Lower Bounds on the OBDD Size of Graphs of Some Popular Functions

Daniel Sawitzki

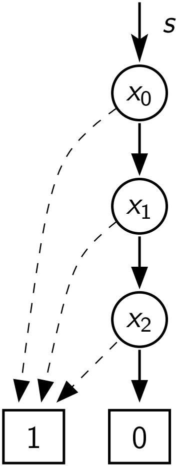
Computer Science 2
University of Dortmund, Germany

SOFSEM 2005
January 22–28, 2005

Overview

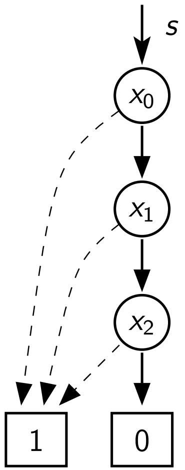
General Branching Programs

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



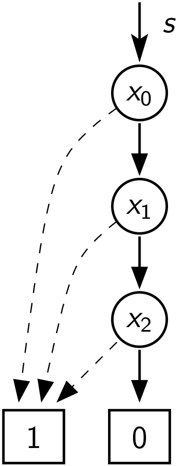
General Branching Programs

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



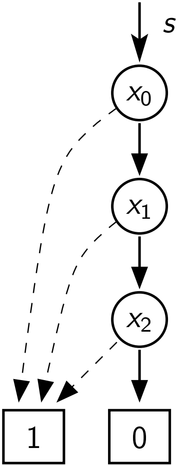
General Branching Programs

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



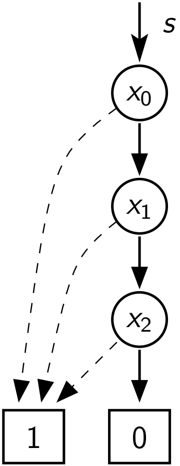
General Branching Programs

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



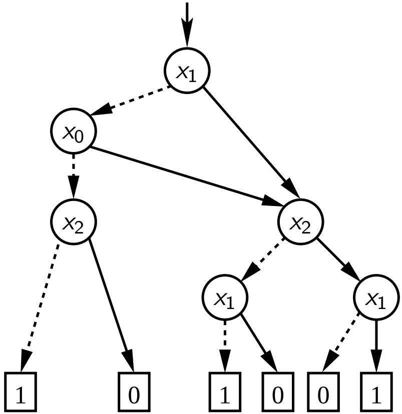
General Branching Programs

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



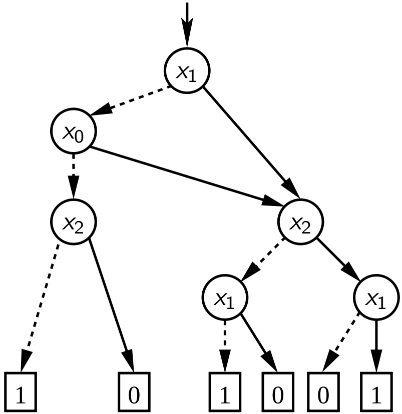
Oblivious Branching Programs and OBDDs

- Let $\pi \in \{x_0, \ldots, x_{n-1}\}^*$.
- 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



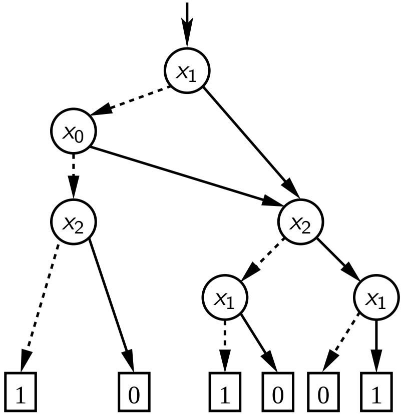
Oblivious Branching Programs and OBDDs

- Let $\pi \in \{x_0, \ldots, x_{n-1}\}^*$.
- 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



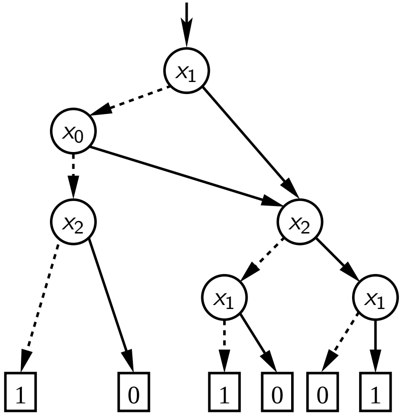
Oblivious Branching Programs and OBDDs

- Let $\pi \in \{x_0, \ldots, x_{n-1}\}^*$.
- 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



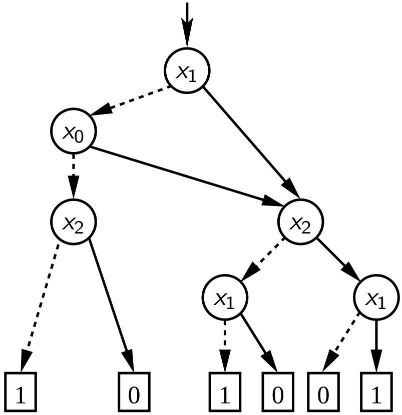
Oblivious Branching Programs and OBDDs

- Let $\pi \in \{x_0, \ldots, x_{n-1}\}^*$.
- 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



