Subexponential parameterized algorithm for computing the cutwidth of a semi-complete digraph

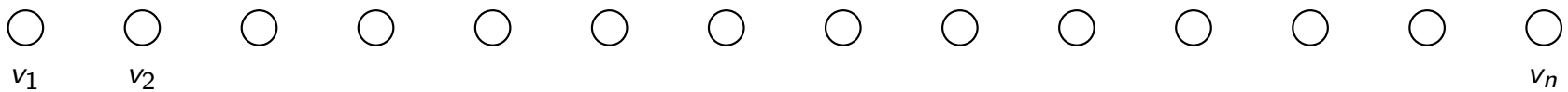
Fedor V. Fomin, Michał Pilipczuk

Department of Informatics, University of Bergen, Norway

ESA 2013, Sophia Antipolis, France,
September 3rd, 2013

Cutwidth of an undirected graph

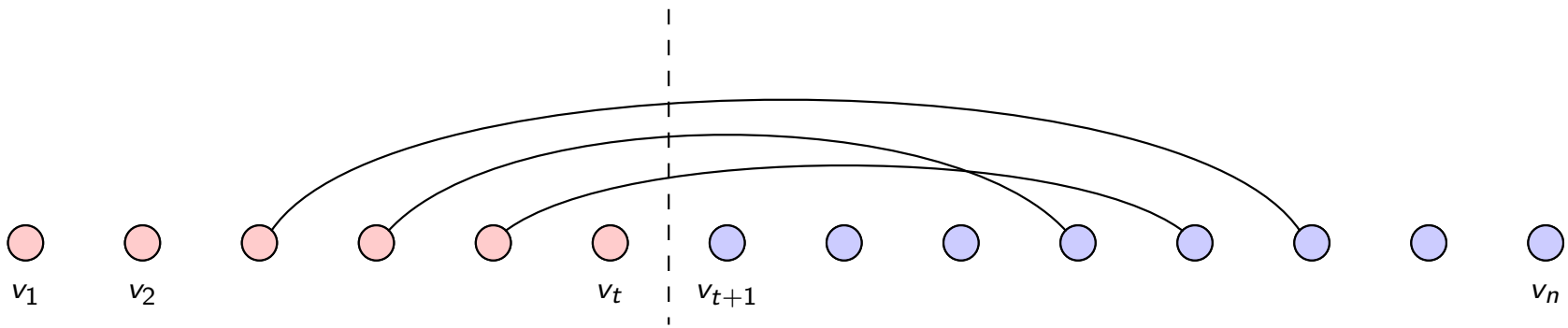
- Let $\sigma = (v_1, v_2, \ldots, v_n)$ be an ordering of $V(G)$.



Cutwidth of an undirected graph

- Let $\sigma = (v_1, v_2, \ldots, v_n)$ be an ordering of $V(G)$.
- *Width* of σ is equal to

$$\max_{t=0,1,\ldots,n} |E(\{v_1, \ldots, v_t\}, \{v_{t+1}, \ldots, v_n\})|.$$

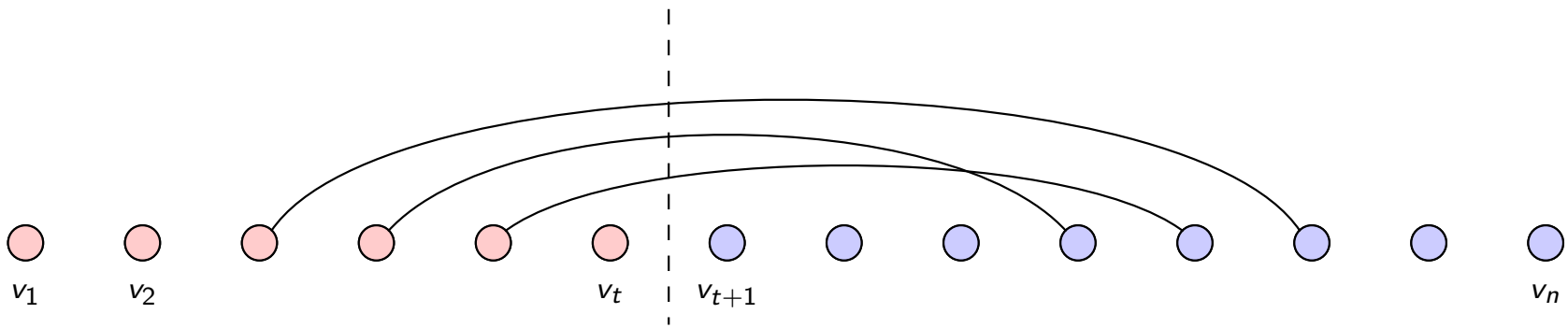


Cutwidth of an undirected graph

- Let $\sigma = (v_1, v_2, \ldots, v_n)$ be an ordering of $V(G)$.
- *Width* of σ is equal to

$$\max_{t=0,1,\ldots,n} |E(\{v_1, \ldots, v_t\}, \{v_{t+1}, \ldots, v_n\})|.$$

- *Cutwidth* of G is minimum width among orderings.

