Rigid Interval Graphs: Vertex Ordering Characterizations and Multi-Sweep Graph Search Recognition Algorithm

Peng Li, Yaokun Wu
(李鹏，吴耀琨)
Shanghai Jiao Tong University

Satellite Seminar
Workshop on Algebraic Combinatorics at SJTU

Sept. 19, 2011

Clique tree

Let G be a graph and let $\mathscr{C}(G)$ be the set of maximal cliques of G. A clique tree T of G is a tree on the vertex set $\mathscr{C}(G)$ such that if C_2 lies in the unique path connecting C_1 and C_3 on T, then $C_2 \supseteq C_1 \cap C_3$. If T is both a clique tree of G and a path, we call it a clique path of G.

A graph has a clique tree if and only if it is chordal and a graph has a clique path if and only if it is an interval graph.

We define a graph to be a rigid interval graph provided it has a unique clique tree and that tree is a path.

Clique tree

Let G be a graph and let $\mathscr{C}(G)$ be the set of maximal cliques of G. A clique tree T of G is a tree on the vertex set $\mathscr{C}(G)$ such that if C_2 lies in the unique path connecting C_1 and C_3 on T, then $C_2 \supseteq C_1 \cap C_3$. If T is both a clique tree of G and a path, we call it a clique path of G.

A graph has a clique tree if and only if it is chordal and a graph has a clique path if and only if it is an interval graph.

We define a graph to be a rigid interval graph provided it has a unique clique tree and that tree is a path.

Clique tree

Let G be a graph and let $\mathscr{C}(G)$ be the set of maximal cliques of G. A clique tree T of G is a tree on the vertex set $\mathscr{C}(G)$ such that if C_2 lies in the unique path connecting C_1 and C_3 on T, then $C_2 \supseteq C_1 \cap C_3$. If T is both a clique tree of G and a path, we call it a clique path of G.

A graph has a clique tree if and only if it is chordal and a graph has a clique path if and only if it is an interval graph.

We define a graph to be a rigid interval graph provided it has a unique clique tree and that tree is a path.

Example 1 (Panda, Das, 2009)

Connected unit interval graphs are rigid interval graphs.

Example 2 (Aspvall, Heggernes, 1994)

A connected chordal graph is a rigid interval graph if at most two of its maximal cliques contain simplicial vertices.

Example 3

Let G be the graph with exactly three maximal cliques $A = \{1, 2, 3\}, B = \{2, 3, 4, 5\}$ and $C = \{3, 5, 6\}$. It is easy to see that G is an interval graph but not any unit interval graph. Note that the only clique tree of G is the path $[A, B, C]$.

Example 4

Let G be the graph with exactly four maximal cliques, $\{4, 1\}$, $\{5, 2\}$, $\{6, 3\}$ and $\{1, 2, 3\}$. Then G has only one clique tree but G is not any interval graph.

Clique ordering characterization

Theorem 5 (Li, W.)

Let G be a graph having a clique path $P = [C_1, C_2, \ldots, C_m]$. Then P is the unique clique tree of G if and only if

$$(C_i \cap C_{i+1}) \setminus C_{i+2} \neq \emptyset \text{ and } (C_{i+1} \cap C_{i+2}) \setminus C_i \neq \emptyset. \quad (1)$$

holds for any $i \in [m-2]$.

Note that Example 1 can be deduced from Theorem 5.

We come to the concept of rigid interval graphs in the course of developing a certifying algorithm for recognizing unit interval graphs.

Clique ordering characterization

Theorem 5 (Li, W.)

Let G be a graph having a clique path $P = [C_1, C_2, \ldots, C_m]$. Then P is the unique clique tree of G if and only if

$$(C_i \cap C_{i+1}) \setminus C_{i+2} \neq \emptyset \text{ and } (C_{i+1} \cap C_{i+2}) \setminus C_i \neq \emptyset. \quad (1)$$

holds for any $i \in [m-2]$.

Note that Example 1 can be deduced from Theorem 5.

We come to the concept of rigid interval graphs in the course of developing a certifying algorithm for recognizing unit interval graphs.

Clique ordering characterization

Theorem 5 (Li, W.)

Let G be a graph having a clique path $P = [C_1, C_2, \ldots, C_m]$. Then P is the unique clique tree of G if and only if

$$(C_i \cap C_{i+1}) \setminus C_{i+2} \neq \emptyset \text{ and } (C_{i+1} \cap C_{i+2}) \setminus C_i \neq \emptyset. \quad (1)$$

holds for any $i \in [m-2]$.