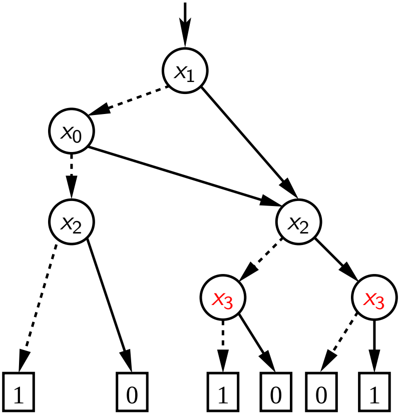
Oblivious Branching Programs and OBDDs

- Let $\pi \in \{x_0, \ldots, x_{n-1}\}^*$.
- 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 $\pi \in \{x_0, \ldots, x_{n-1}\}^*$.
- 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
- Efficient algorithms:
  - Logic design, CAD, model checking
  - Graph algorithms (maximum flow, shortest paths, ...)
  - 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
- Efficient algorithms:
  - Logic design, CAD, model checking
  - Graph algorithms (maximum flow, shortest paths, ...)
  - 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
- Efficient algorithms:
  - Logic design, CAD, model checking
  - Graph algorithms (maximum flow, shortest paths, ...)
  - 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
- Efficient algorithms:
  - Logic design, CAD, model checking
  - Graph algorithms (maximum flow, shortest paths, ...)
  - 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
- Efficient algorithms:
  - Logic design, CAD, model checking
  - Graph algorithms (maximum flow, shortest paths, ...)
  - 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
- Efficient algorithms:
  - Logic design, CAD, model checking
  - Graph algorithms (maximum flow, shortest paths, . . .)
  - 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
- Efficient algorithms:
  - Logic design, CAD, model checking
  - Graph algorithms (maximum flow, shortest paths, . . .)
  - Favored variant: OBDDs due to efficient operations ($\wedge$, $\vee$, $\exists$, $\forall$, SAT, $\equiv$, . . .)

Characteristic Boolean Functions

- Symbolic (graph) algorithms use characteristic functions

$$\chi_P\colon \{0,1\}^{tn} \to \{0,1\}$$

for properties $P \subseteq S = \{0,\dots,2^n-1\}^t$:

$$\chi_P\left(x^{(1)},\dots,x^{(t)}\right) = 1 :\Leftrightarrow \left(\left|x^{(1)}\right|,\dots,\left|x^{(t)}\right|\right) \in P \quad .$$

- Example: Edge set $E \subseteq V^2 = \{v_0,\dots,v_{2^n-1}\}^2$:

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

- Runtime depends essentially on OBDD size of χ_P.
- Technique in symbolic algorithm analysis:
  - Encode simple functions with large OBDDs into characteristic functions.
  - Show lower bounds on runtime/space of symbolic algorithm.

Characteristic Boolean Functions

- Symbolic (graph) algorithms use characteristic functions

$$\chi_P\colon \{0,1\}^{tn} \to \{0,1\}$$

for properties $P \subseteq S = \{0,\dots,2^n-1\}^t$:

$$\chi_P\left(x^{(1)},\dots,x^{(t)}\right) = 1 :\Leftrightarrow \left(\left|x^{(1)}\right|,\dots,\left|x^{(t)}\right|\right) \in P \ .$$

- Example: Edge set $E \subseteq V^2 = \{v_0,\dots,v_{2^n-1}\}^2$:

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

- Runtime depends essentially on OBDD size of χ_P.
- Technique in symbolic algorithm analysis:
  - Encode simple functions with large OBDDs into characteristic functions.
  - Show lower bounds on runtime/space of symbolic algorithm.

Characteristic Boolean Functions

- Symbolic (graph) algorithms use characteristic functions
$$\chi_P\colon \{0,1\}^{tn} \rightarrow \{0,1\}$$
for properties $P \subseteq S = \{0,\ldots,2^n-1\}^t$:
$$\chi_P\left(x^{(1)},\ldots,x^{(t)}\right) = 1 :\Leftrightarrow \left(\left|x^{(1)}\right|,\ldots,\left|x^{(t)}\right|\right) \in P \ .$$
- Example: Edge set $E \subseteq V^2 = \{v_0,\ldots,v_{2^n-1}\}^2$:
$$\chi_E(x,y) = 1 \Leftrightarrow \left(v_{|x|}, v_{|y|}\right) \in E \ .$$
- Runtime depends essentially on OBDD size of χ_P.
- Technique in symbolic algorithm analysis:
  - Encode simple functions with large OBDDs into characteristic functions.
  - Show lower bounds on runtime/space of symbolic algorithm.

Characteristic Boolean Functions

- Symbolic (graph) algorithms use characteristic functions

$$\chi_P\colon \{0,1\}^{tn} \to \{0,1\}$$

for properties $P \subseteq S = \{0,\ldots,2^n-1\}^t$:

$$\chi_P\left(x^{(1)},\ldots,x^{(t)}\right) = 1 :\Leftrightarrow \left(\left|x^{(1)}\right|,\ldots,\left|x^{(t)}\right|\right) \in P \ .$$

- Example: Edge set $E \subseteq V^2 = \{v_0,\ldots,v_{2^n-1}\}^2$:

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

- Runtime depends essentially on OBDD size of χ_P.
- Technique in symbolic algorithm analysis:
  - Encode simple functions with large OBDDs into characteristic functions.
  - Show lower bounds on runtime/space of symbolic algorithm.

