Interval Graphs: Canonical Representations in Logspace

Johannes Köbler *Sebastian Kuhnert*
Bastian Laubner Oleg Verbitsky

Bordeaux, July 7th, 2010

Overview

Theorem (Our main result)

Interval graph isomorphism can be decided in logspace.[1]

Previous results:

- Linear time [Lueker, Booth 79]
- In AC^2 [Klein 96]

First step in both cases: Compute a *perfect elimination order*.

Not obvious how to do that in logspace!

[1] In fact, we compute canonical interval representations. Details follow.

Outline

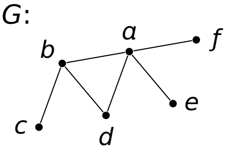
1. Interval graphs
   - Definition
   - Inclusion-maximal cliques
   - Interval hypergraphs
2. Canonical interval representations
   - Step 1: Decomposition into overlap components
   - Step 2: Canonizing overlap components
   - Step 3: Canonizing whole interval hypergraphs
3. Results and open problems

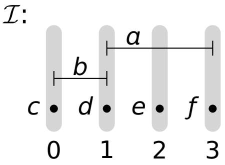
Interval graphs

Definition (Interval Graph)

A graph G is an *interval graph* iff it is (isomorphic to) the intersection graph of a set $\mathcal{I}$ of intervals.
Such an $\mathcal{I}$ is an *interval representation* of G.

- Each interval corresponds to a vertex.
- Two vertices are adjacent iff the corresponding intervals intersect.



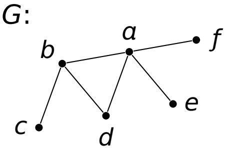


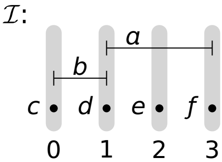
Interval graphs

Definition (Interval Graph)

A graph G is an *interval graph* iff it is (isomorphic to) the intersection graph of a set $\mathcal{I}$ of intervals.
Such an $\mathcal{I}$ is an *interval representation* of G.

- Each interval corresponds to a vertex.
- Two vertices are adjacent iff the corresponding intervals intersect.



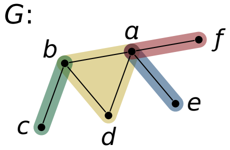


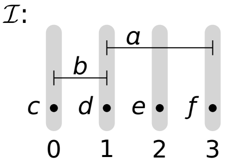
- An interval representation $\mathcal{I}$ of G is *minimal*, if no interval representation of G has fewer points.

Maxcliques

Definition

A *maxclique* is an inclusion-maximal clique.





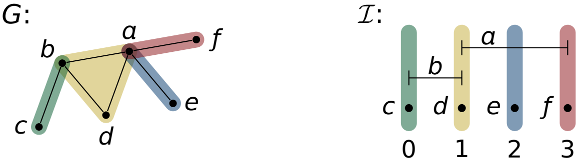
Maxcliques

Definition

A *maxclique* is an inclusion-maximal clique.

Lemma

If $\mathcal{I}$ is a minimal interval representation of G, then the points of $\mathcal{I}$ correspond to the maxcliques of G.



Maxcliques

Definition

A *maxclique* is an inclusion-maximal clique.

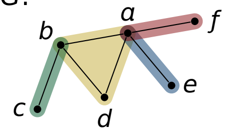
Lemma

If $\mathcal{I}$ is a minimal interval representation of G, then the points of $\mathcal{I}$ correspond to the maxcliques of G.

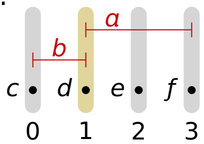
Lemma

Each maxclique of an interval graph G can be represented as the common neighborhood of two vertices.