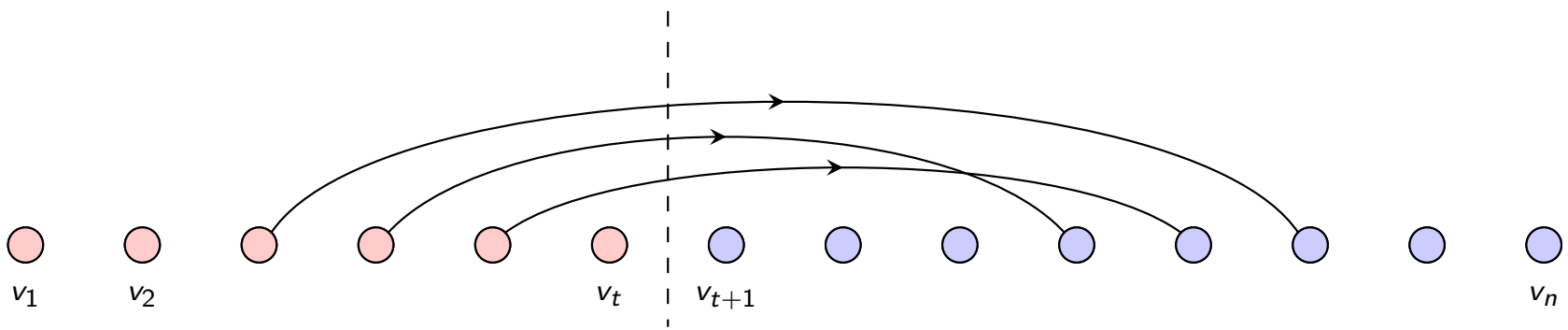


Cutwidth of a directed graph

- Let $\sigma = (v_1, v_2, \ldots, v_n)$ be an ordering of $V(D)$.
- *Width* of σ is equal to

$$\max_{t=0,1,\ldots,n} |E(\{v_1, \ldots, v_t\}, \{v_{t+1}, \ldots, v_n\})|.$$

- *Cutwidth* of D is minimum width among orderings.

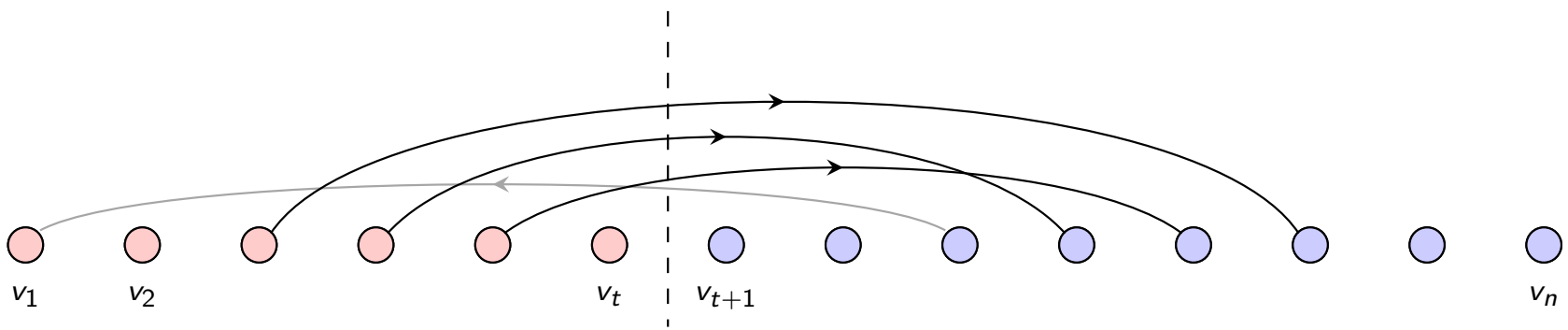


Cutwidth of a directed graph

- Let $\sigma = (v_1, v_2, \ldots, v_n)$ be an ordering of $V(D)$.
- *Width* of σ is equal to

$$\max_{t=0,1,\ldots,n} |E(\{v_1, \ldots, v_t\}, \{v_{t+1}, \ldots, v_n\})|.$$

- *Cutwidth* of D is minimum width among orderings.



Tournaments and semi-complete digraphs

- T is a **tournament**: for all different $u, v \in V(T)$, **exactly** one of arcs (u, v), (v, u) is present.

Tournaments and semi-complete digraphs

- T is a **tournament**: for all different $u, v \in V(T)$, **exactly** one of arcs (u, v), (v, u) is present.
- T is **semi-complete**: for all different $u, v \in V(T)$, **at least** one of arcs (u, v), (v, u) is present.

Tournaments and semi-complete digraphs

- T is a **tournament**: for all different $u, v \in V(T)$, **exactly** one of arcs (u, v), (v, u) is present.
- T is **semi-complete**: for all different $u, v \in V(T)$, **at least** one of arcs (u, v), (v, u) is present.
- **This talk**: only tournaments.

Why cutwidth of tournaments?