Note that Example 1 can be deduced from Theorem 5.

We come to the concept of rigid interval graphs in the course of developing a certifying algorithm for recognizing unit interval graphs.

Vetex ordering characterization

Many graph classes have vertex ordering characterizations and they often ensure corresponding simple graph recognition algorithms.

- Chordal graphs: perfect elimination orderings
- Interval graphs: I-orderings
- Unit interval graphs: UI-orderings
- Rigid interval graphs: ???

Vetex ordering characterization

Many graph classes have vertex ordering characterizations and they often ensure corresponding simple graph recognition algorithms.

- Chordal graphs: perfect elimination orderings
- Interval graphs: I-orderings
- Unit interval graphs: UI-orderings
- Rigid interval graphs: ???

Vetex ordering characterization

Many graph classes have vertex ordering characterizations and they often ensure corresponding simple graph recognition algorithms.

- Chordal graphs: perfect elimination orderings
- Interval graphs: I-orderings
- Unit interval graphs: UI-orderings
- Rigid interval graphs: ???

Vetex ordering characterization

Many graph classes have vertex ordering characterizations and they often ensure corresponding simple graph recognition algorithms.

- Chordal graphs: perfect elimination orderings
- Interval graphs: I-orderings
- Unit interval graphs: UI-orderings
- Rigid interval graphs: ???

Vetex ordering characterization

Many graph classes have vertex ordering characterizations and they often ensure corresponding simple graph recognition algorithms.

- Chordal graphs: perfect elimination orderings
- Interval graphs: I-orderings
- Unit interval graphs: UI-orderings
- Rigid interval graphs: ???

Vetex ordering characterization of interval graphs

An ordering $v_1, v_2, \ldots, v_n$ of $V(G)$ is an I-ordering if $v_i v_j \in E(G)$ implies $v_i v_k \in E(G)$ for any $k \in [i, j]$. An I-ordering corresponds to the from-left-to-right ordering of the left end-points of a set of intervals.

Theorem 6 (Raychaudhuri, 1987)

A graph is an interval graph if and only if it has an I-ordering.

Vetex ordering characterization of interval graphs

An ordering $v_1, v_2, \ldots, v_n$ of $V(G)$ is an I-ordering if $v_i v_j \in E(G)$ implies $v_i v_k \in E(G)$ for any $k \in [i, j]$. An I-ordering corresponds to the from-left-to-right ordering of the left end-points of a set of intervals.

Theorem 6 (Raychaudhuri, 1987)

A graph is an interval graph if and only if it has an I-ordering.

We will present recognition algorithm for rigid interval graphs based on Maximal Neighborhood Search and some vertex ordering characterization of rigid interval graphs.

Graph search algorithm

Let G be a graph on n vertices. A *selection rule* $\mathfrak{S}$ is a map from $2^{V(G)}$ to $2^{V(G)}$ such that $T \cap \mathfrak{S}(T) = \emptyset$ for any $T \in 2^{V(G)}$. A *graph search algorithm* with selection rule $\mathfrak{S}$ determines an ordering τ of $V(G)$ by putting $\tau(1) \in \mathfrak{S}(\emptyset)$ and then selecting $\tau(i+1)$ (arbitrarily) to be a member of $\mathfrak{S}(\{\tau(1), \ldots, \tau(i)\})$ for any $i \in [n-1]$ inductively. Note that depending on how to break tie by choosing an element from all allowed candidates, the output of a graph search algorithm may vary.

A graph search algorithm is an Maximal Neighborhood Search (MNS) type algorithm (Corneil, Krueger, 2008) if for any ordering $v_1, v_2, \ldots, v_n$ of $V(G)$ generated by the algorithm and for any $1 \leq i < j \leq n$, $N_G(v_i) \cap \{v_1, \ldots, v_{i-1}\}$ cannot be a proper subset of $N_G(v_j) \cap \{v_1, \ldots, v_{i-1}\}$.

It is known that applying any MNS type algorithm on a chordal graph will produce a PEO, thus certifying that it is chordal!

