orders $\pi_n \in \Sigma_{\mathcal{N}(n)}$. F has bounded width b w. r. t. Π (F is b -bounded by Π) iff for all $n \in \mathbb{N}$ the OBDD $\pi_n\mathcal{G}_c[f_n]$ contains no more than b nodes labeled with the same variable x_i for $i \in \{0, \dots, \mathcal{N}(n) - 1\}$.

Note that $\pi_n\mathcal{G}[f_n] \leq \pi_n\mathcal{G}_c[f_n] = \mathcal{O}(\mathcal{N}(n)b)$.

Theorem 2 (Sawitzki [23]). Let $F^{(1)} := (f_n^{(1)})_{n \in \mathbb{N}}$ and $F^{(2)} := (f_n^{(2)})_{n \in \mathbb{N}}$ be sequences of functions $f_n^{(1)}, f_n^{(2)} \in B_{k\mathcal{N}(n)}$, $k \in \mathbb{N}$, $\mathcal{N}: \mathbb{N} \rightarrow \mathbb{N}$, defined on variables $x^{(1)}, \dots, x^{(k)} \in \{0, 1\}^{\mathcal{N}(n)}$. Assume that $F^{(1)}$ and $F^{(2)}$ have bounded width b_1 resp. b_2 w. r. t. variable orders $\Pi := (\pi_n)_{n \in \mathbb{N}}$, $\pi_n \in \Sigma_{k\mathcal{N}(n)}$.

1. (Binary Synthesis)

For all $n \in \mathbb{N}$, the OBDD $\pi_n\mathcal{G}[f_n^{(1)} \otimes f_n^{(2)}]$, $\otimes \in B_2$, can be computed in time and space $\mathcal{O}(k\mathcal{N}(n)b_1b_2)$. The resulting sequence $(f_n^{(1)} \otimes f_n^{(2)})_{n \in \mathbb{N}}$ is b_1b_2 -bounded by Π .

2. (Quantification)

Let $X := (X_n)_{n \in \mathbb{N}}$ be a sequence of variable sets $X_n \subseteq \{x_i^{(j)} \mid i \in \{0, \dots, \mathcal{N}(n) - 1\}, j \in \{1, \dots, k\}\}$. For all $n \in \mathbb{N}$, the OBDD $\pi_n\mathcal{G}[(\mathcal{Q}X_n)f_n^{(1)}]$, $\mathcal{Q} \in \{\exists, \forall\}$, can be computed in time and space $\mathcal{O}(|X_n|k\mathcal{N}(n)2^{2b_1})$. The resulting sequence $((\mathcal{Q}X_n)f_n^{(1)})_{n \in \mathbb{N}}$ is 2^{b_1} -bounded by Π .

3. (Argument Reordering)

Let $\rho \in \Sigma_k$ and assume that Π is k -interleaved. For all $n \in \mathbb{N}$, the OBDD $\pi_n\mathcal{G}[\mathcal{R}_\rho(f_n^{(1)})]$ can be computed in time and space $\mathcal{O}(\mathcal{N}(n)b_1k^32^k)$. The resulting sequence $(\mathcal{R}_\rho(f_n^{(1)}))_{n \in \mathbb{N}}$ is b_12^k -bounded by Π .

The resulting width bounds are worst cases. However, because b_1 , b_2 , and k are independent of n , each operation takes linear time and space w. r. t. the number $\mathcal{N}(n)$ of variables. We conclude that bounded-width functions are closed under all operations used by the symbolic APSP-algorithm.

Theorem 3 (Woelfel [26]). Let $F := (f_n)_{n \in \mathbb{N}}$ be a sequence of functions $f_n \in B_{k\mathcal{N}(n)}$, $k \in \mathbb{N}$, $\mathcal{N}: \mathbb{N} \rightarrow \mathbb{N}$, and $\Pi := (\pi_n)_{n \in \mathbb{N}}$ k -interleaved variable orders $\pi_n \in \Sigma_{k\mathcal{N}(n)}$ with increasing bit significance. If for all $n \in \mathbb{N}$ it is $f_n \in \mathbb{T}_{k,\mathcal{N}(n)}^W$ then F is $\mathcal{O}(k^2W)$ -bounded by Π .

Theorem 3 implies that the comparison functions introduced in Sect. 3 are bounded-width functions.