- Plays one of crucial roles in containment theory for tournaments of Chudnovsky, Fradkin, Kim, Scott, and Seymour.

Why cutwidth of tournaments?

- Plays one of crucial roles in containment theory for tournaments of Chudnovsky, Fradkin, Kim, Scott, and Seymour.
- (Chudnovsky, Fradkin, Seymour): If T does not admit D as an immersion, then $\mathbf{ctw}(T) \leqslant f(|D|)$ for some function f.

Why cutwidth of tournaments?

- Plays one of crucial roles in containment theory for tournaments of Chudnovsky, Fradkin, Kim, Scott, and Seymour.
- (Chudnovsky, Fradkin, Seymour): If T does not admit D as an immersion, then $\mathbf{ctw}(T) \leqslant f(|D|)$ for some function f.
- (Chudnovsky, Seymour): The immersion order is a well-quasi-ordering of tournaments.

Why cutwidth of tournaments?

- Plays one of crucial roles in containment theory for tournaments of Chudnovsky, Fradkin, Kim, Scott, and Seymour.
- (Chudnovsky, Fradkin, Seymour): If T does not admit D as an immersion, then $\mathbf{ctw}(T) \leqslant f(|D|)$ for some function f.
- (Chudnovsky, Seymour): The immersion order is a well-quasi-ordering of tournaments.
- (P.): An exact FPT algorithm working in $2^{\mathcal{O}(k)} \cdot n^2$ time, where k is the value of cutwidth.

New results

- A new FPT exact algorithm for cutwidth working in $\mathcal{O}^\star(2^{\mathcal{O}(\sqrt{k \log k})})$ time.

New results

- A new FPT exact algorithm for cutwidth working in $\mathcal{O}^\star(2^{\mathcal{O}(\sqrt{k \log k})})$ time.
- **Byproduct**: a new algorithm for Feedback Arc Set in Tournaments (FAST).

New results

- A new FPT exact algorithm for cutwidth working in $\mathcal{O}^\star(2^{\mathcal{O}(\sqrt{k \log k})})$ time.
- **Byproduct**: a new algorithm for Feedback Arc Set in Tournaments (FAST).
- FAST: delete at most k arcs of a given tournament T to obtain an acyclic digraph.

New results

- A new FPT exact algorithm for cutwidth working in $\mathcal{O}^\star(2^{\mathcal{O}(\sqrt{k \log k})})$ time.
- **Byproduct**: a new algorithm for Feedback Arc Set in Tournaments (FAST).
- FAST: delete at most k arcs of a given tournament T to obtain an acyclic digraph.
  - Equivalently, find an ordering of vertices with at most k forward arcs.

New results

- A new FPT exact algorithm for cutwidth working in $\mathcal{O}^\star(2^{\mathcal{O}(\sqrt{k \log k})})$ time.
- **Byproduct**: a new algorithm for Feedback Arc Set in Tournaments (FAST).
- FAST: delete at most k arcs of a given tournament T to obtain an acyclic digraph.
  - Equivalently, find an ordering of vertices with at most k forward arcs.
- **Running time**: $\mathcal{O}^\star(2^{c\sqrt{k}})$ for $c = \frac{2\pi}{\sqrt{3} \cdot \ln 2} \leqslant 5.24$.

New results

- A new FPT exact algorithm for cutwidth working in $\mathcal{O}^\star(2^{\mathcal{O}(\sqrt{k \log k})})$ time.
- **Byproduct**: a new algorithm for Feedback Arc Set in Tournaments (FAST).
- FAST: delete at most k arcs of a given tournament T to obtain an acyclic digraph.
  - Equivalently, find an ordering of vertices with at most k forward arcs.
- **Running time**: $\mathcal{O}^\star(2^{c\sqrt{k}})$ for $c = \frac{2\pi}{\sqrt{3} \cdot \ln 2} \leqslant 5.24$.
- Simpler than previous algorithms (Alon, Lokshtanov, Saurabh; Karpinski and Schudy; Feige)

Dynamic programming on cuts

- Consider the set of all 2^n partitions (X, Y) of $V(T)$ (*cuts*).

Dynamic programming on cuts

- Consider the set of all 2^n partitions (X, Y) of $V(T)$ (*cuts*).

k-cut

A partition (X, Y) of $V(T)$ is a *k-cut* if there is at most k arcs with head in Y and tail in X.

Dynamic programming on cuts

- Consider the set of all 2^n partitions (X, Y) of $V(T)$ (*cuts*).

k-cut

A partition (X, Y) of $V(T)$ is a *k-cut* if there is at most k arcs with head in Y and tail in X.

- Let $\mathcal{N}$ be the family of all k-cuts of T.

Dynamic programming on cuts

- Consider the set of all 2^n partitions (X, Y) of $V(T)$ (*cuts*).

k-cut

A partition (X, Y) of $V(T)$ is a *k-cut* if there is at most k arcs with head in Y and tail in X.