Characteristic Boolean Functions

- Symbolic (graph) algorithms use characteristic functions

$$\chi_P\colon \{0,1\}^{tn} \to \{0,1\}$$

for properties $P \subseteq S = \{0,\ldots,2^n-1\}^t$:

$$\chi_P\left(x^{(1)},\ldots,x^{(t)}\right) = 1 :\Leftrightarrow \left(\left|x^{(1)}\right|,\ldots,\left|x^{(t)}\right|\right) \in P \ .$$

- Example: Edge set $E \subseteq V^2 = \{v_0,\ldots,v_{2^n-1}\}^2$:

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

- Runtime depends essentially on OBDD size of χ_P.
- Technique in symbolic algorithm analysis:
  - Encode simple functions with large OBDDs into characteristic functions.
  - Show lower bounds on runtime/space of symbolic algorithm.

Characteristic Boolean Functions

- Symbolic (graph) algorithms use characteristic functions

  $$\chi_P\colon \{0,1\}^{tn} \to \{0,1\}$$

  for properties $P \subseteq S = \{0,\dots,2^n-1\}^t$:

  $$\chi_P\left(x^{(1)},\dots,x^{(t)}\right) = 1 :\Leftrightarrow \left(\left|x^{(1)}\right|,\dots,\left|x^{(t)}\right|\right) \in P \enspace .$$

- Example: Edge set $E \subseteq V^2 = \{v_0,\dots,v_{2^n-1}\}^2$:

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

- Runtime depends essentially on OBDD size of χ_P.
- Technique in symbolic algorithm analysis:
  - Encode simple functions with large OBDDs into characteristic functions.
  - Show lower bounds on runtime/space of symbolic algorithm.

Characteristic Functions with Large OBDDs

- How to get them? Start with known exponential lower bounds:
  - Integer Multiplication: $MUL_{n,n-1}(x,y) := (|x| \cdot |y|)_{n-1}$, OBDD size $\geq 2^{n/2}/61$ (Woelfel 2001)
  - Indirect Storage Access and Hidden Weighted Bit
- Try to transfer lower bounds to the function graph.

Definition

Consider m functions $f = (f_0, \ldots, f_{m-1})$ with $f_i \colon \{0,1\}^n \to \{0,1\}$. The function graph $f\text{-}GRAPH$ is defined by

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

Characteristic Functions with Large OBDDs

- How to get them? Start with known exponential lower bounds:
  - Integer Multiplication: $MUL_{n,n-1}(x, y) := (|x| \cdot |y|)_{n-1}$, OBDD size $\geq 2^{n/2}/61$ (Woelfel 2001)
  - Indirect Storage Access and Hidden Weighted Bit
- Try to transfer lower bounds to the function graph.

Definition

Consider m functions $f = (f_0, \ldots, f_{m-1})$ with $f_i \colon \{0,1\}^n \to \{0,1\}$. The function graph $f\text{-}GRAPH$ is defined by

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

Characteristic Functions with Large OBDDs

- How to get them? Start with known exponential lower bounds:
  - Integer Multiplication: $MUL_{n,n-1}(x,y) := (|x| \cdot |y|)_{n-1}$, OBDD size $\geq 2^{n/2}/61$ (Woelfel 2001)
  - Indirect Storage Access and Hidden Weighted Bit
- Try to transfer lower bounds to the function graph.

Definition

Consider m functions $f = (f_0, \ldots, f_{m-1})$ with $f_i \colon \{0,1\}^n \to \{0,1\}$. The function graph $f\text{-}GRAPH$ is defined by

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

Characteristic Functions with Large OBDDs

- How to get them? Start with known exponential lower bounds:
  - Integer Multiplication: $MUL_{n,n-1}(x, y) := (|x| \cdot |y|)_{n-1}$, OBDD size $\geq 2^{n/2}/61$ (Woelfel 2001)
  - Indirect Storage Access and Hidden Weighted Bit
- Try to transfer lower bounds to the function graph.

Definition

Consider m functions $f = (f_0, \ldots, f_{m-1})$ with $f_i\colon \{0,1\}^n \to \{0,1\}$. The function graph $f\text{-}GRAPH$ is defined by

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

Characteristic Functions with Large OBDDs

- How to get them? Start with known exponential lower bounds:
  - Integer Multiplication: $MUL_{n,n-1}(x,y) := (|x| \cdot |y|)_{n-1}$, OBDD size $\geq 2^{n/2}/61$ (Woelfel 2001)
  - Indirect Storage Access and Hidden Weighted Bit
- Try to transfer lower bounds to the function graph.