5.1 Analysis on Graphs with Characteristic Bounded-Width Functions

Consider a sequence $G = (G_n)_{n \in \mathbb{N}}$ of valid input graphs $G_n = (V_n, E_n, c_n)$ for the symbolic APSP-algorithm. Assume that G_n has $N_n := |V_n|$ nodes and maximum edge weight $B_n \in \mathbb{N}_{>0}$. Let $C = (C(x, y, d)_n)_{n \in \mathbb{N}}$ be the sequence of G 's characteristic functions and $S = (S(x, y, d)_n)_{n \in \mathbb{N}}$ be the characteristic functions of G 's shortest path distances $(\text{dist}_n)_{n \in \mathbb{N}}$. Let $\mathcal{N}(n) = \Theta(\log(N_n B_n))$ be the number of bits encoding one node number $|x|$ or distance value $|d| \leq B_n(N_n - 1)$ of G_n . Moreover, assume a sequence $\Pi := (\pi_n)_{n \in \mathbb{N}}$ of interleaved variable orders which read bits of distance values with increasing significance.

Theorem 4. *If both C and S are b -bounded by Π , the symbolic APSP-algorithm computes $S(x, y, d)_n$ from $C(x, y, d)_n$ in time $\mathcal{O}(\log^3(N_n B_n) \cdot \alpha(b))$ and space $\mathcal{O}(\log(N_n B_n) \cdot \alpha(b))$ for all $n \in \mathbb{N}$ and*

$$\alpha(b) := 2^{2^{\mathcal{O}(b^3)}}.$$

Proof. All characteristic functions are defined on a constant number of binary node and distance numbers. Hence, the over-all number of Boolean variables is $\Theta(\log(N_n B_n))$ and reordering causes only a linear width growth to bounded-width functions. We show that all occurring functions are $\alpha(b)$ -bounded by Π . (A similar analysis technique has been used in [16].)

Using Π enables to realize the comparison $(|d| < 2^i)$ by multivariate threshold functions of $\mathbb{T}_{1, \mathcal{N}(n)}^{\mathcal{O}(1)}$ with width bound $\mathcal{O}(1)$ (see Theorem 3). From $S_i(x, y, d)_n = S(x, y, d)_n \wedge (|d| < 2^i)$, $i \in \{1, \dots, \mathcal{N}(n)\}$, and the width bound b of S we conclude that S_i is $\mathcal{O}(b)$ -bounded (see Theorem 2).

It remains to show that each intermediate result is $\alpha(b)$ -bounded. In (1), $S_1(x, y, d)_n$ is initialized by three syntheses involving comparisons and the input OBDD $C(x, y, d)_n$. Analogue to S_i , each occurring OBDD has bounded width $\mathcal{O}(b)$.

In (2), three binary syntheses are performed before the existential quantifications. Being a composition of multivariate threshold functions, the comparison $(|d^{(1)}| + |d^{(2)}| + |d^{(3)}| = |d|)$ is $\mathcal{O}(1)$ -bounded. Due to Theorem 2, each intermediate conjunction result is $\mathcal{O}(b^3)$ -bounded, whereas each quantification result (including H_{i+1}) is $2^{\mathcal{O}(b^3)}$ -bounded.

At next, $H'_{i+1}(x, y, d)_n$ is obtained by restricting $H_{i+1}(x, y, d)_n$ in (3). The conjunctions with $(|d| < 2^{i+1})$ resp. $(|d^{(1)}| < |d|)$ do not change the asymptotical width bound $2^{\mathcal{O}(b^3)}$. Finally, the quantification $(\exists d^{(1)})$ causes one further exponentiation and the new width bound is $2^{ab^3} \cdot 2^{2^{ab^3}} = \alpha(b)$ for an appropriate constant a . This still holds after disjunction with $S_i(x, y, d)_n$ due to S_i being $\mathcal{O}(b)$ -bounded.

Hence, all occurring characteristic functions are $\alpha(b)$ -bounded and have π_n -OBDD-size $\mathcal{O}(\log(N_n B_n) \cdot \alpha(b))$. Due to Theorem 2, each of the $\Theta(\log^2(N_n B_n))$ executed OBDD-operations takes time and space $\mathcal{O}(\log(N_n B_n) \cdot \alpha(b))$, which implies an over-all runtime bound of $\mathcal{O}(\log^3(N_n B_n) \cdot \alpha(b))$. Because only a constant

number of OBDDs has to be stored at any time, the over-all space usage is of the same magnitude as each single OBDD-size. \square

Analogue to this proof, a width bound of $2^{\mathcal{O}(b^4)}$ can be obtained for $P(w,x,y,z)$. Moreover, a less elegant formulation of (4) improves $\alpha(b)$ to $2^{\mathcal{O}(b^3)}$.

