Topological problems in tournaments

Michał Pilipczuk

Department of Informatics, University of Bergen, Norway

Dagstuhl Seminar on Bidimensional Structures,
19^{th} March 2013

Undirected graphs

- **Graph Minors**: Elegant mathematical theory that explains tractability of problems.

Undirected graphs

- **Graph Minors**: Elegant mathematical theory that explains tractability of problems.
- **Natural containment notion**: minor relation

Undirected graphs

- **Graph Minors**: Elegant mathematical theory that explains tractability of problems.
- **Natural containment notion**: minor relation
- **RS Theorem**: minor relation is a well-quasi-ordering.

Undirected graphs

- **Graph Minors**: Elegant mathematical theory that explains tractability of problems.
- **Natural containment notion**: minor relation
- **RS Theorem**: minor relation is a well-quasi-ordering.
  - Algorithmic applications

Undirected graphs

- **Graph Minors**: Elegant mathematical theory that explains tractability of problems.
- **Natural containment notion**: minor relation
- **RS Theorem**: minor relation is a well-quasi-ordering.
  - Algorithmic applications
- FPT algorithm for minor testing.

Undirected graphs

- **Graph Minors**: Elegant mathematical theory that explains tractability of problems.
- **Natural containment notion**: minor relation
- **RS Theorem**: minor relation is a well-quasi-ordering.
  - Algorithmic applications
- FPT algorithm for minor testing.
- **Measure of complexity**: treewidth.

Directed graphs

- Problems at each step.

Directed graphs

- Problems at each step.
- No natural containment notion (topological minor?, immersion?)

Directed graphs

- Problems at each step.
- No natural containment notion (topological minor?, immersion?)
- Containment problems are NP-hard for small, constant size graphs H [Fortune et al.]

Directed graphs

- Problems at each step.
- No natural containment notion (topological minor?, immersion?)
- Containment problems are NP-hard for small, constant size graphs H [Fortune et al.]
- No natural analogue of treewidth.

Tournaments

- **Idea**: find a subclass of directed graphs where a sound containment theory can be constructed.

Tournaments

- **Idea**: find a subclass of directed graphs where a sound containment theory can be constructed.
- The class of **tournaments** has been identified as such by Chudnovsky, Fradkin, Kim, Scott, and Seymour.

Tournaments

- **Idea**: find a subclass of directed graphs where a sound containment theory can be constructed.
- The class of **tournaments** has been identified as such by Chudnovsky, Fradkin, Kim, Scott, and Seymour.
  - The results hold also for a sligthly more general class of *semi-complete* digraphs.

Containment theory for tournaments

- We have all the necessary ingredients:

Containment theory for tournaments

- We have all the necessary ingredients:
  - Two natural containment notions that are WQO.

Containment theory for tournaments

- We have all the necessary ingredients:
  - Two natural containment notions that are WQO.
  - Two natural width parameters and corresponding decomposition-obstacle theorems.

Containment theory for tournaments

- We have all the necessary ingredients:
  - Two natural containment notions that are WQO.
  - Two natural width parameters and corresponding decomposition-obstacle theorems.
  - A number of FPT and XP algorithms.

Containment theory for tournaments

- We have all the necessary ingredients:
  - Two natural containment notions that are WQO.
  - Two natural width parameters and corresponding decomposition-obstacle theorems.
  - A number of FPT and XP algorithms.
- The theory is much simpler than the general undirected case.

Containment theory for tournaments

- We have all the necessary ingredients:
  - Two natural containment notions that are WQO.
  - Two natural width parameters and corresponding decomposition-obstacle theorems.
  - A number of FPT and XP algorithms.
- The theory is much simpler than the general undirected case.
  - Wonderful playground to work with algorithmic structural graph theory...

Containment theory for tournaments

- We have all the necessary ingredients:
  - Two natural containment notions that are WQO.
  - Two natural width parameters and corresponding decomposition-obstacle theorems.
  - A number of FPT and XP algorithms.
- The theory is much simpler than the general undirected case.
  - Wonderful playground to work with algorithmic structural graph theory...
  - ... that still has a lot of interesting open problems.