Definition

Consider m functions $f = (f_0, \dots, f_{m-1})$ with $f_i \colon \{0,1\}^n \to \{0,1\}$. The function graph f-$GRAPH$ is defined by

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

An Exponential Lower Bound

Definition

The Graph of Integer Multiplication $MUL\text{-}GRAPH_n \colon \{0,1\}^{4n} \to \{0,1\}$ is defined by

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

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

Theorem

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

An Exponential Lower Bound

Definition

The Graph of Integer Multiplication $MUL\text{-}GRAPH_n\colon \{0,1\}^{4n} \to \{0,1\}$ is defined by

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

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

Theorem

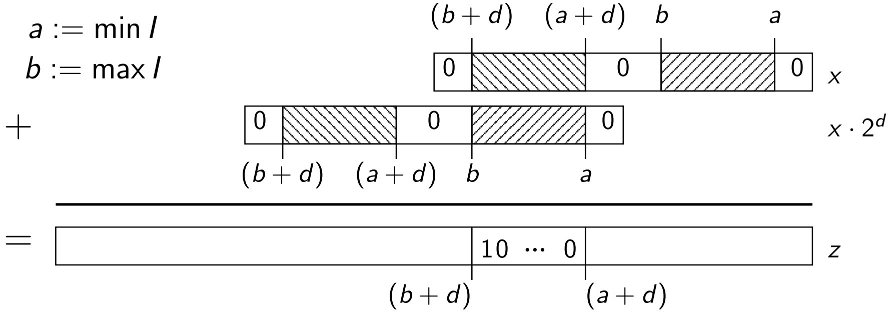
Any π-oblivious BP for $MUL\text{-}GRAPH_n$ whose sequence π contains each variable at most k times has at least size $2^{n/2^{O(k)}}$.

An Exponential Lower Bound—Sketch of Proof (1)

- Choose $I \subset \{0, \ldots, n-1\}$ and $d \in \mathbb{N}$.
- Consider $MUL\text{-}GRAPH_n^* := \bigwedge_{i=a+d}^{b+d} [MUL_{n,i}(x, y) = z_i]$.
- Prove lower bound for subfunction f_n of $MUL\text{-}GRAPH_n^*$.
- Transfer lower bound to $MUL\text{-}GRAPH_n$.

$a := \min I$

$b := \max I$

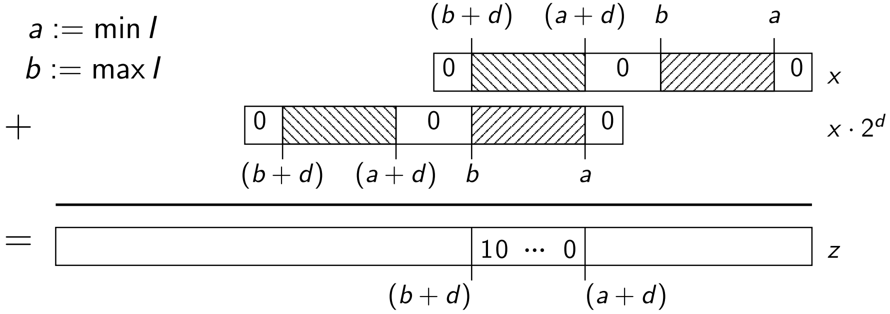


An Exponential Lower Bound—Sketch of Proof (1)

- Choose $I \subset \{0, \ldots, n-1\}$ and $d \in \mathbb{N}$.
- Consider $MUL\text{-}GRAPH_n^* := \bigwedge_{i=a+d}^{b+d} [MUL_{n,i}(x,y) = z_i]$.
- Prove lower bound for subfunction f_n of $MUL\text{-}GRAPH_n^*$.
- Transfer lower bound to $MUL\text{-}GRAPH_n$.

$a := \min I$

$b := \max I$

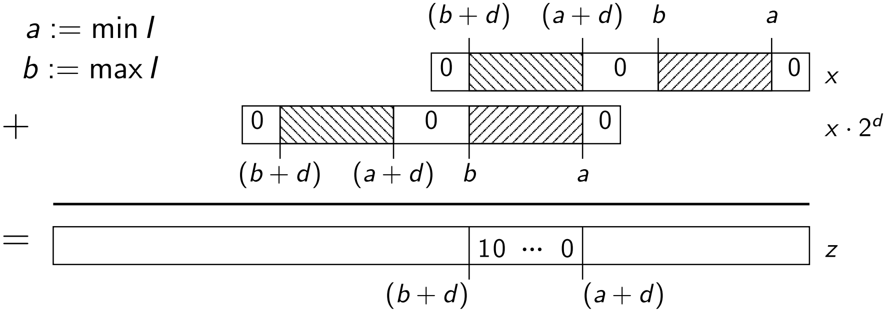


An Exponential Lower Bound—Sketch of Proof (1)

- Choose $I \subset \{0, \ldots, n-1\}$ and $d \in \mathbb{N}$.
- Consider $MUL\text{-}GRAPH_n^* := \bigwedge_{i=a+d}^{b+d} [MUL_{n,i}(x, y) = z_i]$.
- Prove lower bound for subfunction f_n of $MUL\text{-}GRAPH_n^*$.
- Transfer lower bound to $MUL\text{-}GRAPH_n$.