Corollary 1. *If both C and S are b -bounded by Π , the symbolic APSP-algorithm computes $P(w, x, y, z)_n$ from $S(x, y, d)_n$ in time $\mathcal{O}(\log^2(N_n B_n) \cdot 2^{\mathcal{O}(b^4)})$ and space $\mathcal{O}(\log(N_n B_n) \cdot 2^{\mathcal{O}(b^4)})$.*

How to classify this result? It is desirable that symbolic algorithms behave efficiently on “small” input OBDDs, which could be defined most general by being polynomial in the number $\mathcal{N}(n) = \Theta(\log(N_n B_n))$ of Boolean variables. Theorem 4 can be considered as showing this convenient property for the more restricted case of bounded-width functions, whose π_n -OBDD-size is even linear in $\mathcal{N}(n)$; here, “efficiently” means polylogarithmic w. r. t. N_n and B_n .

Hence, the symbolic APSP-algorithm can be considered as being *fixed-parameter tractable* [7] for the parameter b of characteristic b -bounded functions.

5.2 Composition of Graphs with Characteristic Bounded-Width Functions

Having the results on bounded-width functions, we ask what kinds of graphs can be represented by them. Obviously, sequences $G = (G_n)_{n \in \mathbb{N}}$ consisting of a single graph $G_1 = \dots = G_n, n \in \mathbb{N}$, have characteristic bounded-width functions. We already know multivariate threshold functions to have bounded width. These in turn can be used to build many simple sequences like empty, complete, complete bipartite, and grid graphs [23]. From the closedness under OBDD-operations we now conclude the closedness under four graph composition operations.

Let $G^{(i)} := (G_n^{(i)} = (V_n^{(i)}, E_n^{(i)}, c_n^{(i)}))_{n \in \mathbb{N}}, i \in \{1, 2, 3\}$, be sequences of valid input graphs for the symbolic APSP-algorithm with same notation as G in Sect. 5.1. Assume $V_n^{(1)} \cap V_n^{(2)} = \emptyset$ for all $n \in \mathbb{N}$.

Definition 5. *Graph Composition Operations.*

1. $G^{(3)}$ is called the cojoin of $G^{(1)}$ and $G^{(2)}$ iff for all $n \in \mathbb{N}$ it is $V_n^{(3)} = V_n^{(1)} \cup V_n^{(2)}, E_n^{(3)} = E_n^{(1)} \cup E_n^{(2)}$, and $c_n^{(3)}(e) = c_n^{(i)}(e)$ for $e \in E_n^{(i)}$.
2. $G^{(3)}$ is called the \mathcal{A} -join of $G^{(1)}$ and $G^{(2)}$, $\mathcal{A}: \mathbb{N} \rightarrow \mathbb{N}_{>0}$, iff for all $n \in \mathbb{N}$ it is $V_n^{(3)} = V_n^{(1)} \cup V_n^{(2)}, E_n^{(3)} = E_n^{(1)} \cup E_n^{(2)} \cup (V_n^{(1)} \times V_n^{(2)})$, and $c_n^{(3)}(e) = c_n^{(i)}(e)$ for $e \in E_n^{(i)}$ resp. $c_n^{(3)}(e) = \mathcal{A}(n)$ for $e \in V_n^{(1)} \times V_n^{(2)}$.
3. $G^{(3)}$ is called the node substitution of $G^{(1)}$ in $G^{(2)}$ iff for all $n \in \mathbb{N}$ it is $V_n^{(3)} = V_n^{(1)} \times V_n^{(2)}$,

$$E_n^{(3)} = \left\{ ((t, u), (v, w)) \mid ((t, v) \in E_n^{(1)} \wedge (u = w)) \vee (u, w) \in E_n^{(2)} \right\},$$

and $c_n^{(3)}$ weights edge $((t, u), (v, w))$ with $c_n^{(1)}(t, v)$ if $u = w$ resp. $c_n^{(2)}(u, w)$ if $(u, w) \in E_n^{(2)}$.

