

Algorithms on Implicit Networks

New Results on the OBDD-Size of Graphs

Robin Nunkesser, Daniel Sawitzki, and Philipp Woelfel

Computer Science 2
University of Dortmund, Germany

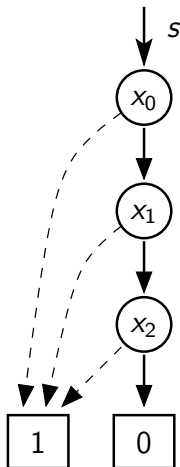
Jahreskolloquium DFG-Schwerpunkt 1126, 10.–12. März 2005

Overview

- 1 Branching Programs
- 2 Results for Graph Classes
- 3 Results for Graphs of Functions
- 4 Implicit Algorithm Analysis
- 5 Summary

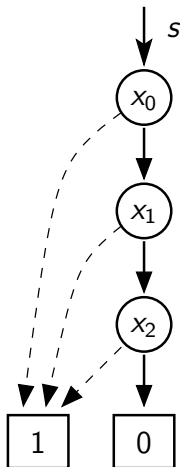
General Branching Programs

- **Branching Programs (BPs)** represent Boolean functions $f: \{0, 1\}^n \rightarrow \{0, 1\}$ on variables $x_0, \dots, x_{n-1} \in \{0, 1\}$.
- BP P is acyclic digraph with **inner nodes** and **sinks**.
- Inner nodes: Labeled with variable, left by 0- and 1-edge.
- Sinks correspond to $f(x_0, \dots, x_{n-1})$.
- Pointer marks **source node** s .



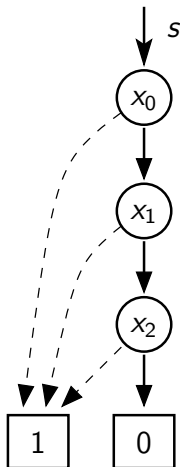
General Branching Programs

- **Branching Programs (BPs)** represent Boolean functions $f: \{0, 1\}^n \rightarrow \{0, 1\}$ on variables $x_0, \dots, x_{n-1} \in \{0, 1\}$.
- BP P is acyclic digraph with **inner nodes** and **sinks**.
 - Inner nodes: Labeled with variable, left by 0- and 1-edge.
 - Sinks correspond to $f(x_0, \dots, x_{n-1})$.
 - Pointer marks **source node** s .



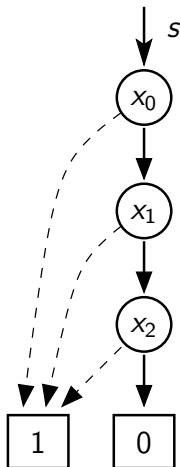
General Branching Programs

- **Branching Programs (BPs)** represent Boolean functions $f: \{0, 1\}^n \rightarrow \{0, 1\}$ on variables $x_0, \dots, x_{n-1} \in \{0, 1\}$.
- BP P is acyclic digraph with **inner nodes** and **sinks**.
- Inner nodes: Labeled with variable, left by 0- and 1-edge.
- Sinks correspond to $f(x_0, \dots, x_{n-1})$.
- Pointer marks **source node** s .



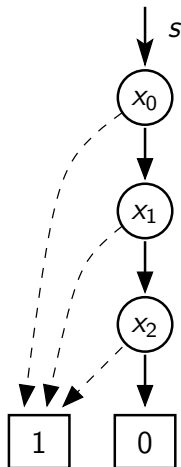
General Branching Programs

- **Branching Programs (BPs)** represent Boolean functions $f: \{0, 1\}^n \rightarrow \{0, 1\}$ on variables $x_0, \dots, x_{n-1} \in \{0, 1\}$.
- BP P is acyclic digraph with **inner nodes** and **sinks**.
- Inner nodes: Labeled with variable, left by 0- and 1-edge.
- Sinks correspond to $f(x_0, \dots, x_{n-1})$.
- Pointer marks **source node** s .



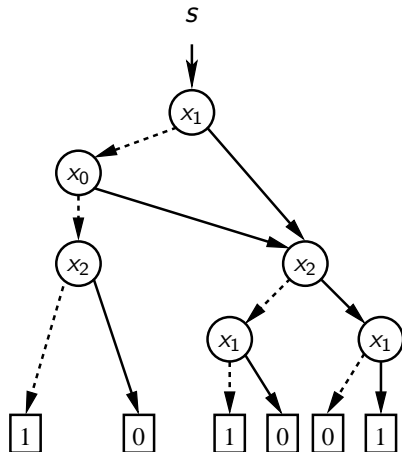
General Branching Programs

- **Branching Programs (BPs)** represent Boolean functions $f: \{0, 1\}^n \rightarrow \{0, 1\}$ on variables $x_0, \dots, x_{n-1} \in \{0, 1\}$.
- BP P is acyclic digraph with **inner nodes** and **sinks**.
- Inner nodes: Labeled with variable, left by 0- and 1-edge.
- Sinks correspond to $f(x_0, \dots, x_{n-1})$.
- Pointer marks **source node** s .



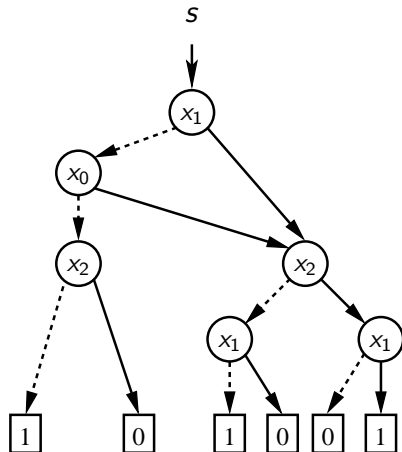
Oblivious Branching Programs and OBDDs

- Let $\pi \in \{x_0, \dots, x_{n-1}\}^*$.
- Consider variables $\sigma(p)$ on path p .
- BP P is π -oblivious $\Leftrightarrow \sigma(p)$ is subsequence of π for any path p .
- Example: $\pi = (x_1, x_0, x_2, x_1)$
- Bryant (1985): Ordered Binary Decision Diagrams (OBDDs)
- Restriction to permutations π



