Kernelization techniques, Feedback Arc Set in Tournaments and Modular Decomposition

Christophe Paul

CNRS - LIRMM - Université Montpellier II, France

January 29, 2010

Joint work with

S. Bessy, F. Fomin, S. Gaspers, A. Perez, S. Saurabh, S. Thomassé

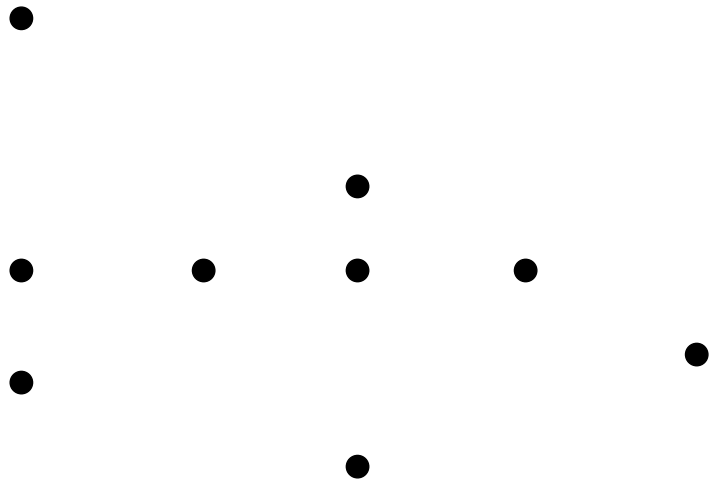
Introduction to kernelization and lower bound techniques

A linear kernel for Feedback Arc Set in Tournament (FAST)

Other modular decomposition based kernels

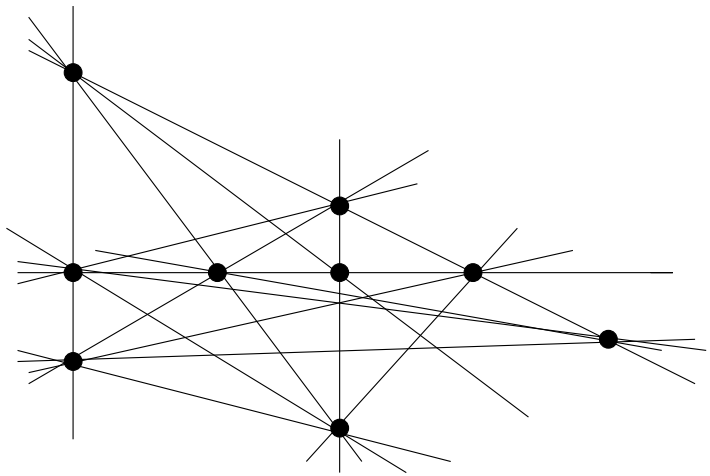
A FIRST EXAMPLE

Can we cover a set of n points in the plane by at most k lines ?



A FIRST EXAMPLE

Can we cover a set of n points in the plane by at most k lines ?

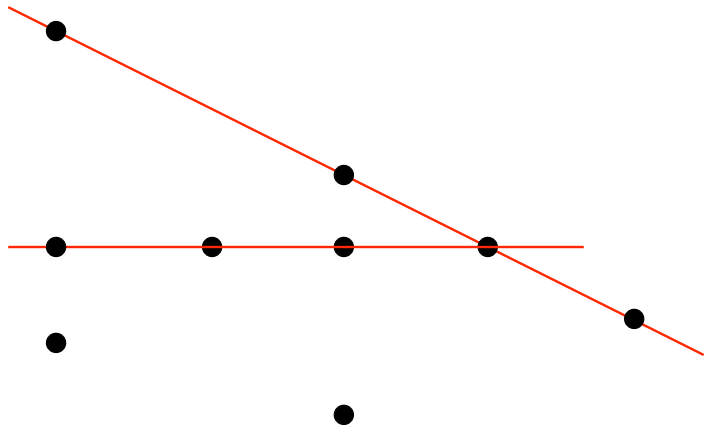


Observation

We can restrict the set of lines to those that are generated by the set of pair of points of S.

A FIRST EXAMPLE

Can we cover a set of n points in the plane by at most k lines ?

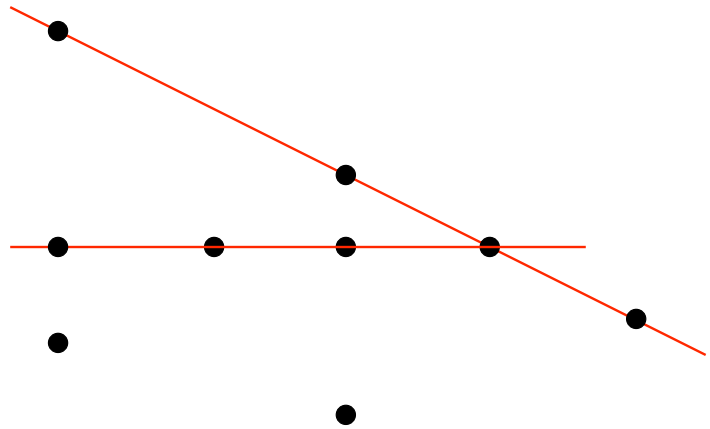


Reduction rule [Sunflower]

If a line contains at least $k+1$ points, then remove the points it covers and decrease k by 1.

A FIRST EXAMPLE

Can we cover a set of n points in the plane by at most k lines ?



Reduction rule [Sunflower]

If a line contains at least $k+1$ points, then remove the points it covers and decrease k by 1.

Théorème

Any instance of the point cover problem can be reduced in polynomial time to an equivalent instance of size at most k^2 points.

KERNELIZATION

A kernelization of a parameterized problem (Q, κ) is a polytime algorithm that transforms

$$\text{an instance } (I, k) \quad \longrightarrow \quad \text{an reduced instance } (I', k')$$

such that

- (I, k) is a YES-instance $\iff$ (I', k') is a YES-instance
- $|I'| \leqslant g(k)$
- $k' \leqslant k$

KERNELIZATION

A kernelization of a parameterized problem (Q, κ) is a polytime algorithm that transforms

$$\text{an instance } (I, k) \quad \longrightarrow \quad \text{an reduced instance } (I', k')$$

such that

- (I, k) is a YES-instance $\iff$ (I', k') is a YES-instance
- $|I'| \leqslant g(k)$
- $k' \leqslant k$ - or more generally $k' \leqslant h(k)$

KERNELIZATION

A kernelization of a parameterized problem (Q, κ) is a polytime algorithm that transforms

an instance (I, k) $\longrightarrow$ an reduced instance (I', k')

such that

- (I, k) is a YES-instance $\iff$ (I', k') is a YES-instance
- $|I'| \leqslant g(k)$
- $k' \leqslant k$ - or more generally $k' \leqslant h(k)$

Theorem [Folkore] A parameterized problem (Q, κ) is FPT if and only if it has a kernelization

KERNELIZATION

A kernelization of a parameterized problem (Q, κ) is a polytime algorithm that transforms

an instance (I, k) $\longrightarrow$ an reduced instance (I', k')

such that

- (I, k) is a YES-instance $\iff$ (I', k') is a YES-instance
- $|I'| \leqslant g(k)$
- $k' \leqslant k$ - or more generally $k' \leqslant h(k)$

Theorem [Folkore] A parameterized problem (Q, κ) is FPT if and only if it has a kernelization

Proof

$\Leftarrow$ Compute the kernel and solve by brute force

KERNELIZATION

A kernelization of a parameterized problem (Q, κ) is a polytime algorithm that transforms

an instance (I, k) $\longrightarrow$ an reduced instance (I', k')

such that

- (I, k) is a YES-instance $\iff$ (I', k') is a YES-instance
- $|I'| \leqslant g(k)$
- $k' \leqslant k$ - or more generally $k' \leqslant h(k)$

Theorem [Folkore] A parameterized problem (Q, κ) is FPT if and only if it has a kernelization

Proof

$\Leftarrow$ Compute the kernel and solve by brute force

$\Rightarrow$
- if $g(k) \leqslant n$, then solve in time $f(k).n^{O(1)} \leqslant n^{O(1)}$
- if $n < g(k)$, then (I, k) is a kernel.

KERNELIZATION

A kernelization of a parameterized problem (Q, κ) is a polytime algorithm that transforms

an instance (I, k) $\longrightarrow$ an reduced instance (I', k')

such that

- (I, k) is a YES-instance $\iff$ (I', k') is a YES-instance
- $|I'| \leqslant g(k)$
- $k' \leqslant k$ - or more generally $k' \leqslant h(k)$

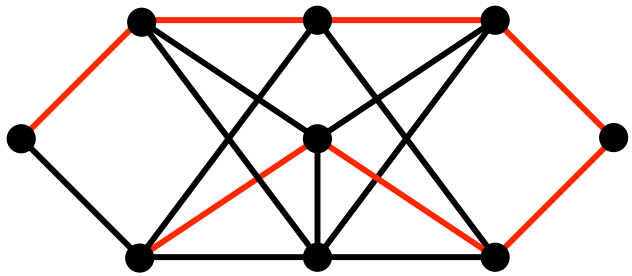
Theorem [Folkore] A parameterized problem (Q, κ) is FPT if and only if it has a kernelization

BUT this gives only exponential size kernel !

