

The Complexity of Problems on Implicitly Represented Inputs

Daniel Sawitzki

University of Dortmund, Computer Science 2

DFG Jahres-Kolloquium 2006, Aachen

Contents

- 1** Introduction
- 2** P-Complete Problems
- 3** Fixed-Parameter Intractability
- 4** Space Complexity
- 5** Summary

Implicit Data Representation

Definition

An **implicit** data representation avoids explicit enumeration of single elements.

- Focus: Representation by Boolean functions
- Consider string $I \in \{0, 1\}^n$ of length $n = 2^m$.
- Define the **characteristic function** $\chi_I: \{0, 1\}^m \rightarrow \{0, 1\}$ of I by

$$\chi_I(x) = I_{|x|}$$

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

Implicit Data Representation

Definition

An **implicit** data representation avoids explicit enumeration of single elements.

- Focus: Representation by Boolean functions
- Consider string $I \in \{0, 1\}^n$ of length $n = 2^m$.
- Define the **characteristic function** $\chi_I: \{0, 1\}^m \rightarrow \{0, 1\}$ of I by

$$\chi_I(x) = I_{|x|}$$

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

Implicit Data Representation

Definition

An **implicit** data representation avoids explicit enumeration of single elements.

- Focus: Representation by Boolean functions
- Consider string $I \in \{0, 1\}^n$ of length $n = 2^m$.
- Define the **characteristic function** $\chi_I: \{0, 1\}^m \rightarrow \{0, 1\}$ of I by

$$\chi_I(x) = I_{|x|}$$

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

Implicit Algorithms and OBDDs

- Popular data structure for Boolean functions:
Ordered Binary Decision Diagrams (OBDDs)
- OBDDs of structured functions are known to be very succinct.
- OBDDs offer efficient algorithms for functional operations.
- \Rightarrow **Implicit/symbolic algorithms:**
 - Represent input $I \in \{0,1\}^n$ implicitly as OBDD of χ_I .
 - Compute OBDD of χ_O for output $O \in \{0,1\}^*$.
 - Process implicit data via functional OBDD operations.
- Hope: Efficient (sublinear) heuristic on large but structured problem instances

Implicit Algorithms and OBDDs

- Popular data structure for Boolean functions:
Ordered Binary Decision Diagrams (OBDDs)
- OBDDs of structured functions are known to be very succinct.
- OBDDs offer efficient algorithms for functional operations.
- \Rightarrow **Implicit/symbolic algorithms:**
 - Represent input $I \in \{0,1\}^n$ implicitly as OBDD of χ_I .
 - Compute OBDD of χ_O for output $O \in \{0,1\}^*$.
 - Process implicit data via functional OBDD operations.
- Hope: Efficient (sublinear) heuristic on large but structured problem instances

Implicit Algorithms and OBDDs

- Popular data structure for Boolean functions:
Ordered Binary Decision Diagrams (OBDDs)
- OBDDs of structured functions are known to be very succinct.
- OBDDs offer efficient algorithms for functional operations.
- \Rightarrow **Implicit/symbolic algorithms:**
 - Represent input $I \in \{0,1\}^n$ implicitly as OBDD of χ_I .
 - Compute OBDD of χ_O for output $O \in \{0,1\}^*$.
 - Process implicit data via functional OBDD operations.
- Hope: Efficient (sublinear) heuristic on large but structured problem instances

Implicit Algorithms and OBDDs

- Popular data structure for Boolean functions:
Ordered Binary Decision Diagrams (OBDDs)
- OBDDs of structured functions are known to be very succinct.
- OBDDs offer efficient algorithms for functional operations.
- \Rightarrow **Implicit/symbolic algorithms:**
 - Represent input $I \in \{0, 1\}^n$ implicitly as OBDD of χ_I .
 - Compute OBDD of χ_O for output $O \in \{0, 1\}^*$.
 - Process implicit data via functional OBDD operations.
- Hope: Efficient (sublinear) heuristic on large but structured problem instances

Implicit Algorithms and OBDDs

- Popular data structure for Boolean functions:
Ordered Binary Decision Diagrams (OBDDs)
- OBDDs of structured functions are known to be very succinct.
- OBDDs offer efficient algorithms for functional operations.
- \Rightarrow **Implicit/symbolic algorithms:**
 - Represent input $I \in \{0, 1\}^n$ implicitly as OBDD of χ_I .
 - Compute OBDD of χ_O for output $O \in \{0, 1\}^*$.
 - Process implicit data via functional OBDD operations.
- Hope: Efficient (sublinear) heuristic on large but structured problem instances

Implicit Algorithms and OBDDs

- Popular data structure for Boolean functions:
Ordered Binary Decision Diagrams (OBDDs)
- OBDDs of structured functions are known to be very succinct.
- OBDDs offer efficient algorithms for functional operations.
- \Rightarrow **Implicit/symbolic algorithms:**
 - Represent input $I \in \{0, 1\}^n$ implicitly as OBDD of χ_I .
 - Compute OBDD of χ_O for output $O \in \{0, 1\}^*$.
 - Process implicit data via functional OBDD operations.
- Hope: Efficient (sublinear) heuristic on large but structured problem instances

Implicit Algorithms and OBDDs

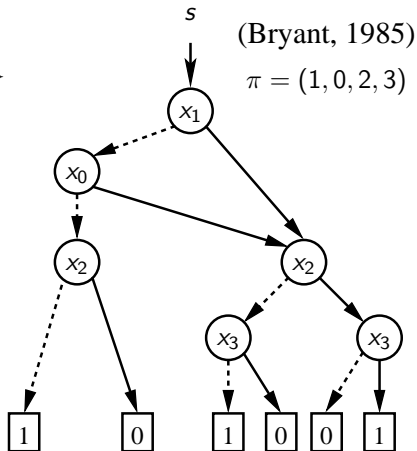
- Popular data structure for Boolean functions:
Ordered Binary Decision Diagrams (OBDDs)
- OBDDs of structured functions are known to be very succinct.
- OBDDs offer efficient algorithms for functional operations.
- \Rightarrow **Implicit/symbolic algorithms:**
 - Represent input $I \in \{0, 1\}^n$ implicitly as OBDD of χ_I .
 - Compute OBDD of χ_O for output $O \in \{0, 1\}^*$.
 - Process implicit data via functional OBDD operations.
- Hope: Efficient (sublinear) heuristic on large but structured problem instances