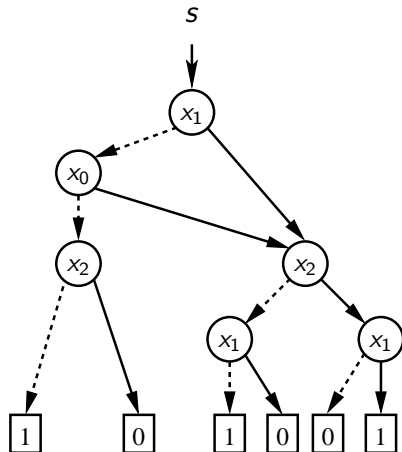
Oblivious Branching Programs and OBDDs

- Let $\pi \in \{x_0, \dots, x_{n-1}\}^*$.
- Consider variables $\sigma(p)$ on path p .
- BP P is π -oblivious $\Leftrightarrow \sigma(p)$ is subsequence of π for any path p .
- Example: $\pi = (x_1, x_0, x_2, x_1)$
- Bryant (1985): Ordered Binary Decision Diagrams (OBDDs)
- Restriction to permutations π



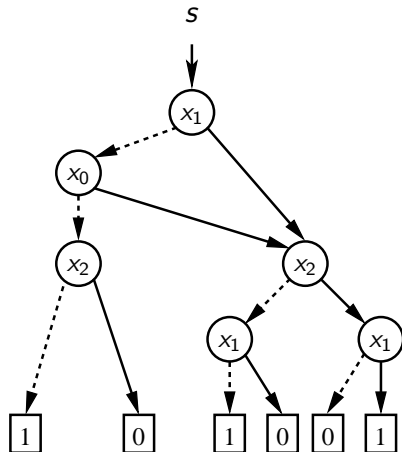
Oblivious Branching Programs and OBDDs

- Let $\pi \in \{x_0, \dots, x_{n-1}\}^*$.
- Consider variables $\sigma(p)$ on path p .
- BP P is **π -oblivious** : \Leftrightarrow $\sigma(p)$ is subsequence of π for any path p .
- Example: $\pi = (x_1, x_0, x_2, x_1)$
- Bryant (1985): **Ordered Binary Decision Diagrams (OBDDs)**
- Restriction to permutations π



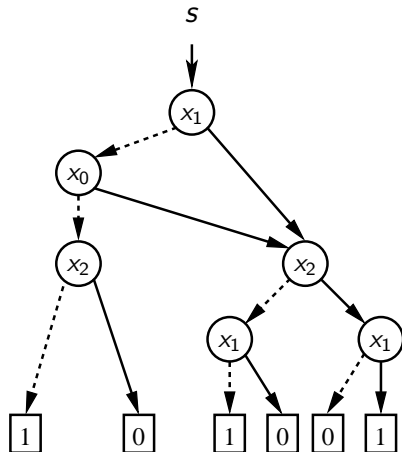
Oblivious Branching Programs and OBDDs

- Let $\pi \in \{x_0, \dots, x_{n-1}\}^*$.
- Consider variables $\sigma(p)$ on path p .
- BP P is **π -oblivious** : \Leftrightarrow $\sigma(p)$ is subsequence of π for any path p .
- Example: $\pi = (x_1, x_0, x_2, x_1)$
- Bryant (1985): **Ordered Binary Decision Diagrams (OBDDs)**
- Restriction to permutations π



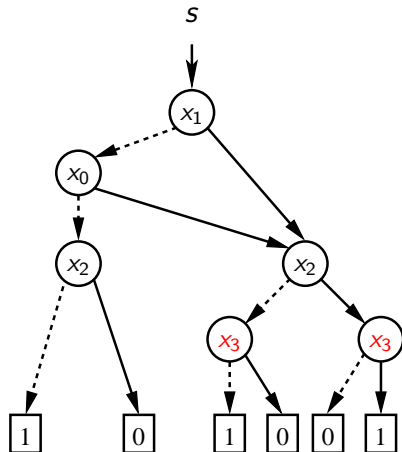
Oblivious Branching Programs and OBDDs

- Let $\pi \in \{x_0, \dots, x_{n-1}\}^*$.
- Consider variables $\sigma(p)$ on path p .
- BP P is **π -oblivious** : \Leftrightarrow $\sigma(p)$ is subsequence of π for any path p .
- Example: $\pi = (x_1, x_0, x_2, x_1)$
- Bryant (1985): **Ordered Binary Decision Diagrams (OBDDs)**
- Restriction to permutations π



Oblivious Branching Programs and OBDDs

- Let $\pi \in \{x_0, \dots, x_{n-1}\}^*$.
- Consider variables $\sigma(p)$ on path p .
- BP P is **π -oblivious** : \Leftrightarrow $\sigma(p)$ is subsequence of π for any path p .
- Example: $\pi = (x_1, x_0, x_2, x_1)$
- Bryant (1985): **Ordered Binary Decision Diagrams (OBDDs)**
- Restriction to permutations π



Applications of Branching Programs

- Complexity theory:
 - Lower bounds on space complexity of computations
 - Tradeoffs on time and space for sequential machines
- Compact graph representation
- Efficient algorithms:
 - CAD, model checking
 - Implicit graph algorithms
(max. flows, shortest paths, top. sorting, ...)
 - Favored variant: OBDDs due to efficient operations
(\wedge , \vee , \exists , \forall , SAT, \equiv , ...)

Applications of Branching Programs

- Complexity theory:
 - Lower bounds on space complexity of computations
 - Tradeoffs on time and space for sequential machines
- Compact graph representation
- Efficient algorithms:
 - CAD, model checking
 - Implicit graph algorithms
(max. flows, shortest paths, top. sorting, ...)
 - Favored variant: OBDDs due to efficient operations
(\wedge , \vee , \exists , \forall , SAT, \equiv , ...)

Applications of Branching Programs

- Complexity theory:
 - Lower bounds on space complexity of computations
 - Tradeoffs on time and space for sequential machines
- Compact graph representation
- Efficient algorithms:
 - CAD, model checking
 - Implicit graph algorithms
(max. flows, shortest paths, top. sorting, ...)
 - Favored variant: OBDDs due to efficient operations
(\wedge , \vee , \exists , \forall , SAT, \equiv , ...)