$a := \min I$

$b := \max I$

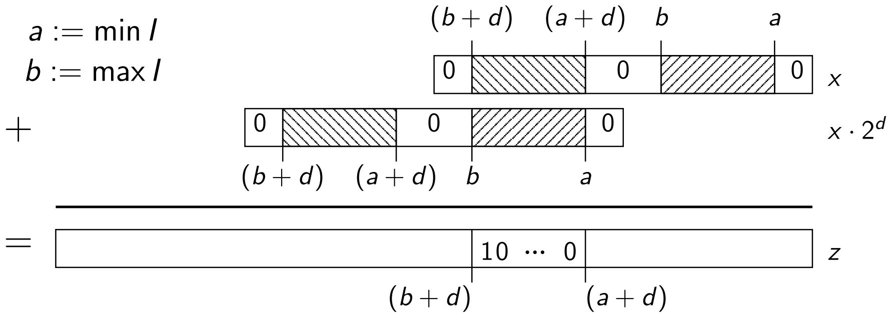


An Exponential Lower Bound—Sketch of Proof (1)

- Choose $I \subset \{0, \ldots, n-1\}$ and $d \in \mathbb{N}$.
- Consider $MUL\text{-}GRAPH_n^* := \bigwedge_{i=a+d}^{b+d} [MUL_{n,i}(x, y) = z_i]$.
- Prove lower bound for subfunction f_n of $MUL\text{-}GRAPH_n^*$.
- Transfer lower bound to $MUL\text{-}GRAPH_n$.

$a := \min I$

$b := \max I$



An Exponential Lower Bound—Sketch of Proof (2)

- Subfunction f_n depends only on $\{x_i \mid i \in I\} =: x^{(1)}$ and $\{x_{i+d} \mid i \in I\} =: x^{(2)}$.
- It is $f_n\left(x^{(1)}, x^{(2)}\right) = 1 \Leftrightarrow \forall i \colon x_i^{(1)} \neq x_i^{(2)}$.
- Alice gets $x^{(1)}$, Bob gets $x^{(2)}$
  $\Rightarrow$ Comm. complexity $\geq \log\left(\text{rank}(C)\right) = |I|$

$x^{(1)}$	0	?	0 0	?	0	?	?	1	
$x^{(2)}$	1	?	1 1	?	1	?	?	1	

1	0	0	0	0	0	0	0	0	0

$x^{(1)}/x^{(2)}$	$\cdots$	1101	1110	1111
0000	$\cdots$	0	0	1
0001	$\cdots$	0	1	0
0010	$\cdots$	1	0	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

An Exponential Lower Bound—Sketch of Proof (2)

- Subfunction f_n depends only on $\{x_i \mid i \in I\} =: x^{(1)}$ and $\{x_{i+d} \mid i \in I\} =: x^{(2)}$.
- It is $f_n\left(x^{(1)}, x^{(2)}\right) = 1 \Leftrightarrow \forall i\colon x_i^{(1)} \neq x_i^{(2)}$.
- Alice gets $x^{(1)}$, Bob gets $x^{(2)}$
  $\Rightarrow$ Comm. complexity $\geq \log(\text{rank}(C)) = |I|$

$x^{(1)}$	0	?	0	0	?	0	?	?	1
$x^{(2)}$	1	?	1	1	?	1	?	?	1

1	0	0	0	0	0	0	0	0	0

$x^{(1)}/x^{(2)}$	$\cdots$	1101	1110	1111
0000	$\cdots$	0	0	1
0001	$\cdots$	0	1	0
0010	$\cdots$	1	0	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

An Exponential Lower Bound—Sketch of Proof (2)

- Subfunction f_n depends only on $\{x_i \mid i \in I\} =: x^{(1)}$ and $\{x_{i+d} \mid i \in I\} =: x^{(2)}$.
- It is $f_n\left(x^{(1)}, x^{(2)}\right) = 1 \Leftrightarrow \forall i\colon x_i^{(1)} \neq x_i^{(2)}$.
- Alice gets $x^{(1)}$, Bob gets $x^{(2)}$
  $\Rightarrow$ Comm. complexity $\geq \log\left(\text{rank}(C)\right) = |I|$

$x^{(1)}$	0	?	0	0	?	0	?	?	1
$x^{(2)}$	1	?	1	1	?	1	?	?	1
1	0	0	0	0	0	0	0	0	0

$x^{(1)}/x^{(2)}$	$\cdots$	1101	1110	1111
0000	$\cdots$	0	0	1
0001	$\cdots$	0	1	0
0010	$\cdots$	1	0	0
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$

An Exponential Lower Bound—Sketch of Proof (3)

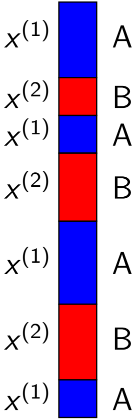
- Special choice of $x^{(1)} = \{x_i \mid i \in I\}$ and $x^{(2)} = \{x_{i+d} \mid i \in I\}$ such that
  1. π contains $\leq 4k-1$ switches between $x^{(1)}$ and $x^{(2)}$.
  2. $|I| \geq n/2^{8k} - 3$,

  (Alon and Maass 1988)
- π-oblivious BP P for f_n yields protocol of length $(4k-1) \cdot \log(\text{size}(P))$.
- Comm. complex. $\geq |I| \geq n/2^{8k} - 3$ $\Rightarrow \text{size}(P) \geq 2^{n/2^{O(k)}}$.
- Transfer result to $MUL\text{-}GRAPH_n$. □

An Exponential Lower Bound—Sketch of Proof (3)

- Special choice of $x^{(1)} = \{x_i \mid i \in I\}$ and $x^{(2)} = \{x_{i+d} \mid i \in I\}$ such that
  1. π contains $\leq 4k - 1$ switches between $x^{(1)}$ and $x^{(2)}$.
  2. $|I| \geq n/2^{8k} - 3$,