Containment theory for tournaments

- We have all the necessary ingredients:
  - Two natural containment notions that are WQO.
  - Two natural width parameters and corresponding decomposition-obstacle theorems.
  - A number of FPT and XP algorithms.
- The theory is much simpler than the general undirected case.
  - Wonderful playground to work with algorithmic structural graph theory...
  - ... that still has a lot of interesting open problems.
- **Goal**: make an overview of the topic.

Tournaments

- **Simple digraph**: no loops or multiple arcs.

Tournaments

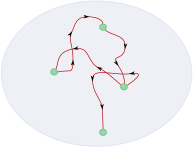
- **Simple digraph**: no loops or multiple arcs.
- **Tournament**: $\forall v, w \in V(T)$, **exactly** one of (v, w) and (w, v) belongs to $E(T)$.

Tournaments

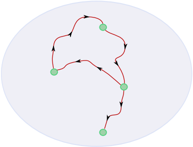
- **Simple digraph**: no loops or multiple arcs.
- **Tournament**: $\forall v, w \in V(T)$, **exactly** one of (v, w) and (w, v) belongs to $E(T)$.
- **Semi-complete digraph**: $\forall v, w \in V(T)$, **at least** one of (v, w) and (w, v) belongs to $E(T)$.

Containment notions

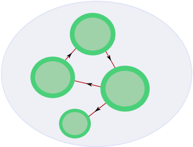
Immersion
vertices map to vertices,
arcs to arc-disjoint paths



Topological subgraph
vertices map to vertices,
arcs to vertex-disjoint paths



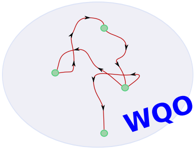
Minor
vertices map to vertex-disjoint strongly
connected subgraphs, arcs to arcs



Containment notions

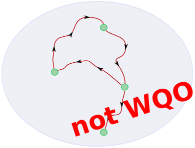
Immersion

vertices map to vertices,
arcs to arc-disjoint paths



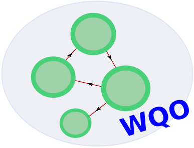
Topological subgraph

vertices map to vertices,
arcs to vertex-disjoint paths

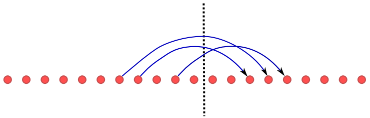
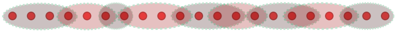


Minor

vertices map to vertex-disjoint strongly
connected subgraphs, arcs to arcs



Containment notions vs. width parameters

Immersion	**Cutwidth** 
Topological subgraph	**Pathwidth**
Minor	

Decomposition-obstacle theorems

Cutwidth vs. immersion

If T does not contain H as an immersion, then $\mathbf{ctw}(T) \leqslant f(|H|)$.

Pathwidth vs. top. subgraph/minor

If T does not contain H as a topological subgraph/minor, then $\mathbf{pw}(T) \leqslant f(|H|)$.

Decomposition-obstacle theorems

Cutwidth vs. immersion

If T does not contain H as an immersion, then $\mathbf{ctw}(T) \leqslant f(|H|)$.

Pathwidth vs. top. subgraph/minor

If T does not contain H as a topological subgraph/minor, then $\mathbf{pw}(T) \leqslant f(|H|)$.

- **Important**: the proofs prove decomposition-obstacle theorems

Decomposition-obstacle theorems

Cutwidth vs. immersion

If T does not contain H as an immersion, then $\mathbf{ctw}(T) \leqslant f(|H|)$.

Pathwidth vs. top. subgraph/minor

If T does not contain H as a topological subgraph/minor, then $\mathbf{pw}(T) \leqslant f(|H|)$.