Does every parameterized problem have a polynomial kernel ?

LONGEST PATH

- A graph $G = (V, E)$, a parameter $k \in \mathbb{N}$
- Does G contain a path of length k ?



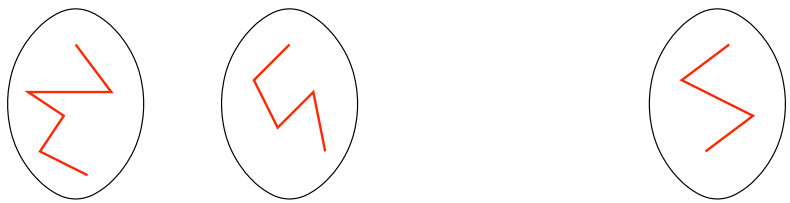
Longest Path is NP-Complete (cf. hamiltonian path) but is FPT by Color Coding [Alon, Yuster and Zwick])

LONGEST PATH

Hypothesis

LONGEST PATH has a poly-size kernelization $\mathcal{A}$

- consider an instance (G, k) built from a series of t instances
  $(G, k) = (G_1, k) \oplus (G_2, k) \oplus \ \ldots \ \oplus (G_t, k)$

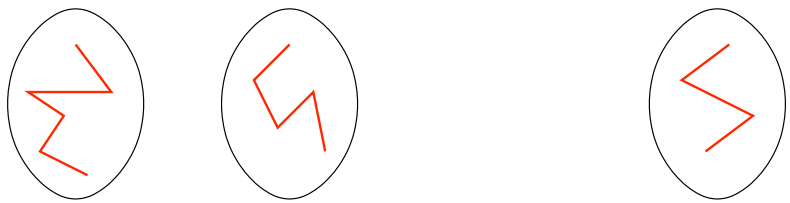


LONGEST PATH

Hypothesis

LONGEST PATH has a poly-size kernelization $\mathcal{A}$

- consider an instance (G, k) built from a series of t instances
  $(G, k) = (G_1, k) \oplus (G_2, k) \oplus \ \ldots \ \oplus (G_t, k)$



Observation : (G, k) is a YES instance iff $\exists i$ such (G_i, k) is a YES instance.

Question / intuition :

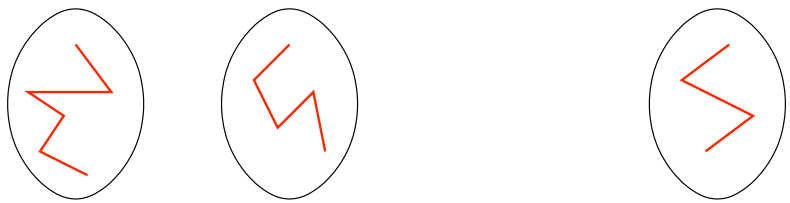
Is it possible that a polytime algorithm $\mathcal{A}$ computes an equivalent instance of size $k^{O(1)}$ (a value possibly much smaller than t) ?

LONGEST PATH

Hypothesis

LONGEST PATH has a poly-size kernelization $\mathcal{A}$

- consider an instance (G, k) built from a series of t instances
  $(G, k) = (G_1, k) \oplus (G_2, k) \oplus \ \ldots \ \oplus (G_t, k)$



Observation : (G, k) is a YES instance iff $\exists i$ such (G_i, k) is a YES instance.

Question / intuition :

Is it possible that a polytime algorithm $\mathcal{A}$ identifies which G_i is a YES instance (if any) ?

DISTILLATION ALGORITHMS

A OR-distillation algorithm $\mathcal{A}$ for a decision problem Q:

- receives as input a sequence of instances $(x_1, \ldots x_t)$

DISTILLATION ALGORITHMS

A OR-distillation algorithm $\mathcal{A}$ for a decision problem Q:

- receives as input a sequence of instances $(x_1, \ldots x_t)$
- runs in time $poly(\sum_{i=1}^{t} |x_i|)$,

DISTILLATION ALGORITHMS

A OR-distillation algorithm $\mathcal{A}$ for a decision problem Q:

- receives as input a sequence of instances $(x_1, \ldots x_t)$
- runs in time $poly(\sum_{i=1}^{t} |x_i|)$,
- computes an instance y such that
  1. y is a YES instance $\Leftrightarrow \exists i \in [t]$, x_i is a YES instance;
  2. $|y|$ is polynomial in $\max_{i \in [t]} |x_i|$.

DISTILLATION ALGORITHMS

A OR-distillation algorithm $\mathcal{A}$ for a decision problem Q:

- receives as input a sequence of instances $(x_1, \ldots x_t)$
- runs in time $poly(\sum_{i=1}^{t} |x_i|)$,
- computes an instance y such that
  1. y is a YES instance $\Leftrightarrow \exists i \in [t]$, x_i is a YES instance;
  2. $|y|$ is polynomial in $\max_{i \in [t]} |x_i|$.

Conjecture [Bodlaender, Downey, Fellows, Hermelin]

No NP-complete problem has a OR-distillation algorithm.

DISTILLATION ALGORITHMS

A OR-distillation algorithm $\mathcal{A}$ for a decision problem Q:

- receives as input a sequence of instances $(x_1, \ldots x_t)$
- runs in time $poly(\sum_{i=1}^{t} |x_i|)$,
- computes an instance y such that
  1. y is a YES instance $\Leftrightarrow \exists i \in [t]$, x_i is a YES instance;
  2. $|y|$ is polynomial in $\max_{i \in [t]} |x_i|$.

Conjecture [Bodlaender, Downey, Fellows, Hermelin]

No NP-complete problem has a OR-distillation algorithm.

Theorem [Fortnow et Santhanam]

The OR-distillation conjecture holds, unless $PH = \Sigma_p^3$

or-COMPOSITIONAL ALGORITHMS

An OR-compositional algorithm for a parameterized problem (Q, κ):

- receives a series of instances $((x_1, k), \ldots (x_t, k))$;

or-COMPOSITIONAL ALGORITHMS

An OR-compositional algorithm for a parameterized problem (Q, κ):

- receives a series of instances $((x_1, k), \ldots (x_t, k))$;
- runs in time $poly(\sum_{i=1}^{t} |x_i| + k)$,

or-COMPOSITIONAL ALGORITHMS

An OR-compositional algorithm for a parameterized problem (Q, κ):

- receives a series of instances $((x_1, k), \ldots (x_t, k))$;
- runs in time $poly(\sum_{i=1}^{t} |x_i| + k)$,
- computes an instance (y, k') such that
  1. (y, k') is a YES instance $\Leftrightarrow \exists i \in [t]$, (x_i, k) is a YES instance;
  2. k' is polynomial in k.

or-COMPOSITIONAL ALGORITHMS

An OR-compositional algorithm for a parameterized problem (Q, κ):

- receives a series of instances $((x_1, k), \ldots (x_t, k))$;
- runs in time $poly(\sum_{i=1}^{t} |x_i| + k)$,
- computes an instance (y, k') such that
  1. (y, k') is a YES instance $\Leftrightarrow \exists i \in [t]$, (x_i, k) is a YES instance;
  2. k' is polynomial in k.

Observation : The existence of a poly kernel for Longest Path would imply the existence of an OR-compositional algorithm.

or-COMPOSITIONAL ALGORITHMS

An OR-compositional algorithm for a parameterized problem (Q, κ):

- receives a series of instances $((x_1, k), \ldots (x_t, k))$;
- runs in time $poly(\sum_{i=1}^{t} |x_i| + k)$,
- computes an instance (y, k') such that
  1. (y, k') is a YES instance $\Leftrightarrow \exists i \in [t]$, (x_i, k) is a YES instance;
  2. k' is polynomial in k.

Theorem [Bodlaender, Downey, Fellows, Hermelin]

Let (Q, κ) be an OR-composable parameterized problem such that decision problem Q is NP-Complete.
If (Q, κ) has a polynomial kernel, then Q has OR-distillation algorithm.

or-COMPOSITIONAL ALGORITHMS

An OR-compositional algorithm for a parameterized problem (Q, κ):

- receives a series of instances $((x_1, k), \ldots (x_t, k))$;
- runs in time $poly(\sum_{i=1}^{t} |x_i| + k)$,
- computes an instance (y, k') such that
  1. (y, k') is a YES instance $\Leftrightarrow \exists i \in [t]$, (x_i, k) is a YES instance;
  2. k' is polynomial in k.