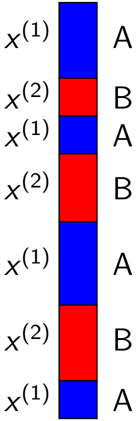
  (Alon and Maass 1988)
- π-oblivious BP P for f_n yields protocol of length $(4k-1) \cdot \log(\text{size}(P))$.
- Comm. complex. $\geq |I| \geq n/2^{8k} - 3$ $\Rightarrow \text{size}(P) \geq 2^{n/2^{O(k)}}$.
- Transfer result to $MUL\text{-}GRAPH_n$. □



An Exponential Lower Bound—Sketch of Proof (3)

- Special choice of $x^{(1)} = \{x_i \mid i \in I\}$ and $x^{(2)} = \{x_{i+d} \mid i \in I\}$ such that
  1. π contains $\leq 4k-1$ switches between $x^{(1)}$ and $x^{(2)}$.
  2. $|I| \geq n/2^{8k} - 3$,

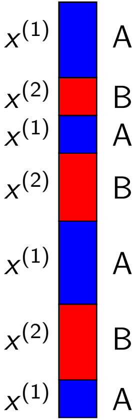
  (Alon and Maass 1988)
- π-oblivious BP P for f_n yields protocol of length $(4k-1) \cdot \log(\text{size}(P))$.
- Comm. complex. $\geq |I| \geq n/2^{8k} - 3$ $\Rightarrow \text{size}(P) \geq 2^{n/2^{O(k)}}$.
- Transfer result to $MUL\text{-}GRAPH_n$. □



An Exponential Lower Bound—Sketch of Proof (3)

- Special choice of $x^{(1)} = \{x_i \mid i \in I\}$ and $x^{(2)} = \{x_{i+d} \mid i \in I\}$ such that
  1. π contains $\leq 4k - 1$ switches between $x^{(1)}$ and $x^{(2)}$.
  2. $|I| \geq n/2^{8k} - 3$,

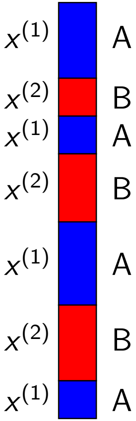
  (Alon and Maass 1988)
- π-oblivious BP P for f_n yields protocol of length $(4k - 1) \cdot \log(\text{size}(P))$.
- Comm. complex. $\geq |I| \geq n/2^{8k} - 3$ $\Rightarrow \text{size}(P) \geq 2^{n/2^{O(k)}}$.
- Transfer result to $MUL\text{-}GRAPH_n$. □



An Exponential Lower Bound—Sketch of Proof (3)

- Special choice of $x^{(1)} = \{x_i \mid i \in I\}$ and $x^{(2)} = \{x_{i+d} \mid i \in I\}$ such that
  1. π contains $\leq 4k-1$ switches between $x^{(1)}$ and $x^{(2)}$.
  2. $|I| \geq n/2^{8k} - 3$,

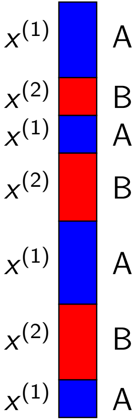
  (Alon and Maass 1988)
- π-oblivious BP P for f_n yields protocol of length $(4k-1) \cdot \log(\text{size}(P))$.
- Comm. complex. $\geq |I| \geq n/2^{8k} - 3$
  $\Rightarrow \text{size}(P) \geq 2^{n/2^{O(k)}}$.
- Transfer result to $MUL\text{-}GRAPH_n$. □



An Exponential Lower Bound—Sketch of Proof (3)

- Special choice of $x^{(1)} = \{x_i \mid i \in I\}$ and $x^{(2)} = \{x_{i+d} \mid i \in I\}$ such that
  1. π contains $\leq 4k - 1$ switches between $x^{(1)}$ and $x^{(2)}$.
  2. $|I| \geq n/2^{8k} - 3$,

  (Alon and Maass 1988)
- π-oblivious BP P for f_n yields protocol of length $(4k - 1) \cdot \log(\text{size}(P))$.
- Comm. complex. $\geq |I| \geq n/2^{8k} - 3$ $\Rightarrow \text{size}(P) \geq 2^{n/2^{O(k)}}$.
- Transfer result to $MUL\text{-}GRAPH_n$. □



Analysis of a Symbolic APSP-Algorithm

- Input: $\chi_c(x, y, d) = 1 :\Leftrightarrow c(v_{|x|}, v_{|y|}) = |d|$
- Output: $\chi_\Delta(x, y, d) = 1 :\Leftrightarrow \Delta(v_{|x|}, v_{|y|}) = |d|$
- Computes intermediate functions $S_i(x, y, d)$:
  - S_i covers paths with at most 2^i edges.
  - $S_i(x, y, d) = 1 \Leftrightarrow$ Shortest $v_{|x|}$–$v_{|y|}$-path has cost $|d|$.
- Construct graph G_n such that
  - χ_c and χ_Δ have small OBDDs,
  - $S_i(x, y, d)$ contains $MUL\text{-}GRAPH_n(x, y, d)$.
- $\Rightarrow$ APSP-algorithm needs time $2^{\Omega(n)}$. □