- **Important**: the proofs prove decomposition-obstacle theorems
- that can be turned into approximation algorithms: the algorithm either gives a decomposition, or an appropriate combinatorial obstacle.

Overview of algorithms

Problem	CFKSS	FP, SODA 2013	P, STACS 2013	FP, arxiv
Cutwidth approx.	$O(n^3)$ (width $O(k^2)$)			
Cutwidth exact	$f(k) \cdot n^3$, non-u, non-c			
Pathwidth approx.	$n^{f(k)}$ time (width $O(k^2)$)			
Pathwidth exact	$n^{f(k)}$ non-u, non-c			
Immersion test	$f(\|H\|) \cdot n^3$			
T. subgraph/minor test	$n^{f(\|H\|)}$			

Overview of algorithms

Problem	CFKSS	FP, SODA 2013	P, STACS 2013	FP, arxiv
Cutwidth approx.	$O(n^3)$ (width $O(k^2)$)			
Cutwidth exact	$f(k) \cdot n^3$, non-u, non-c			
Pathwidth approx.	$n^{f(k)}$ time (width $O(k^2)$)	$2^{O(k \log k)} \cdot n^3 \log n$ (width $O(k^2)$)		
Pathwidth exact	$n^{f(k)}$ non-u, non-c	$f(k) \cdot n^3 \log n$ non-u, non-c		
Immersion test	$f(\|H\|) \cdot n^3$			
T. subgraph/minor test	$n^{f(\|H\|)}$	$f(\|H\|) \cdot n^3 \log n$		

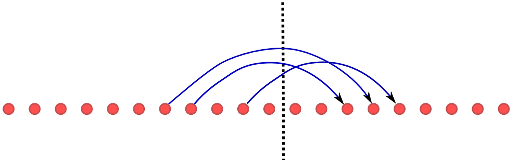
Overview of algorithms

Problem	CFKSS	FP, SODA 2013	P, STACS 2013	FP, arxiv
Cutwidth approx.	$O(n^3)$ (width $O(k^2)$)		$O(n^2)$ (width $O(k^2)$)	
Cutwidth exact	$f(k) \cdot n^3$, non-u, non-c		$2^{O(k)} \cdot n^2$	
Pathwidth approx.	$n^{f(k)}$ time (width $O(k^2)$)	$2^{O(k \log k)} \cdot n^3 \log n$ (width $O(k^2)$)	$O(kn^2)$ (width $7k$)	
Pathwidth exact	$n^{f(k)}$ non-u, non-c	$f(k) \cdot n^3 \log n$ non-u, non-c	$2^{O(k \log k)} \cdot n^2$	
Immersion test	$f(\|H\|) \cdot n^3$		$2^{O(\|H\|^2 \log \|H\|)} \cdot n^2$	
T. subgraph/minor test	$n^{f(\|H\|)}$	$f(\|H\|) \cdot n^3 \log n$	$2^{O(\|H\| \log \|H\|)} \cdot n^2$	

Overview of algorithms

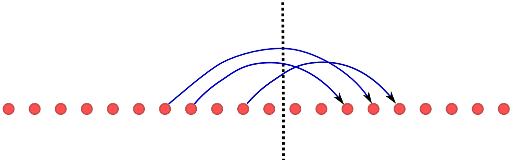
Problem	CFKSS	FP, SODA 2013	P, STACS 2013	FP, arxiv
Cutwidth approx.	$O(n^3)$ (width $O(k^2)$)		$O(n^2)$ (width $O(k^2)$)	
Cutwidth exact	$f(k) \cdot n^3$, non-u, non-c		$2^{O(k)} \cdot n^2$	$O^\star(2^{O(\sqrt{k \log k})})$
Pathwidth approx.	$n^{f(k)}$ time (width $O(k^2)$)	$2^{O(k \log k)} \cdot n^3 \log n$ (width $O(k^2)$)	$O(kn^2)$ (width $7k$)	
Pathwidth exact	$n^{f(k)}$ non-u, non-c	$f(k) \cdot n^3 \log n$ non-u, non-c	$2^{O(k \log k)} \cdot n^2$	
Immersion test	$f(\|H\|) \cdot n^3$		$2^{O(\|H\|^2 \log \|H\|)} \cdot n^2$	
T. subgraph/minor test	$n^{f(\|H\|)}$	$f(\|H\|) \cdot n^3 \log n$	$2^{O(\|H\| \log \|H\|)} \cdot n^2$	

Cutwidth



- Let $T = (V, E)$ be a tournament and $\sigma = (v_1, v_2, \ldots, v_n)$ be an ordering of V.