Theorem [Bodlaender, Downey, Fellows, Hermelin]

Let (Q, κ) be an OR-composable parameterized problem such that decision problem Q is NP-Complete.
If (Q, κ) has a polynomial kernel, then Q has OR-distillation algorithm.

Corollary

Unless $PH = \Sigma_p^3$, LONGEST PATH does not have a polynomial kernel.

Introduction to kernelization and lower bound techniques

A linear kernel for Feedback Arc Set in Tournament (FAST)

Other modular decomposition based kernels

k-FEEDBACK ARC SET IN TOURNAMENT (k-FAST)

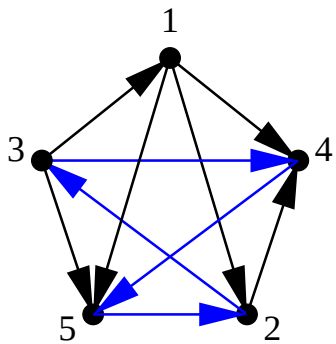
Input: A tournament $T = (V, A)$ and a parameter k

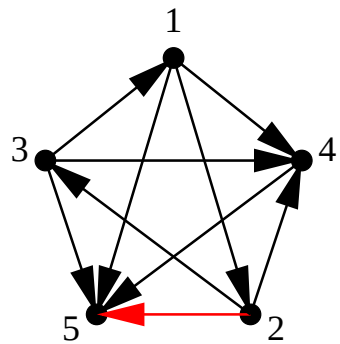
Question: Can T be transformed into an acyclic tournament by at most k arc reversals?

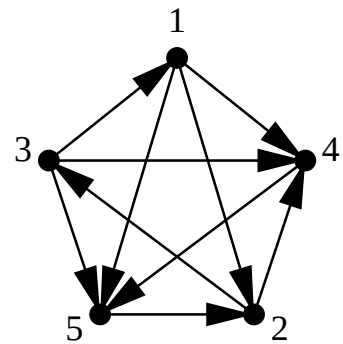
k-FEEDBACK ARC SET IN TOURNAMENT (k-FAST)

Input: A tournament $T = (V, A)$ and a parameter k

Question: Can T be transformed into an acyclic tournament by at most k arc reversals?



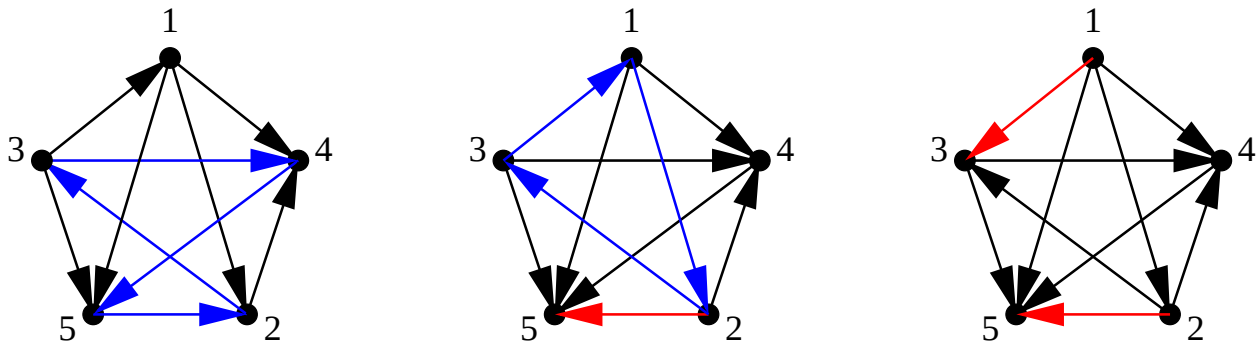




k-FEEDBACK ARC SET IN TOURNAMENT (k-FAST)

Input: A tournament $T = (V, A)$ and a parameter k

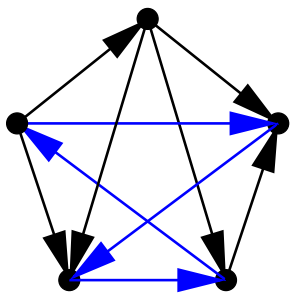
Question: Can T be transformed into an acyclic tournament by at most k arc reversals?

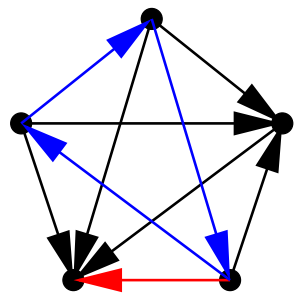


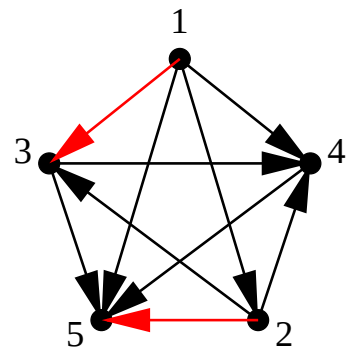
k-FEEDBACK ARC SET IN TOURNAMENT (k-FAST)

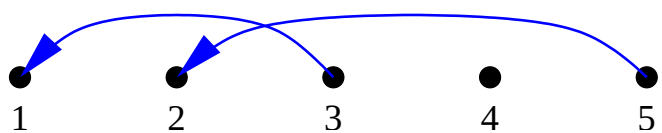
Input: A tournament $T = (V, A)$ and a parameter k

Question: Can T be transformed into an acyclic tournament by at most k arc reversals?









$\longrightarrow$ find a vertex ordering π with at most k backward edges

KNOWN RESULTS

- NP-Complete [Alon'06] [Charbit et al.'07]
- FTP [Raman, Saurabh'06] [Alon et al.'09]
- $O(k^2)$ vertex kernel [Dom et al.'06], [Alon et al.'09]

KNOWN RESULTS

- NP-Complete [Alon'06] [Charbit et al.'07]
- FTP [Raman, Saurabh'06] [Alon et al.'09]
- $O(k^2)$ vertex kernel [Dom et al.'06], [Alon et al.'09]

- $(1+\epsilon)$-approximation scheme [Kenyon-Mathieu, Schudy'07]

KNOWN RESULTS

- NP-Complete [Alon'06] [Charbit et al.'07]
- FTP [Raman, Saurabh'06] [Alon et al.'09]
- $O(k^2)$ vertex kernel [Dom et al.'06], [Alon et al.'09]

- $(1+\epsilon)$-approximation scheme [Kenyon-Mathieu, Schudy'07]

Our result

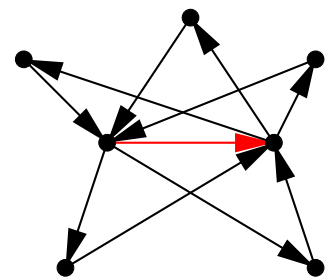
k-FAST has a $O(k)$ vertex kernel

Obs.: A tournament is acyclic iff there is no (directed) triangle

Obs.: A tournament is acyclic iff there is no (directed) triangle

Rule 1 [irrelevant vertex] If a vertex v is not contained in any triangle, then delete v

Rule 2 [sunflower] If there is an arc belonging to more that k distinct triangles, then reverse it and decrease k by 1

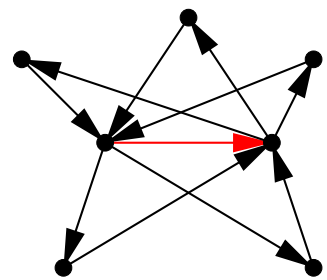


Obs.: A tournament is acyclic iff there is no (directed) triangle

Rule 1 [irrelevant vertex] If a vertex v is not contained in any triangle, then delete v

A reduced tournament contains no source nor sink

Rule 2 [sunflower] If there is an arc belonging to more that k distinct triangles, then reverse it and decrease k by 1

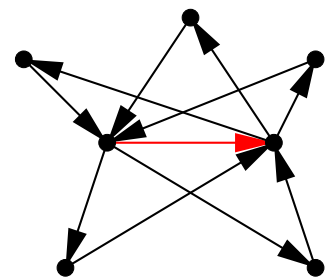


Obs.: A tournament is acyclic iff there is no (directed) triangle

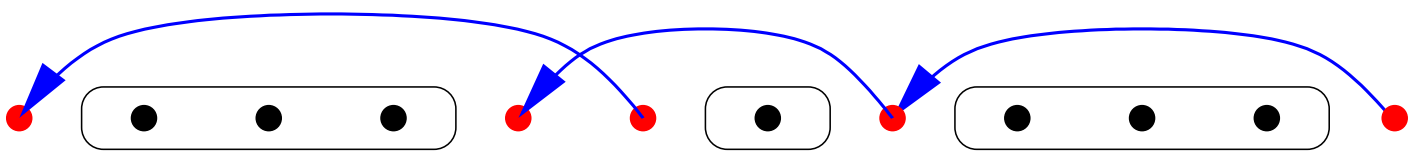
Rule 1 [irrelevant vertex] If a vertex v is not contained in any triangle, then delete v