Implicit Algorithms and OBDDs

- Popular data structure for Boolean functions:
Ordered Binary Decision Diagrams (OBDDs)
- OBDDs of structured functions are known to be very succinct.
- OBDDs offer efficient algorithms for functional operations.
- \Rightarrow **Implicit/symbolic algorithms:**
 - Represent input $I \in \{0, 1\}^n$ implicitly as OBDD of χ_I .
 - Compute OBDD of χ_O for output $O \in \{0, 1\}^*$.
 - Process implicit data via functional OBDD operations.
- Hope: Efficient (sublinear) heuristic on large but structured problem instances

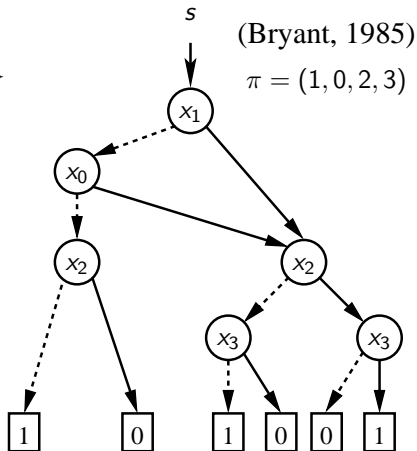
Ordered Binary Decision Diagrams (OBDDs)

- Data structure for $f: \{0, 1\}^m \rightarrow \{0, 1\}$ with Vars. $x_0, \dots, x_{m-1} \in \{0, 1\}$
- OBDD \mathcal{G}_f is acyclic digraph having inner nodes and sinks.
- Inner nodes: Variable label, 0- and 1-edge
- Sink represents value $f(x_0, \dots, x_{m-1})$.
- Evaluation starts at source s .
- Reads vars. w. r. t. $\pi \in \Sigma_m$.



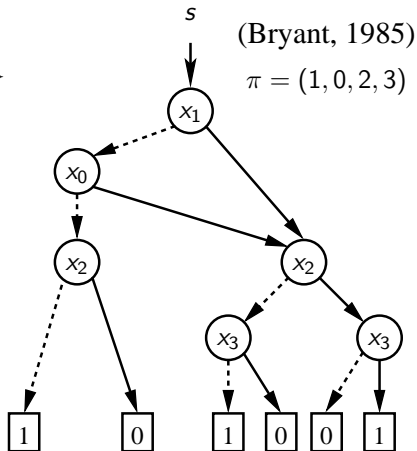
Ordered Binary Decision Diagrams (OBDDs)

- Data structure for $f: \{0, 1\}^m \rightarrow \{0, 1\}$ with Vars. $x_0, \dots, x_{m-1} \in \{0, 1\}$
- OBDD \mathcal{G}_f is acyclic digraph having **inner nodes** and **sinks**.
- Inner nodes: Variable label, 0- and 1-edge
- Sink represents value $f(x_0, \dots, x_{m-1})$.
- Evaluation starts at **source** s .
- Reads vars. w. r. t. $\pi \in \Sigma_m$.



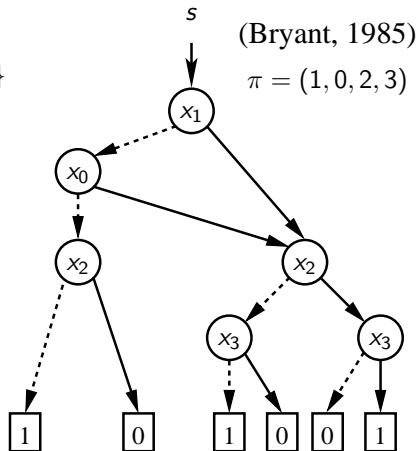
Ordered Binary Decision Diagrams (OBDDs)

- Data structure for $f: \{0, 1\}^m \rightarrow \{0, 1\}$ with Vars. $x_0, \dots, x_{m-1} \in \{0, 1\}$
- OBDD \mathcal{G}_f is acyclic digraph having **inner nodes** and **sinks**.
- Inner nodes: Variable label, 0- and 1-edge
- Sink represents value $f(x_0, \dots, x_{m-1})$.
- Evaluation starts at **source** s .
- Reads vars. w. r. t. $\pi \in \Sigma_m$.



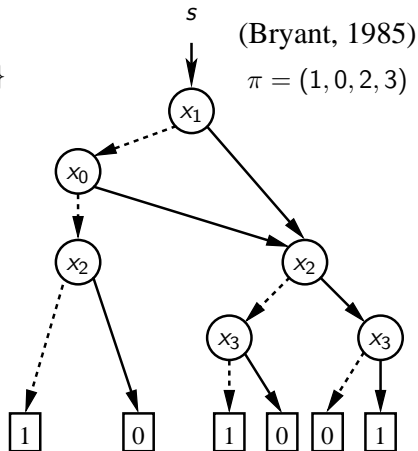
Ordered Binary Decision Diagrams (OBDDs)

- Data structure for $f: \{0, 1\}^m \rightarrow \{0, 1\}$ with Vars. $x_0, \dots, x_{m-1} \in \{0, 1\}$
- OBDD \mathcal{G}_f is acyclic digraph having **inner nodes** and **sinks**.
- Inner nodes: Variable label, 0- and 1-edge
- Sink represents value $f(x_0, \dots, x_{m-1})$.
- Evaluation starts at **source** s .
- Reads vars. w. r. t. $\pi \in \Sigma_m$.