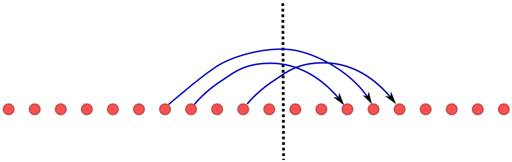
Cutwidth



- Let $T = (V, E)$ be a tournament and $\sigma = (v_1, v_2, \ldots, v_n)$ be an ordering of V.
- We define *width* of σ to be

$$\max_{0<i<n} |E(\{v_1, v_2, \ldots, v_i\}, \{v_{i+1}, v_{i+2}, \ldots, v_n\})|$$

Cutwidth

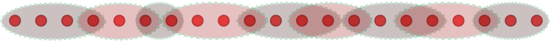


- Let $T = (V, E)$ be a tournament and $\sigma = (v_1, v_2, \ldots, v_n)$ be an ordering of V.
- We define *width* of σ to be

$$\max_{0<i<n} |E(\{v_1, v_2, \ldots, v_i\}, \{v_{i+1}, v_{i+2}, \ldots, v_n\})|$$

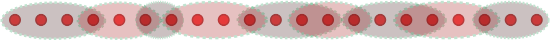
- *Cutwidth* of T, denoted $\mathbf{ctw}(T)$, is minimum width among orderings of V.

Pathwidth



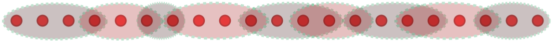
- A *path decomposition* $\mathcal{P}$ of $T = (V, E)$ is a sequence $[W_1, W_2, \ldots, W_p]$ of *bags* such that:

Pathwidth



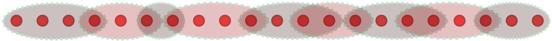
- A *path decomposition* $\mathcal{P}$ of $T = (V, E)$ is a sequence $[W_1, W_2, \ldots, W_p]$ of *bags* such that:
  - $W_i \subseteq V$ and $\bigcup_{i=1}^{p} W_i = V$;

Pathwidth



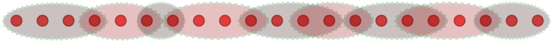
- A *path decomposition* $\mathcal{P}$ of $T = (V, E)$ is a sequence $[W_1, W_2, \ldots, W_p]$ of *bags* such that:
  - $W_i \subseteq V$ and $\bigcup_{i=1}^{p} W_i = V$;
  - for every vertex v, the set of bags containing v form an interval I_v in $\mathcal{P}$;

Pathwidth



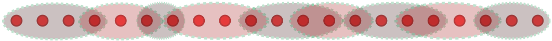
- A *path decomposition* $\mathcal{P}$ of $T = (V, E)$ is a sequence $[W_1, W_2, \ldots, W_p]$ of *bags* such that:
  - $W_i \subseteq V$ and $\bigcup_{i=1}^{p} W_i = V$;
  - for every vertex v, the set of bags containing v form an interval I_v in $\mathcal{P}$;
  - for every arc (v, w), intervals I_v and I_w either have nonempty intersection, or I_v appears later on $\mathcal{P}$ than I_w.