A reduced tournament contains no source nor sink

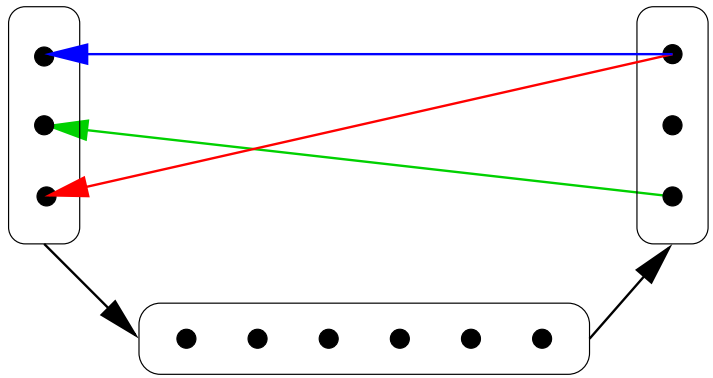
Rule 2 [sunflower] If there is an arc belonging to more that k distinct triangles, then reverse it and decrease k by 1



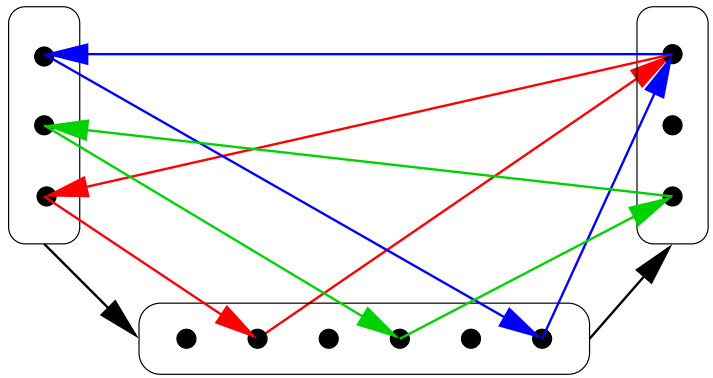
The span $s(\overrightarrow{uv})$ of a backward arc of a reduced tournament is $\leqslant 2k+2$



Rule 3 [acyclic module] Let M be a maximal acyclic module. If there are at most $p = |M|$ arcs from $N^+(M)$ to $N^-(M)$, then reverse all these arcs and decrease k by p.



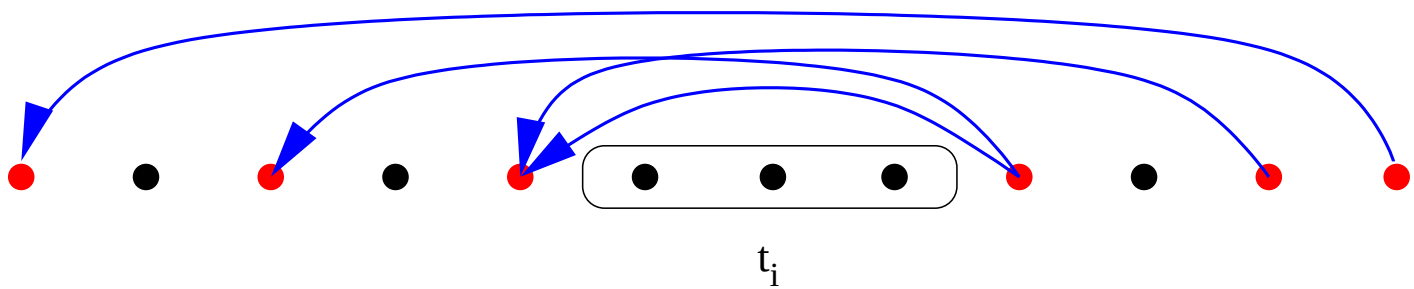
Rule 3 [acyclic module] Let M be a maximal acyclic module. If there are at most $p = |M|$ arcs from $N^+(M)$ to $N^-(M)$, then reverse all these arcs and decrease k by p.



Rule 3 [acyclic module] Let M be a maximal acyclic module. If there are at most $p = |M|$ arcs from $N^+(M)$ to $N^-(M)$, then reverse all these arcs and decrease k by p.

acyclic module Rule $\Rightarrow$

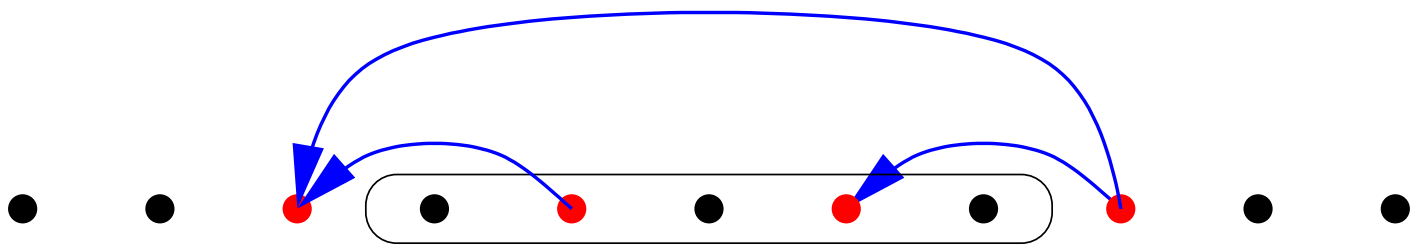
$$\sum t_i^2 \leqslant \sum s(\overrightarrow{uv})$$



Rule 3 [acyclic module] Let M be a maximal acyclic module. If there are at most $p = |M|$ arcs from $N^+(M)$ to $N^-(M)$, then reverse all these arcs and decrease k by p.

sunflower Rule $\Rightarrow$

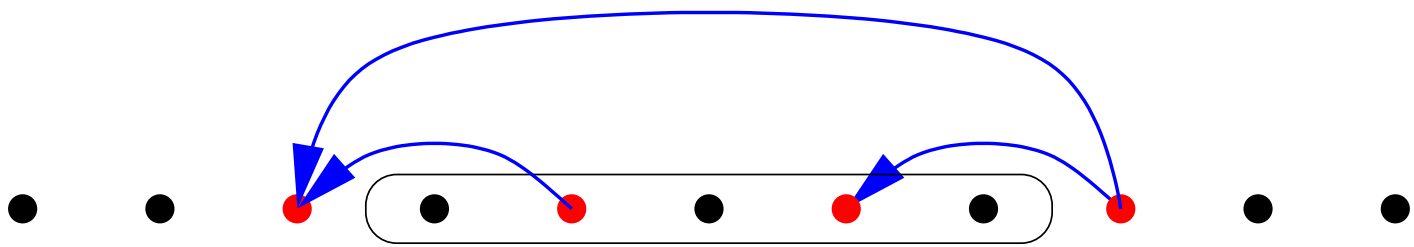
$$\sum t_i^2 \leqslant \sum s(\overrightarrow{uv}) \leqslant k(2k+2)$$



Rule 3 [acyclic module] Let M be a maximal acyclic module. If there are at most $p = |M|$ arcs from $N^+(M)$ to $N^-(M)$, then reverse all these arcs and decrease k by p.

sunflower Rule $\Rightarrow$

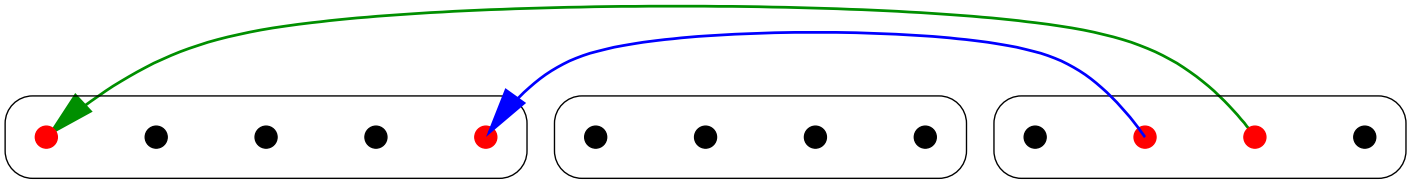
$$\sum t_i^2 \leqslant \sum s(\overrightarrow{uv}) \leqslant k(2k+2)$$



Theorem:

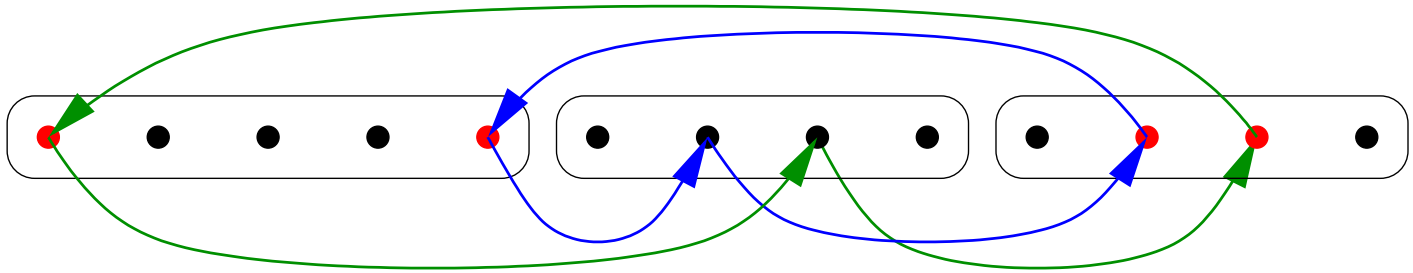
k-FAST has a vertex kernel of size $2k + \sum t_i = O(k\sqrt{k})$.