Applications of Branching Programs

- Complexity theory:
 - Lower bounds on space complexity of computations
 - Tradeoffs on time and space for sequential machines
- Compact graph representation
- Efficient algorithms:
 - CAD, model checking
 - Implicit graph algorithms
(max. flows, shortest paths, top. sorting, ...)
 - Favored variant: OBDDs due to efficient operations
(\wedge , \vee , \exists , \forall , SAT, \equiv , ...)

Applications of Branching Programs

- Complexity theory:
 - Lower bounds on space complexity of computations
 - Tradeoffs on time and space for sequential machines
- Compact graph representation
- Efficient algorithms:
 - CAD, model checking
 - Implicit graph algorithms
(max. flows, shortest paths, top. sorting, ...)
 - Favored variant: OBDDs due to efficient operations
(\wedge , \vee , \exists , \forall , SAT, \equiv , ...)

Applications of Branching Programs

- Complexity theory:
 - Lower bounds on space complexity of computations
 - Tradeoffs on time and space for sequential machines
- Compact graph representation
- Efficient algorithms:
 - CAD, model checking
 - Implicit graph algorithms
(max. flows, shortest paths, top. sorting, ...)
 - Favored variant: OBDDs due to efficient operations
(\wedge , \vee , \exists , \forall , SAT, \equiv , ...)

Applications of Branching Programs

- Complexity theory:
 - Lower bounds on space complexity of computations
 - Tradeoffs on time and space for sequential machines
- Compact graph representation
- Efficient algorithms:
 - CAD, model checking
 - Implicit graph algorithms
(max. flows, shortest paths, top. sorting, ...)
 - Favored variant: OBDDs due to efficient operations
(\wedge , \vee , \exists , \forall , SAT, \equiv , ...)

Applications of Branching Programs

- Complexity theory:
 - Lower bounds on space complexity of computations
 - Tradeoffs on time and space for sequential machines
- Compact graph representation
- Efficient algorithms:
 - CAD, model checking
 - Implicit graph algorithms
(max. flows, shortest paths, top. sorting, ...)
 - Favored variant: OBDDs due to efficient operations
(\wedge , \vee , \exists , \forall , SAT, \equiv , ...)

Applications of Branching Programs

- Complexity theory:
 - Lower bounds on space complexity of computations
 - Tradeoffs on time and space for sequential machines
- Compact graph representation
- Efficient algorithms:
 - CAD, model checking
 - Implicit graph algorithms
(max. flows, shortest paths, top. sorting, ...)
 - Favored variant: OBDDs due to efficient operations
(\wedge , \vee , \exists , \forall , SAT, \equiv , ...)

Implicit Graph Representation with OBDDs

Consider digraph $G = (V, E)$ with nodes $v_0, \dots, v_{|V|-1}$,
 $\log |V| = n \in \mathbb{N}$

Definition

Characteristic Boolean function $\chi_E: \{0, 1\}^{2n} \rightarrow \{0, 1\}$ is defined by

$$\chi_E(x, y) = 1 :\Leftrightarrow (v_{|x|}, v_{|y|}) \in E .$$

OBDD-size of χ_E ?

- For any f on n vars.: OBDD-size(f) $\leq (2 + o(1))2^n/n$
- \Rightarrow OBDD-size(χ_E) = $\mathcal{O}(|V|^2 / \log |V|)$
- OBDD-size(χ_E) = $\mathcal{O}(|E| \log |V|)$

Implicit Graph Representation with OBDDs

Consider digraph $G = (V, E)$ with nodes $v_0, \dots, v_{|V|-1}$,
 $\log |V| = n \in \mathbb{N}$

Definition

Characteristic Boolean function $\chi_E: \{0, 1\}^{2n} \rightarrow \{0, 1\}$ is defined by

$$\chi_E(x, y) = 1 :\Leftrightarrow (v_{|x|}, v_{|y|}) \in E .$$

OBDD-size of χ_E ?

- For any f on n vars.: $\text{OBDD-size}(f) \leq (2 + o(1))2^n/n$
- $\Rightarrow \text{OBDD-size}(\chi_E) = \mathcal{O}(|V|^2 / \log |V|)$
- $\text{OBDD-size}(\chi_E) = \mathcal{O}(|E| \log |V|)$

Implicit Graph Representation with OBDDs

Consider digraph $G = (V, E)$ with nodes $v_0, \dots, v_{|V|-1}$,
 $\log |V| = n \in \mathbb{N}$

Definition

Characteristic Boolean function $\chi_E: \{0, 1\}^{2n} \rightarrow \{0, 1\}$ is defined by

$$\chi_E(x, y) = 1 :\Leftrightarrow (v_{|x|}, v_{|y|}) \in E .$$

OBDD-size of χ_E ?

- For any f on n vars.: $\text{OBDD-size}(f) \leq (2 + o(1))2^n/n$
- $\Rightarrow \text{OBDD-size}(\chi_E) = \mathcal{O}(|V|^2 / \log |V|)$
- $\text{OBDD-size}(\chi_E) = \mathcal{O}(|E| \log |V|)$

Implicit Graph Representation with OBDDs

Consider digraph $G = (V, E)$ with nodes $v_0, \dots, v_{|V|-1}$,
 $\log |V| = n \in \mathbb{N}$

Definition

Characteristic Boolean function $\chi_E: \{0, 1\}^{2n} \rightarrow \{0, 1\}$ is defined by

$$\chi_E(x, y) = 1 :\Leftrightarrow (v_{|x|}, v_{|y|}) \in E .$$

OBDD-size of χ_E ?

- For any f on n vars.: $\text{OBDD-size}(f) \leq (2 + o(1))2^n/n$
- $\Rightarrow \text{OBDD-size}(\chi_E) = \mathcal{O}(|V|^2/\log |V|)$
- $\text{OBDD-size}(\chi_E) = \mathcal{O}(|E| \log |V|)$

Implicit Graph Representation with OBDDs

Consider digraph $G = (V, E)$ with nodes $v_0, \dots, v_{|V|-1}$,
 $\log |V| = n \in \mathbb{N}$

Definition

Characteristic Boolean function $\chi_E: \{0, 1\}^{2n} \rightarrow \{0, 1\}$ is defined by

$$\chi_E(x, y) = 1 :\Leftrightarrow (v_{|x|}, v_{|y|}) \in E .$$

OBDD-size of χ_E ?