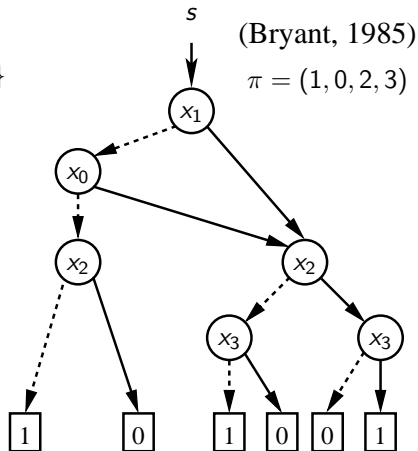
Ordered Binary Decision Diagrams (OBDDs)

- Data structure for $f: \{0, 1\}^m \rightarrow \{0, 1\}$ with Vars. $x_0, \dots, x_{m-1} \in \{0, 1\}$
- OBDD \mathcal{G}_f is acyclic digraph having **inner nodes** and **sinks**.
- Inner nodes: Variable label, 0- and 1-edge
- Sink represents value $f(x_0, \dots, x_{m-1})$.
- Evaluation starts at **source** s .
- Reads vars. w. r. t. $\pi \in \Sigma_m$.



Ordered Binary Decision Diagrams (OBDDs)

- Data structure for $f: \{0, 1\}^m \rightarrow \{0, 1\}$ with Vars. $x_0, \dots, x_{m-1} \in \{0, 1\}$
- OBDD \mathcal{G}_f is acyclic digraph having **inner nodes** and **sinks**.
- Inner nodes: Variable label, 0- and 1-edge
- Sink represents value $f(x_0, \dots, x_{m-1})$.
- Evaluation starts at **source** s .
- Reads vars. w. r. t. $\pi \in \Sigma_m$.



Algorithmic Properties of OBDDs

- Every function f on m vars. has at most OBDD size $\mathcal{O}(2^m/m)$.
- Hope for structured functions: OBDD size $\text{poly}(m)$
- Efficient algorithms for all functional operations (binary operators, quantifications, satisfiability, ...)
- Sequence of $\Omega(m)$ operations may cause exponential blow-up.

Algorithmic Properties of OBDDs

- Every function f on m vars. has at most OBDD size $\mathcal{O}(2^m/m)$.
- Hope for structured functions: OBDD size $\text{poly}(m)$
- Efficient algorithms for all functional operations (binary operators, quantifications, satisfiability, ...)
- Sequence of $\Omega(m)$ operations may cause exponential blow-up.

Algorithmic Properties of OBDDs

- Every function f on m vars. has at most OBDD size $\mathcal{O}(2^m/m)$.
- Hope for structured functions: OBDD size $\text{poly}(m)$
- Efficient algorithms for all functional operations (binary operators, quantifications, satisfiability, ...)
- Sequence of $\Omega(m)$ operations may cause exponential blow-up.

Algorithmic Properties of OBDDs

- Every function f on m vars. has at most OBDD size $\mathcal{O}(2^m/m)$.
- Hope for structured functions: OBDD size $\text{poly}(m)$
- Efficient algorithms for all functional operations (binary operators, quantifications, satisfiability, ...)
- Sequence of $\Omega(m)$ operations may cause exponential blow-up.

Implicit Graph Algorithms: An Example

Example: An implicit BFS algorithm on χ_G for

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

$i := 0; R_0(x) := (|x| = s)$

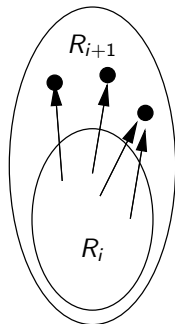
repeat

$N(x) := (\exists y)[\chi_G(y, x) \wedge R_i(y) \wedge \overline{R_i(x)}]$

$R_{i+1}(x) := R_i(x) \vee N(x)$

$i := i + 1$

until $R_i = R_{i-1}$



State of Affairs

- Situation until 2002:
 - OBDDs well established in CAD, Model Checking, ...
 - Pure heuristics for mostly application-specific problems
 - No theoretical analyses of time/space
- Recent contributions:
 - Implicit algorithms for many graph-theoretic problems (Flows, Shortest Paths, Topological Sorting, ...)
 - Polylogarithmic upper bounds for structured instances
- This talk:
 - Lower bound for P-complete problems
 - Fixed-parameter intractability of some graph problems
 - Space complexity of some graph problems

State of Affairs

- Situation until 2002:
 - OBDDs well established in CAD, Model Checking, . . .
 - Pure heuristics for mostly application-specific problems
 - No theoretical analyses of time/space
- Recent contributions:
 - Implicit algorithms for many graph-theoretic problems (Flows, Shortest Paths, Topological Sorting, . . .)
 - Polylogarithmic upper bounds for structured instances
- This talk:
 - Lower bound for P-complete problems
 - Fixed-parameter intractability of some graph problems
 - Space complexity of some graph problems

State of Affairs

- Situation until 2002:
 - OBDDs well established in CAD, Model Checking, . . .
 - Pure heuristics for mostly application-specific problems
 - No theoretical analyses of time/space
- Recent contributions:
 - Implicit algorithms for many graph-theoretic problems (Flows, Shortest Paths, Topological Sorting, . . .)
 - Polylogarithmic upper bounds for structured instances
- This talk:
 - Lower bound for P-complete problems
 - Fixed-parameter intractability of some graph problems
 - Space complexity of some graph problems

