Computing kernels of graphs

Adèle Pass-Lanneau
Internship supervised by Frédéric Meunier

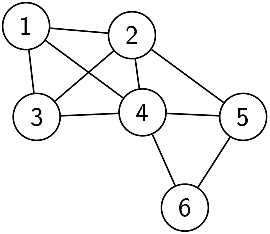
Ecole polytechnique - CERMICS

July 11, 2016

Plan

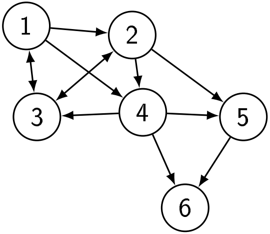
Plan

Graph theory basics



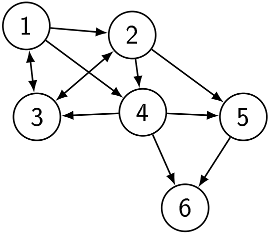
- Vertices, edges

Graph theory basics



- Vertices, edges
- Orientation, arcs

Graph theory basics



- Vertices, edges
- Orientation, arcs
- Neighbors/Adjacent vertices
- Successor
- Sink: vertex without successor
- Circuit

Kernel

Definition

Let $D = (V, A)$ be a directed graph. A subset of vertices K is a *kernel* of D when:

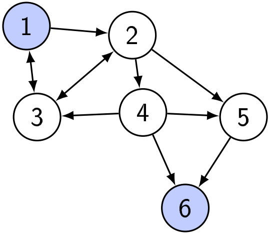
- K *is stable:* it contains no pair of adjacent vertices
- K *is absorbing:* every vertex $v \notin K$ has a successor in K, i.e.,$\forall v \notin K$, there is a vertex $k \in K$ such that $(v, k) \in A$.

Kernel

Definition

Let $D = (V, A)$ be a directed graph. A subset of vertices K is a *kernel* of D when:

- *K is stable:* it contains no pair of adjacent vertices
- *K is absorbing:* every vertex $v \notin K$ has a successor in K, i.e.,$\forall v \notin K$, there is a vertex $k \in K$ such that $(v, k) \in A$.



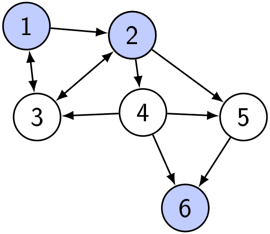
- $\{1, 6\}$ is not an absorbing subset of D: vertex 2 has no successor in $\{1, 6\}$

Kernel

Definition

Let $D = (V, A)$ be a directed graph. A subset of vertices K is a *kernel* of D when:

- K *is stable:* it contains no pair of adjacent vertices
- K *is absorbing:* every vertex $v \notin K$ has a successor in K, i.e.,$\forall v \notin K$, there is a vertex $k \in K$ such that $(v, k) \in A$.



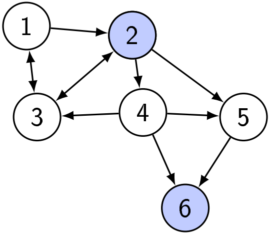
- $\{1, 6\}$ is not an absorbing subset of D: vertex 2 has no successor in $\{1, 6\}$
- $\{1, 2, 6\}$ is absorbing but not stable because of the arc $(1, 2)$

Kernel

Definition

Let $D = (V, A)$ be a directed graph. A subset of vertices K is a *kernel* of D when:

- K *is stable:* it contains no pair of adjacent vertices
- K *is absorbing:* every vertex $v \notin K$ has a successor in K, i.e.,$\forall v \notin K$, there is a vertex $k \in K$ such that $(v, k) \in A$.



