

Permutation graphs — an introduction

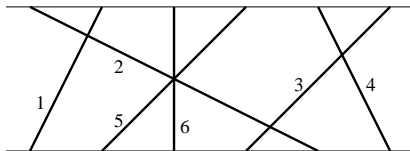
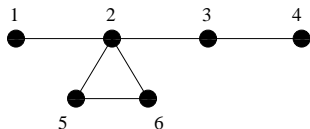
Ioan Todinca

LIFO - Université d'Orléans

Algorithms and permutations, february 2012



Permutation graphs

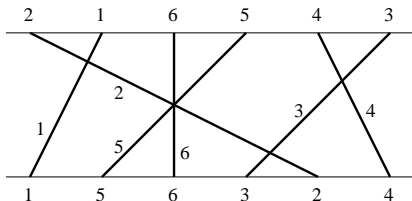
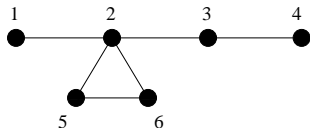


- Optimisation algorithms
use, as input, the intersection model (realizer)
- Recognition algorithms
output the intersection(s) model(s)

Plan of the talk

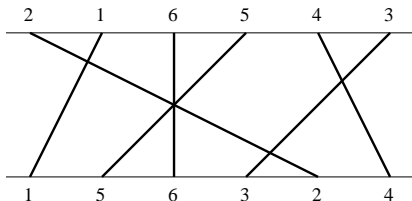
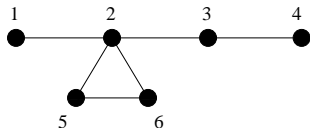
1. Relationship with other graph classes
2. Optimisation problems :
MAXINDEPENDENTSET/MAXCLIQUE/COLORING ;
TREEWIDTH
3. Recognition algorithm
4. Encoding all realizers via modular decomposition
5. Conclusion

Definition and basic properties



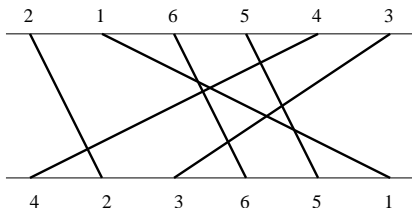
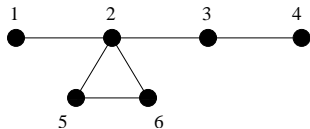
- **Realizer** : (π_1, π_2)
- One can "reverse" the realizer upside-down or right-left :
 $(\pi_1, \pi_2) \sim (\pi_2, \pi_1) \sim (\overline{\pi_1}, \overline{\pi_2}) \sim (\overline{\pi_2}, \overline{\pi_1})$
- **Complements of permutation graphs are permutation graphs.**
 \rightarrow Reverse the ordering of the bottoms of the segments :
 $(\pi_1, \pi_2) \rightarrow (\pi_1, \overline{\pi_2})$

Definition and basic properties



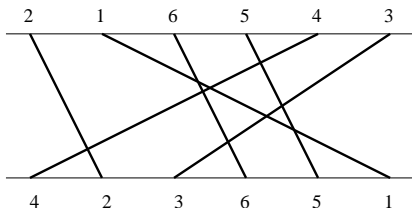
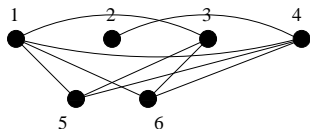
- **Realizer** : (π_1, π_2)
- One can "reverse" the realizer upside-down or right-left :
 $(\pi_1, \pi_2) \sim (\pi_2, \pi_1) \sim (\overline{\pi_1}, \overline{\pi_2}) \sim (\overline{\pi_2}, \overline{\pi_1})$
- **Complements of permutation graphs are permutation graphs.**
 \rightarrow Reverse the ordering of the bottoms of the segments :
 $(\pi_1, \pi_2) \rightarrow (\pi_1, \overline{\pi_2})$