State of Affairs

- Situation until 2002:
 - OBDDs well established in CAD, Model Checking, . . .
 - Pure heuristics for mostly application-specific problems
 - No theoretical analyses of time/space
- Recent contributions:
 - Implicit algorithms for many graph-theoretic problems (Flows, Shortest Paths, Topological Sorting, . . .)
 - Polylogarithmic upper bounds for structured instances
- This talk:
 - Lower bound for P-complete problems
 - Fixed-parameter intractability of some graph problems
 - Space complexity of some graph problems

State of Affairs

- Situation until 2002:
 - OBDDs well established in CAD, Model Checking, ...
 - Pure heuristics for mostly application-specific problems
 - No theoretical analyses of time/space
- Recent contributions:
 - Implicit algorithms for many graph-theoretic problems (Flows, Shortest Paths, Topological Sorting, ...)
 - Polylogarithmic upper bounds for structured instances
- This talk:
 - Lower bound for P-complete problems
 - Fixed-parameter intractability of some graph problems
 - Space complexity of some graph problems

State of Affairs

- Situation until 2002:
 - OBDDs well established in CAD, Model Checking, . . .
 - Pure heuristics for mostly application-specific problems
 - No theoretical analyses of time/space
- Recent contributions:
 - Implicit algorithms for many graph-theoretic problems (Flows, Shortest Paths, Topological Sorting, . . .)
 - Polylogarithmic upper bounds for structured instances
- This talk:
 - Lower bound for P-complete problems
 - Fixed-parameter intractability of some graph problems
 - Space complexity of some graph problems

State of Affairs

- Situation until 2002:
 - OBDDs well established in CAD, Model Checking, . . .
 - Pure heuristics for mostly application-specific problems
 - No theoretical analyses of time/space
- Recent contributions:
 - Implicit algorithms for many graph-theoretic problems (Flows, Shortest Paths, Topological Sorting, . . .)
 - Polylogarithmic upper bounds for structured instances
- This talk:
 - Lower bound for P-complete problems
 - Fixed-parameter intractability of some graph problems
 - Space complexity of some graph problems

State of Affairs

- Situation until 2002:
 - OBDDs well established in CAD, Model Checking, . . .
 - Pure heuristics for mostly application-specific problems
 - No theoretical analyses of time/space
- Recent contributions:
 - Implicit algorithms for many graph-theoretic problems (Flows, Shortest Paths, Topological Sorting, . . .)
 - Polylogarithmic upper bounds for structured instances
- This talk:
 - Lower bound for P-complete problems
 - Fixed-parameter intractability of some graph problems
 - Space complexity of some graph problems

State of Affairs

- Situation until 2002:
 - OBDDs well established in CAD, Model Checking, . . .
 - Pure heuristics for mostly application-specific problems
 - No theoretical analyses of time/space
- Recent contributions:
 - Implicit algorithms for many graph-theoretic problems (Flows, Shortest Paths, Topological Sorting, . . .)
 - Polylogarithmic upper bounds for structured instances
- This talk:
 - Lower bound for P-complete problems
 - Fixed-parameter intractability of some graph problems
 - Space complexity of some graph problems

State of Affairs

- Situation until 2002:
 - OBDDs well established in CAD, Model Checking, . . .
 - Pure heuristics for mostly application-specific problems
 - No theoretical analyses of time/space
- Recent contributions:
 - Implicit algorithms for many graph-theoretic problems (Flows, Shortest Paths, Topological Sorting, . . .)
 - Polylogarithmic upper bounds for structured instances
- This talk:
 - Lower bound for P-complete problems
 - Fixed-parameter intractability of some graph problems
 - Space complexity of some graph problems

State of Affairs

- Situation until 2002:
 - OBDDs well established in CAD, Model Checking, . . .
 - Pure heuristics for mostly application-specific problems
 - No theoretical analyses of time/space
- Recent contributions:
 - Implicit algorithms for many graph-theoretic problems (Flows, Shortest Paths, Topological Sorting, . . .)
 - Polylogarithmic upper bounds for structured instances
- This talk:
 - Lower bound for P-complete problems
 - Fixed-parameter intractability of some graph problems
 - Space complexity of some graph problems

Contents

- 1 Introduction
- 2 P-Complete Problems**
- 3 Fixed-Parameter Intractability
- 4 Space Complexity
- 5 Summary

The Number of Functional Operations

- Efficient implicit algos. execute **few** operations on **small** data structures.
- Many works just consider the number of operations (SCCs, Gentilini et al., SODA'03).
- General goal: Design algorithms with $\mathcal{O}(\log^k n)$ operations.
- **Impossible** for P-complete problem (unless $P=NC$)!

The Number of Functional Operations

- Efficient implicit algos. execute **few** operations on **small** data structures.
- Many works just consider the number of operations (SCCs, Gentilini et al., SODA'03).
- General goal: Design algorithms with $\mathcal{O}(\log^k n)$ operations.
- **Impossible** for P-complete problem (unless $P=NC$)!

The Number of Functional Operations

- Efficient implicit algos. execute **few** operations on **small** data structures.
- Many works just consider the number of operations (SCCs, Gentilini et al., SODA'03).
- General goal: Design algorithms with $\mathcal{O}(\log^k n)$ operations.
- **Impossible** for P-complete problem (unless $P=NC$)!

The Number of Functional Operations

- Efficient implicit algos. execute **few** operations on **small** data structures.
- Many works just consider the number of operations (SCCs, Gentilini et al., SODA'03).
- General goal: Design algorithms with $\mathcal{O}(\log^k n)$ operations.
- **Impossible** for P-complete problem (unless $P=NC$)!

A Framework for Implicit Algorithms

Definition

A **Symbolic Register Access Machine (SRAM)** is a RAM with additional **symbolic regs.** S_0, S_1, \dots each holding a Boolean function $f: \{0, 1\}^m \rightarrow \{0, 1\}$. Each functional operation has cost 1. [...]