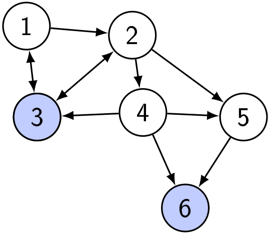
- $\{1, 6\}$ is not an absorbing subset of D: vertex 2 has no successor in $\{1, 6\}$
- $\{1, 2, 6\}$ is absorbing but not stable because of the arc $(1, 2)$
- $\{2, 6\}$ is absorbing and stable: it is a kernel of D

Kernel

Definition

Let $D = (V, A)$ be a directed graph. A subset of vertices K is a *kernel* of D when:

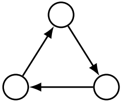
- *K is stable:* it contains no pair of adjacent vertices
- *K is absorbing:* every vertex $v \notin K$ has a successor in K, i.e.,$\forall v \notin K$, there is a vertex $k \in K$ such that $(v, k) \in A$.



- $\{1, 6\}$ is not an absorbing subset of D: vertex 2 has no successor in $\{1, 6\}$
- $\{1, 2, 6\}$ is absorbing but not stable because of the arc $(1, 2)$
- $\{2, 6\}$ is absorbing and stable: it is a kernel of D
- $\{3, 6\}$ is another kernel of D

Existence of a kernel

- Graphs can have several kernels (even exponentially many)
- Others have no kernel at all, for example odd circuits

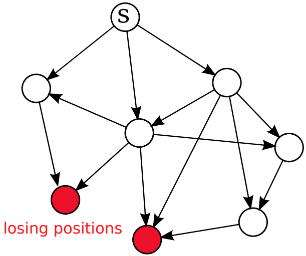


- Existence theorems are proven for some families of graphs

  Graphs without circuits, perfect graphs (Boros-Gurvich), graphs without odd circuit (Richardson), line-graphs (Maffray)...

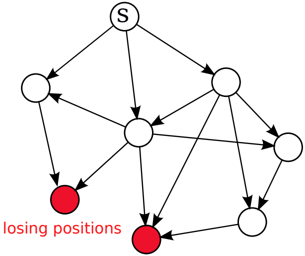
Why kernels ?

Related to winning strategies in game theory
Initially introduced by Von Neumann and Morgenstern in their *Theory of Games and Economic Behavior* in 1944



Consider a graph without circuit, with a token placed on a starting vertex s.
Two-player game: at his turn, each player moves the token from the current vertex to a successor of his.
The player who can't move anymore is the loser.

Why kernels ?



A game corresponds to a path $v_0 = s \to v_1 \to \dots \to v_n$ with v_n a sink. Player 1 chooses v_{2p+1} $\forall p$.

Let K be a kernel of the digraph. Then:

- All sinks are in K.
- $\forall v_{2p} \notin K$, Player 1 has a strategy to impose $v_{2p+1} \in K$.
- If $s \notin K$, Player 1 knows how to win !

Example of a Nim game

A Nim game:

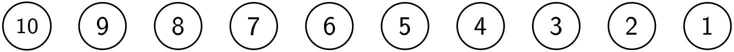
- stack of 10 sticks
- two players playing in turns
- each of them removes 1, 2 or 3 sticks
- the loser is the player who removes the last stick

Example of a Nim game

A Nim game:

- stack of 10 sticks
- two players playing in turns
- each of them removes 1, 2 or 3 sticks
- the loser is the player who removes the last stick

Equivalent to moving a token in the appropriate state graph.

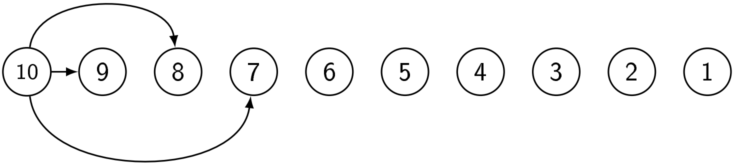


Example of a Nim game

A Nim game:

- stack of 10 sticks
- two players playing in turns
- each of them removes 1, 2 or 3 sticks
- the loser is the player who removes the last stick

Equivalent to moving a token in the appropriate state graph.