- Let $\mathcal{N}$ be the family of all k-cuts of T.
- (X_2, Y_2) *extends* (X_1, Y_1) if $X_2 = X_1 \cup \{v\}$ and, equivalently, $Y_2 = Y_1 \setminus \{v\}$.

Dynamic programming on cuts

- Let D be a digraph with $V(D) = \mathcal{N}$, where we put an arc from (X_1, Y_1) to (X_2, Y_2) if (X_2, Y_2) extends (X_1, Y_1).

Dynamic programming on cuts

- Let D be a digraph with $V(D) = \mathcal{N}$, where we put an arc from (X_1, Y_1) to (X_2, Y_2) if (X_2, Y_2) extends (X_1, Y_1).
- **Observation 1**. Paths from $(\emptyset, V(T))$ to $(V(T), \emptyset)$ correspond to orderings of width at most k.

Dynamic programming on cuts

- Let D be a digraph with $V(D) = \mathcal{N}$, where we put an arc from (X_1, Y_1) to (X_2, Y_2) if (X_2, Y_2) extends (X_1, Y_1).
- **Observation 1**. Paths from $(\emptyset, V(T))$ to $(V(T), \emptyset)$ correspond to orderings of width at most k.
- **Observation 2**. If we put weights $|E(X_1, \{v\})|$ on the arc from (X_1, Y_1) to (X_2, Y_2), then paths from $(\emptyset, V(T))$ to $(V(T), \emptyset)$ of total weight at most k correspond to orderings with at most k forward arcs.

Dynamic programming on cuts

- Let D be a digraph with $V(D) = \mathcal{N}$, where we put an arc from (X_1, Y_1) to (X_2, Y_2) if (X_2, Y_2) extends (X_1, Y_1).
- **Observation 1**. Paths from $(\emptyset, V(T))$ to $(V(T), \emptyset)$ correspond to orderings of width at most k.
- **Observation 2**. If we put weights $|E(X_1, \{v\})|$ on the arc from (X_1, Y_1) to (X_2, Y_2), then paths from $(\emptyset, V(T))$ to $(V(T), \emptyset)$ of total weight at most k correspond to orderings with at most k forward arcs.
- **Corollary**. If we are able to construct D fast, then we are able to solve both problems fast.

The strategy

- **Brute-force**: Check all the cuts, $\mathcal{O}^\star(2^n)$ time.

The strategy

- **Brute-force**: Check all the cuts, $\mathcal{O}^\star(2^n)$ time.
- **Our insight**: If T has cutwidth at most k, or admits a feedback arc set of size at most k, then $|\mathcal{N}|$ has size subexponential in terms of k.

The strategy

- **Brute-force**: Check all the cuts, $\mathcal{O}^\star(2^n)$ time.
- **Our insight**: If T has cutwidth at most k, or admits a feedback arc set of size at most k, then $|\mathcal{N}|$ has size subexponential in terms of k.
  - $|\mathcal{N}| = \mathcal{O}(2^{c\sqrt{k}} \cdot n)$ for tournaments admitting a feedback arc set of size at most k;

The strategy

- **Brute-force**: Check all the cuts, $\mathcal{O}^\star(2^n)$ time.
- **Our insight**: If T has cutwidth at most k, or admits a feedback arc set of size at most k, then $|\mathcal{N}|$ has size subexponential in terms of k.
  - $|\mathcal{N}| = \mathcal{O}(2^{c\sqrt{k}} \cdot n)$ for tournaments admitting a feedback arc set of size at most k;
  - $|\mathcal{N}| = \mathcal{O}(2^{\mathcal{O}(\sqrt{k \log k})} \cdot n)$ for tournaments with cutwidth at most k.

The strategy

- **Fact**: k-cuts may be enumerated with polynomial-time delay.

The strategy

- **Fact**: k-cuts may be enumerated with polynomial-time delay.
  - Standard branching.

The strategy

- **Fact**: k-cuts may be enumerated with polynomial-time delay.
  - Standard branching.
- **Algorithms**:

The strategy

- **Fact**: k-cuts may be enumerated with polynomial-time delay.
  - Standard branching.
- **Algorithms**:
  - Enumerate k-cuts with polynomial delay.

The strategy

- **Fact**: k-cuts may be enumerated with polynomial-time delay.
  - Standard branching.
- **Algorithms**:
  - Enumerate k-cuts with polynomial delay.
  - If we exceed the combinatorial bound, just terminate the enumeration and answer NO.

The strategy

- **Fact**: k-cuts may be enumerated with polynomial-time delay.
  - Standard branching.
- **Algorithms**:
  - Enumerate k-cuts with polynomial delay.
  - If we exceed the combinatorial bound, just terminate the enumeration and answer NO.
  - Otherwise, construct D and run DFS/Dijkstra.

Bounding $|\mathcal{N}|$ for small FAST

- Let T be a tournament that admits an ordering σ with at most k forward arcs, and let T' be the transitive tournament constructed from T by reversing these forward arcs.

Bounding $|\mathcal{N}|$ for small FAST

- Let T be a tournament that admits an ordering σ with at most k forward arcs, and let T' be the transitive tournament constructed from T by reversing these forward arcs.
- **Observation**: Every k-cut of T is a $2k$-cut of T'.