Definition and basic properties



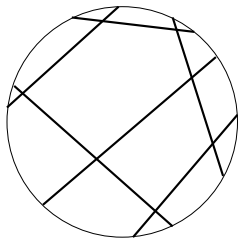
- **Realizer** : (π_1, π_2)
- One can "reverse" the realizer upside-down or right-left :
 $(\pi_1, \pi_2) \sim (\pi_2, \pi_1) \sim (\overline{\pi_1}, \overline{\pi_2}) \sim (\overline{\pi_2}, \overline{\pi_1})$
- **Complements of permutation graphs are permutation graphs.**
 → Reverse the ordering of the bottoms of the segments :
 $(\pi_1, \pi_2) \rightarrow (\pi_1, \overline{\pi_2})$

Definition and basic properties

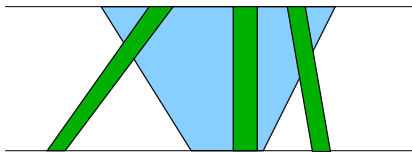


- **Realizer** : (π_1, π_2)
- One can "reverse" the realizer upside-down or right-left :
 $(\pi_1, \pi_2) \sim (\pi_2, \pi_1) \sim (\overline{\pi_1}, \overline{\pi_2}) \sim (\overline{\pi_2}, \overline{\pi_1})$
- **Complements of permutation graphs are permutation graphs.**
 → Reverse the ordering of the bottoms of the segments :
 $(\pi_1, \pi_2) \rightarrow (\pi_1, \overline{\pi_2})$

More intersection graph classes



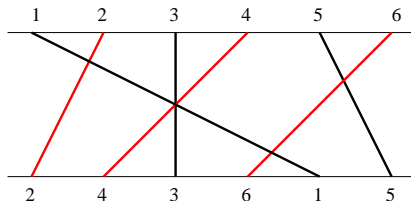
Circle graphs



Trapezoid graphs

Books on graph classes : [Golumbic '80 ; Brandstädt, Le, Spinrad '99 ; Spinrad 2003]

MAXINDEPENDENTSET via Dynamic Programming

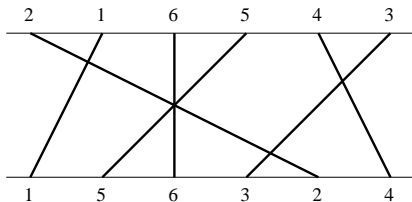
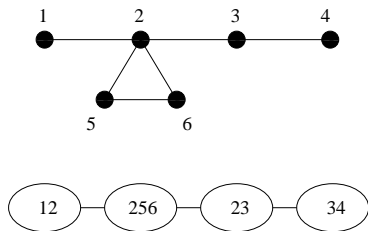


- Dynamic programming from left to right :

$$MIS[i] = 1 + \max_{j \text{ left to } i} MIS[j]$$

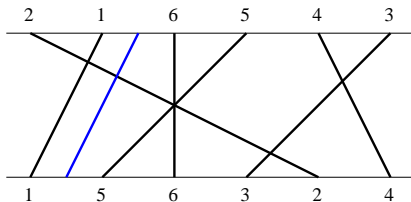
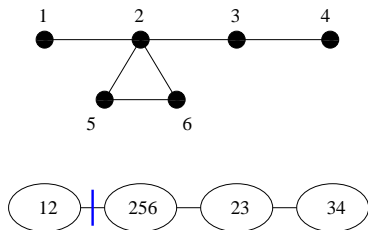
- MAXINDEPENDENTSET corresponds to the longest increasing sequence in a permutation — $O(n \log n)$
- MAXCLIQUE : longest decreasing sequence
- COLORING : chromatic number = max clique (perfect graphs)