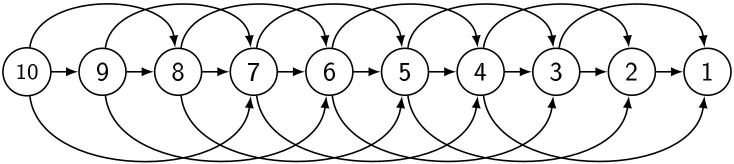


Example of a Nim game

A Nim game:

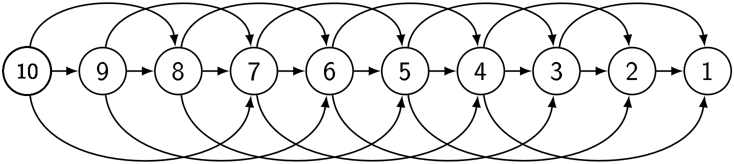
- stack of 10 sticks
- two players playing in turns
- each of them removes 1, 2 or 3 sticks
- the loser is the player who removes the last stick

Equivalent to moving a token in the appropriate state graph.



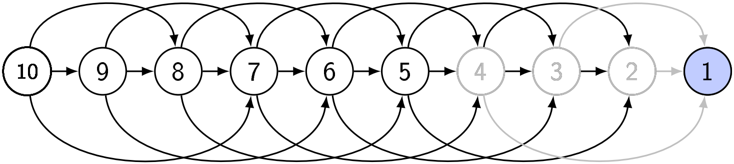
Kernel and Nim game

Now we compute a kernel of this graph...



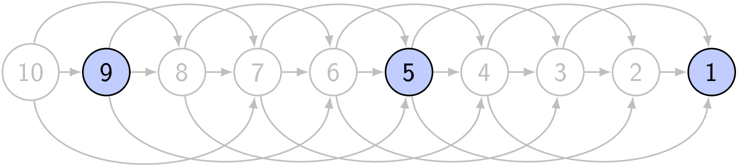
Kernel and Nim game

Now we compute a kernel of this graph...



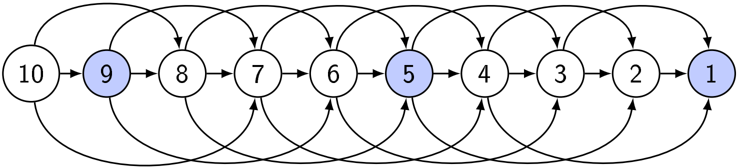
Kernel and Nim game

Now we compute a kernel of this graph...



Kernel and Nim game

Now we compute a kernel of this graph...



Winning strategy for player 1:

Always put player 2 in a state $s \in K$

$\iff$ Always leave a number of sticks $\equiv 1$ [4]

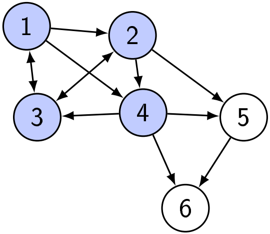
Motivation

- Graphs do not always have a kernel.
  Deciding if a graph has a kernel is NP-complete in general (Chvatal 1973)
- In some graph families, a kernel do always exist.
  Thus the decision problem is trivial, but **computing one kernel may still be hard** (similar to Nash equilibrium)
- Our goal: find if computing a kernel is polynomial for families where kernel existence is guaranteed but computation complexity is unknown.
  $\longrightarrow$ *chordal clique-acyclic graphs*

Cliques

A clique of a graph is a subset of vertices that are adjacent two by two.

A kernel of a clique is a single vertex (an absorbing vertex).



A digraph has a *clique-acyclic orientation* if every clique has an absorbing vertex.

Chordal graphs

A graph is *chordal* is it has no cycle of length ≥ 4 without a chord.

Theorem

Chordal clique-acyclic digraphs have a kernel.

Special case of Boros-Gurvich theorem (1996) on perfect graphs.

Not an algorithmic proof: no polynomial algorithm to compute a kernel.

Problem statement

Problem

Find a polynomial algorithm to compute a kernel of a chordal clique-acyclic graph.

Plan

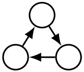
An easy case: graph without $\leftrightarrow$

Input: Chordal clique-acyclic graph without *reversible* arcs.