- SRAM model captures capabilities of all typical implicit (OBDD-based) algorithms.

Theorem

SRAM on input χ_I with time $t(n)$ and $m \leq k \log n$ variables can be simulated by PRAM in parallel time $\mathcal{O}((t(n))^2 \cdot \log^2 n)$ with $\mathcal{O}(n^k)$ processors on $I \in \{0, 1\}^n$.

A Framework for Implicit Algorithms

Definition

A **Symbolic Register Access Machine (SRAM)** is a RAM with additional **symbolic regs.** S_0, S_1, \dots each holding a Boolean function $f: \{0, 1\}^m \rightarrow \{0, 1\}$. Each functional operation has cost 1. [...]

- SRAM model captures capabilities of all typical implicit (OBDD-based) algorithms.

Theorem

SRAM on input χ_I with time $t(n)$ and $m \leq k \log n$ variables can be simulated by PRAM in parallel time $\mathcal{O}((t(n))^2 \cdot \log^2 n)$ with $\mathcal{O}(n^k)$ processors on $I \in \{0, 1\}^n$.

A Framework for Implicit Algorithms

Definition

A **Symbolic Register Access Machine (SRAM)** is a RAM with additional **symbolic regs.** S_0, S_1, \dots each holding a Boolean function $f: \{0, 1\}^m \rightarrow \{0, 1\}$. Each functional operation has cost 1. [...]

- SRAM model captures capabilities of all typical implicit (OBDD-based) algorithms.

Theorem

SRAM on input χ_I with time $t(n)$ and $m \leq k \log n$ variables can be simulated by PRAM in parallel time $\mathcal{O}((t(n))^2 \cdot \log^2 n)$ with $\mathcal{O}(n^k)$ processors on $I \in \{0, 1\}^n$.

Result for P-Complete Problems

Theorem

P-complete problems have no PRAMs with time $\mathcal{O}(\log^k n)$ on $\mathcal{O}(n^k)$ processors unless $P = NC$.

Corollary

P-complete problems have no implicit algorithms with $\mathcal{O}(\log^k n)$ functional operations on $\leq k \log n$ variables unless $P = NC$.

- Example: Flow maximization is P-complete.
- Open: Is 0-1 flow maximization P-complete?
- \Rightarrow No polylog. implicit algo. yet.

Result for P-Complete Problems

Theorem

P-complete problems have no PRAMs with time $\mathcal{O}(\log^k n)$ on $\mathcal{O}(n^k)$ processors unless $P = NC$.

Corollary

P-complete problems have no implicit algorithms with $\mathcal{O}(\log^k n)$ functional operations on $\leq k \log n$ variables unless $P = NC$.

- Example: Flow maximization is P-complete.
- Open: Is 0-1 flow maximization P-complete?
- \Rightarrow No polylog. implicit algo. yet.

Result for P-Complete Problems

Theorem

P-complete problems have no PRAMs with time $\mathcal{O}(\log^k n)$ on $\mathcal{O}(n^k)$ processors unless $P = NC$.

Corollary

P-complete problems have no implicit algorithms with $\mathcal{O}(\log^k n)$ functional operations on $\leq k \log n$ variables unless $P = NC$.

- Example: Flow maximization is P-complete.
- Open: Is 0-1 flow maximization P-complete?
- \Rightarrow No polylog. implicit algo. yet.

Result for P-Complete Problems

Theorem

P-complete problems have no PRAMs with time $\mathcal{O}(\log^k n)$ on $\mathcal{O}(n^k)$ processors unless $P = NC$.

Corollary

P-complete problems have no implicit algorithms with $\mathcal{O}(\log^k n)$ functional operations on $\leq k \log n$ variables unless $P = NC$.

- Example: Flow maximization is P-complete.
- Open: Is 0-1 flow maximization P-complete?
- \Rightarrow No polylog. implicit algo. yet.

Result for P-Complete Problems

Theorem

P-complete problems have no PRAMs with time $\mathcal{O}(\log^k n)$ on $\mathcal{O}(n^k)$ processors unless $P = NC$.

Corollary

P-complete problems have no implicit algorithms with $\mathcal{O}(\log^k n)$ functional operations on $\leq k \log n$ variables unless $P = NC$.

- Example: Flow maximization is P-complete.
- Open: Is 0-1 flow maximization P-complete?
- \Rightarrow No polylog. implicit algo. yet.

Contents

- 1 Introduction
- 2 P-Complete Problems
- 3 Fixed-Parameter Intractability**
- 4 Space Complexity
- 5 Summary

s - t -Connectivity in OBDD-represented Graphs

- Input: $\chi_G(x, y) = 1 \Leftrightarrow (v_{|x|}, v_{|y|}) \in E, s, t \in V$
- Feigenbaum et al. (STACS'98): PSPACE-hard!
 - Technique: Construct small OBDD for configuration transition relation of pol. space bounded TM.
 - For $\Pi \in PSPACE$, TM M_Π and input $I \in \{0, 1\}^m$: Construct OBDD $\chi_{\Pi, I}$ of size $\mathcal{O}(\text{poly}(m))$.
 - Ask if start config. is connected to accepting config.
- W. r. t. graph size: No $\mathcal{O}(\log^k |V|)$ -algorithm.
- Question: Which input OBDD properties might enable polynomial complexity?

s - t -Connectivity in OBDD-represented Graphs

- Input: $\chi_G(x, y) = 1 \Leftrightarrow (v_{|x|}, v_{|y|}) \in E, s, t \in V$
- Feigenbaum et al. (STACS'98): PSPACE-hard!
 - Technique: Construct small OBDD for configuration transition relation of pol. space bounded TM.
 - For $\Pi \in PSPACE$, TM M_Π and input $I \in \{0, 1\}^m$: Construct OBDD $\chi_{\Pi, I}$ of size $\mathcal{O}(\text{poly}(m))$.
 - Ask if start config. is connected to accepting config.
- W. r. t. graph size: No $\mathcal{O}(\log^k |V|)$ -algorithm.
- Question: Which input OBDD properties might enable polynomial complexity?