ACYCLIC MODULE RULE REVISTED



$$\overrightarrow{A} = \overrightarrow{E}\ (\text{external arcs}) \uplus \overrightarrow{L}(\ \text{local arcs})$$

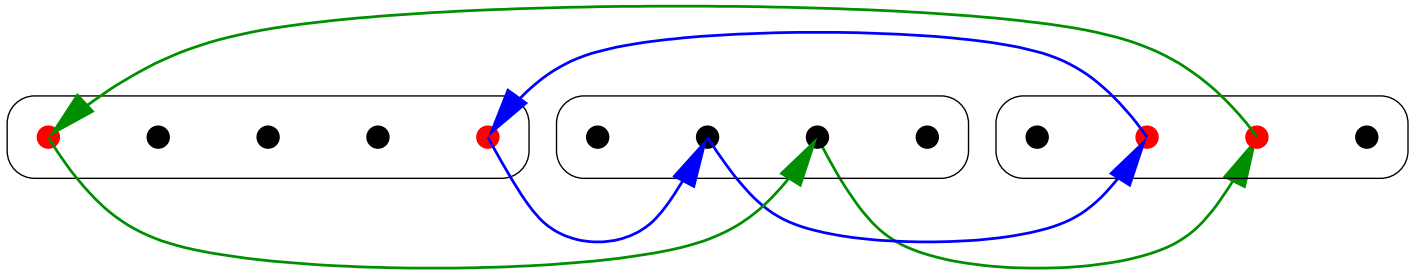
ACYCLIC MODULE RULE REVISTED



$$\overrightarrow{A} = \overrightarrow{E}\ (\text{external arcs}) \uplus \overrightarrow{L}(\ \text{local arcs})$$

The backward arcs of $\overrightarrow{E}$ are disjointly certified by arcs of $\overrightarrow{E}$

ACYCLIC MODULE RULE REVISTED



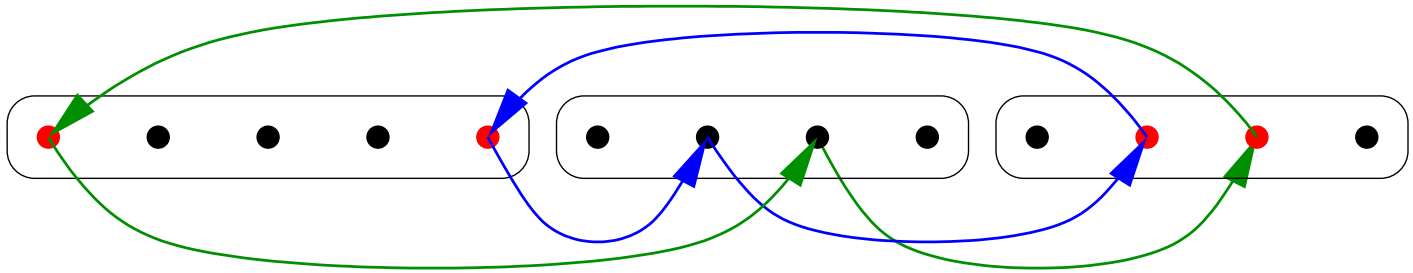
$$\overrightarrow{A} = \overrightarrow{E}\ (\text{external arcs}) \uplus \overrightarrow{L}(\ \text{local arcs})$$

The backward arcs of $\overrightarrow{E}$ are disjointly certified by arcs of $\overrightarrow{E}$

Définition

A vertex partition $\mathcal{P}$ is safe if *backwards*$(\overrightarrow{E})$ certified within $\overrightarrow{E}$

ACYCLIC MODULE RULE REVISTED



$$\overrightarrow{A} = \overrightarrow{E}\ (\text{external arcs}) \uplus \overrightarrow{L}(\ \text{local arcs})$$

The backward arcs of $\overrightarrow{E}$ are disjointly certified by arcs of $\overrightarrow{E}$

Définition

A vertex partition $\mathcal{P}$ is safe if *backwards*$(\overrightarrow{E})$ certified within $\overrightarrow{E}$

Rule 3 [safe partition] If $\mathcal{P}$ is a safe partition
then reverse *backwards*$(\overrightarrow{E})$ and decrease k by $|$*backwards*$(\overrightarrow{E})|$.

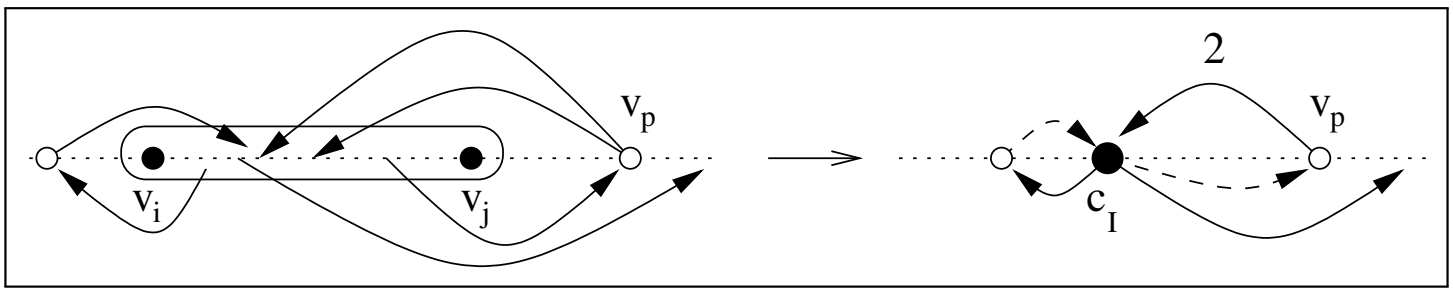
GOAL

1. Find a safe partition and reduce the tournament by the safe partition rule
2. Estimate the size of a reduced tournament

GOAL

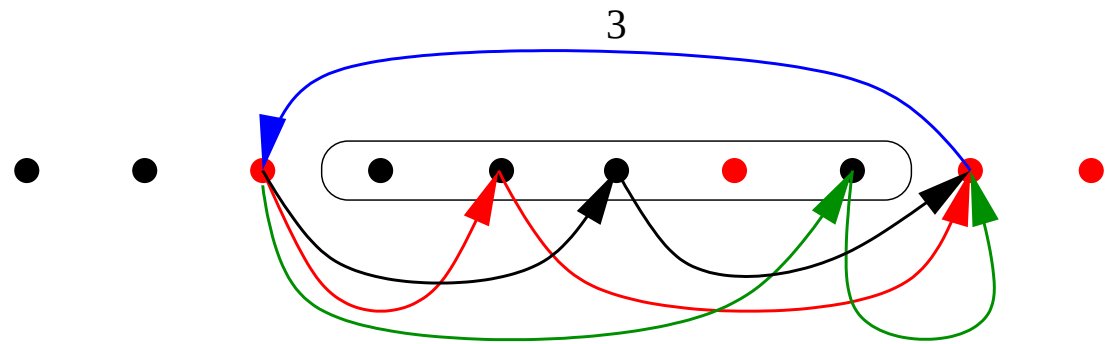
1. Find a safe partition and reduce the tournament by the safe partition rule
2. Estimate the size of a reduced tournament

Interval contraction



CERTIFICATE

A certificate $c(\overrightarrow{uv})$ for a backward arc $\overrightarrow{uv}$ of weight ω is a collection of ω arc-disjoint forward paths in the span of $\overrightarrow{uv}$



If $\overrightarrow{B}$ is a set of backward-arcs, then

$$c(\overrightarrow{B}) = \{c(\overrightarrow{uv}) \text{ pairwise disjoint } \mid \overrightarrow{uv} \in \overrightarrow{B}\}$$

Lemma: If T is an ordered backward weighted tournament st.

$$\forall\ I = [v_i \dots v_j] \ : \quad 2\ .\ \omega(I) \leqslant |I| - 1$$

then there exists a certificate $c(\overrightarrow{B} = \{\text{backward-arcs of } T\})$

Lemma: If T is an ordered backward weighted tournament st.