- For any f on n vars.: $\text{OBDD-size}(f) \leq (2 + o(1))2^n/n$
- $\Rightarrow \text{OBDD-size}(\chi_E) = \mathcal{O}(|V|^2 / \log |V|)$
- $\text{OBDD-size}(\chi_E) = \mathcal{O}(|E| \log |V|)$

Results for Graph Classes

Theorem (Nunkesser and Woelfel, 2005)

*Bounds on the OBDD-size of fundamental graph classes
($N := |V|$).*

	<i>Lower ($-\mathcal{O}(1)$)</i>	<i>Upper</i>
<i>Cographs</i>	$1.832 \frac{N}{\log N}$	$3N \lceil \log N \rceil$

Results for Graph Classes

Theorem (Nunkesser and Woelfel, 2005)

Bounds on the OBDD-size of fundamental graph classes
 ($N := |V|$).

	<i>Lower</i> ($-\mathcal{O}(1)$)	<i>Upper</i>
<i>Cographs</i>	$1.832 \frac{N}{\log N}$	$3N \lceil \log N \rceil$
<i>P_4-sparse</i>	$1.832 \frac{N}{\log N}$	$18N \lceil \log N \rceil - 10N + \mathcal{O}(1)$

Results for Graph Classes

Theorem (Nunkesser and Woelfel, 2005)

Bounds on the OBDD-size of fundamental graph classes
 ($N := |V|$).

	<i>Lower</i> ($-\mathcal{O}(1)$)	<i>Upper</i>
<i>Cographs</i>	$1.832 \frac{N}{\log N}$	$3N \lceil \log N \rceil$
<i>P_4-sparse</i>	$1.832 \frac{N}{\log N}$	$18N \lceil \log N \rceil - 10N + \mathcal{O}(1)$
<i>P_4-extendible</i>	$1.832 \frac{N}{\log N}$	$270N \lceil \log N \rceil - 178N + \mathcal{O}(1)$

Results for Graph Classes

Theorem (Nunkesser and Woelfel, 2005)

Bounds on the OBDD-size of fundamental graph classes
 ($N := |V|$).

	<i>Lower</i> ($-\mathcal{O}(1)$)	<i>Upper</i>
<i>Cographs</i>	$1.832 \frac{N}{\log N}$	$3N \lceil \log N \rceil$
<i>P_4-sparse</i>	$1.832 \frac{N}{\log N}$	$18N \lceil \log N \rceil - 10N + \mathcal{O}(1)$
<i>P_4-extendible</i>	$1.832 \frac{N}{\log N}$	$270N \lceil \log N \rceil - 178N + \mathcal{O}(1)$
<i>Interval</i>	$(1 - \epsilon)N$	$(32 + o(1)) \frac{N^{3/2}}{\log^{3/4} N}$

Results for Graph Classes

Theorem (Nunkesser and Woelfel, 2005)

Bounds on the OBDD-size of fundamental graph classes
 ($N := |V|$).

	<i>Lower</i> ($-\mathcal{O}(1)$)	<i>Upper</i>
<i>Cographs</i>	$1.832 \frac{N}{\log N}$	$3N \lceil \log N \rceil$
<i>P_4-sparse</i>	$1.832 \frac{N}{\log N}$	$18N \lceil \log N \rceil - 10N + \mathcal{O}(1)$
<i>P_4-extendible</i>	$1.832 \frac{N}{\log N}$	$270N \lceil \log N \rceil - 178N + \mathcal{O}(1)$
<i>Interval</i>	$(1 - \epsilon)N$	$(32 + o(1)) \frac{N^{3/2}}{\log^{3/4} N}$
<i>Bipartite</i>	$(1/8 - \epsilon) \frac{N^2}{\log N}$	—

Graphs of Boolean Functions

Definition

Consider m functions $f = (f_0, \dots, f_{m-1})$ with $f_i: \{0, 1\}^n \rightarrow \{0, 1\}$.
 The **graph of f** called f -GRAPH is defined by

$$f\text{-GRAPH}(x_0, \dots, x_{n-1}, y_0, \dots, y_{m-1}) := \bigwedge_{i=0}^{m-1} [f_i(x) = y_i] .$$

- Fundamental functions with exp. OBDD-size:
 - Middle Bit of Multiplication, Indirect Storage Access, Hidden Weighted Bit
- Do exp. lower bounds carry over to graphs of functions?

Not in general! But for MUL, ISA, and HWB.

Graphs of Boolean Functions

Definition

Consider m functions $f = (f_0, \dots, f_{m-1})$ with $f_i: \{0, 1\}^n \rightarrow \{0, 1\}$.
 The **graph of f** called f -GRAPH is defined by

$$f\text{-GRAPH}(x_0, \dots, x_{n-1}, y_0, \dots, y_{m-1}) := \bigwedge_{i=0}^{m-1} [f_i(x) = y_i] .$$

- Fundamental functions with exp. OBDD-size:
 - Middle Bit of Multiplication, Indirect Storage Access, Hidden Weighted Bit
- Do exp. lower bounds carry over to graphs of functions?

Not in general! But for MUL, ISA, and HWB.

Graphs of Boolean Functions

Definition

Consider m functions $f = (f_0, \dots, f_{m-1})$ with $f_i: \{0, 1\}^n \rightarrow \{0, 1\}$.
 The **graph of f** called f -GRAPH is defined by

$$f\text{-GRAPH}(x_0, \dots, x_{n-1}, y_0, \dots, y_{m-1}) := \bigwedge_{i=0}^{m-1} [f_i(x) = y_i] \ .$$

- Fundamental functions with exp. OBDD-size:
 - Middle Bit of Multiplication, Indirect Storage Access, Hidden Weighted Bit
- Do exp. lower bounds carry over to graphs of functions?

Not in general! But for MUL, ISA, and HWB.