Graph search algorithm

Let G be a graph on n vertices. A *selection rule* $\mathfrak{S}$ is a map from $2^{V(G)}$ to $2^{V(G)}$ such that $T \cap \mathfrak{S}(T) = \emptyset$ for any $T \in 2^{V(G)}$. A *graph search algorithm* with selection rule $\mathfrak{S}$ determines an ordering τ of $V(G)$ by putting $\tau(1) \in \mathfrak{S}(\emptyset)$ and then selecting $\tau(i+1)$ (arbitrarily) to be a member of $\mathfrak{S}(\{\tau(1), \ldots, \tau(i)\})$ for any $i \in [n-1]$ inductively. Note that depending on how to break tie by choosing an element from all allowed candidates, the output of a graph search algorithm may vary.

A graph search algorithm is an Maximal Neighborhood Search (MNS) type algorithm (Corneil, Krueger, 2008) if for any ordering $v_1, v_2, \ldots, v_n$ of $V(G)$ generated by the algorithm and for any $1 \leq i < j \leq n$, $N_G(v_i) \cap \{v_1, \ldots, v_{i-1}\}$ cannot be a proper subset of $N_G(v_j) \cap \{v_1, \ldots, v_{i-1}\}$.

It is known that applying any MNS type algorithm on a chordal graph will produce a PEO, thus certifying that it is chordal!

Graph search algorithm

Let G be a graph on n vertices. A *selection rule* $\mathfrak{S}$ is a map from $2^{V(G)}$ to $2^{V(G)}$ such that $T \cap \mathfrak{S}(T) = \emptyset$ for any $T \in 2^{V(G)}$. A *graph search algorithm* with selection rule $\mathfrak{S}$ determines an ordering τ of $V(G)$ by putting $\tau(1) \in \mathfrak{S}(\emptyset)$ and then selecting $\tau(i+1)$ (arbitrarily) to be a member of $\mathfrak{S}(\{\tau(1), \ldots, \tau(i)\})$ for any $i \in [n-1]$ inductively. Note that depending on how to break tie by choosing an element from all allowed candidates, the output of a graph search algorithm may vary.

A graph search algorithm is an Maximal Neighborhood Search (MNS) type algorithm (Corneil, Krueger, 2008) if for any ordering $v_1, v_2, \ldots, v_n$ of $V(G)$ generated by the algorithm and for any $1 \leq i < j \leq n$, $N_G(v_i) \cap \{v_1, \ldots, v_{i-1}\}$ cannot be a proper subset of $N_G(v_j) \cap \{v_1, \ldots, v_{i-1}\}$.

It is known that applying any MNS type algorithm on a chordal graph will produce a PEO, thus certifying that it is chordal!

Graph search algorithm

Let G be a graph on n vertices. A *selection rule* $\mathfrak{S}$ is a map from $2^{V(G)}$ to $2^{V(G)}$ such that $T \cap \mathfrak{S}(T) = \emptyset$ for any $T \in 2^{V(G)}$. A *graph search algorithm* with selection rule $\mathfrak{S}$ determines an ordering τ of $V(G)$ by putting $\tau(1) \in \mathfrak{S}(\emptyset)$ and then selecting $\tau(i+1)$ (arbitrarily) to be a member of $\mathfrak{S}(\{\tau(1), \ldots, \tau(i)\})$ for any $i \in [n-1]$ inductively. Note that depending on how to break tie by choosing an element from all allowed candidates, the output of a graph search algorithm may vary.

A graph search algorithm is an Maximal Neighborhood Search (MNS) type algorithm (Corneil, Krueger, 2008) if for any ordering $v_1, v_2, \ldots, v_n$ of $V(G)$ generated by the algorithm and for any $1 \leq i < j \leq n$, $N_G(v_i) \cap \{v_1, \ldots, v_{i-1}\}$ cannot be a proper subset of $N_G(v_j) \cap \{v_1, \ldots, v_{i-1}\}$.

It is known that applying any MNS type algorithm on a chordal graph will produce a PEO, thus certifying that it is chordal!

Graph search algorithm

Let G be a graph on n vertices. A *selection rule* $\mathfrak{S}$ is a map from $2^{V(G)}$ to $2^{V(G)}$ such that $T \cap \mathfrak{S}(T) = \emptyset$ for any $T \in 2^{V(G)}$. A *graph search algorithm* with selection rule $\mathfrak{S}$ determines an ordering τ of $V(G)$ by putting $\tau(1) \in \mathfrak{S}(\emptyset)$ and then selecting $\tau(i+1)$ (arbitrarily) to be a member of $\mathfrak{S}(\{\tau(1), \ldots, \tau(i)\})$ for any $i \in [n-1]$ inductively. Note that depending on how to break tie by choosing an element from all allowed candidates, the output of a graph search algorithm may vary.