$$\forall\ I = [v_i \dots v_j]\ :\quad 2\ .\ \omega(I) \leqslant |I| - 1$$

then there exists a certificate $c(\overrightarrow{B} = \{\text{backward-arcs of } T\})$

Proof sketch (by induction)

1. $[v_1 \dots v_n]$ is the only critical interval : $2\ .\ \omega(I) = |I| - 1$

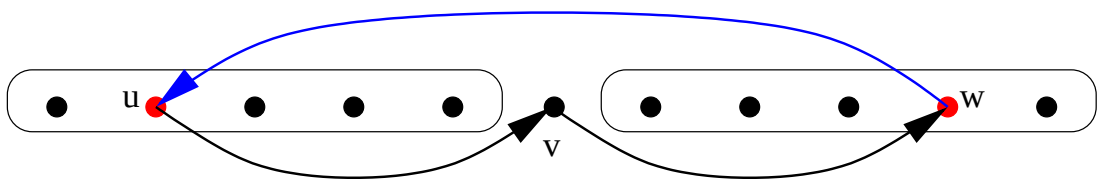
Lemma: If T is an ordered backward weighted tournament st.

$$\forall\ I = [v_i \dots v_j]\ :\quad 2\ .\ \omega(I) \leqslant |I| - 1$$

then there exists a certificate $c(\overrightarrow{B} = \{\text{backward-arcs of } T\})$

Proof sketch (by induction)

1. $[v_1 \dots v_n]$ is the only critical interval : $2\ .\ \omega(I) = |I| - 1$
   - Remove a vertex v non incident to any backward-arc (it exists)
   - choose a backward-arc $\overrightarrow{wu}$ over u of maximal span, reverse it



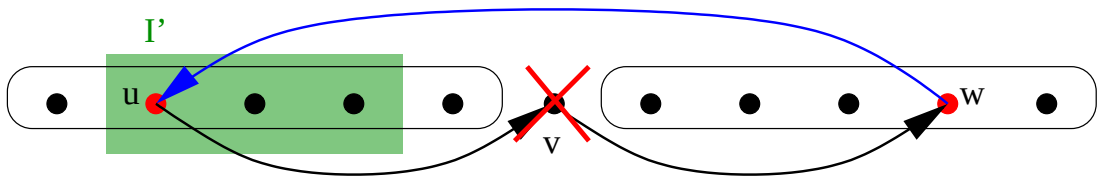
Lemma: If T is an ordered backward weighted tournament st.

$$\forall\ I = [v_i \dots v_j]\ :\quad 2\ .\ \omega(I) \leqslant |I| - 1$$

then there exists a certificate $c(\overrightarrow{B} = \{\text{backward-arcs of } T\})$

Proof sketch (by induction)

1. $[v_1 \dots v_n]$ is the only critical interval : $2\ .\ \omega(I) = |I| - 1$
   - Remove a vertex v non incident to any backward-arc (it exists)
   - choose a backward-arc $\overrightarrow{wu}$ over u of maximal span, reverse it
   - let I' be an interval not containing v



$$2\ .\ \omega(I') \leqslant |I'| - 1$$

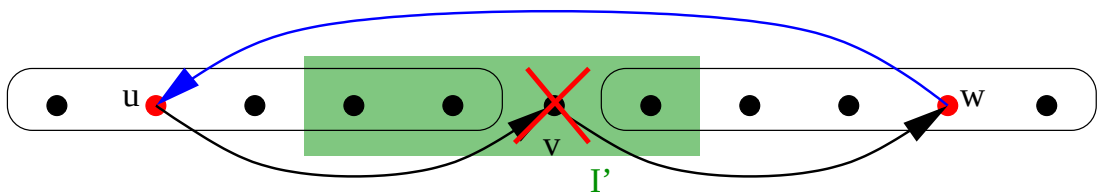
Lemma: If T is an ordered backward weighted tournament st.

$$\forall\ I = [v_i \dots v_j]\ :\quad 2\ .\ \omega(I) \leqslant |I| - 1$$

then there exists a certificate $c(\overrightarrow{B} = \{\text{backward-arcs of } T\})$

Proof sketch (by induction)

1. $[v_1 \dots v_n]$ is the only critical interval : $2\ .\ \omega(I) = |I| - 1$
   - Remove a vertex v non incident to any backward-arc (it exists)
   - choose a backward-arc $\overrightarrow{wu}$ over u of maximal span, reverse it
   - let I' be an interval containing v but not u and w



$$2\ .\ \omega(I') \leqslant |I'| - 1$$

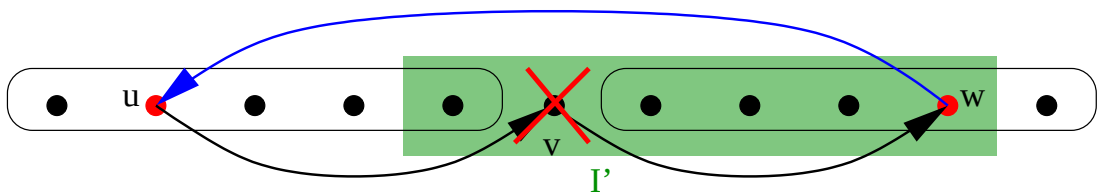
Lemma: If T is an ordered backward weighted tournament st.

$$\forall\ I = [v_i \dots v_j]\ :\quad 2\ .\ \omega(I) \leqslant |I| - 1$$

then there exists a certificate $c(\overrightarrow{B} = \{\text{backward-arcs of } T\})$

Proof sketch (by induction)

1. $[v_1 \dots v_n]$ is the only critical interval : $2\ .\ \omega(I) = |I| - 1$
   - Remove a vertex v non incident to any backward-arc (it exists)
   - choose a backward-arc $\overrightarrow{wu}$ over u of maximal span, reverse it
   - let I' be an interval containing v, and u or w



$$2\ .\ \omega(I') \leqslant |I'| - 1$$

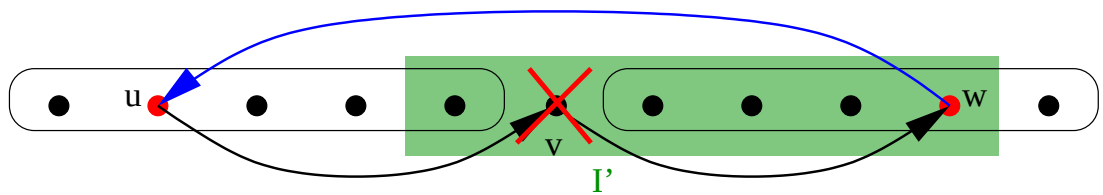
Lemma: If T is an ordered backward weighted tournament st.

$$\forall\ I = [v_i \dots v_j]\ :\quad 2\ .\ \omega(I) \leqslant |I| - 1$$

then there exists a certificate $c(\overrightarrow{B} = \{\text{backward-arcs of } T\})$

Proof sketch (by induction)

1. $[v_1 \dots v_n]$ is the only critical interval : $2\ .\ \omega(I) = |I| - 1$
   - Remove a vertex v non incident to any backward-arc (it exists)
   - choose a backward-arc $\overrightarrow{wu}$ over u of maximal span, reverse it
   - let I' be an interval



$$2\ .\ \omega(I') \leqslant |I'| - 1$$

By its choice, $\overrightarrow{wu}$ cannot be used once reversed

Lemma: If T is an ordered backward weighted tournament st.

$$\forall\ I = [v_i \dots v_j]\ :\quad 2\ .\ \omega(I) \leqslant |I| - 1$$

then there exists a certificate $c(\overrightarrow{B} = \{\text{backward-arcs of } T\})$

Proof sketch (by induction)

1. $[v_1 \dots v_n]$ is the only critical interval : $2\ .\ \omega(I) = |I| - 1$
2. if there exists a critical interval $I \neq V$,
   then $T[I]$ and T/I can be disjointly certified

Lemma: If T is an ordered backward weighted tournament st.

$$|V(T)| \geqslant 2p+1 \quad \text{and} \quad |\overrightarrow{B} = \{\text{backward-arcs of } T\}| \leqslant p$$

then T is not reduced by the safe partition rule, which can be applied in polytime

Lemma: If T is an ordered backward weighted tournament st.

$$|V(T)| \geqslant 2p+1 \quad \text{and} \quad |\overrightarrow{B} = \{\text{backward-arcs of } T\}| \leqslant p$$

then T is not reduced by the safe partition rule, which can be applied in polytime

Proof idea (by induction)

- If every interval I satisfies $2 \,.\, \omega(I) \leqslant |I| - 1$, then the partition into singletons is safe

Lemma: If T is an ordered backward weighted tournament st.

$$|V(T)| \geqslant 2p+1 \quad \text{and} \quad |\overrightarrow{B} = \{\text{backward-arcs of } T\}| \leqslant p$$

then T is not reduced by the safe partition rule, which can be applied in polytime

Proof idea (by induction)

- If every interval I satisfies $2 \,.\, \omega(I) \leqslant |I| - 1$, then the partition into singletons is safe
- Let I be an interval st $2 \,.\, \omega(I) > |I| - 1$ then

contract I into v_I, find a safe partition $\mathcal{P}$ of T/I
$\mathcal{P}'$ obtained by substituting v_I by I is a safe partition of T

Theorem: For every fixed $\epsilon > 0$, there exists a vertex kernel for k-FAST with at most $(2 + \epsilon)k$ vertices.