Graphs of Boolean Functions

Definition

Consider m functions $f = (f_0, \dots, f_{m-1})$ with $f_i: \{0, 1\}^n \rightarrow \{0, 1\}$.
 The **graph of f** called f -GRAPH is defined by

$$f\text{-GRAPH}(x_0, \dots, x_{n-1}, y_0, \dots, y_{m-1}) := \bigwedge_{i=0}^{m-1} [f_i(x) = y_i] .$$

- Fundamental functions with exp. OBDD-size:
 - Middle Bit of Multiplication, Indirect Storage Access, Hidden Weighted Bit
- Do exp. lower bounds carry over to graphs of functions?

Not in general! But for MUL, ISA, and HWB.

Integer Multiplication

Definition

The **Integer Multiplication** function $MUL_{n,i}: \{0, 1\}^{2n} \rightarrow \{0, 1\}$ is defined by

$$MUL_{n,i}(x, y) = (|x| \cdot |y|)_i$$

for $x, y \in \{0, 1\}^n$.

Theorem (Woelfel, 2001)

$$\text{OBDD-size}(MUL_{n,n-1}) \geq 2^{n/2}/61.$$

Integer Multiplication

Definition

The **Integer Multiplication** function $MUL_{n,i}: \{0, 1\}^{2n} \rightarrow \{0, 1\}$ is defined by

$$MUL_{n,i}(x, y) = (|x| \cdot |y|)_i$$

for $x, y \in \{0, 1\}^n$.

Theorem (Woelfel, 2001)

$OBDD\text{-size}(MUL_{n,n-1}) \geq 2^{n/2}/61$.

The Graph of Integer Multiplication

Definition

The **Graph of Integer Multiplication**

$MUL-GRAPH_n: \{0, 1\}^{4n} \rightarrow \{0, 1\}$ is defined by

$$MUL-GRAPH_n(x, y, z) = 1 :\Leftrightarrow (|x| \cdot |y| = |z|)$$

for $x, y \in \{0, 1\}^n$ and $z \in \{0, 1\}^{2n}$.

Theorem (S., 2005)

Any π -oblivious BP for $MUL-GRAPH_n$ whose sequence π contains each variable at most k times has at least size $2^{n/2^{O(k)}}$.

$\Rightarrow \text{OBDD-size}(MUL-GRAPH_n) = 2^{\Omega(n)}$.

The Graph of Integer Multiplication

Definition

The **Graph of Integer Multiplication**

$MUL-GRAPH_n: \{0, 1\}^{4n} \rightarrow \{0, 1\}$ is defined by

$$MUL-GRAPH_n(x, y, z) = 1 \Leftrightarrow (|x| \cdot |y| = |z|)$$

for $x, y \in \{0, 1\}^n$ and $z \in \{0, 1\}^{2n}$.

Theorem (S., 2005)

Any π -oblivious BP for $MUL-GRAPH_n$ whose sequence π contains each variable at most k times has at least size $2^{n/2^{O(k)}}$.

$\Rightarrow \text{OBDD-size}(MUL-GRAPH_n) = 2^{\Omega(n)}$.

The Graph of Integer Multiplication

Definition

The **Graph of Integer Multiplication**

$MUL-GRAPH_n: \{0, 1\}^{4n} \rightarrow \{0, 1\}$ is defined by

$$MUL-GRAPH_n(x, y, z) = 1 \Leftrightarrow (|x| \cdot |y| = |z|)$$

for $x, y \in \{0, 1\}^n$ and $z \in \{0, 1\}^{2n}$.

Theorem (S., 2005)

Any π -oblivious BP for $MUL-GRAPH_n$ whose sequence π contains each variable at most k times has at least size $2^{n/2^{O(k)}}$.

$\Rightarrow \text{OBDD-size}(MUL-GRAPH_n) = 2^{\Omega(n)}$.

A Restricted Implicit APSP Algorithm

All-pairs shortest-paths in weighted digraph $G = (V, E, c)$.

- Input: $\chi_c(x, y, d) = 1 \Leftrightarrow c(v_{|x|}, v_{|y|}) = |d|$
- Output: $\chi_\Delta(x, y, d) = 1 \Leftrightarrow \Delta(v_{|x|}, v_{|y|}) = |d|$

Algorithm \mathcal{A} for $c: E \rightarrow \mathbb{N}_{>0}$ (S., 2004):

Theorem

\mathcal{A} computes χ_Δ from χ_c in time $O(\log^3(|V| \cdot c^{\max}))$ if both χ_c, χ_Δ have constant-width OBDDs.

Does this property hold for trivial unrestricted algorithms?

A Restricted Implicit APSP Algorithm

All-pairs shortest-paths in weighted digraph $G = (V, E, c)$.

- Input: $\chi_c(x, y, d) = 1 \Leftrightarrow c(v_{|x|}, v_{|y|}) = |d|$
- Output: $\chi_\Delta(x, y, d) = 1 \Leftrightarrow \Delta(v_{|x|}, v_{|y|}) = |d|$

Algorithm \mathcal{A} for $c: E \rightarrow \mathbb{N}_{>0}$ (S., 2004):

Theorem

\mathcal{A} computes χ_Δ from χ_c in time $O(\log^3(|V| \cdot c^{\max}))$ if both χ_c, χ_Δ have constant-width OBDDs.

Does this property hold for trivial unrestricted algorithms?

A Restricted Implicit APSP Algorithm

All-pairs shortest-paths in weighted digraph $G = (V, E, c)$.

- Input: $\chi_c(x, y, d) = 1 \Leftrightarrow c(v_{|x|}, v_{|y|}) = |d|$
- Output: $\chi_\Delta(x, y, d) = 1 \Leftrightarrow \Delta(v_{|x|}, v_{|y|}) = |d|$

Algorithm \mathcal{A} for $c: E \rightarrow \mathbb{N}_{>0}$ (S., 2004):

Theorem

\mathcal{A} computes χ_Δ from χ_c in time $\mathcal{O}(\log^3(|V| \cdot c^{\max}))$ if both χ_c, χ_Δ have constant-width OBDDs.

Does this property hold for trivial unrestricted algorithms?

A Restricted Implicit APSP Algorithm

All-pairs shortest-paths in weighted digraph $G = (V, E, c)$.