Bounding $|\mathcal{N}|$ for small FAST

- Let T be a tournament that admits an ordering σ with at most k forward arcs, and let T' be the transitive tournament constructed from T by reversing these forward arcs.
- **Observation**: Every k-cut of T is a $2k$-cut of T'.
- Hence, to bound the number of k-cuts of T it suffices to bound the number of $2k$-cuts of T'.

Bounding $|\mathcal{N}|$ for small FAST

- Let T be a tournament that admits an ordering σ with at most k forward arcs, and let T' be the transitive tournament constructed from T by reversing these forward arcs.
- **Observation**: Every k-cut of T is a $2k$-cut of T'.
- Hence, to bound the number of k-cuts of T it suffices to bound the number of $2k$-cuts of T'.
- How many k-cuts can a transitive tournament have?

Bounding $|\mathcal{N}|$ for a transitive tournament

- We relate k-cuts of T to *partition numbers*.

Bounding $|\mathcal{N}|$ for a transitive tournament

- We relate k-cuts of T to *partition numbers*.
- Partition number $p(k)$ is the number of different multisets of positive integers summing up to k. $(10 = 1 + 1 + 1 + 3 + 4)$

Bounding $|\mathcal{N}|$ for a transitive tournament

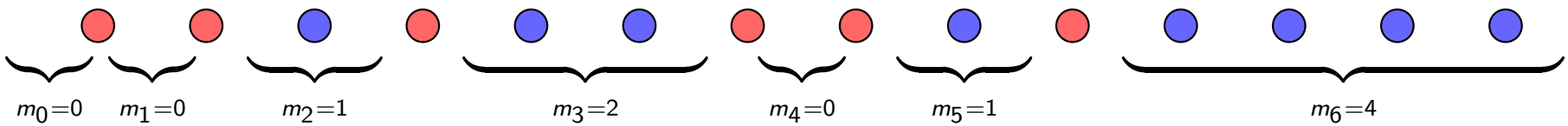
- We relate k-cuts of T to *partition numbers*.
- Partition number $p(k)$ is the number of different multisets of positive integers summing up to k. $(10 = 1 + 1 + 1 + 3 + 4)$
- $p(k) \leqslant \frac{A}{k+1} \cdot e^{C \cdot \sqrt{k}}$ for some constant A, where $C = \pi \cdot \sqrt{\frac{2}{3}}$ (Hardy, Ramanujan, 1919).

Bounding $|\mathcal{N}|$ for a transitive tournament

- Let σ be the ordering of $V(T)$ s.t. $(u, v) \in E(T)$ iff $u <_\sigma v$, and let (X, Y) be a k-cut. Let $|X| = a$.

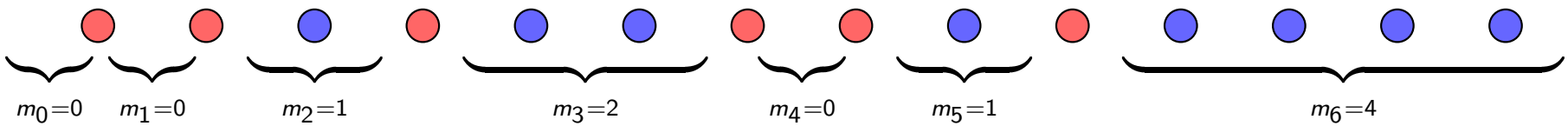
Bounding $|\mathcal{N}|$ for a transitive tournament

- Let σ be the ordering of $V(T)$ s.t. $(u, v) \in E(T)$ iff $u <_\sigma v$, and let (X, Y) be a k-cut. Let $|X| = a$.
- Let $m_0, m_1, \ldots, m_a$ be the numbers of elements of Y in the consecutive intervals between elements of X.



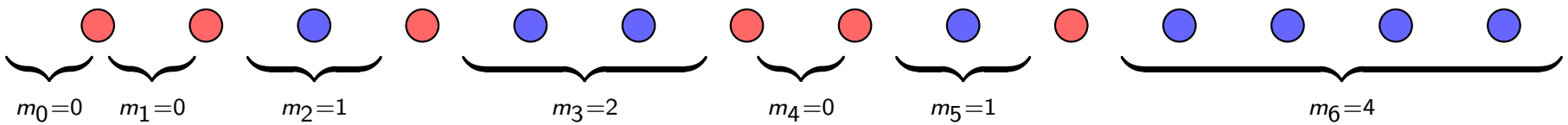
Bounding $|\mathcal{N}|$ for a transitive tournament

- Let σ be the ordering of $V(T)$ s.t. $(u, v) \in E(T)$ iff $u <_\sigma v$, and let (X, Y) be a k-cut. Let $|X| = a$.
- Let $m_0, m_1, \ldots, m_a$ be the numbers of elements of Y in the consecutive intervals between elements of X.
- A vertex counted in m_i is the head of exactly $a - i$ arcs from X to Y.