A graph search algorithm is an Maximal Neighborhood Search (MNS) type algorithm (Corneil, Krueger, 2008) if for any ordering $v_1, v_2, \ldots, v_n$ of $V(G)$ generated by the algorithm and for any $1 \leq i < j \leq n$, $N_G(v_i) \cap \{v_1, \ldots, v_{i-1}\}$ cannot be a proper subset of $N_G(v_j) \cap \{v_1, \ldots, v_{i-1}\}$.

It is known that applying any MNS type algorithm on a chordal graph will produce a PEO, thus certifying that it is chordal!

Graph search algorithm

Let G be a graph on n vertices. A *selection rule* $\mathfrak{S}$ is a map from $2^{V(G)}$ to $2^{V(G)}$ such that $T \cap \mathfrak{S}(T) = \emptyset$ for any $T \in 2^{V(G)}$. A *graph search algorithm* with selection rule $\mathfrak{S}$ determines an ordering τ of $V(G)$ by putting $\tau(1) \in \mathfrak{S}(\emptyset)$ and then selecting $\tau(i+1)$ (arbitrarily) to be a member of $\mathfrak{S}(\{\tau(1), \ldots, \tau(i)\})$ for any $i \in [n-1]$ inductively. Note that depending on how to break tie by choosing an element from all allowed candidates, the output of a graph search algorithm may vary.

A graph search algorithm is an Maximal Neighborhood Search (MNS) type algorithm (Corneil, Krueger, 2008) if for any ordering $v_1, v_2, \ldots, v_n$ of $V(G)$ generated by the algorithm and for any $1 \leq i < j \leq n$, $N_G(v_i) \cap \{v_1, \ldots, v_{i-1}\}$ cannot be a proper subset of $N_G(v_j) \cap \{v_1, \ldots, v_{i-1}\}$.

It is known that applying any MNS type algorithm on a chordal graph will produce a PEO, thus certifying that it is chordal!

LBFS

The Lexicographic Breadth First Search (LBFS) algorithm (Rose, Tarjan and Lueker, 1976) is an MNS type algorithm which favors vertices whose visited neighbors appear as early as possible. It has an easily implementable and linear time algorithm.

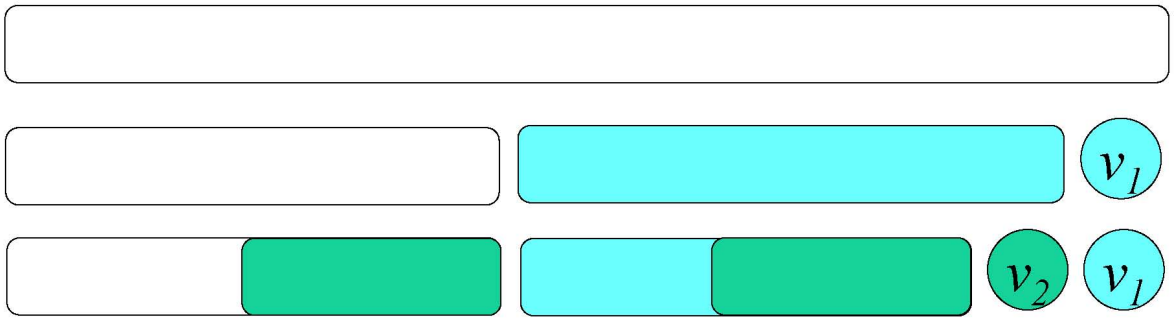

Figure: Partition refinement and LBFS ordering.

Note that general MNS type algorithms may not be any breadth-first search algorithm.

LBFS

The Lexicographic Breadth First Search (LBFS) algorithm (Rose, Tarjan and Lueker, 1976) is an MNS type algorithm which favors vertices whose visited neighbors appear as early as possible. It has an easily implementable and linear time algorithm.