TREewidth via dynamic programming on scanlines



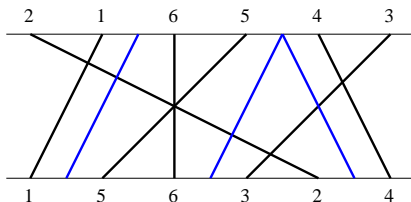
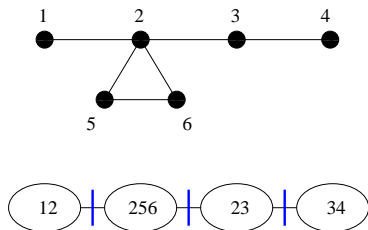
- Minimal separators correspond to [scanlines](#)
- Bags correspond to areas between two scanlines
- TREewidth can be solved in polynomial time [Bodlaender, Kloks, Kratsch 95 ; Meister 2010]

TREEWIDTH via dynamic programming on scanlines



- Minimal separators correspond to **scanlines**
- Bags correspond to areas between two scanlines
- TREEWIDTH can be solved in polynomial time [Bodlaender, Kloks, Kratsch 95 ; Meister 2010]

TREewidth via dynamic programming on scanlines



- Minimal separators correspond to **scanlines**
- Bags correspond to areas between two scanlines
- TREewidth can be solved in polynomial time [Bodlaender, Kloks, Kratsch 95 ; Meister 2010]

Recognition algorithm

Theorem ([Pnueli, Lempel, Even '71], see also [Golumbic '80])

*G is a permutation graph **if and only if** G and \overline{G} are comparability graphs.*

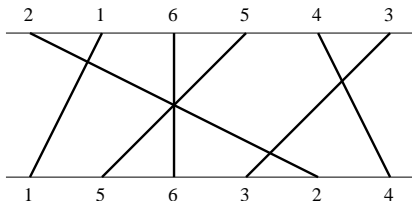
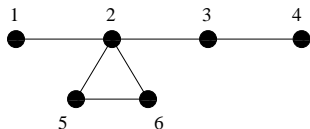
Algorithm

1. Find a transitive orientation of G and one of \overline{G}
2. Construct an intersection model for G

In $O(n + m)$ time by [McConnell, Spinrad '99]

permutation \subseteq comparability \cap co-comparability

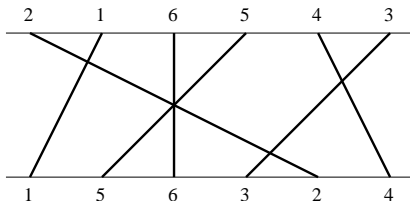
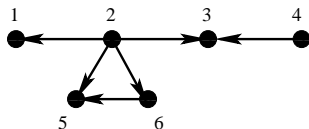
Transitive orientation of a permutation graph G : orient edges according to the top endpoints of the segments.



If $xy, yz \in E$ and $\pi_1(x) < \pi_1(y) < \pi_1(z)$ then $xz \in E$.

permutation \subseteq comparability \cap co-comparability

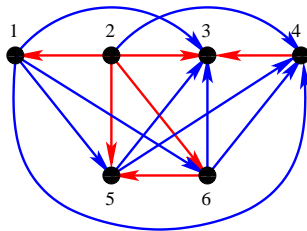
Transitive orientation of a permutation graph G : orient edges according to the top endpoints of the segments.



If $xy, yz \in E$ and $\pi_1(x) < \pi_1(y) < \pi_1(z)$ then $xz \in E$.

comparability \cap co-comparability \subseteq permutation

Let E_{tr} be a transitive orientation of G and F_{tr} a transitive orientation of its complement.

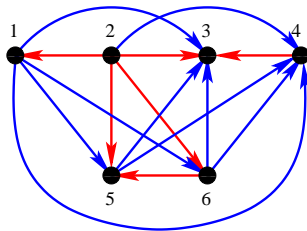


Lemma

$E_{tr} \cup F_{tr}$ induces a total ordering $\pi_1(E_{tr} \cup F_{tr})$ on the vertex set.

comparability \cap co-comparability \subseteq permutation

Let E_{tr} be a transitive orientation of G and F_{tr} a transitive orientation of its complement.