- Input: $\chi_c(x, y, d) = 1 \Leftrightarrow c(v_{|x|}, v_{|y|}) = |d|$
- Output: $\chi_\Delta(x, y, d) = 1 \Leftrightarrow \Delta(v_{|x|}, v_{|y|}) = |d|$

Algorithm \mathcal{A} for $c: E \rightarrow \mathbb{N}_{>0}$ (S., 2004):

Theorem

\mathcal{A} computes χ_Δ from χ_c in time $\mathcal{O}(\log^3(|V| \cdot c^{\max}))$ if both χ_c, χ_Δ have constant-width OBDDs.

Does this property hold for trivial unrestricted algorithms?

A Restricted Implicit APSP Algorithm

All-pairs shortest-paths in weighted digraph $G = (V, E, c)$.

- Input: $\chi_c(x, y, d) = 1 \Leftrightarrow c(v_{|x|}, v_{|y|}) = |d|$
- Output: $\chi_\Delta(x, y, d) = 1 \Leftrightarrow \Delta(v_{|x|}, v_{|y|}) = |d|$

Algorithm \mathcal{A} for $c: E \rightarrow \mathbb{N}_{>0}$ (S., 2004):

Theorem

\mathcal{A} computes χ_Δ from χ_c in time $\mathcal{O}(\log^3(|V| \cdot c^{\max}))$ if both χ_c, χ_Δ have constant-width OBDDs.

Does this property hold for trivial unrestricted algorithms?

An Unrestricted Trivial Implicit APSP Algorithm

Trivial algorithm \mathcal{B} :

- Works for arbitrary $c: E \rightarrow \mathbb{N}_0$.
- Considers paths lengths $2^0, 2^1, \dots, 2^i, \dots, 2^{\lceil \log |V| \rceil}$:
 $S_i(x, y, d) = 1 \Leftrightarrow$ Shortest $v_{|x|} \rightarrow v_{|y|}$ -path has cost $|d|$.

Analysis technique: Construct counterexample G_n such that

- Input χ_c and output χ_Δ have constant-width OBDDs,
- some $S_i(x, y, d)$ contains $MUL-GRAPH_n$.

\Rightarrow Algo. \mathcal{B} needs time $2^{\Omega(n)}$

□

An Unrestricted Trivial Implicit APSP Algorithm

Trivial algorithm \mathcal{B} :

- Works for arbitrary $c: E \rightarrow \mathbb{N}_0$.
- Considers paths lengths $2^0, 2^1, \dots, 2^i, \dots, 2^{\lceil \log |V| \rceil}$:
 $S_i(x, y, d) = 1 \Leftrightarrow$ Shortest $v_{|x|} \rightarrow v_{|y|}$ -path has cost $|d|$.

Analysis technique: Construct counterexample G_n such that

- Input χ_c and output χ_Δ have constant-width OBDDs,
- some $S_i(x, y, d)$ contains $MUL\text{-}GRAPH_n$.

\Rightarrow Algo. \mathcal{B} needs time $2^{\Omega(n)}$



An Unrestricted Trivial Implicit APSP Algorithm

Trivial algorithm \mathcal{B} :

- Works for arbitrary $c: E \rightarrow \mathbb{N}_0$.
- Considers paths lengths $2^0, 2^1, \dots, 2^i, \dots, 2^{\lceil \log |V| \rceil}$:
 $S_i(x, y, d) = 1 \Leftrightarrow$ Shortest $v_{|x|} \rightarrow v_{|y|}$ -path has cost $|d|$.

Analysis technique: Construct counterexample G_n such that

- Input χ_c and output χ_Δ have constant-width OBDDs,
- some $S_i(x, y, d)$ contains $MUL\text{-}GRAPH_n$.

\Rightarrow Algo. \mathcal{B} needs time $2^{\Omega(n)}$

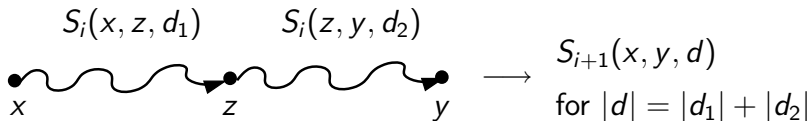


An Unrestricted Trivial Implicit APSP Algorithm

Trivial algorithm \mathcal{B} :

- Works for arbitrary $c: E \rightarrow \mathbb{N}_0$.
- Considers paths lengths $2^0, 2^1, \dots, 2^i, \dots, 2^{\lceil \log |V| \rceil}$:

$$S_i(x, y, d) = 1 \Leftrightarrow \text{Shortest } v_{|x|} \rightarrow v_{|y|}\text{-path has cost } |d| .$$



Analysis technique: Construct counterexample G_n such that

- Input χ_c and output χ_Δ have constant-width OBDDs,
- some $S_i(x, y, d)$ contains $MUL\text{-}GRAPH_n$.

\Rightarrow Algo. \mathcal{B} needs time $2^{\Omega(n)}$

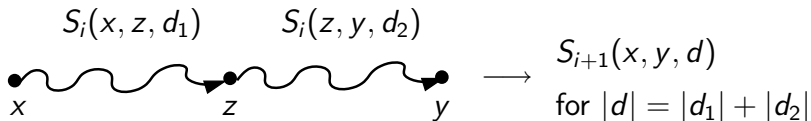
□

An Unrestricted Trivial Implicit APSP Algorithm

Trivial algorithm \mathcal{B} :

- Works for arbitrary $c: E \rightarrow \mathbb{N}_0$.
- Considers paths lengths $2^0, 2^1, \dots, 2^i, \dots, 2^{\lceil \log |V| \rceil}$:

$$S_i(x, y, d) = 1 \Leftrightarrow \text{Shortest } v_{|x|} \rightarrow v_{|y|}\text{-path has cost } |d| .$$



Analysis technique: Construct counterexample G_n such that

- Input χ_c and output χ_Δ have constant-width OBDDs,
- some $S_i(x, y, d)$ contains $MUL\text{-}GRAPH_n$.

