Representation of Graphs by OBDDs

Robin Nunkesser[1] Philipp Wölfel[2]

[1]Department of Computer Science, University of Dortmund

[2]Department of Computer Science, University of Toronto

ISAAC 2005

Outline

1. Introduction
   - Motivation
   - Graph Representation with OBDDs
   - Overview of the Results
2. Some Upper Bounds
   - Cographs
   - Unit Interval Graphs

Motivation

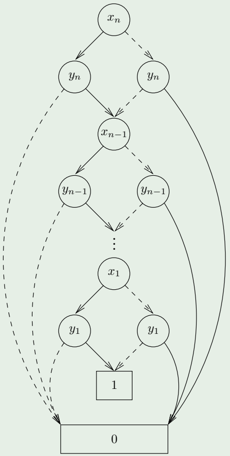
Space requirement and adjacency query time

	adjacency matrix	adjacency list	OBDDs
space	$\Theta\left(n^2\right)$	$\mathcal{O}\left(n+m\right)$	?
adjacency query	$\mathcal{O}\left(1\right)$	$\mathcal{O}\left(n\right)$	$\mathcal{O}\left(\log n\right)$

- Very huge graphs cannot be stored explicitly.
- Implicit representations are generally not generic.
- OBDDs offer an implicit and generic representation of graphs.
- Special OBDD Algorithms e.g. for network flow maximization [Hachtel and Somenzi(1997), Sawitzki(2004a)], topological sorting [Woelfel(2003)] or for finding shortest paths in networks [Sawitzki(2004b)] have been devised.

Graph Representation with OBDDs

Example of an OBDD



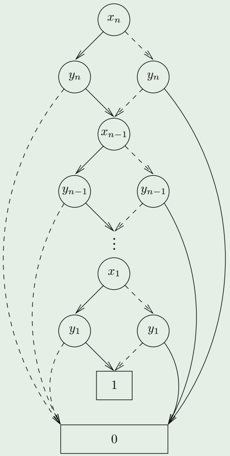
Explanation

- OBDDs represent boolean functions with a directed, acyclic graph.
- The example represents the boolean function $\mathrm{EQUAL}(x, y) \in B_{2n}$ with

$$\mathrm{EQUAL}(x, y) = 1 \iff x = y.$$

Graph Representation with OBDDs

Example of an OBDD



Definition

A graph $G = (V, E)$ is represented by an OBDD, if the OBDD represents the characteristic function of G's edges $\chi_E : E \to \{0, 1\}$, where

$$\chi_E(v_1, v_2) = 1 \iff \{v_1, v_2\} \in E.$$

(The nodes have to be encoded by bit strings)

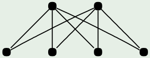
Overview of the Results

Upper and lower bounds for the size of the OBDDs

graph class	upper bound	lower bound
cograph, P_4-reducible, P_4-sparse, P_4-extendible	$\mathcal{O}(n \log n)$	$\Omega(n/\log n)$
unit interval graph	$\mathcal{O}(n/\sqrt{\log n})$	$\Omega(n/\log n)$
interval graph	$\mathcal{O}(n^{3/2}/\log^{3/4} n)$	$\Omega(n)$
undirected graph	$\mathcal{O}(n^2/\log n)$ $\mathcal{O}(m \log n)$	$\Omega(n^2/\log n)$
bipartite graph	$\mathcal{O}(n^2/\log n)$ $\mathcal{O}(m \log n)$	$\Omega(n^2/\log n)$

Cographs

Example of a cograph



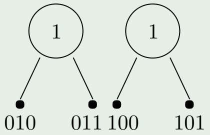
Definition

A graph is a *cograph*, if it contains no induced path P_4 ().

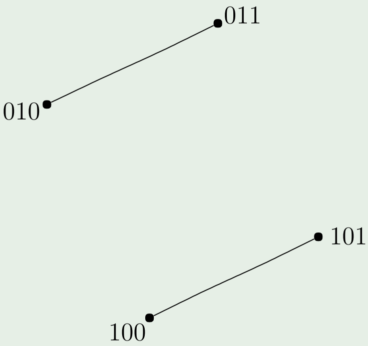
- Cographs can be constructed from single vertex graphs by *graph unions* and *graph joins*.
- $G_1 \cup G_2 = (V(G_1) \cup V(G_2), E(G_1) \cup E(G_2))$
- $G_1 + G_2 = (V(G_1) \cup V(G_2), E(G_1) \cup E(G_2)$
  $\cup \{\{v, w\} | v \in V(G_1), w \in V(G_2)\})$
- Cographs have a unique tree representation reflecting the operations generating the cograph.