4. $G^{(3)}$ is called the product of $G^{(1)}$ and $G^{(2)}$ iff for all $n \in \mathbb{N}$ it is $V_n^{(3)} = V_n^{(1)} \times V_n^{(2)}$,

$$E_n^{(3)} = \left\{ ((t, u), (v, w)) \mid ((t, v) \in E_n^{(1)} \wedge (u = w)) \vee ((t = v) \wedge (u, w) \in E_n^{(2)}) \right\},$$

and $c_n^{(3)}$ weights edge $((t, u), (v, w))$ with $c_n^{(1)}(t, v)$ if $u = w$ resp. $c_n^{(2)}(u, w)$ if $t = v$.

Theorem 5. Let $G^{(3)}$ be the product of $G^{(1)}$ and $G^{(2)}$. If $C^{(i)}$ and $S^{(i)}$ are b -bounded by Π for $i \in \{1, 2\}$, then $C^{(3)}$ is $\mathcal{O}(b^2)$ -bounded by Π and $S^{(3)}$ is $2^{\mathcal{O}(b^2)}$ -bounded by Π .

Proof. Let $(x^{(1)}, x^{(2)})$ denote the binary node number of $(v_{|x^{(1)}|}, v_{|x^{(2)}|}) \in V_n^{(3)}$ corresponding to argument x of a characteristic function. A shortest path in the product $G_n^{(3)}$ is composed of shortest paths in $G_n^{(1)}$ and $G_n^{(2)}$. We express $C^{(3)}$ and $S^{(3)}$ in terms of $C^{(1)}$, $C^{(2)}$, $S^{(1)}$, and $S^{(2)}$:

$$C^{(3)}(x^{(1)}, x^{(2)}, y^{(1)}, y^{(2)}, d)_n = [C^{(1)}(x^{(1)}, y^{(1)}, d)_n \wedge (x^{(2)} = y^{(2)})] \vee [(x^{(1)} = y^{(1)}) \wedge C^{(2)}(x^{(2)}, y^{(2)}, d)_n],$$

$$S^{(3)}(x^{(1)}, x^{(2)}, y^{(1)}, y^{(2)}, d)_n = (\exists d^{(1)}, d^{(2)}) [(|d^{(1)}| + |d^{(2)}| = |d|) \wedge S^{(1)}(x^{(1)}, y^{(1)}, d^{(1)})_n \wedge S^{(2)}(x^{(2)}, y^{(2)}, d^{(2)})_n].$$

Analogue to S_i in Theorem 4, we conclude $C^{(3)}$ to be $\mathcal{O}(b^2)$ -bounded. For $S^{(3)}$, the quantifiers are applied to an intermediate result of width $\mathcal{O}(b^2)$ and cause an exponentiation leading to the final bound of $2^{\mathcal{O}(b^2)}$. \square

The same width bounds can be obtained for the case of node substitution, whereas both $C^{(3)}$ and $S^{(3)}$ are $\mathcal{O}(b^2)$ -bounded if a cojoin or \mathcal{A} -join has been applied. That is, the efficiency results of Theorem 4 also hold for complex graphs builded from basic ones having characteristic bounded-width functions.

For the complete proofs, further composition operations, and a more comprehensive discussion of graphs with characteristic bounded-width functions, the reader is referred to [23].

6 A Reason for Restricting to Positive Edge Weights

The proof of Theorem 4 makes use of the fact that the intermediate results $S_i(x, y, d)$ can be expressed in terms of the final result $S(x, y, d)$ and the $\mathcal{O}(1)$ -bounded comparison $(|d| < 2^i)$ by $S_i(x, y, d) = S(x, y, d) \wedge (|d| < 2^i)$. For the correctness of the APSP-algorithm it is essential that only empty paths have length 0 due to the strictly positive edge weights.

Instead of computing $S(x, y, d)$ by iterating over the number $i \in \{1, \dots, n\}$ of considered distance bits, we could iteratively double the number of edges of considered paths. A corresponding recursion would be

$$\begin{aligned}
 H_{i+1}(x, y, d) &:= (\exists x^{(1)}, d^{(1)}, d^{(2)}) [(|d^{(1)}| + |d^{(2)}| = |d|) \\
 &\quad \wedge S_i(x, x^{(1)}, d^{(1)}) \wedge S_i(x^{(1)}, y, d^{(2)})] , \\
 S_{i+1}(x, y, d) &:= H_{i+1}(x, y, d) \wedge \overline{(\exists d^{(1)}) [(|d^{(1)}| < |d|) \wedge H_{i+1}(x, y, d^{(1)})] } ,
 \end{aligned}$$

where $S_i(u, v, d)$ now represents all shortest paths consisting of no more than 2^i edges. The resulting algorithm is able to handle graphs with general integral edge weights that contain no negative cycles. Nevertheless, it does not provide a counterpart to Theorem 4: There are graphs $G^* := (G_n^*)_{n \in \mathbb{N}}$ fulfilling the bounded-width conditions, but whose intermediate functions S_i have not bounded width in general.