G:

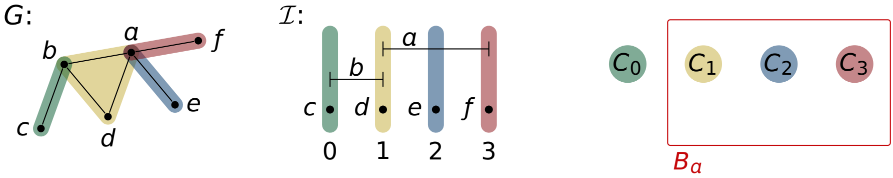


$\mathcal{I}$:



Bundles of maxcliques

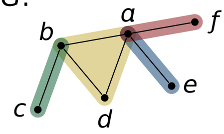
- For a vertex $v \in V(G)$, the *bundle* B_v is the set of those maxcliques that contain v.



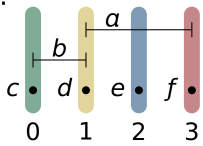
Bundles of maxcliques

- For a vertex $v \in V(G)$, the *bundle B_v* is the set of those maxcliques that contain v.
- The *bundle hypergraph $\mathcal{B}_G$* of G has the maxcliques as vertices and the bundles as hyperedges.

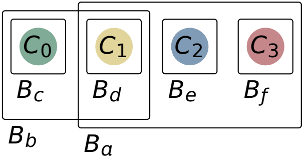
G:



$\mathcal{I}$:



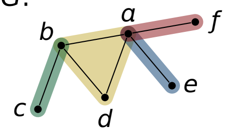
$\mathcal{B}_G$:



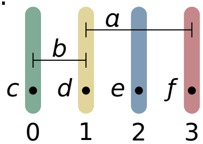
Bundles of maxcliques

- For a vertex $v \in V(G)$, the *bundle B_v* is the set of those maxcliques that contain v.
- The *bundle hypergraph $\mathcal{B}_G$* of G has the maxcliques as vertices and the bundles as hyperedges.

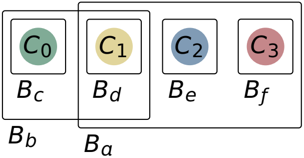
G:



$\mathcal{I}$:



$\mathcal{B}_G$:

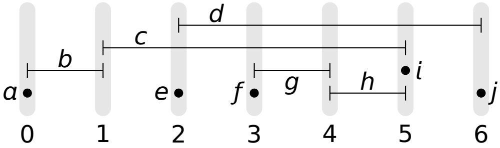


Lemma

For every minimal interval representation $\mathcal{I}$ of G:
$\mathcal{I} \cong \mathcal{B}_G$ as hypergraphs.

Interval hypergraphs

- A hypergraph $\mathcal{H}$ is an *interval hypergraph* iff it is isomorphic to a set of intervals $\mathcal{I}$.
- An isomorphism from $\mathcal{H}$ to $\mathcal{I}$ induces an *interval labeling ℓ* that maps hyperedges to intervals.



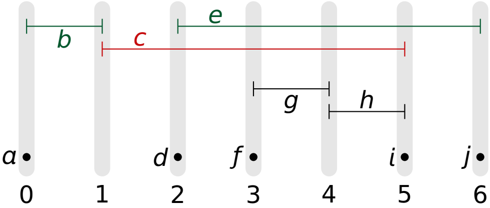
- We compute *canonical interval labelings* in logspace: For each interval hypergraph $\mathcal{H}$ we compute an interval labeling $\ell_{\mathcal{H}}$ such that $\mathcal{H} \cong \mathcal{K} \Leftrightarrow \mathcal{H}^{\ell_{\mathcal{H}}} = \mathcal{K}^{\ell_{\mathcal{K}}}$.