Unique Tree Representation

Unique tree representation

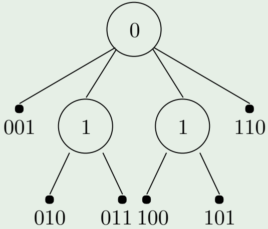


Corresponding cograph

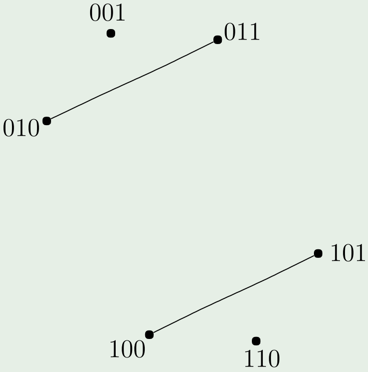


Unique Tree Representation

Unique tree representation

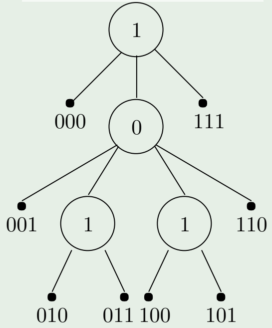


Corresponding cograph

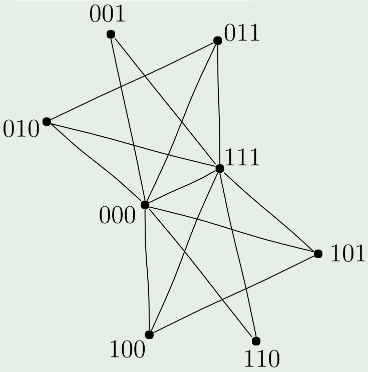


Unique Tree Representation

Unique tree representation

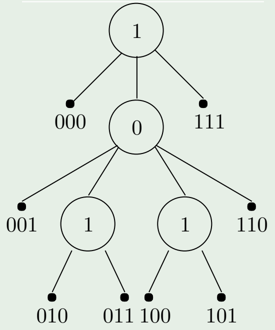


Corresponding cograph



Upper Bound

Unique tree representation

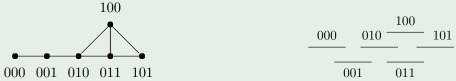


- Adjacency is determined by the lca in the tree representation.
- We encode the vertices according to a preorder traversal.

1. Store the binary encoding x of the first vertex while reading it bit by bit. **OBDD-size:** $\sum_{i=0}^{\lceil \log n \rceil} \left\lceil \frac{n}{2^i} \right\rceil = \mathcal{O}(n)$
2. Read the binary encoding y of the second vertex bit by bit and store the lowest node that may be lca for the so far read encodings ($x < y$, $x > y$ or $x = y$ has also to be stored). **OBDD-size:** $\mathcal{O}(n \log n)$

Unit Interval Graphs

Example of an unit interval graph and the corresponding intervals



1. Alternatingly read the bits of x and y.
2. Determine the sets V_x and V_y of the vertices, whose encoding starts with x and y, respectively.
    - No vertex in V_x is adjacent to a vertex in V_y: Output 0
    - Every vertex in V_x is adjacent to every vertex in V_y: Output 1
    - There is a pair of vertices in $V_x \times V_y$, that is adjacent and a pair that is not adjacent: Store the bits of x and y read and go to 1.

- **OBDD-size**: $\mathcal{O}(n/\sqrt{\log n})$ (we mainly have to estimate the number of occurrences of the last case)

Upper Bound

- Assume w.l.o.g. that $|x| < |y|$ and for the sake of simplicity that we have read as many bits of x as of y.
- Let $(|\alpha_1|, |\beta_1|), \ldots, (|\alpha_p|, |\beta_p|)$ be the lexicographical ordered list of all assignments to the first k bits of x (assignment α_i) and y (assignment β_i) where the last case occurs.

Claim

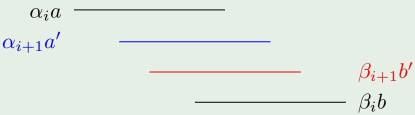
Let $(|\alpha_i|, |\beta_i|)$ be as described above, then $\forall 1 \leq i \leq p : |\beta_i| \leq |\beta_{i+1}|$.

- We obtain $p \leq (2^k - 1) + (2^k - 1) + 1$ from this claim.
- Summing this over all k and using a trivial upper bound for large values of k leads to $\mathcal{O}(n/\sqrt{\log n})$.

Proof of the Claim

- Assume $|\alpha_i| < |\alpha_{i+1}|$ (otherwise the claim is trivial).
- For the sake of contradiction assume that $|\beta_i| > |\beta_{i+1}|$ and therefore $|\alpha_i| < |\alpha_{i+1}| \leq |\beta_{i+1}| < |\beta_i|$.
- As the last case occurs, there have to be assignments a and b to the remaining variables, such that the vertices $\alpha_i a$ and $\beta_i b$ are adjacent.
- Consider additional arbitrary assignments a' and b' to the remaining variables for α_{i+1} and β_{i+1}.
- Obviously, then $|\alpha_i a| < |\alpha_{i+1} a'| < |\beta_i b|$ and $|\alpha_i a| < |\beta_{i+1} b'| < |\beta_i b|$.

The position of the corresponding intervals



- Hence, $|\alpha_{i+1} a'|$ and $|\beta_{i+1} b'|$ are also adjacent, which is contradiction.

Summary

- OBDDs offer a general and implicit representation of graphs.
- For sufficiently structured graphs this representation is advantageous with respect to space requirements.

Thank you!