Analysis of a Symbolic APSP-Algorithm

- Input: $\chi_c(x, y, d) = 1 :\Leftrightarrow c(v_{|x|}, v_{|y|}) = |d|$
- Output: $\chi_\Delta(x, y, d) = 1 :\Leftrightarrow \Delta(v_{|x|}, v_{|y|}) = |d|$
- Computes intermediate functions $S_i(x, y, d)$:
  - S_i covers paths with at most 2^i edges.
  - $S_i(x, y, d) = 1 \Leftrightarrow$ Shortest $v_{|x|}$–$v_{|y|}$-path has cost $|d|$.
- Construct graph G_n such that
  - χ_c and χ_Δ have small OBDDs,
  - $S_i(x, y, d)$ contains $\mathit{MUL\text{-}GRAPH}_n(x, y, d)$.
- $\Rightarrow$ APSP-algorithm needs time $2^{\Omega(n)}$. □

Analysis of a Symbolic APSP-Algorithm

- Input: $\chi_c(x, y, d) = 1 :\Leftrightarrow c(v_{|x|}, v_{|y|}) = |d|$
- Output: $\chi_\Delta(x, y, d) = 1 :\Leftrightarrow \Delta(v_{|x|}, v_{|y|}) = |d|$
- Computes intermediate functions $S_i(x, y, d)$:
  - S_i covers paths with at most 2^i edges.
  - $S_i(x, y, d) = 1 \Leftrightarrow$ Shortest $v_{|x|}$–$v_{|y|}$-path has cost $|d|$.
- Construct graph G_n such that
  - χ_c and χ_Δ have small OBDDs,
  - $S_i(x, y, d)$ contains $MUL\text{-}GRAPH_n(x, y, d)$.
- $\Rightarrow$ APSP-algorithm needs time $2^{\Omega(n)}$. □

Analysis of a Symbolic APSP-Algorithm

- Input: $\chi_c(x, y, d) = 1 :\Leftrightarrow c(v_{|x|}, v_{|y|}) = |d|$
- Output: $\chi_\Delta(x, y, d) = 1 :\Leftrightarrow \Delta(v_{|x|}, v_{|y|}) = |d|$
- Computes intermediate functions $S_i(x, y, d)$:
  - S_i covers paths with at most 2^i edges.
  - $S_i(x, y, d) = 1 \Leftrightarrow$ Shortest $v_{|x|}$–$v_{|y|}$-path has cost $|d|$.
- Construct graph G_n such that
  - χ_c and χ_Δ have small OBDDs,
  - $S_i(x, y, d)$ contains $MUL\text{-}GRAPH_n(x, y, d)$.
- $\Rightarrow$ APSP-algorithm needs time $2^{\Omega(n)}$. □

Analysis of a Symbolic APSP-Algorithm

- Input: $\chi_c(x, y, d) = 1 :\Leftrightarrow c(v_{|x|}, v_{|y|}) = |d|$
- Output: $\chi_\Delta(x, y, d) = 1 :\Leftrightarrow \Delta(v_{|x|}, v_{|y|}) = |d|$
- Computes intermediate functions $S_i(x, y, d)$:
  - S_i covers paths with at most 2^i edges.
  - $S_i(x, y, d) = 1 \Leftrightarrow$ Shortest $v_{|x|}$–$v_{|y|}$-path has cost $|d|$.
- Construct graph G_n such that
  - χ_c and χ_Δ have small OBDDs,
  - $S_i(x, y, d)$ contains $MUL\text{-}GRAPH_n(x, y, d)$.
- $\Rightarrow$ APSP-algorithm needs time $2^{\Omega(n)}$. □

Analysis of a Symbolic APSP-Algorithm

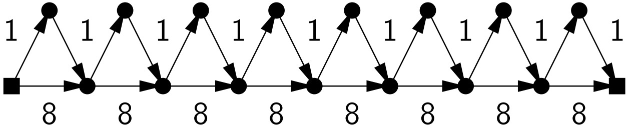
- Input: $\chi_c(x, y, d) = 1 :\Leftrightarrow c(v_{|x|}, v_{|y|}) = |d|$
- Output: $\chi_\Delta(x, y, d) = 1 :\Leftrightarrow \Delta(v_{|x|}, v_{|y|}) = |d|$
- Computes intermediate functions $S_i(x, y, d)$:
  - S_i covers paths with at most 2^i edges.
  - $S_i(x, y, d) = 1 \Leftrightarrow$ Shortest $v_{|x|}$–$v_{|y|}$-path has cost $|d|$.
- Construct graph G_n such that
  - χ_c and χ_Δ have small OBDDs,
  - $S_i(x, y, d)$ contains $MUL\text{-}GRAPH_n(x, y, d)$.
- $\Rightarrow$ APSP-algorithm needs time $2^{\Omega(n)}$. □

Analysis of a Symbolic APSP-Algorithm

- Input: $\chi_c(x, y, d) = 1 :\Leftrightarrow c(v_{|x|}, v_{|y|}) = |d|$
- Output: $\chi_\Delta(x, y, d) = 1 :\Leftrightarrow \Delta(v_{|x|}, v_{|y|}) = |d|$
- Computes intermediate functions $S_i(x, y, d)$:
  - S_i covers paths with at most 2^i edges.
  - $S_i(x, y, d) = 1 \Leftrightarrow$ Shortest $v_{|x|}$–$v_{|y|}$-path has cost $|d|$.
- Construct graph G_n such that
  - χ_c and χ_Δ have small OBDDs,
  - $S_i(x, y, d)$ contains $MUL\text{-}GRAPH_n(x, y, d)$.
- $\Rightarrow$ APSP-algorithm needs time $2^{\Omega(n)}$. □