\Rightarrow Algo. \mathcal{B} needs time $2^{\Omega(n)}$

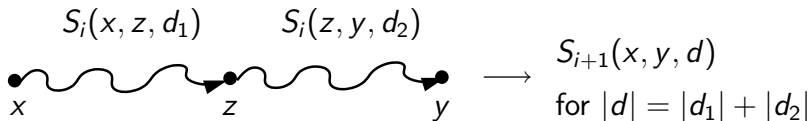


An Unrestricted Trivial Implicit APSP Algorithm

Trivial algorithm \mathcal{B} :

- Works for arbitrary $c: E \rightarrow \mathbb{N}_0$.
- Considers paths lengths $2^0, 2^1, \dots, 2^i, \dots, 2^{\lceil \log |V| \rceil}$:

$$S_i(x, y, d) = 1 \Leftrightarrow \text{Shortest } v_{|x|} \rightarrow v_{|y|}\text{-path has cost } |d| .$$



Analysis technique: Construct counterexample G_n such that

- Input χ_c and output χ_Δ have constant-width OBDDs,
- some $S_i(x, y, d)$ contains $MUL\text{-}GRAPH_n$.

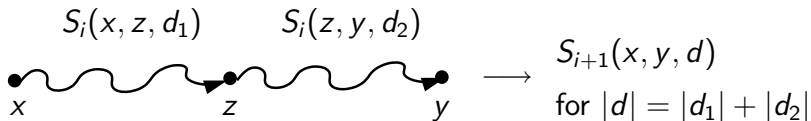
\Rightarrow Algo. \mathcal{B} needs time $2^{\Omega(n)}$



An Unrestricted Trivial Implicit APSP Algorithm

Trivial algorithm \mathcal{B} :

- Works for arbitrary $c: E \rightarrow \mathbb{N}_0$.
- Considers paths lengths $2^0, 2^1, \dots, 2^i, \dots, 2^{\lceil \log |V| \rceil}$:
 $S_i(x, y, d) = 1 \Leftrightarrow$ Shortest $v_{|x|} \rightarrow v_{|y|}$ -path has cost $|d|$.



Analysis technique: Construct counterexample G_n such that

- Input χ_c and output χ_Δ have constant-width OBDDs,
- some $S_i(x, y, d)$ contains $MUL-GRAPH_n$.

\Rightarrow Algo. \mathcal{B} needs time $2^{\Omega(n)}$

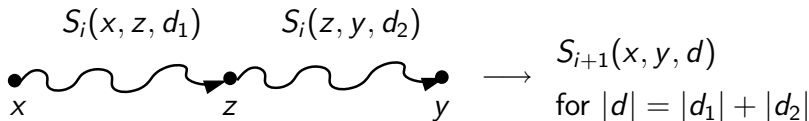


An Unrestricted Trivial Implicit APSP Algorithm

Trivial algorithm \mathcal{B} :

- Works for arbitrary $c: E \rightarrow \mathbb{N}_0$.
- Considers paths lengths $2^0, 2^1, \dots, 2^i, \dots, 2^{\lceil \log |V| \rceil}$:

$$S_i(x, y, d) = 1 \Leftrightarrow \text{Shortest } v_{|x|} \rightarrow v_{|y|}\text{-path has cost } |d| .$$



Analysis technique: Construct counterexample G_n such that

- Input χ_c and output χ_Δ have constant-width OBDDs,
- some $S_i(x, y, d)$ contains $MUL\text{-}GRAPH_n$.

\Rightarrow Algo. \mathcal{B} needs time $2^{\Omega(n)}$

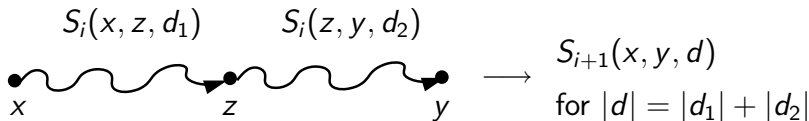


An Unrestricted Trivial Implicit APSP Algorithm

Trivial algorithm \mathcal{B} :

- Works for arbitrary $c: E \rightarrow \mathbb{N}_0$.
- Considers paths lengths $2^0, 2^1, \dots, 2^i, \dots, 2^{\lceil \log |V| \rceil}$:

$$S_i(x, y, d) = 1 \Leftrightarrow \text{Shortest } v_{|x|} \rightarrow v_{|y|}\text{-path has cost } |d| .$$



Analysis technique: Construct counterexample G_n such that

- Input χ_c and output χ_Δ have constant-width OBDDs,
- some $S_i(x, y, d)$ contains $MUL\text{-}GRAPH_n$.

\Rightarrow Algo. \mathcal{B} needs time $2^{\Omega(n)} = |V|^{\Omega(1)}$

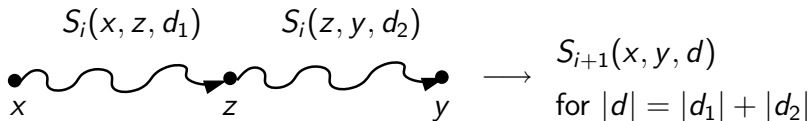
□

An Unrestricted Trivial Implicit APSP Algorithm

Trivial algorithm \mathcal{B} :

- Works for arbitrary $c: E \rightarrow \mathbb{N}_0$.
- Considers paths lengths $2^0, 2^1, \dots, 2^i, \dots, 2^{\lceil \log |V| \rceil}$:

$$S_i(x, y, d) = 1 \Leftrightarrow \text{Shortest } v_{|x|} \rightarrow v_{|y|}\text{-path has cost } |d| .$$



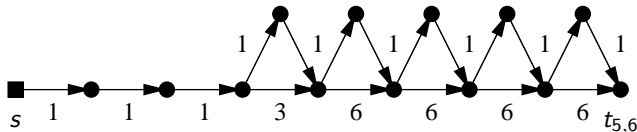
Analysis technique: Construct counterexample G_n such that

- Input χ_c and output χ_Δ have constant-width OBDDs,
- some $S_i(x, y, d)$ contains $MUL\text{-}GRAPH_n$.