It can be computed in polytime.

Theorem: For every fixed $\epsilon > 0$, there exists a vertex kernel for k-FAST with at most $(2+\epsilon)k$ vertices.

It can be computed in polytime.

Proof idea (by induction)

- compute an approximation $\overrightarrow{B}$ of size at most $(1+\epsilon/2).k$

Theorem: For every fixed $\epsilon > 0$, there exists a vertex kernel for k-FAST with at most $(2+\epsilon)k$ vertices.

It can be computed in polytime.

Proof idea (by induction)

- compute an approximation $\overrightarrow{B}$ of size at most $(1+\epsilon/2).k$
- order T with respect to $\overrightarrow{B}$

Theorem: For every fixed $\epsilon > 0$, there exists a vertex kernel for k-FAST with at most $(2+\epsilon)k$ vertices.

It can be computed in polytime.

Proof idea (by induction)

- compute an approximation $\overrightarrow{B}$ of size at most $(1+\epsilon/2).k$
- order T with respect to $\overrightarrow{B}$
- if $|V(T)| \leqslant (2+\epsilon).k$, then apply the safe partition rule and then the other rules

Theorem: For every fixed $\epsilon > 0$, there exists a vertex kernel for k-FAST with at most $(2+\epsilon)k$ vertices.

It can be computed in polytime.

Proof idea (by induction)

- compute an approximation $\overrightarrow{B}$ of size at most $(1+\epsilon/2).k$
- order T with respect to $\overrightarrow{B}$
- if $|V(T)| \leqslant (2+\epsilon).k$, then apply the safe partition rule and then the other rules
- when it stops, either we have the kernel or $k = 0$

Introduction to kernelization and lower bound techniques

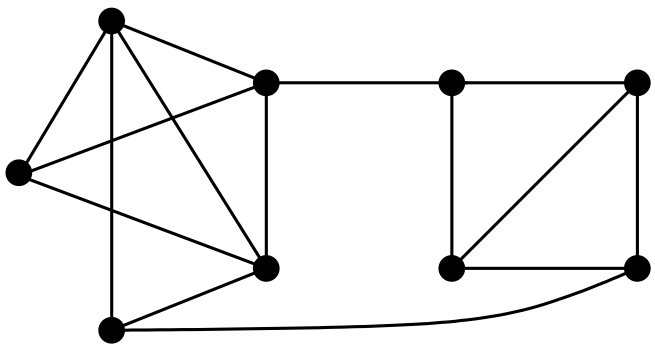
A linear kernel for Feedback Arc Set in Tournament (FAST)

Other modular decomposition based kernels

CLUSTER EDITING

Given a graph $G = (V, E)$, find $F \subseteq V \times V$ such that

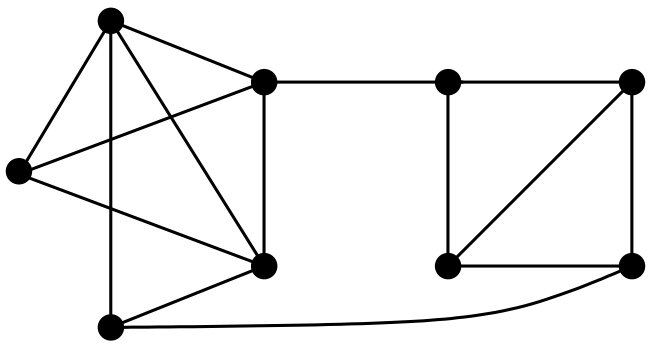
- $|F| \leqslant k$ et
- $H = (V, E \triangle F)$ is the disjoint union of cliques

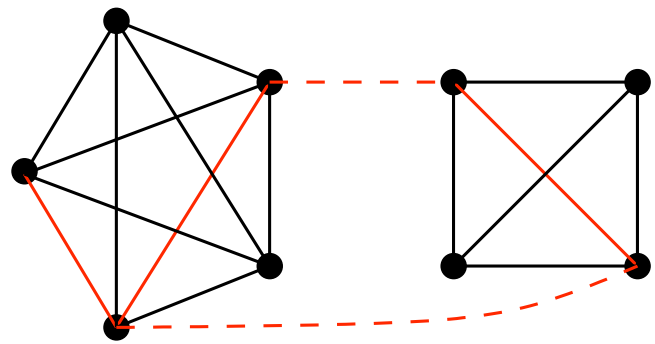


CLUSTER EDITING

Given a graph $G = (V, E)$, find $F \subseteq V \times V$ such that

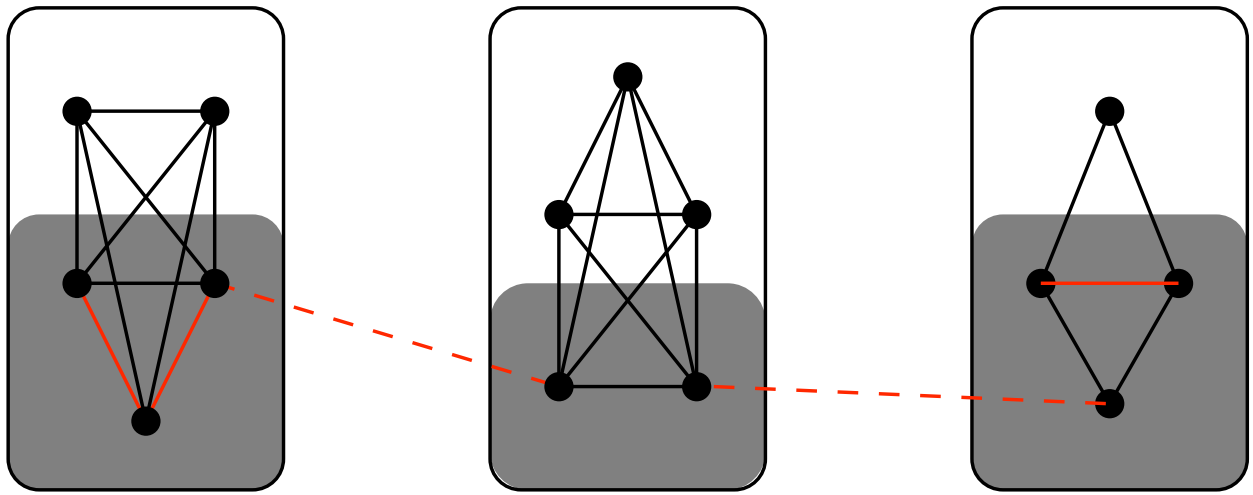
- $|F| \leqslant k$ et
- $H = (V, E \triangle F)$ is the disjoint union of cliques





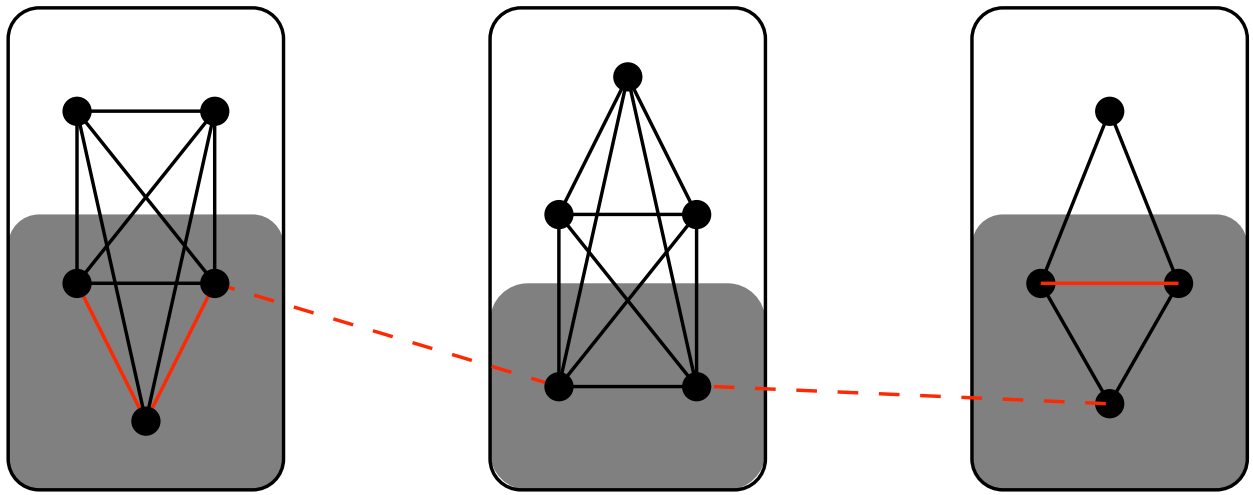
A vertex is **affected** if it is incident to an edited edge F.