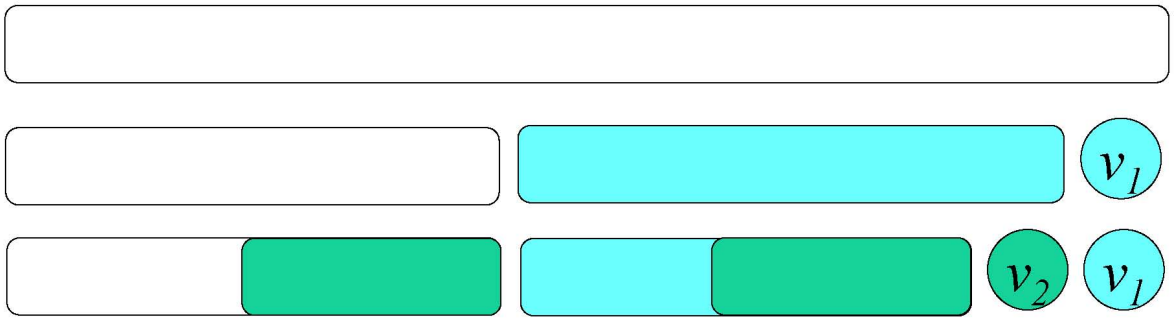

Figure: Partition refinement and LBFS ordering.

Note that general MNS type algorithms may not be any breadth-first search algorithm.

LBFS

The Lexicographic Breadth First Search (LBFS) algorithm (Rose, Tarjan and Lueker, 1976) is an MNS type algorithm which favors vertices whose visited neighbors appear as early as possible. It has an easily implementable and linear time algorithm.

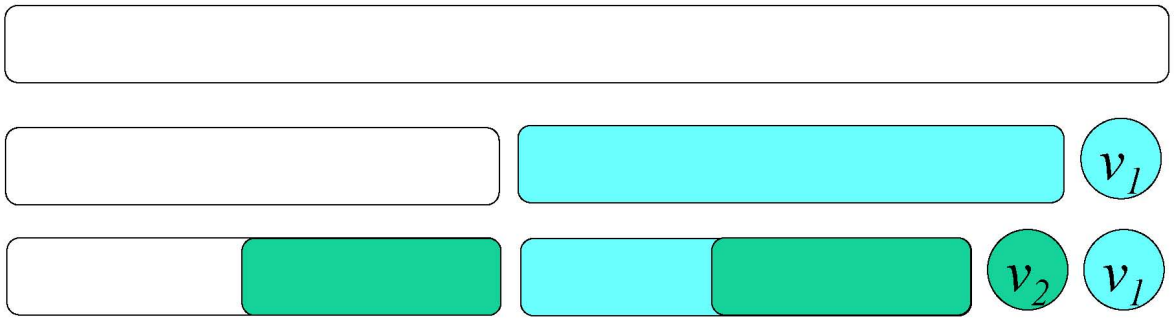

Figure: Partition refinement and LBFS ordering.

Note that general MNS type algorithms may not be any breadth-first search algorithm.

Rigid interval ordering

Any vertex ordering of a graph produced by an MNS type algorithm is called an MNS ordering. An ordering $v_1, \ldots, v_n$ is consecutive provided $v_i v_{i+1} \in E(G)$ for any $i \in [n-1]$ and is a perfect clique slice ordering (PCSO) if it is an MNS ordering, and for any $i \in [n-1]$,

$$\{v_j : \ j > i, N(v_j) \cap \{v_1, \ldots, v_i\} \text{ is maximal under set inclusion } \}$$

is a clique. We refer to an ordering as a rigid interval ordering provided it is both a consecutive I-ordering and a PCSO.

Theorem 7 (Li, W.)

A graph G is a rigid interval graph if and only if it has a rigid interval ordering.

Rigid interval ordering

Any vertex ordering of a graph produced by an MNS type algorithm is called an MNS ordering. An ordering $v_1, \ldots, v_n$ is consecutive provided $v_i v_{i+1} \in E(G)$ for any $i \in [n-1]$ and is a perfect clique slice ordering (PCSO) if it is an MNS ordering, and for any $i \in [n-1]$,

$$\{v_j : j > i, N(v_j) \cap \{v_1, \ldots, v_i\} \text{ is maximal under set inclusion } \}$$

is a clique. We refer to an ordering as a rigid interval ordering provided it is both a consecutive I-ordering and a PCSO.

Theorem 7 (Li, W.)

A graph G is a rigid interval graph if and only if it has a rigid interval ordering.