The construction of G^* makes use of the fact that shortest paths may consist of many edges. If the shortest paths have more than 2^i edges, the functions S_i have to represent longer ones, which can be chosen such that the bounded width of S_i would also imply bounded width for the multiplication function MUL_n . This contradicts exponential lower bounds on the π -OBDD-size of MUL_n for every variable order π . For a detailed discussion, the reader is referred to [23].

There is no symbolic algorithm known to the author which is able to handle general integral edge weights and that has properties comparable to Theorem 4.

7 Related Problems

The symbolic APSP-algorithm can be easily adapted to compute the edges of almost shortest-paths of small stretch [5, 6]. This is done by replacing $P(w, x, y, z)$ (see (6)) by a function $P^{a,b}(w, x, y, z)$ representing edges $(v_{|w|}, v_{|x|})$ on $v_{|y|}^-v_{|z|}^-$ -paths \bar{p} of length $\|\bar{p}\| \leq a \cdot \text{dist}(v_{|y|}, v_{|z|}) + b$.

$$\begin{aligned}
 P^{a,b}(w, x, y, z) &:= (\exists d, d^{(1)}, d^{(2)}, d^{(3)}) [(|d^{(1)}| + |d^{(2)}| + |d^{(3)}| \leq a \cdot |d| + b) \\
 &\quad \wedge S(y, w, d^{(1)}) \wedge C(w, x, d^{(2)}) \wedge S(x, z, d^{(3)}) \wedge S(y, z, d)] .
 \end{aligned}$$

Due to the results on the bounded width of multivariate threshold functions, Corollary 1 holds for $P^{a,b}$, too.

Finally, we consider a dynamic scenario: After the computation of $S(x, y, d)$, the weights of an edge set $E' \subseteq E$ are decreased resulting in the new symbolic graph $C'(x, y, d)$. If every graph path contains at most one updated edge, the new distances $S'(x, y, d)$ can be computed by $\Theta(\log(NB))$ OBDD-operations. $v_{|x|}^-v_{|y|}^-$ -paths of length $|d|$ containing a decreased edge $(v_{x^{(1)}}, v_{x^{(2)}})$ are expressed as

$$\begin{aligned}
 F(x, y, d) &:= (\exists x^{(1)}, x^{(2)}, d^{(1)}, d^{(2)}, d^{(3)}) [(|d^{(1)}| + |d^{(2)}| + |d^{(3)}| = |d|) \\
 &\quad \wedge S(x, x^{(1)}, d^{(1)}) \wedge C'(x^{(1)}, x^{(2)}, d^{(2)}) \wedge S(x^{(2)}, y, d^{(3)})] .
 \end{aligned}$$

To obtain $S'(x, y, d)$, we just have to select the smallest $|d|$ with $S(x, y, d) \vee F(x, y, d) = 1$ similar to (4). Again, bounded width of C' and S imply time and space bounds as in Theorem 4.

8 Conclusions

We presented a symbolic algorithm for the all-pairs shortest-paths problem. The algorithm works on OBDD-representations of loopless directed graphs $G = (V, E, c)$ with strictly positive integral edge weights. It computes the lengths and edges of shortest paths by performing a polylogarithmic number of OBDD-operations w. r. t. $N := |V|$ and $B := \max\{c(e) \mid e \in E\}$.

In order to investigate runtime and space usage, bounded-width functions have been introduced, which have small OBDDs and allow efficient OBDD-operations. The algorithm is proved to have polylogarithmic runtime and space usage w. r. t. N and B on graphs whose characteristic functions have bounded width. This property is closed under important graph composition operations. In contrast, the bounded-width of input and output does not guarantee efficiency for symbolic algorithms which iterate over the number of edges of paths instead of their length.

Finally, adaptations of the symbolic APSP-algorithm to dynamic edge weights and almost shortest paths of small stretch have been briefly discussed.

Acknowledgments

Thanks to Thomas Franke and Ingo Wegener for proofreading and helpful discussions.

The author's technical reports can be obtained via his homepage.