1. A positive instance has at most $2k$ affected vertices



A vertex is **affected** if it is incident to an edited edge F.

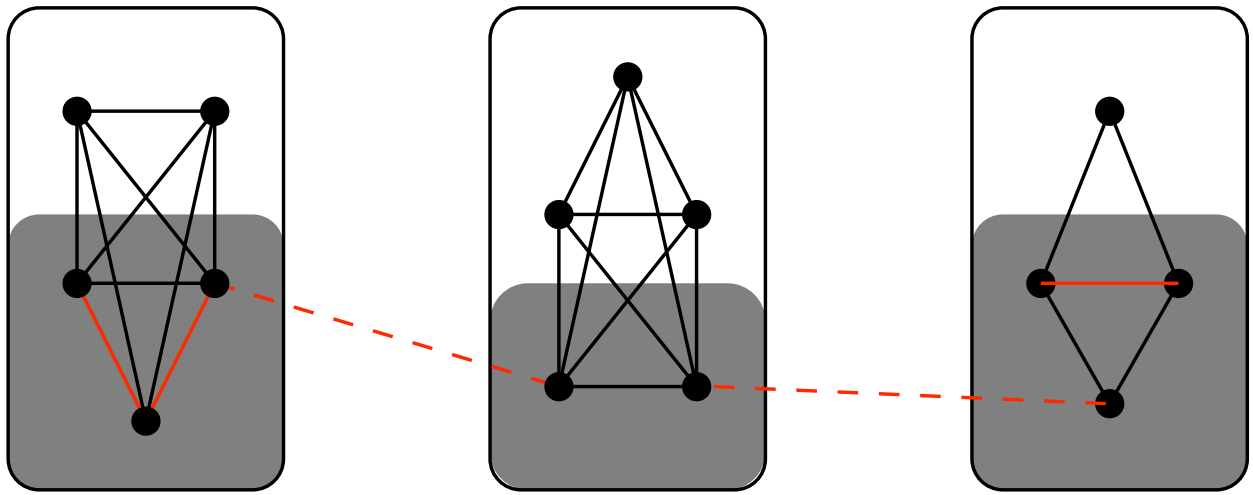
1. A positive instance has at most $2k$ affected vertices
2. A k-edition create at most $2k$-clusters



- The set S of non-affected vertices of a cluster is a clique-module in G

A vertex is **affected** if it is incident to an edited edge F.

1. A positive instance has at most $2k$ affected vertices
2. A k-edition create at most $2k$-clusters



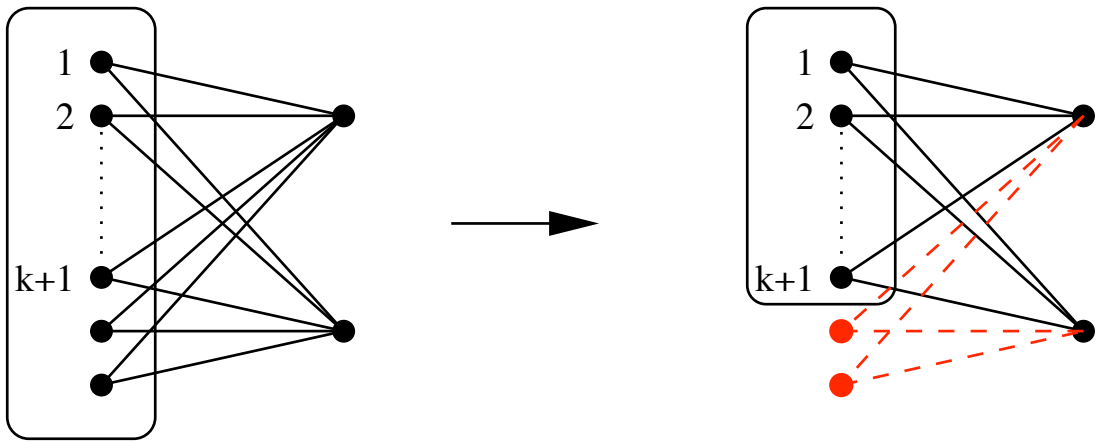
- The set S of non-affected vertices of a cluster is a clique-module in G
- A critical clique is a maximal clique-module of G

REDUCTION RULES

1. Remove clique connected components.

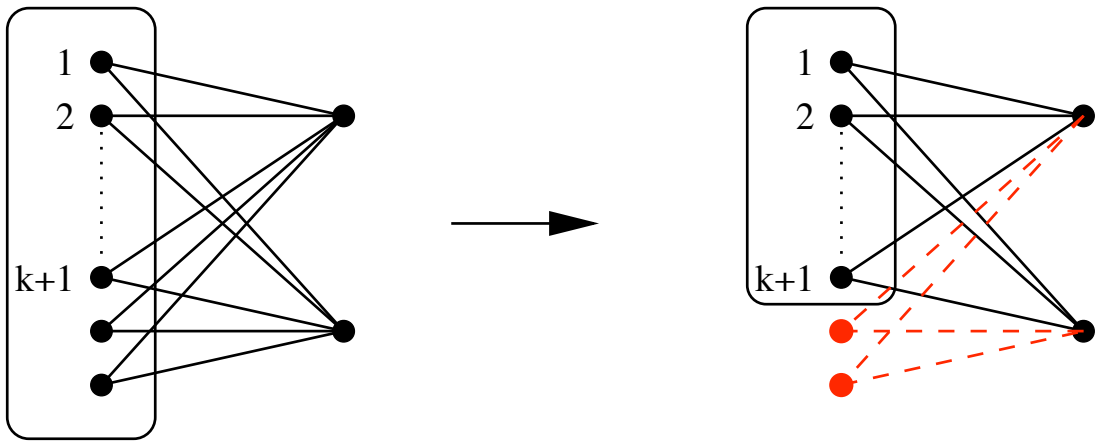
REDUCTION RULES

1. Remove clique connected components.
2. If G has a critical clique K of size $|K| > k+1$, shrink K down to $k+1$ vertices



REDUCTION RULES

1. Remove clique connected components.
2. If G has a critical clique K of size $|K| > k+1$, shrink K down to $k+1$ vertices

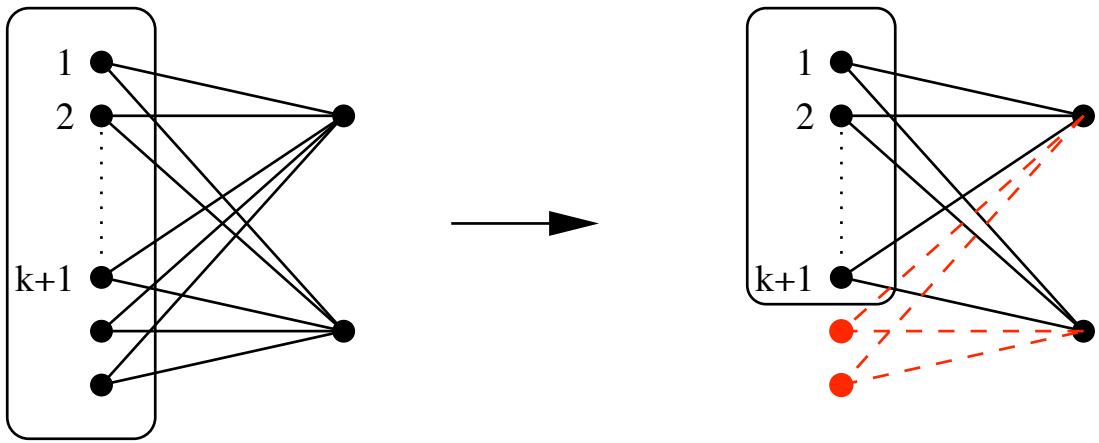


Theorem

k-Cluster Editing has a $4k^2 + 2k$ vertex kernel (it can be computed in linear time).

REDUCTION RULES

1. Remove clique connected components.
2. If G has a critical clique K of size $|K| > k+1$, shrink K down to $k+1$ vertices



Theorem

k-Cluster Editing has a $4k^2 + 2k$ vertex kernel (it can be computed in linear time).

$\longrightarrow$ a $4k$ vertex kernel is known [Guo'07, Fellows et al.'07]

OTHER EXAMPLES...

- Bicluster Editing
- Closest 3-Leaf Power
- Flip Consensus Tree
- Cograph Editing

OTHER EXAMPLES...

- Bicluster Editing
- Closest 3-Leaf Power
- Flip Consensus Tree
- Cograph Editing

THANK YOU !