Rigid interval ordering

Any vertex ordering of a graph produced by an MNS type algorithm is called an MNS ordering. An ordering $v_1, \ldots, v_n$ is consecutive provided $v_i v_{i+1} \in E(G)$ for any $i \in [n-1]$ and is a perfect clique slice ordering (PCSO) if it is an MNS ordering, and for any $i \in [n-1]$,

$$\{v_j : j > i, N(v_j) \cap \{v_1, \ldots, v_i\} \text{ is maximal under set inclusion } \}$$

is a clique. We refer to an ordering as a rigid interval ordering provided it is both a consecutive I-ordering and a PCSO.

Theorem 7 (Li, W.)

A graph G is a rigid interval graph if and only if it has a rigid interval ordering.

Rigid interval ordering

Any vertex ordering of a graph produced by an MNS type algorithm is called an MNS ordering. An ordering $v_1, \ldots, v_n$ is consecutive provided $v_i v_{i+1} \in E(G)$ for any $i \in [n-1]$ and is a perfect clique slice ordering (PCSO) if it is an MNS ordering, and for any $i \in [n-1]$,

$$\{v_j : j > i, N(v_j) \cap \{v_1, \ldots, v_i\} \text{ is maximal under set inclusion } \}$$

is a clique. We refer to an ordering as a rigid interval ordering provided it is both a consecutive I-ordering and a PCSO.

Theorem 7 (Li, W.)

A graph G is a rigid interval graph if and only if it has a rigid interval ordering.

Multi-sweep graph search I

For any graph search algorithm $\mathcal{A}$ with a selection rule $\mathfrak{S}$, any graph G and $u \in V(G)$ satisfying $u \in \mathfrak{S}(\emptyset)$, we write $\mathcal{A}(G, u)$ for the search algorithm applied to G with a selection rule $\mathfrak{S}'$ such that $\mathfrak{S}'(\emptyset) = \{u\}$ and $\mathfrak{S}' = \mathfrak{S}$ elsewhere.

Any output of $\mathcal{A}(G, u)$ is an output of $\mathcal{A}$ applied on G.

Multi-sweep graph search II

For the purpose of devising certain multi-sweep LBFS algorithms, Ma and Simon independently propose a special kind of LBFS, called LBFS+. In the same fashion, for any graph search algorithm $\mathcal{A}$ with selection rule $\mathfrak{S}$, let us propose here the algorithm $\mathcal{A}+$, which is adapted from $\mathcal{A}$ by introducing some tie-breaking rule according to a given vertex ordering.

$\mathcal{A}+(G,\tau)$:
{ Input: a graph G and an ordering τ of $V(G)$;
Output : an ordering σ of $V(G)$.}
For any $i \in [|V(G)|]$, after $\sigma(1),\ldots,\sigma(i-1)$ are determined, choose $\sigma(i)$ to be the vertex in $\mathfrak{S}(\{\sigma(1),\ldots,\sigma(i-1)\})$ that appears last in τ.

Multi-sweep graph search II

For the purpose of devising certain multi-sweep LBFS algorithms, Ma and Simon independently propose a special kind of LBFS, called LBFS+. In the same fashion, for any graph search algorithm $\mathcal{A}$ with selection rule $\mathfrak{S}$, let us propose here the algorithm $\mathcal{A}$+, which is adapted from $\mathcal{A}$ by introducing some tie-breaking rule according to a given vertex ordering.

$\mathcal{A}+(G,\tau)$:
{ Input: a graph G and an ordering τ of $V(G)$;
Output : an ordering σ of $V(G)$.}
For any $i \in [|V(G)|]$, after $\sigma(1),\ldots,\sigma(i-1)$ are determined, choose $\sigma(i)$ to be the vertex in $\mathfrak{S}(\{\sigma(1),\ldots,\sigma(i-1)\})$ that appears last in τ.

Three-sweep recognition algorithm

Three-sweep MNS algorithm for recognizing rigid interval graphs based on an MNS type algorithm $\mathcal{A}$:
{ Input: a graph G on n vertices;
Output: a statement declaring whether or not G is a rigid interval graph.}
Step 1. Generate an MNS ordering δ of $V(G)$;
Step 2. Do $\mathcal{A}(G, \delta(n))$ yielding sweep σ;
Step 3. Do $\mathcal{A}+(G, \sigma)$ yielding sweep τ;
Step 4. If τ is a rigid interval ordering of G, then conclude that G is a rigid interval graph; else, conclude that G is not a rigid interval graph.

How to check PCSO?