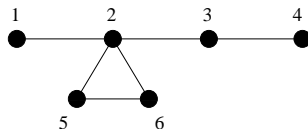
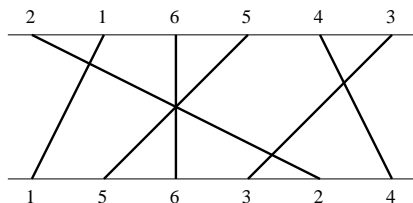


Lemma

$E_{tr} \cup F_{tr}$ induces a total ordering $\pi_1(E_{tr} \cup F_{tr})$ on the vertex set.
 $rev(E_{tr}) \cup F_{tr}$ induces another total ordering $\pi_2(rev(E_{tr}) \cup F_{tr})$.

A realizer of G

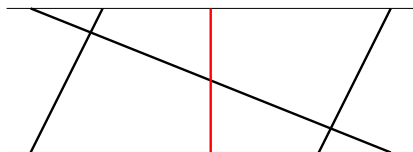
Permutations $\pi_1(E_{tr} \cup F_{tr})$ and $\pi_2(\text{rev}(E_{tr}) \cup F_{tr})$ form a realizer of G .



Segments x and y intersect iff $(xy) \in E_{tr}$ and $(yx) \in \text{rev}(E_{tr})$ or vice-versa ; equivalently, iff $xy \in E$.

Modules and common intervals

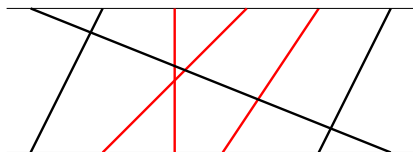
- Substituting a segment (vertex) by the realizer of a permutation graph (module) produces a new permutation graph.



- A common interval of π_1 and π_2 forms a module in G
- Strong modules** correspond exactly to **strong common intervals** [de Mongolfier 2003]
- A graph is a permutation graphs **iff** all prime nodes in the modular decomposition are permutation graphs.

Modules and common intervals

- Substituting a segment (vertex) by the realizer of a permutation graph (module) produces a new permutation graph.



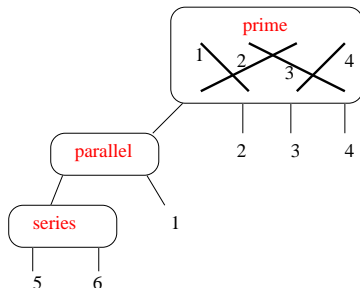
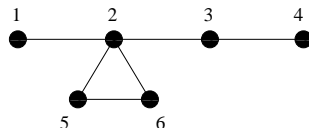
- A common interval of π_1 and π_2 forms a module in G
 - Strong modules** correspond exactly to **strong common intervals** [de Mongolfier 2003]
- A graph is a permutation graphs **iff** all prime nodes in the modular decomposition are permutation graphs.

Encoding realizers

Theorem ([Gallai '67])

A *prime* permutation graph has a *unique* realizer, up to reversals.

The modular decomposition tree + realizers of prime nodes encode all possible realizers of G , cf. [Crespelle, Paul 2010].



Conclusion

Summary

- Many optimization problems become polynomial on permutation graphs
- Representations (intersection models) based on modular decompositions

Some questions

- Is BANDWIDTH polynomial or *NP*-complete on permutation graphs?
- What about subgraph isomorphism from parametrized point of view?

Conclusion

Summary

- Many optimization problems become polynomial on permutation graphs
- Representations (intersection models) based on modular decompositions

Some questions

- Is BANDWIDTH polynomial or *NP*-complete on permutation graphs?
- What about subgraph isomorphism from parametrized point of view?

Thank you ! Your questions ?