Pathwidth



- A *path decomposition* $\mathcal{P}$ of $T = (V, E)$ is a sequence $[W_1, W_2, \ldots, W_p]$ of *bags* such that:
  - $W_i \subseteq V$ and $\bigcup_{i=1}^{p} W_i = V$;
  - for every vertex v, the set of bags containing v form an interval I_v in $\mathcal{P}$;
  - for every arc (v, w), intervals I_v and I_w either have nonempty intersection, or I_v appears later on $\mathcal{P}$ than I_w.
- Width of $\mathcal{P}$ is $\max |W_i| - 1$.

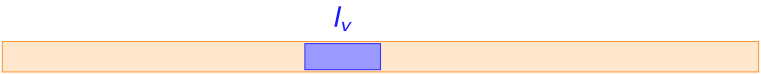
Pathwidth



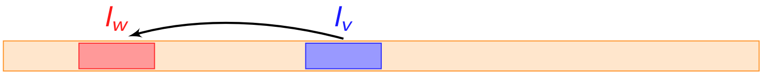
- A *path decomposition* $\mathcal{P}$ of $T = (V, E)$ is a sequence $[W_1, W_2, \ldots, W_p]$ of *bags* such that:
  - $W_i \subseteq V$ and $\bigcup_{i=1}^{p} W_i = V$;
  - for every vertex v, the set of bags containing v form an interval I_v in $\mathcal{P}$;
  - for every arc (v, w), intervals I_v and I_w either have nonempty intersection, or I_v appears later on $\mathcal{P}$ than I_w.
- Width of $\mathcal{P}$ is $\max |W_i| - 1$.
- Pathwidth of T, denoted $\mathbf{pw}(T)$, is minimum width among the path decompositions of T.

The degree ordering approach

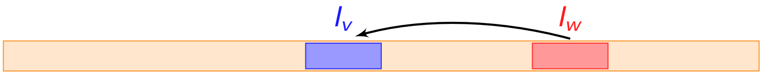
The degree ordering approach



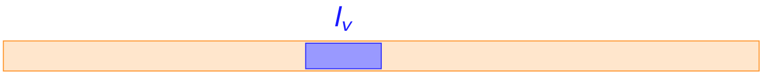
The degree ordering approach



The degree ordering approach

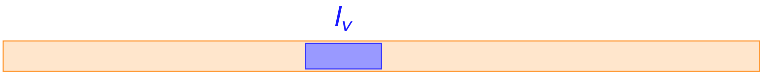


The degree ordering approach



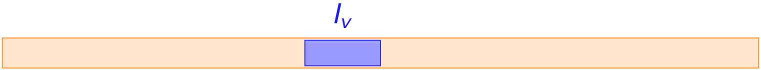
- Outdegree of a vertex approximately indicates, where the vertex should be placed in the optimal path decomposition.

The degree ordering approach



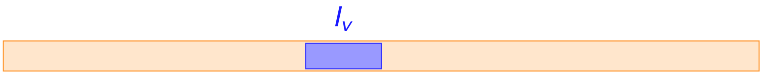
- Outdegree of a vertex approximately indicates, where the vertex should be placed in the optimal path decomposition.
  - Possible aberrations due to vertices sharing bags, but their number is limited by the width of the decomposition.

The degree ordering approach



- Outdegree of a vertex approximately indicates, where the vertex should be placed in the optimal path decomposition.
  - Possible aberrations due to vertices sharing bags, but their number is limited by the width of the decomposition.
- Ordering of vertices by the outdegree should give a crude idea of how the optimum decomposition looks.

The degree ordering approach

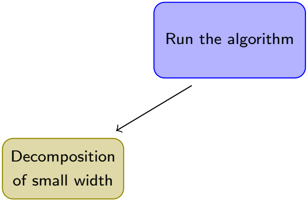


- Outdegree of a vertex approximately indicates, where the vertex should be placed in the optimal path decomposition.
  - Possible aberrations due to vertices sharing bags, but their number is limited by the width of the decomposition.
- Ordering of vertices by the outdegree should give a crude idea of how the optimum decomposition looks.
- In particular, there should not be many vertices with similar outdegrees.