Outline

1. Interval graphs
   - Definition
   - Inclusion-maximal cliques
   - Interval hypergraphs
2. Canonical interval representations
   - Step 1: Decomposition into overlap components
   - Step 2: Canonizing overlap components
   - Step 3: Canonizing whole interval hypergraphs
3. Results and open problems

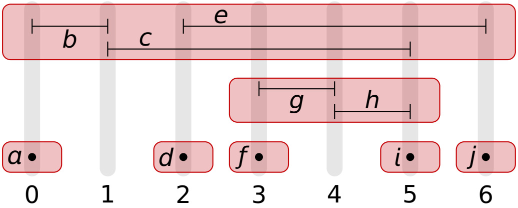
Step 1: Decomposition into overlap components

- Two hyperedges A, B *overlap* ($A \between B$), iff $A \cap B \notin \{\emptyset, A, B\}$



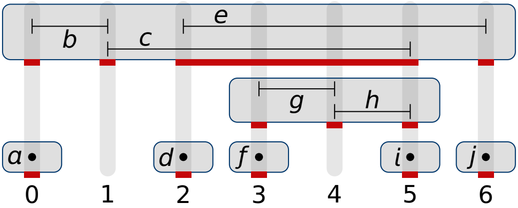
Step 1: Decomposition into overlap components

- Two hyperedges A, B *overlap* ($A \between B$), iff $A \cap B \notin \{\emptyset, A, B\}$
- Two hyperedges are in the same *overlap component*, iff they are connected by an overlap-path.



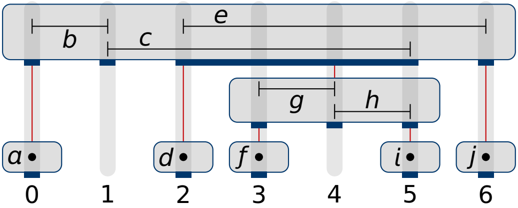
Step 1: Decomposition into overlap components

- Two hyperedges A, B *overlap* ($A \between B$), iff $A \cap B \notin \{\emptyset, A, B\}$
- Two hyperedges are in the same *overlap component*, iff they are connected by an overlap-path.
- Two points are in the same *slot* of an overlap component, iff they belong to the same edges of this component.



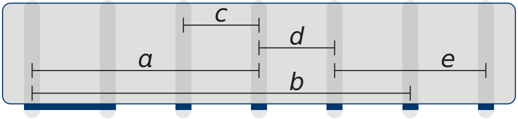
Step 1: Decomposition into overlap components

- Two hyperedges A, B *overlap* ($A \between B$), iff $A \cap B \notin \{\emptyset, A, B\}$
- Two hyperedges are in the same *overlap component*, iff they are connected by an overlap-path.
- Two points are in the same *slot* of an overlap component, iff they belong to the same edges of this component.
- Overlap components form a tree: Each component is located at a slot of its parent.



Step 2: Canonizing overlap components

Challenge: compute a canonical interval labeling for a single overlap component

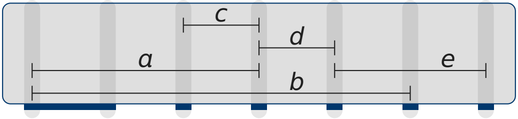


Step 2: Canonizing overlap components

Challenge: compute a canonical interval labeling for a single overlap component

Lemma

The interval representation of an overlap component is unique up to reversing. Both admissible interval labelings can be computed in logspace.



Step 2: Canonizing overlap components

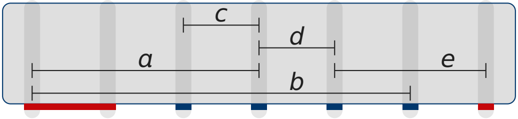
Challenge: compute a canonical interval labeling for a single overlap component

Lemma

The interval representation of an overlap component is unique up to reversing. Both admissible interval labelings can be computed in logspace.

Sketch of the algorithm:

1. Identify the two side-slots, take one as leftmost



Step 2: Canonizing overlap components

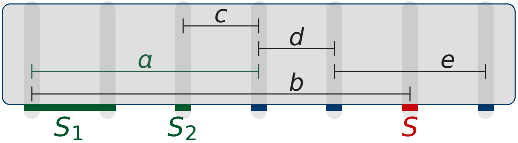
Challenge: compute a canonical interval labeling for a single overlap component

Lemma

The interval representation of an overlap component is unique up to reversing. Both admissible interval labelings can be computed in logspace.