- Claim 1: it has no circuit
- Claim 2: computing a kernel of a graph without circuit is easy

No circuit?

1. The graph has no reversible arc
   $\Longrightarrow$ the graph has no 2-circuit
2. The graph is clique-acyclic: every clique has an absorbing vertex
   $\Longrightarrow$ the graph has no 3-circuit



3. The graph is chordal
   $\Longrightarrow$ if it has a circuit of length ≥ 3, it has a 3-circuit
   $\Longrightarrow$ the graph has no circuit of length ≥ 3

Chordal clique-acyclic graph without *reversible* arcs has no circuit.

Computing a kernel of a graph without circuit

We did it already for the Nim graph!

Reminder: all sinks must be in the kernel

Algorithm:

- Put all sinks in K
- Remove all their neighbors
- Reiterate

An easy case: graph without $\leftrightarrow$

Input: Chordal clique-acyclic graph without *reversible* arcs.

- Claim 1: it has no circuit $\checkmark$
- Claim 2: computing a kernel of a graph without circuit is easy OK $\checkmark$

Polynomial algorithm in this case

An easy case: graph without $\leftrightarrow$

Input: Chordal clique-acyclic graph without *reversible* arcs.

- Claim 1: it has no circuit ✓
- Claim 2: computing a kernel of a graph without circuit is easy OK ✓

Polynomial algorithm in this case

With $\leftrightarrow$:

- potentially no sink...
- replacing $\leftrightarrow$ by $\rightarrow$ creates circuits...

Our theorem in general case

Theorem

A kernel of a chordal clique-acyclic digraph can be computed in polynomial time.

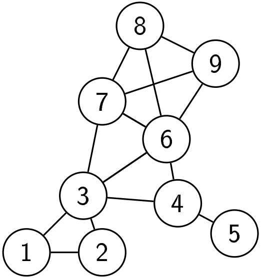
Introducing the clique tree

Structural property of chordal graphs:
Efficiently represented by a **tree** whose nodes are **cliques** of the graph.

A special case of **tree decomposition** (Halin 1976)

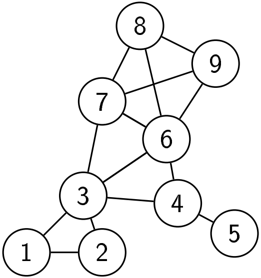
Building a clique tree

The chordal graph

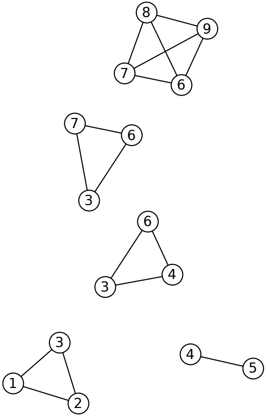


Building a clique tree

The chordal graph

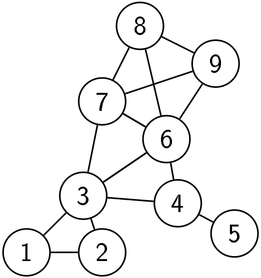


Its maximal cliques

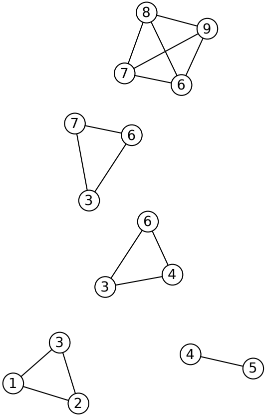


Building a clique tree

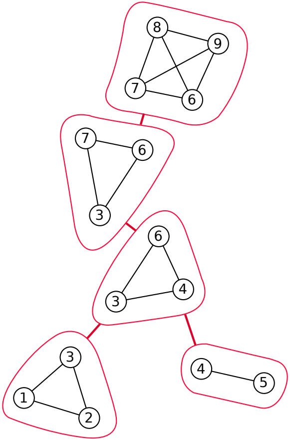
The chordal graph



Its maximal cliques



One clique tree



Clique tree: formal definition

A *clique tree* of a connected chordal digraph D is a tree $\mathcal{T} = (\mathcal{C}, \mathcal{E})$ such that:
- the node set $\mathcal{C}$ is the collection of all maximal cliques of D
- for every vertex $v \in V$, the subgraph of $\mathcal{T}$ induced by all cliques of $\mathcal{C}$ containing v is a subtree of $\mathcal{T}$.