s - t -Connectivity in OBDD-represented Graphs

- Input: $\chi_G(x, y) = 1 \Leftrightarrow (v_{|x|}, v_{|y|}) \in E, s, t \in V$
- Feigenbaum et al. (STACS'98): PSPACE-hard!
 - Technique: Construct small OBDD for configuration transition relation of pol. space bounded TM.
 - For $\Pi \in PSPACE$, TM M_Π and input $I \in \{0, 1\}^m$: Construct OBDD $\chi_{\Pi, I}$ of size $\mathcal{O}(\text{poly}(m))$.
 - Ask if start config. is connected to accepting config.
- W. r. t. graph size: No $\mathcal{O}(\log^k |V|)$ -algorithm.
- Question: Which input OBDD properties might enable polynomial complexity?

s - t -Connectivity in OBDD-represented Graphs

- Input: $\chi_G(x, y) = 1 \Leftrightarrow (v_{|x|}, v_{|y|}) \in E, s, t \in V$
- Feigenbaum et al. (STACS'98): PSPACE-hard!
 - Technique: Construct small OBDD for configuration transition relation of pol. space bounded TM.
 - For $\Pi \in PSPACE$, TM M_Π and input $I \in \{0, 1\}^m$: Construct OBDD $\chi_{\Pi, I}$ of size $\mathcal{O}(\text{poly}(m))$.
 - Ask if start config. is connected to accepting config.
- W. r. t. graph size: No $\mathcal{O}(\log^k |V|)$ -algorithm.
- Question: Which input OBDD properties might enable polynomial complexity?

s - t -Connectivity in OBDD-represented Graphs

- Input: $\chi_G(x, y) = 1 \Leftrightarrow (v_{|x|}, v_{|y|}) \in E, s, t \in V$
- Feigenbaum et al. (STACS'98): PSPACE-hard!
 - Technique: Construct small OBDD for configuration transition relation of pol. space bounded TM.
 - For $\Pi \in PSPACE$, TM M_Π and input $I \in \{0, 1\}^m$: Construct OBDD $\chi_{\Pi, I}$ of size $\mathcal{O}(\text{poly}(m))$.
 - Ask if start config. is connected to accepting config.
- W. r. t. graph size: No $\mathcal{O}(\log^k |V|)$ -algorithm.
- Question: Which input OBDD properties might enable polynomial complexity?

s - t -Connectivity in OBDD-represented Graphs

- Input: $\chi_G(x, y) = 1 \Leftrightarrow (v_{|x|}, v_{|y|}) \in E, s, t \in V$
- Feigenbaum et al. (STACS'98): PSPACE-hard!
 - Technique: Construct small OBDD for configuration transition relation of pol. space bounded TM.
 - For $\Pi \in PSPACE$, TM M_Π and input $I \in \{0, 1\}^m$: Construct OBDD $\chi_{\Pi, I}$ of size $\mathcal{O}(\text{poly}(m))$.
 - Ask if start config. is connected to accepting config.
- W. r. t. graph size: No $\mathcal{O}(\log^k |V|)$ -algorithm.
- Question: Which input OBDD properties might enable polynomial complexity?

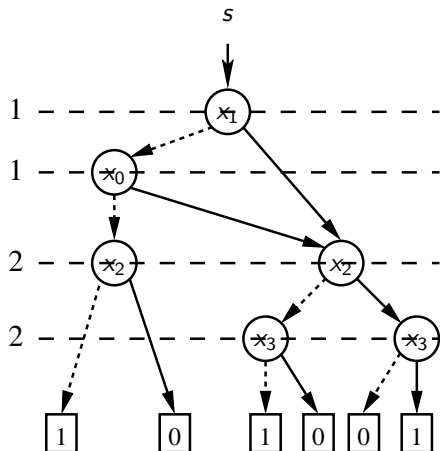
s - t -Connectivity in OBDD-represented Graphs

- Input: $\chi_G(x, y) = 1 \Leftrightarrow (v_{|x|}, v_{|y|}) \in E, s, t \in V$
- Feigenbaum et al. (STACS'98): PSPACE-hard!
 - Technique: Construct small OBDD for configuration transition relation of pol. space bounded TM.
 - For $\Pi \in PSPACE$, TM M_Π and input $I \in \{0, 1\}^m$: Construct OBDD $\chi_{\Pi, I}$ of size $\mathcal{O}(\text{poly}(m))$.
 - Ask if start config. is connected to accepting config.
- W. r. t. graph size: No $\mathcal{O}(\log^k |V|)$ -algorithm.
- Question: Which input OBDD properties might enable polynomial complexity?

Definition of OBDD Width

Definition

The OBDD **width** is the maximum number of nodes labeled the same variable.



OBDD Width as Fixed Parameter

- Efficient algorithms for inputs with small OBDD width W ?
- For width W of χ_G and some function α :
- Parameterized complexity $\mathcal{O}(\log^k |V| \cdot \alpha(W))$ possible?
- Feigenbaum proof: $W = \mathcal{O}(1) \Rightarrow$ No FPT-algo. for s - t -conn.
- New: Fixed-parameter intractability for further problems on OBDD-represented graphs.

OBDD Width as Fixed Parameter

- Efficient algorithms for inputs with small OBDD width W ?
- For width W of χ_G and some function α :
 - Parameterized complexity $\mathcal{O}(\log^k |V| \cdot \alpha(W))$ possible?
 - Feigenbaum proof: $W = \mathcal{O}(1) \Rightarrow$ No FPT-algo. for s - t -conn.
 - New: Fixed-parameter intractability for further problems on OBDD-represented graphs.