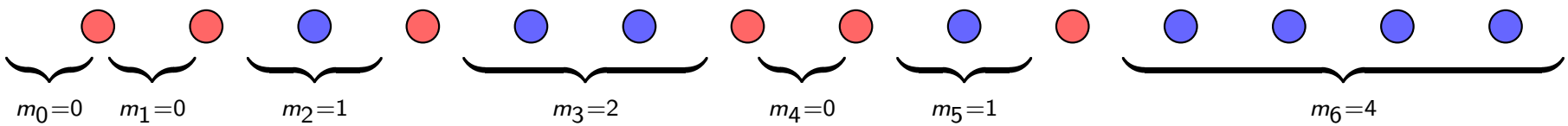
Bounding $|\mathcal{N}|$ for a transitive tournament

- Let σ be the ordering of $V(T)$ s.t. $(u, v) \in E(T)$ iff $u <_\sigma v$, and let (X, Y) be a k-cut. Let $|X| = a$.
- Let $m_0, m_1, \ldots, m_a$ be the numbers of elements of Y in the consecutive intervals between elements of X.
- A vertex counted in m_i is the head of exactly $a - i$ arcs from X to Y.
- Then $1 \cdot m_{a-1} + 2 \cdot m_{a-2} + 3 \cdot m_{a-3} + \ldots + a \cdot m_0 = k' \leqslant k$.



Bounding $|\mathcal{N}|$ for a transitive tournament

- Let σ be the ordering of $V(T)$ s.t. $(u, v) \in E(T)$ iff $u <_\sigma v$, and let (X, Y) be a k-cut. Let $|X| = a$.
- Let $m_0, m_1, \ldots, m_a$ be the numbers of elements of Y in the consecutive intervals between elements of X.
- A vertex counted in m_i is the head of exactly $a - i$ arcs from X to Y.
- Then $1 \cdot m_{a-1} + 2 \cdot m_{a-2} + 3 \cdot m_{a-3} + \ldots + a \cdot m_0 = k' \leqslant k$.
- $(X, Y) \to \{m_{a-1} \times 1, m_{a-2} \times 2, \ldots, m_0 \times a\}$, and (X, Y) can be easily reconstructed from the partition.



Bounding $|\mathcal{N}|$ for a transitive tournament

- The number of k-cuts with $|X| = a$ is at most $p(0) + p(1) + p(2) + \ldots + p(k) \leqslant A \cdot e^{C \cdot \sqrt{k}}$.

Bounding $|\mathcal{N}|$ for a transitive tournament

- The number of k-cuts with $|X| = a$ is at most $p(0) + p(1) + p(2) + \ldots + p(k) \leqslant A \cdot e^{C \cdot \sqrt{k}}$.
- Summing through different a gives n overhead.

Bounding $|\mathcal{N}|$ for a transitive tournament

- The number of k-cuts with $|X| = a$ is at most $p(0) + p(1) + p(2) + \ldots + p(k) \leqslant A \cdot e^{C \cdot \sqrt{k}}$.
- Summing through different a gives n overhead.
- For tournaments of small cutwidth we need to be a little bit smarter.

Bad pairs lemma

- Let (X, Y) be partition of $\{1, 2, \ldots, n\}$.

Bad pairs lemma

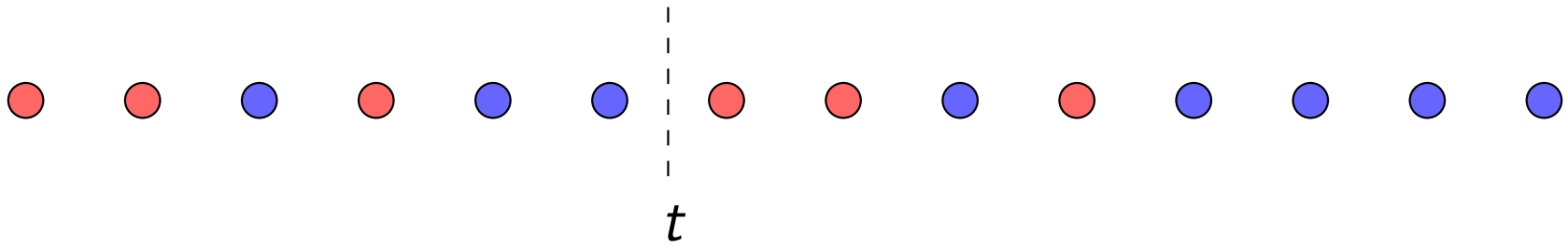
- Let (X, Y) be partition of $\{1, 2, \ldots, n\}$.
- A pair of indices (a, b) is *bad* if $a < b$, $a \in Y$ and $b \in X$.

Bad pairs lemma

- Let (X, Y) be partition of $\{1, 2, \ldots, n\}$.
- A pair of indices (a, b) is *bad* if $a < b$, $a \in Y$ and $b \in X$.