Three-sweep recognition algorithm

Three-sweep MNS algorithm for recognizing rigid interval graphs based on an MNS type algorithm $\mathcal{A}$:
{ Input: a graph G on n vertices;
Output: a statement declaring whether or not G is a rigid interval graph.}
Step 1. Generate an MNS ordering δ of $V(G)$;
Step 2. Do $\mathcal{A}(G, \delta(n))$ yielding sweep σ;
Step 3. Do $\mathcal{A}+(G, \sigma)$ yielding sweep τ;
Step 4. If τ is a rigid interval ordering of G, then conclude that G is a rigid interval graph; else, conclude that G is not a rigid interval graph.

How to check PCSO?

Another vertex ordering characterization

Theorem 8 (Li, W.)

A graph G is a rigid interval graph if and only if it has two consecutive orderings σ and τ such that σ is an I-ordering, τ is an MNS ordering and $\tau(1) = \sigma(n)$.

Four-sweep recognition algorithm

Four-sweep MNS algorithm for recognizing rigid interval graphs based on an MNS type algorithm $\mathcal{A}$:
{ Input: a graph G on n vertices;
Output: a statement declaring whether or not G is a rigid interval graph.}
Step 1. Generate an arbitrary MNS ordering δ of $V(G)$;
Step 2. Do $\mathcal{A}(G, \delta(n))$ yielding sweep π;
Step 3. Do $\mathcal{A}+(G, \pi)$ yielding sweep σ;
Step 4. Do $\mathcal{A}+(G, \sigma)$ yielding sweep τ;
Step 5. If σ is a consecutive I-ordering of G and τ is consecutive, then conclude that G is a rigid interval graph; else, conclude that G is not any rigid interval graph.

This algorithm can be easily implemented in linear time by appealing to LBFS.

Four-sweep recognition algorithm

Four-sweep MNS algorithm for recognizing rigid interval graphs based on an MNS type algorithm $\mathcal{A}$:
{ Input: a graph G on n vertices;
Output: a statement declaring whether or not G is a rigid interval graph.}
Step 1. Generate an arbitrary MNS ordering δ of $V(G)$;
Step 2. Do $\mathcal{A}(G, \delta(n))$ yielding sweep π;
Step 3. Do $\mathcal{A}+(G, \pi)$ yielding sweep σ;
Step 4. Do $\mathcal{A}+(G, \sigma)$ yielding sweep τ;
Step 5. If σ is a consecutive I-ordering of G and τ is consecutive, then conclude that G is a rigid interval graph; else, conclude that G is not any rigid interval graph.

This algorithm can be easily implemented in linear time by appealing to LBFS.

Questions

- Can all rigid interval graphs be generated in some easy way from unit interval graphs?
- Is there a good intersection model for rigid interval graph?
- Is there any other good vertex ordering characterization of rigid interval graphs?
- Can we find a simple certifying algorithm for recognizing rigid interval graphs?
- Are there good vertex ordering characterizations of all chordal graphs with a unique clique tree? What about their recognition algorithms?

Questions

- Can all rigid interval graphs be generated in some easy way from unit interval graphs?
- Is there a good intersection model for rigid interval graph?
- Is there any other good vertex ordering characterization of rigid interval graphs?
- Can we find a simple certifying algorithm for recognizing rigid interval graphs?
- Are there good vertex ordering characterizations of all chordal graphs with a unique clique tree? What about their recognition algorithms?

Questions

- Can all rigid interval graphs be generated in some easy way from unit interval graphs?
- Is there a good intersection model for rigid interval graph?
- Is there any other good vertex ordering characterization of rigid interval graphs?
- Can we find a simple certifying algorithm for recognizing rigid interval graphs?
- Are there good vertex ordering characterizations of all chordal graphs with a unique clique tree? What about their recognition algorithms?

Questions

- Can all rigid interval graphs be generated in some easy way from unit interval graphs?
- Is there a good intersection model for rigid interval graph?
- Is there any other good vertex ordering characterization of rigid interval graphs?
- Can we find a simple certifying algorithm for recognizing rigid interval graphs?
- Are there good vertex ordering characterizations of all chordal graphs with a unique clique tree? What about their recognition algorithms?

Questions

- Can all rigid interval graphs be generated in some easy way from unit interval graphs?
- Is there a good intersection model for rigid interval graph?
- Is there any other good vertex ordering characterization of rigid interval graphs?
- Can we find a simple certifying algorithm for recognizing rigid interval graphs?
- Are there good vertex ordering characterizations of all chordal graphs with a unique clique tree? What about their recognition algorithms?