OBDD Width as Fixed Parameter

- Efficient algorithms for inputs with small OBDD width W ?
- For width W of χ_G and some function α :
- Parameterized complexity $\mathcal{O}(\log^k |V| \cdot \alpha(W))$ possible?
- Feigenbaum proof: $W = \mathcal{O}(1) \Rightarrow$ No FPT-algo. for s - t -conn.
- New: Fixed-parameter intractability for further problems on OBDD-represented graphs.

OBDD Width as Fixed Parameter

- Efficient algorithms for inputs with small OBDD width W ?
- For width W of χ_G and some function α :
- Parameterized complexity $\mathcal{O}(\log^k |V| \cdot \alpha(W))$ possible?
- Feigenbaum proof: $W = \mathcal{O}(1) \Rightarrow$ No FPT-algo. for s - t -conn.
- New: Fixed-parameter intractability for further problems on OBDD-represented graphs.

OBDD Width as Fixed Parameter

- Efficient algorithms for inputs with small OBDD width W ?
- For width W of χ_G and some function α :
- Parameterized complexity $\mathcal{O}(\log^k |V| \cdot \alpha(W))$ possible?
- Feigenbaum proof: $W = \mathcal{O}(1) \Rightarrow$ No FPT-algo. for s - t -conn.
- New: Fixed-parameter intractability for further problems on OBDD-represented graphs.

OBDD Width as Fixed Parameter

- Efficient algorithms for inputs with small OBDD width W ?
- For width W of χ_G and some function α :
- Parameterized complexity $\mathcal{O}(\log^k |V| \cdot \alpha(W))$ possible?
- Feigenbaum proof: $W = \mathcal{O}(1) \Rightarrow$ No FPT-algo. for s - t -conn.
- New: Fixed-parameter intractability for further problems on OBDD-represented graphs.

Theorem (FPT w. r. t. input OBDD width)

Neither s - t -Conn., Acyclicity, Connectivity, Bipartiteness, Eulerian-Cycle, Planarity, SSSP, APSP, MaxFlow, nor MST on OBDD-represented graphs has an FPT-algo., unless $P=PSPACE$.

OBDD Width as Fixed Parameter

- Efficient algorithms for inputs with small OBDD width W ?
- For width W of χ_G and some function α :
- Parameterized complexity $\mathcal{O}(\log^k |V| \cdot \alpha(W))$ possible?
- Feigenbaum proof: $W = \mathcal{O}(1) \Rightarrow$ No FPT-algo. for s - t -conn.
- New: Fixed-parameter intractability for further problems on OBDD-represented graphs.

Theorem (FPT w. r. t. input and output OBDD width)

*Neither **SSSP**, **APSP**, **MaxFlow**, nor **MST** on OBDD-represented graphs has an FPT-algo., unless $P=PSPACE$.*

Contents

- 1 Introduction
- 2 P-Complete Problems
- 3 Fixed-Parameter Intractability
- 4 Space Complexity**
- 5 Summary

Upper Bounds

- So far: We cannot expect polynomial time on constant width OBDDs.
- Implicit graph algorithms with polynomial space (i. e., polylog. in $|V|$)?
- For decision problems: Nondeterministic (counter-)example construction storing $\mathcal{O}(1)$ nodes/edges at once.
- For optimization problems with polynomial output OBDD: Enumerate all polynomial-size solutions.

Upper Bounds

- So far: We cannot expect polynomial time on constant width OBDDs.
- Implicit graph algorithms with polynomial space (i. e., polylog. in $|V|$)?
- For decision problems: Nondeterministic (counter-)example construction storing $\mathcal{O}(1)$ nodes/edges at once.
- For optimization problems with polynomial output OBDD: Enumerate all polynomial-size solutions.

Upper Bounds

- So far: We cannot expect polynomial time on constant width OBDDs.
- Implicit graph algorithms with polynomial space (i. e., polylog. in $|V|$)?
- For decision problems: Nondeterministic (counter-)example construction storing $\mathcal{O}(1)$ nodes/edges at once.
- For optimization problems with polynomial output OBDD: Enumerate all polynomial-size solutions.

Upper Bounds

- So far: We cannot expect polynomial time on constant width OBDDs.
- Implicit graph algorithms with polynomial space (i. e., polylog. in $|V|$)?
- For decision problems: Nondeterministic (counter-)example construction storing $\mathcal{O}(1)$ nodes/edges at once.
- For optimization problems with polynomial output OBDD: Enumerate all polynomial-size solutions.

Theorem

s-t-Conn., Acyclicity, Connectivity, Bipartiteness, and Eulerian-Cycle [. . .] on OBDD-represented graphs are in PSPACE.

Upper Bounds

- So far: We cannot expect polynomial time on constant width OBDDs.
- Implicit graph algorithms with polynomial space (i. e., polylog. in $|V|$)?
- For decision problems: Nondeterministic (counter-)example construction storing $\mathcal{O}(1)$ nodes/edges at once.
- For optimization problems with polynomial output OBDD: Enumerate all polynomial-size solutions.

Theorem

SSSP, APSP, MaxFlow, and MST [...] on OBDD-represented graphs have polynomial space algorithms w. r. t. input and output OBDD size.

Lower Bounds

- Is space dependency on output OBDD size necessary?
- Technique: Encode OBDD-hard functions into structured graph instances.

Theorem

Maximum Flow, Shortest Paths, and Restricted Reachability on OBDD-represented graphs have exponential space complexity.

Consider OBDD $\chi_G(x, y)$ for $G = (V, E)$

Definition

Restricted Reachability: Compute $\{\chi_{R_i}(x)\}_i$ for

$$\chi_{R_i}(x) = 1 :\Leftrightarrow v_x \text{ reachable from } s \in V \text{ via at most } 2^i \text{ edges.}$$

Lower Bounds

- Is space dependency on output OBDD size necessary?
- Technique: Encode OBDD-hard functions into structured graph instances.