References

1. R.I. Bahar, E.A. Frohm, C.M. Gaona, G.D. Hachtel, E. Macii, A. Pardo, and F. Somenzi. Algebraic decision diagrams and their applications. In *ICCAD'93*, pages 188–191. IEEE Press, 1993.
2. R. Bloem, H.N. Gabow, and F. Somenzi. An algorithm for strongly connected component analysis in $n \log n$ symbolic steps. In *FMCAD'00*, volume 1954 of *LNCS*, pages 37–54. Springer, 2000.
3. R.E. Bryant. Symbolic manipulation of Boolean functions using a graphical representation. In *DAC'85*, pages 688–694. ACM Press, 1985.
4. R.E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, 35:677–691, 1986.
5. E. Cohen and U. Zwick. All-pairs small-stretch paths. *Journal of Algorithms*, 38:335–353, 2001.
6. D. Dor, S. Halperin, and U. Zwick. All-pairs almost shortest paths. *SIAM Journal on Computing*, 29:1740–1759, 2000.
7. R.G. Downey and M.R. Fellows. *Parameterized Complexity*. Springer, Berlin Heidelberg New-York, 1999.

8. J. Feigenbaum, S. Kannan, M.Y. Vardi, and M. Viswanathan. Complexity of problems on graphs represented as OBDDs. *Chicago Journal of Theoretical Computer Science*, 1999, 1999.
9. R. Gentilini, C. Piazza, and A. Policriti. Computing strongly connected components in a linear number of symbolic steps. In *SODA'03*, pages 573–582. ACM Press, 2003.
10. R. Gentilini and A. Policriti. Biconnectivity on symbolically represented graphs: A linear solution. In *ISAAC'03*, volume 2906 of *LNCS*, pages 554–564. Springer, 2003.
11. G.D. Hachtel, M. Hermida, A. Pardo, M. Poncino, and F. Somenzi. Re-Encoding sequential circuits to reduce power dissipation. In *ICCAD'94*, pages 70–73. IEEE Press, 1994.
12. G.D. Hachtel and F. Somenzi. *Logic Synthesis and Verification Algorithms*. Kluwer Academic Publishers, Boston, 1996.
13. 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.
14. R. Hojati, H. Touati, R.P. Kurshan, and R.K. Brayton. Efficient ω -regular language containment. In *CAV'93*, volume 663 of *LNCS*, pages 396–409. Springer, 1993.
15. H. Jin, A. Kuehlmann, and F. Somenzi. Fine-grain conjunction scheduling for symbolic reachability analysis. In *TACAS'02*, volume 2280 of *LNCS*, pages 312–326. Springer, 2002.
16. M. Krause. BDD-Based cryptanalysis of keystream generators. In *EUROCRYPT'02*, volume 2332 of *LNCS*, pages 222–237. Springer, 2002.
17. I. Moon, J.H. Kukula, K. Ravi, and F. Somenzi. To split or to conjoin: The question in image computation. In *DAC'00*, pages 23–28. ACM Press, 2000.
18. K. Ravi, R. Bloem, and F. Somenzi. A comparative study of symbolic algorithms for the computation of fair cycles. In *FMCAD'00*, volume 1954 of *LNCS*, pages 143–160. Springer, 2000.
19. D. Sawitzki. Experimental studies of symbolic shortest-path algorithms. In *WEA'04*, volume 3059 of *LNCS*, pages 482–497. Springer, 2004.
20. D. Sawitzki. Implicit flow maximization by iterative squaring. In *SOFSEM'04*, volume 2932 of *LNCS*, pages 301–313. Springer, 2004.
21. D. Sawitzki. Implicit flow maximization on grid networks. Technical report, Universität Dortmund, 2004.
22. D. Sawitzki. Implicit maximization of flows over time. Technical report, Universität Dortmund, 2004.
23. D. Sawitzki. On graphs with characteristic bounded-width functions. Technical report, Universität Dortmund, 2004.
24. I. Wegener. *Branching Programs and Binary Decision Diagrams*. SIAM, Philadelphia, 2000.
25. P. Woelfel. The OBDD-size of cographs. Internal report, Universität Dortmund, 2003.
26. P. Woelfel. Symbolic topological sorting with OBDDs. In *MFCS'03*, volume 2747 of *LNCS*, pages 671–680. Springer, 2003.
27. A. Xie and P.A. Beerel. Implicit enumeration of strongly connected components. In *ICCAD'99*, pages 37–40. ACM Press, 1999.