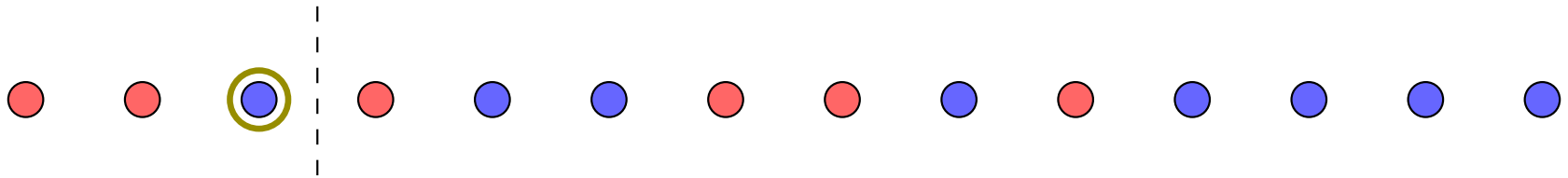
Lemma

Assume that for every t, the number of bad pairs (a, b) with $a \leqslant t < b$ is at most k. Then the total number of bad pairs is at most $k \cdot (1 + \ln k)$.



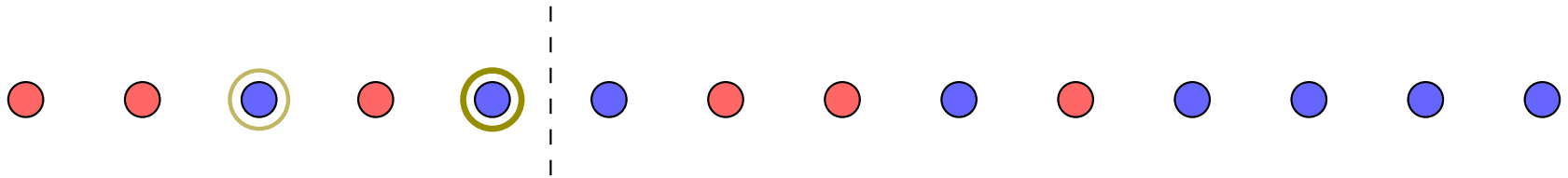
Bad pairs lemma: proof

- How many bad pairs can have the first element of Y at the first coordinate? At most k.



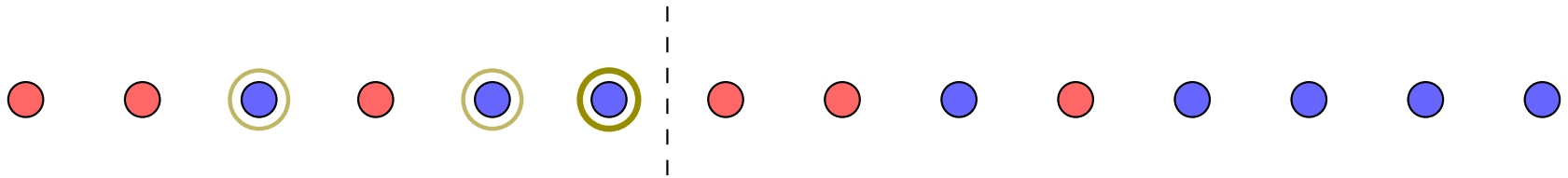
Bad pairs lemma: proof

- How many bad pairs can have the first element of Y at the first coordinate? At most k.
- How many bad pairs can have the second element of Y at the first coordinate? At most $k/2$.



Bad pairs lemma: proof

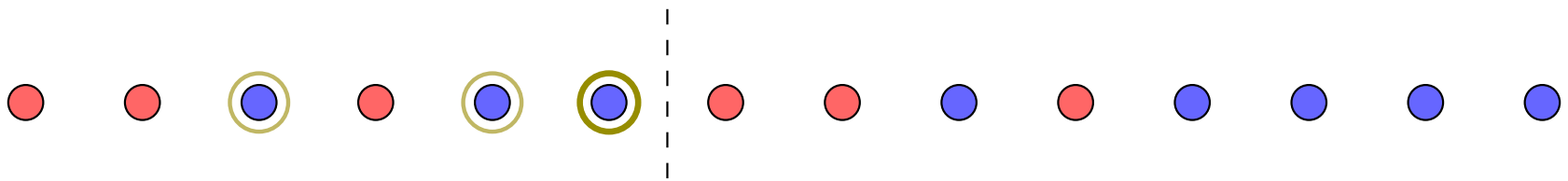
- How many bad pairs can have the first element of Y at the first coordinate? At most k.
- How many bad pairs can have the second element of Y at the first coordinate? At most $k/2$.



Bad pairs lemma: proof

- How many bad pairs can have the first element of Y at the first coordinate? At most k.
- How many bad pairs can have the second element of Y at the first coordinate? At most $k/2$.
- The number of bad pairs is hence bounded by

$$k \cdot (1 + \frac{1}{2} + \frac{1}{3} + \ldots + \frac{1}{k}) \leqslant k \cdot (1 + \ln k).$$



Bounding $|\mathcal{N}|$ for small cutwidth

- Let σ be an ordering of width at most k, and let T' be a transitive tournament s.t. $(b, a) \in E(T')$ iff $a <_\sigma b$.