\Rightarrow Algo. \mathcal{B} needs time $2^{\Omega(n)} = |V|^{\Omega(1)} = \omega(\log^3(|V| \cdot c^{\max}))$. \square

Construction of G_n

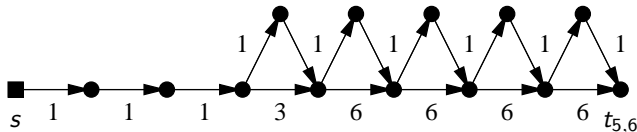
- Union of graphs $G_n(i, j)$ sharing node s .
- Paths (s, \dots, t_{ij}) of 2^n edges, bridged by **shortcut edges**.
- Shortest $s \rightarrow t_{ij}$ -path has weight $2^n + i$.
- S_n must not use shortcuts \Rightarrow path weight $i \cdot j$.
- $\Rightarrow \mathcal{B}$ computes $S_n(x, y, d) \approx \text{MUL-GRAPH}_n(x, y, d)$.



Example: $G_3(5, 6)$

Construction of G_n

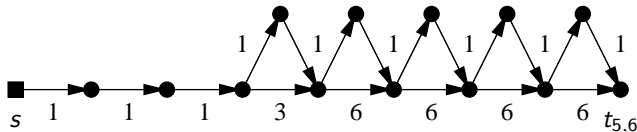
- Union of graphs $G_n(i, j)$ sharing node s .
- Paths $(s, \dots, t_{i,j})$ of 2^n edges, bridged by **shortcut edges**.
- Shortest $s \rightarrow t_{i,j}$ -path has weight $2^n + i$.
- S_n must not use shortcuts \Rightarrow path weight $i \cdot j$.
- $\Rightarrow \mathcal{B}$ computes $S_n(x, y, d) \approx \text{MUL-GRAPH}_n(x, y, d)$.



Example: $G_3(5, 6)$

Construction of G_n

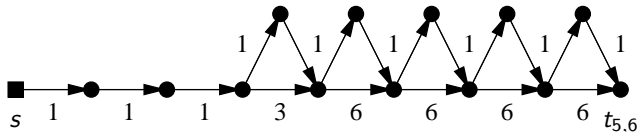
- Union of graphs $G_n(i, j)$ sharing node s .
- Paths $(s, \dots, t_{i,j})$ of 2^n edges, bridged by **shortcut edges**.
- Shortest $s \rightarrow t_{i,j}$ -path has weight $2^n + i$.
- S_n must not use shortcuts \Rightarrow path weight $i \cdot j$.
- $\Rightarrow \mathcal{B}$ computes $S_n(x, y, d) \approx \text{MUL-GRAPH}_n(x, y, d)$.



Example: $G_3(5, 6)$

Construction of G_n

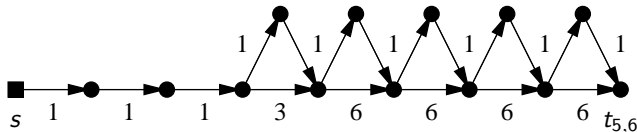
- Union of graphs $G_n(i, j)$ sharing node s .
- Paths $(s, \dots, t_{i,j})$ of 2^n edges, bridged by **shortcut edges**.
- Shortest $s \rightarrow t_{i,j}$ -path has weight $2^n + i$.
- S_n must not use shortcuts \Rightarrow path weight $i \cdot j$.
- $\Rightarrow \mathcal{B}$ computes $S_n(x, y, d) \approx \text{MUL-GRAPH}_n(x, y, d)$.



Example: $G_3(5, 6)$

Construction of G_n

- Union of graphs $G_n(i, j)$ sharing node s .
- Paths $(s, \dots, t_{i,j})$ of 2^n edges, bridged by **shortcut edges**.
- Shortest $s \rightarrow t_{i,j}$ -path has weight $2^n + i$.
- S_n must not use shortcuts \Rightarrow path weight $i \cdot j$.
- $\Rightarrow \mathcal{B}$ computes $S_n(x, y, d) \approx \text{MUL-GRAPH}_n(x, y, d)$.



Example: $G_3(5, 6)$

Summary

- Digraphs can be represented by the OBDD of their characteristic function.
- First results on the OBDD-size of fundamental graph classes (Cographs, P_4 -sparse, P_4 -extendible, Interval, Bipartite).
- The graph of multiplication has exp. OBDD-size.
- Similar results for Indirect Storage Access and Hidden Weighted Bit (S., 2005).
- *MUL-GRAPH* can be encoded into APSP-instances to separate the complexities of different implicit approaches.

Summary

- Digraphs can be represented by the OBDD of their characteristic function.
- First results on the OBDD-size of fundamental graph classes (Cographs, P_4 -sparse, P_4 -extendible, Interval, Bipartite).
- The graph of multiplication has exp. OBDD-size.
- Similar results for Indirect Storage Access and Hidden Weighted Bit (S., 2005).
- *MUL-GRAPH* can be encoded into APSP-instances to separate the complexities of different implicit approaches.

Summary

- Digraphs can be represented by the OBDD of their characteristic function.
- First results on the OBDD-size of fundamental graph classes (Cographs, P_4 -sparse, P_4 -extendible, Interval, Bipartite).
- The graph of multiplication has exp. OBDD-size.
- Similar results for Indirect Storage Access and Hidden Weighted Bit (S., 2005).
- *MUL-GRAPH* can be encoded into APSP-instances to separate the complexities of different implicit approaches.

Summary

- Digraphs can be represented by the OBDD of their characteristic function.
- First results on the OBDD-size of fundamental graph classes (Cographs, P_4 -sparse, P_4 -extendible, Interval, Bipartite).
- The graph of multiplication has exp. OBDD-size.
- Similar results for Indirect Storage Access and Hidden Weighted Bit (S., 2005).
- *MUL-GRAPH* can be encoded into APSP-instances to separate the complexities of different implicit approaches.

Summary

- Digraphs can be represented by the OBDD of their characteristic function.
- First results on the OBDD-size of fundamental graph classes (Cographs, P_4 -sparse, P_4 -extendible, Interval, Bipartite).
- The graph of multiplication has exp. OBDD-size.
- Similar results for Indirect Storage Access and Hidden Weighted Bit (S., 2005).
- *MUL-GRAPH* can be encoded into APSP-instances to separate the complexities of different implicit approaches.

“That’s all Folks!”