Existence and computability (Habib 1995): Every chordal graph has a clique tree which is computable in polynomial time.

Result

- Using clique tree structure, we propose an algorithm to compute a kernel of chordal clique-acyclic graph
- It is polynomial of complexity $\leq O(n^4)$
- Now the question is to generalize it to other families of graphs

Plan

Natural generalization

The theorem still holds for a graph verifying $(H1), (H2), (H3)$.

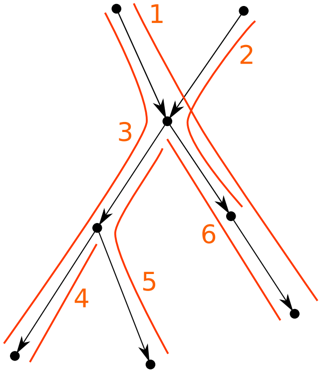
Generalized assumptions

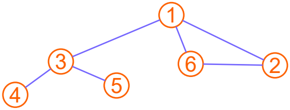
- **(H1)** $\exists$ tree decomposition such that the intersection of two bags is a clique
- **(H2)** $\forall$ subgraph B of a bag,
  B has a kernel computable in polynomial time
- **(H3)** $\forall$ subgraph B of a bag, $\forall v$ fixed vertex in B,
  in polynomial time: compute a kernel of B containing v if exists, or report that none exists otherwise

Extension of chordal case

DE graphs

- Consider a collection of paths on a directed tree
- Path are vertices of the DE graph
- Two vertices are adjacent if the paths share an edge
- The collection of paths is the *representation* of the DE graph





Stable marriages

- Two sets: men and women
- Every person ranks individuals from the other gender
- A matching is a set of married couples
- A matching is stable if *there is no pair* (m, w) *not married together such that both of them prefer the other to their current husband/wife*

Algorithm of Gale and Shapley: a stable marriage is computed in poly time for any instance

Stable marriages

- Two sets: men and women
- Every person ranks individuals from the other gender
- A matching is a set of married couples
- A matching is stable if *there is no pair* (m, w) *not married together such that both of them prefer the other to their current husband/wife*

Algorithm of Gale and Shapley: a stable marriage is computed in poly time for any instance

What about DE graphs ?

DE graphs

DE graphs have a tree decomposition verifying $(H1)$
Question: do DE bags verify $(H2), (H3)$?

One-to-one correspondance between:

- a DE bag with clique-acyclic orientation without reversible arc
- an instance of the stable marriage problem

A kernel of the DE graph is a stable marriage of the Gale-Shapley instance.

DE graphs

DE graphs have a tree decomposition verifying $(H1)$
Question: do DE bags verify $(H2), (H3)$?

One-to-one correspondance between:

- a DE bag with clique-acyclic orientation without reversible arc
- an instance of the stable marriage problem

A kernel of the DE graph is a stable marriage of the Gale-Shapley instance.

Thanks to stable marriages, $(H1), (H2), (H3)$ can be verified for DE clique-acyclic without reversible arc. The generalized algorithm is applicable.

Conclusion and open questions

Results:

- Algorithm for chordal graphs
- Generalization - applicable for some DE graphs

What's next:

- Refine the algorithm to compute all kernels? if possible.
  Would allow us to solve existence problem.
- Extend the results for DE graphs to the case of reversible arcs (weakly stable marriages)
- Characterize graph families that verify $(H1), (H2), (H3)$
- Look for minimal hypothesis instead of $(H1), (H2), (H3)$ (we would appreciate to weaken $(H3)$)

Lunch time

Thank you for your attention !