Sketch of the algorithm:

1. Identify the two side-slots, take one as leftmost
2. For each hyperedge E and slots $S_1, S_2 \subseteq E$, $S \not\subseteq E$: S_1 is left (right) of S iff S_2 is left (right) of S



Step 2: Canonizing overlap components

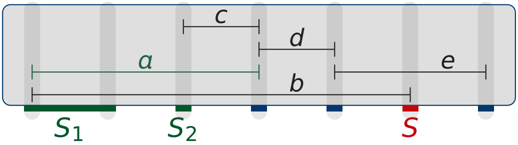
Challenge: compute a canonical interval labeling for a single overlap component

Lemma

The interval representation of an overlap component is unique up to reversing. Both admissible interval labelings can be computed in logspace.

Sketch of the algorithm:

1. Identify the two side-slots, take one as leftmost
2. For each hyperedge E and slots $S_1, S_2 \subseteq E$, $S \not\subseteq E$: S_1 is left (right) of S iff S_2 is left (right) of S
   - This yields a linear order on the slots [Laubner 09]



Step 2: Canonizing overlap components

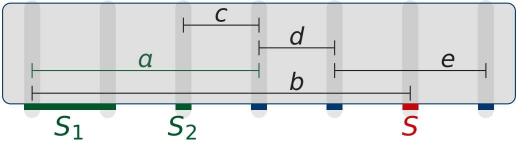
Challenge: compute a canonical interval labeling for a single overlap component

Lemma

The interval representation of an overlap component is unique up to reversing. Both admissible interval labelings can be computed in logspace.

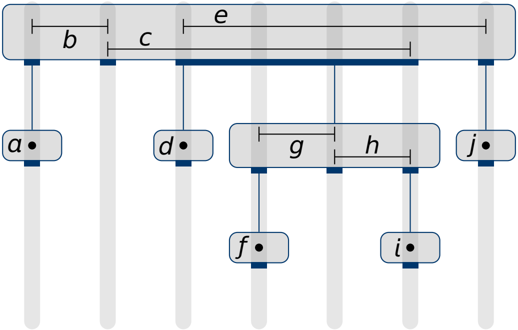
Sketch of the algorithm:

1. Identify the two side-slots, take one as leftmost
2. For each hyperedge E and slots $S_1, S_2 \subseteq E$, $S \not\subseteq E$: S_1 is left (right) of S iff S_2 is left (right) of S
   - This yields a linear order on the slots [Laubner 09]
   - Compute this order as undirected reachability in an auxiliary graph [Reingold 05]



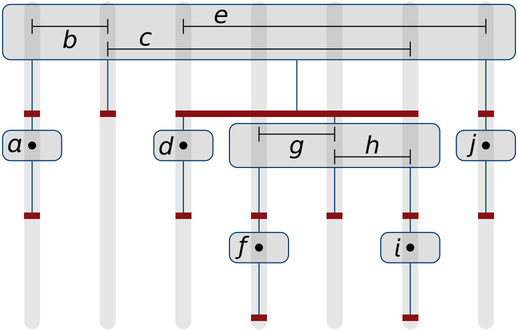
Step 3: Canonizing whole interval hypergraphs

- Idea: Canonize the tree of overlap components
- Complication: Restrictions on the order of children



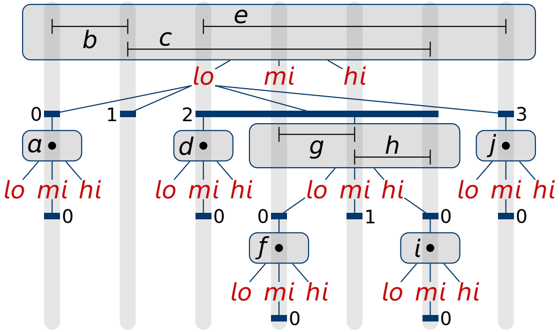
Step 3: Canonizing whole interval hypergraphs