Theorem

Maximum Flow, Shortest Paths, and Restricted Reachability on OBDD-represented graphs have exponential space complexity.

Consider OBDD $\chi_G(x, y)$ for $G = (V, E)$

Definition

Restricted Reachability: Compute $\{\chi_{R_i}(x)\}_i$ for

$\chi_{R_i}(x) = 1 \Leftrightarrow v_x$ reachable from $s \in V$ via at most 2^i edges.

Lower Bounds

- Is space dependency on output OBDD size necessary?
- Technique: Encode OBDD-hard functions into structured graph instances.

Theorem

Maximum Flow, Shortest Paths, and Restricted Reachability on OBDD-represented graphs have exponential space complexity.

Consider OBDD $\chi_G(x, y)$ for $G = (V, E)$

Definition

Restricted Reachability: Compute $\{\chi_{R_i}(x)\}_i$ for

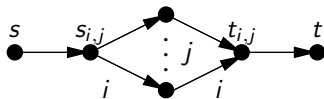
$$\chi_{R_i}(x) = 1 :\Leftrightarrow v_x \text{ reachable from } s \in V \text{ via at most } 2^i \text{ edges.}$$

Example: Maximum Flow

- Instance has components $G_{i,j}$ with max. flow $i \cdot j$.
- Characteristic function of max. flow F on edge $(s, s_{i,j})$:

$$\chi_F(s, s_{i,j}, a) = 1 \Leftrightarrow (i \cdot j = a)$$

- Inherently contains the **graph of multiplication**—exp. OBDD-size for any var. order!

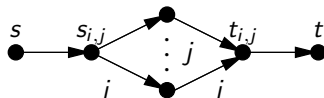


Example: Maximum Flow

- Instance has components $G_{i,j}$ with max. flow $i \cdot j$.
- Characteristic function of max. flow F on edge $(s, s_{i,j})$:

$$\chi_F(s, s_{i,j}, a) = 1 \Leftrightarrow (i \cdot j = a)$$

- Inherently contains the **graph of multiplication**—exp. OBDD-size for any var. order!

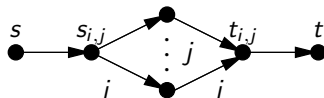


Example: Maximum Flow

- Instance has components $G_{i,j}$ with max. flow $i \cdot j$.
- Characteristic function of max. flow F on edge $(s, s_{i,j})$:

$$\chi_F(s, s_{i,j}, a) = 1 \Leftrightarrow (i \cdot j = a)$$

- Inherently contains the **graph of multiplication**—exp. OBDD-size for any var. order!



Contents

- 1 Introduction
- 2 P-Complete Problems
- 3 Fixed-Parameter Intractability
- 4 Space Complexity
- 5 Summary**

Summary

- P-complete problems cannot be solved by $\mathcal{O}(\log^k n)$ functional operations using $k \log n$ variables (unless $P=NC$).
- Fundamental graph problems have no OBDD-based FPT algorithms w. r. t. fixed input and output OBDD width (unless $P=PSPACE$).
- SSSP, APSP, MaxFlow, and (restricted) Reachability have exp. space complexity.
- \Rightarrow Practical success of OBDDs has to be explained by further instance properties.
- Alternative conjecture: Also in practice, OBDD-based heuristics are (weak) exponential.

Summary

- P-complete problems cannot be solved by $\mathcal{O}(\log^k n)$ functional operations using $k \log n$ variables (unless $P=NC$).
- Fundamental graph problems have no OBDD-based FPT algorithms w. r. t. fixed input and output OBDD width (unless $P=PSPACE$).
- SSSP, APSP, MaxFlow, and (restricted) Reachability have exp. space complexity.
- \Rightarrow Practical success of OBDDs has to be explained by further instance properties.
- Alternative conjecture: Also in practice, OBDD-based heuristics are (weak) exponential.

Summary

- P-complete problems cannot be solved by $\mathcal{O}(\log^k n)$ functional operations using $k \log n$ variables (unless $P=NC$).
- Fundamental graph problems have no OBDD-based FPT algorithms w. r. t. fixed input and output OBDD width (unless $P=PSPACE$).
- SSSP, APSP, MaxFlow, and (restricted) Reachability have exp. space complexity.
- \Rightarrow Practical success of OBDDs has to be explained by further instance properties.
- Alternative conjecture: Also in practice, OBDD-based heuristics are (weak) exponential.

Summary

- P-complete problems cannot be solved by $\mathcal{O}(\log^k n)$ functional operations using $k \log n$ variables (unless $P=NC$).
- Fundamental graph problems have no OBDD-based FPT algorithms w. r. t. fixed input and output OBDD width (unless $P=PSPACE$).
- SSSP, APSP, MaxFlow, and (restricted) Reachability have exp. space complexity.
- \Rightarrow Practical success of OBDDs has to be explained by further instance properties.
- Alternative conjecture: Also in practice, OBDD-based heuristics are (weak) exponential.

Summary

- P-complete problems cannot be solved by $\mathcal{O}(\log^k n)$ functional operations using $k \log n$ variables (unless $P=NC$).
- Fundamental graph problems have no OBDD-based FPT algorithms w. r. t. fixed input and output OBDD width (unless $P=PSPACE$).
- SSSP, APSP, MaxFlow, and (restricted) Reachability have exp. space complexity.
- \Rightarrow Practical success of OBDDs has to be explained by further instance properties.
- Alternative conjecture: Also in practice, OBDD-based heuristics are (weak) exponential.

“That’s all Folks!”

References at <http://ls2-www.cs.uni-dortmund.de/spp1126/>