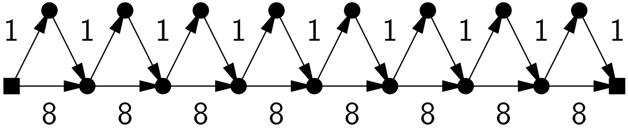
Analysis of a Symbolic APSP-Algorithm

- Input: $\chi_c(x, y, d) = 1 :\Leftrightarrow c(v_{|x|}, v_{|y|}) = |d|$
- Output: $\chi_\Delta(x, y, d) = 1 :\Leftrightarrow \Delta(v_{|x|}, v_{|y|}) = |d|$
- Computes intermediate functions $S_i(x, y, d)$:
  - S_i covers paths with at most 2^i edges.
  - $S_i(x, y, d) = 1 \Leftrightarrow$ Shortest $v_{|x|}$–$v_{|y|}$-path has cost $|d|$.
- Construct graph G_n such that
  - χ_c and χ_Δ have small OBDDs,
  - $S_i(x, y, d)$ contains $MUL\text{-}GRAPH_n(x, y, d)$.
- $\Rightarrow$ APSP-algorithm needs time $2^{\Omega(n)}$. □



Analysis of a Symbolic APSP-Algorithm

- Input: $\chi_c(x, y, d) = 1 :\Leftrightarrow c(v_{|x|}, v_{|y|}) = |d|$
- Output: $\chi_\Delta(x, y, d) = 1 :\Leftrightarrow \Delta(v_{|x|}, v_{|y|}) = |d|$
- Computes intermediate functions $S_i(x, y, d)$:
  - S_i covers paths with at most 2^i edges.
  - $S_i(x, y, d) = 1 \Leftrightarrow$ Shortest $v_{|x|}$–$v_{|y|}$-path has cost $|d|$.
- Construct graph G_n such that
  - χ_c and χ_Δ have small OBDDs,
  - $S_i(x, y, d)$ contains $MUL\text{-}GRAPH_n(x, y, d)$.
- $\Rightarrow$ APSP-algorithm needs time $2^{\Omega(n)}$. □



Summary

- Symbolic (graph) algorithms use branching programs for characteristic functions.
- Algorithm analysis needs charact. functions with exponential BP size.
- Exp. lower bounds for $MUL_{n,n-1}$ can be transferred to $MUL\text{-}GRAPH_n$.
- *MUL-GRAPH* can be encoded into APSP-instances to separate the complexities of different symbolic approaches.
- Similar results hold for Indirect Storage Access and Hidden Weighted Bit.

Summary

- Symbolic (graph) algorithms use branching programs for characteristic functions.
- Algorithm analysis needs charact. functions with exponential BP size.
- Exp. lower bounds for $MUL_{n,n-1}$ can be transferred to *MUL-GRAPH*$_n$.
- *MUL-GRAPH* can be encoded into APSP-instances to separate the complexities of different symbolic approaches.
- Similar results hold for Indirect Storage Access and Hidden Weighted Bit.

Summary

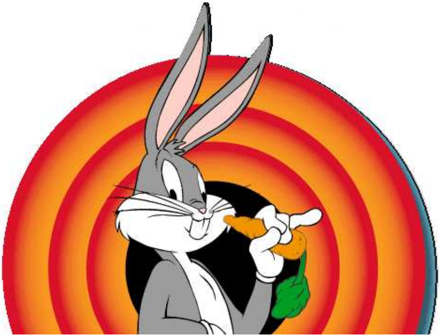
- Symbolic (graph) algorithms use branching programs for characteristic functions.
- Algorithm analysis needs charact. functions with exponential BP size.
- Exp. lower bounds for $MUL_{n,n-1}$ can be transferred to $MUL\text{-}GRAPH_n$.
- *MUL-GRAPH* can be encoded into APSP-instances to separate the complexities of different symbolic approaches.
- Similar results hold for Indirect Storage Access and Hidden Weighted Bit.

Summary

- Symbolic (graph) algorithms use branching programs for characteristic functions.
- Algorithm analysis needs charact. functions with exponential BP size.
- Exp. lower bounds for $MUL_{n,n-1}$ can be transferred to $MUL\text{-}GRAPH_n$.
- *MUL-GRAPH* can be encoded into APSP-instances to separate the complexities of different symbolic approaches.
- Similar results hold for Indirect Storage Access and Hidden Weighted Bit.

Summary

- Symbolic (graph) algorithms use branching programs for characteristic functions.
- Algorithm analysis needs charact. functions with exponential BP size.
- Exp. lower bounds for $MUL_{n,n-1}$ can be transferred to $MUL\text{-}GRAPH_n$.
- *MUL-GRAPH* can be encoded into APSP-instances to separate the complexities of different symbolic approaches.
- Similar results hold for Indirect Storage Access and Hidden Weighted Bit.



“That’s all Folks!”