Bounding $|\mathcal{N}|$ for small cutwidth

- Let σ be an ordering of width at most k, and let T' be a transitive tournament s.t. $(b, a) \in E(T')$ iff $a <_\sigma b$.
- Let (X, Y) be a k-cut of T. Note that arcs of T' from X to Y correspond to bad pairs.

Bounding $|\mathcal{N}|$ for small cutwidth

- Let σ be an ordering of width at most k, and let T' be a transitive tournament s.t. $(b, a) \in E(T')$ iff $a <_\sigma b$.
- Let (X, Y) be a k-cut of T. Note that arcs of T' from X to Y correspond to bad pairs.
- For every t, the number of arcs from X to Y in T' that cross the vertical line at t is at most $2k$:

Bounding $|\mathcal{N}|$ for small cutwidth

- Let σ be an ordering of width at most k, and let T' be a transitive tournament s.t. $(b, a) \in E(T')$ iff $a <_\sigma b$.
- Let (X, Y) be a k-cut of T. Note that arcs of T' from X to Y correspond to bad pairs.
- For every t, the number of arcs from X to Y in T' that cross the vertical line at t is at most $2k$:
  - at most k that were present in T;

Bounding $|\mathcal{N}|$ for small cutwidth

- Let σ be an ordering of width at most k, and let T' be a transitive tournament s.t. $(b, a) \in E(T')$ iff $a <_\sigma b$.
- Let (X, Y) be a k-cut of T. Note that arcs of T' from X to Y correspond to bad pairs.
- For every t, the number of arcs from X to Y in T' that cross the vertical line at t is at most $2k$:
  - at most k that were present in T;
  - at most k arcs crossing the line at t can have different directions in T and T'.

Bounding $|\mathcal{N}|$ for small cutwidth

- Let σ be an ordering of width at most k, and let T' be a transitive tournament s.t. $(b, a) \in E(T')$ iff $a <_\sigma b$.
- Let (X, Y) be a k-cut of T. Note that arcs of T' from X to Y correspond to bad pairs.
- For every t, the number of arcs from X to Y in T' that cross the vertical line at t is at most $2k$:
  - at most k that were present in T;
  - at most k arcs crossing the line at t can have different directions in T and T'.
- Bad Pairs Lemma $\Rightarrow$ (X, Y) is a $2k \cdot (1 + \ln 2k)$-cut of T'.

Bounding $|\mathcal{N}|$ for small cutwidth

- Let σ be an ordering of width at most k, and let T' be a transitive tournament s.t. $(b, a) \in E(T')$ iff $a <_\sigma b$.
- Let (X, Y) be a k-cut of T. Note that arcs of T' from X to Y correspond to bad pairs.
- For every t, the number of arcs from X to Y in T' that cross the vertical line at t is at most $2k$:
  - at most k that were present in T;
  - at most k arcs crossing the line at t can have different directions in T and T'.
- Bad Pairs Lemma $\Rightarrow$ (X, Y) is a $2k \cdot (1 + \ln 2k)$-cut of T'.
- The number of such cuts is bounded by $\mathcal{O}(2^{\mathcal{O}(\sqrt{k \log k})} \cdot n)$.

Conclusions

- We can also do a related Optimal Linear Arrangement problem in $\mathcal{O}^\star(2^{\mathcal{O}(k^{1/3}\sqrt{\log k})})$ time (see the paper).

Conclusions

- We can also do a related Optimal Linear Arrangement problem in $\mathcal{O}^\star(2^{\mathcal{O}(k^{1/3}\sqrt{\log k})})$ time (see the paper).
- All the presented algorithms may be implemented in linear time, i.e., the polynomial factor is n^2 (see the thesis).

Conclusions

- We can also do a related Optimal Linear Arrangement problem in $\mathcal{O}^\star(2^{\mathcal{O}(k^{1/3}\sqrt{\log k})})$ time (see the paper).
- All the presented algorithms may be implemented in linear time, i.e., the polynomial factor is n^2 (see the thesis).
- **Open**: Remove $\sqrt{\log k}$ from the exponent.

Conclusions

- We can also do a related Optimal Linear Arrangement problem in $\mathcal{O}^\star(2^{\mathcal{O}(k^{1/3}\sqrt{\log k})})$ time (see the paper).
- All the presented algorithms may be implemented in linear time, i.e., the polynomial factor is n^2 (see the thesis).
- **Open**: Remove $\sqrt{\log k}$ from the exponent.
- **Open**: A constant-factor approximation for cutwidth of a tournament.

Conclusions

- We can also do a related Optimal Linear Arrangement problem in $\mathcal{O}^\star(2^{\mathcal{O}(k^{1/3}\sqrt{\log k})})$ time (see the paper).
- All the presented algorithms may be implemented in linear time, i.e., the polynomial factor is n^2 (see the thesis).
- **Open**: Remove $\sqrt{\log k}$ from the exponent.
- **Open**: A constant-factor approximation for cutwidth of a tournament.
- **Thank you!**