- Idea: Canonize the tree of overlap components
- Complication: Restrictions on the order of children
- Solution: Slot-nodes and connector-nodes, colors



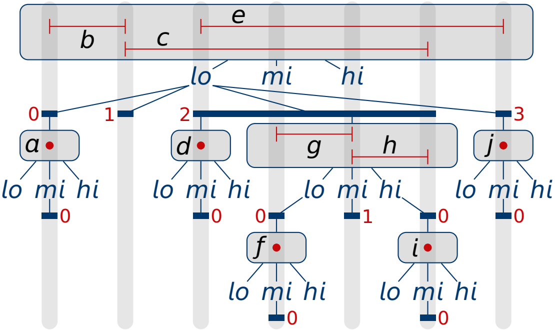
Step 3: Canonizing whole interval hypergraphs

- Idea: Canonize the tree of overlap components
- Complication: Restrictions on the order of children
- Solution: Slot-nodes and connector-nodes, colors
  - Asymmetric components: All slots are *low*, colored by distance to left side
  - Symmetric components: Slots are *low/middle/high* and colored by distance to the nearest side



Step 3: Canonizing whole interval hypergraphs

- Idea: Canonize the tree of overlap components
- Complication: Restrictions on the order of children
- Solution: Slot-nodes and connector-nodes, colors
  - Asymmetric components: All slots are *low*, colored by distance to left side
  - Symmetric components: Slots are *low/middle/high* and colored by distance to the nearest side



Outline

Results

Theorem

Canonical interval labelings of interval (hyper)graphs can be computed in logspace.

Corollary

Recognition, isomorphism and automorphism problems of interval (hyper)graphs and convex graphs are in logspace.

Theorem

For proper/unit interval graphs, canonical proper/unit interval representations can be computed in logspace.

Theorem

Recognition, isomorphism and automorphism problems of the mentioned graph classes are hard for logspace.

Open problems

- Circular arc graphs:
  Intersection graphs of arcs on a circle
  - Recognition in linear time [Kaplan, Nussbaum 06]
  - But: Algorithms require different techniques
- Rooted directed path graphs:
  Intersection graphs of paths in a rooted directed tree
  - Maxcliques can be recognized
  - But: Ordering of maxcliques within overlap components fails
- Generalizations to 2 dimensions:
  - Boxicity 2 graphs are isomorphism complete [Uehara 08]
  - What about (unit) squares/circles?

Thank you!

Literature

Kaplan, Haim, Yahav Nussbaum (2006).
'A simpler linear-time recognition of circular-arc graphs'.
In: *Algorithm Theory. Proc. 10th SWAT*. Berlin et al.: Springer, pp. 41–52.

Klein, Philip N. (1996).
'Efficient parallel algorithms for chordal graphs'.
In: *SIAM J. Comput.* 25.4, pp. 797–827.

Laubner, Bastian (2009).
Capturing polynomial time on interval graphs. Nov. 19, 2009.
arXiv: `0911.3799v1`. Accepted for LICS'10.

Lueker, George S., Kellogg S. Booth (1979).
'A linear time algorithm for deciding interval graph isomorphism'.
In: *J. ACM* 26.2 (Apr. 1979), pp. 183–195.

Reingold, Omer (2005). 'Undirected ST-connectivity in log-space'.
In: *Proc. 37th STOC*, pp. 376–385. Conference version of [Reingold 08].

— (2008). 'Undirected connectivity in log-space'.
In: *J. ACM* 55.4 (Sept. 2008), 17:1–17:24.

Uehara, Ryuhei (2008). 'Simple geometrical intersection graphs'.
In: *WALCOM: Algorithms and Computation*. Berlin et al.: Springer, pp. 25–33.