Remarks

- We have designed a simple three-sweep MNS certifying algorithm for recognizing unit interval graphs. (Earlier work by Hell and Huang (2005) only applies to LBFS.)
- We have designed a 4-sweep LBFS recognition algorithm for interval graphs. (Earlier work by Corneil, Olariu, Stewart (2009) uses more complicated 6-sweep LBFS search.)
- We even obtain recently a 4-sweep LBFS certifying algorithm for recognizing interval graphs. (Earlier work by Kratsch, McConnell, Mehlhorn, Spinrad (2006) involves MPQ tree.)
- The algorithm analysis is complicated.

Remarks

- We have designed a simple three-sweep MNS certifying algorithm for recognizing unit interval graphs. (Earlier work by Hell and Huang (2005) only applies to LBFS.)
- We have designed a 4-sweep LBFS recognition algorithm for interval graphs. (Earlier work by Corneil, Olariu, Stewart (2009) uses more complicated 6-sweep LBFS search.)
- We even obtain recently a 4-sweep LBFS certifying algorithm for recognizing interval graphs. (Earlier work by Kratsch, McConnell, Mehlhorn, Spinrad (2006) involves MPQ tree.)
- The algorithm analysis is complicated.

Remarks

- We have designed a simple three-sweep MNS certifying algorithm for recognizing unit interval graphs. (Earlier work by Hell and Huang (2005) only applies to LBFS.)
- We have designed a 4-sweep LBFS recognition algorithm for interval graphs. (Earlier work by Corneil, Olariu, Stewart (2009) uses more complicated 6-sweep LBFS search.)
- We even obtain recently a 4-sweep LBFS certifying algorithm for recognizing interval graphs. (Earlier work by Kratsch, McConnell, Mehlhorn, Spinrad (2006) involves MPQ tree.)
- The algorithm analysis is complicated.

Remarks

- We have designed a simple three-sweep MNS certifying algorithm for recognizing unit interval graphs. (Earlier work by Hell and Huang (2005) only applies to LBFS.)
- We have designed a 4-sweep LBFS recognition algorithm for interval graphs. (Earlier work by Corneil, Olariu, Stewart (2009) uses more complicated 6-sweep LBFS search.)
- We even obtain recently a 4-sweep LBFS certifying algorithm for recognizing interval graphs. (Earlier work by Kratsch, McConnell, Mehlhorn, Spinrad (2006) involves MPQ tree.)
- The algorithm analysis is complicated.

A by-product of our algorithm analysis

A vertex v of G is *admissible* if there are no vertices x and y such that there is an x, v-path avoiding the closed neighborhood of y and there is an y, v-path avoiding the closed neighborhood of x. The set of vertices of G which are noth simplicial and admissible is denoted by $AS(G)$.

Corollary 9 (Li, W.)

Let G be a connected AT-free graph such that none of its induced subgraph is a 4-cycle or a claw. If $AS(G) \neq \emptyset$, then G is a unit interval graph. In particular, G cannot contain any induced 5-cycle.

A by-product of our algorithm analysis

A vertex v of G is *admissible* if there are no vertices x and y such that there is an x, v-path avoiding the closed neighborhood of y and there is an y, v-path avoiding the closed neighborhood of x. The set of vertices of G which are noth simplicial and admissible is denoted by $AS(G)$.

Corollary 9 (Li, W.)

Let G be a connected AT-free graph such that none of its induced subgraph is a 4-cycle or a claw. If $AS(G) \neq \emptyset$, then G is a unit interval graph. In particular, G cannot contain any induced 5-cycle.

Referring to the 6-sweep LBFS interval graph recognition algorithm of Corneil et al.

The word 'simple' has many different meanings when applied to an algorithm, some of which are illustrated nicely in this case. I believe that the algorithm below would be quite simple to implement, and I include it for that reason. However, the correctness proof is extraordinarily long (much longer than the proof of the algorithm based on PQ-trees), ... Examples like this show us that the God of graph theory may be different from the God of graph algorithms; which would you include in a justification of the statement that interval graphs can be recognized in linear time? – J.P. Spinrad, Efficient Graph Representations*, Fields Institute Monographs, 19, American Mathematical Society, 2003.*

Thank You!