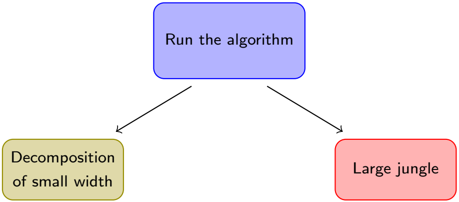
Obstacles: approach of Fradkin and Seymour

Run the algorithm

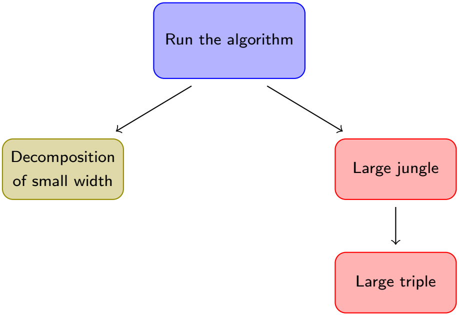
Obstacles: approach of Fradkin and Seymour



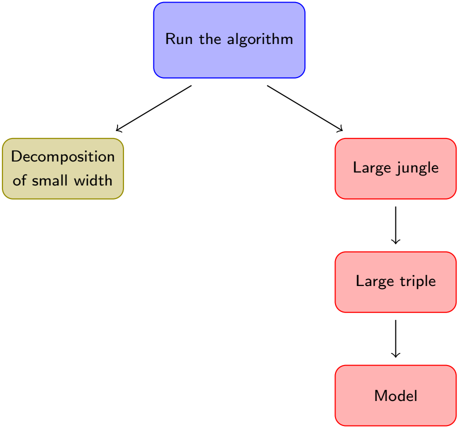
Obstacles: approach of Fradkin and Seymour



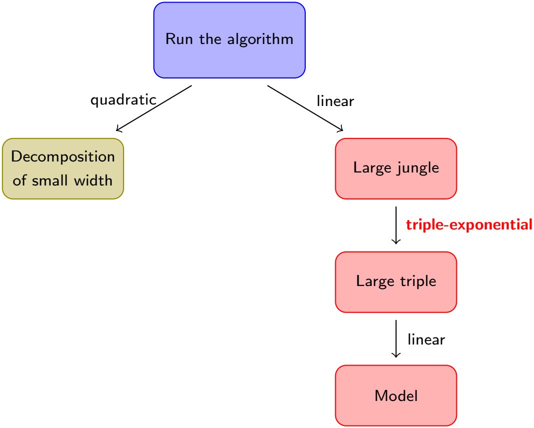
Obstacles: approach of Fradkin and Seymour



Obstacles: approach of Fradkin and Seymour



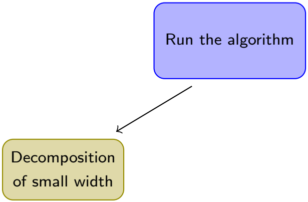
Obstacles: approach of Fradkin and Seymour



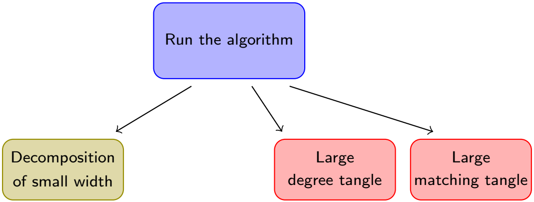
Obstacles: degree ordering

Run the algorithm

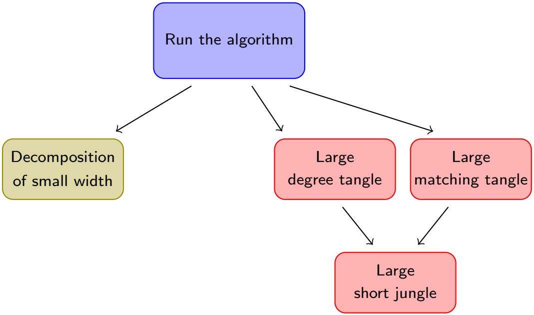
Obstacles: degree ordering



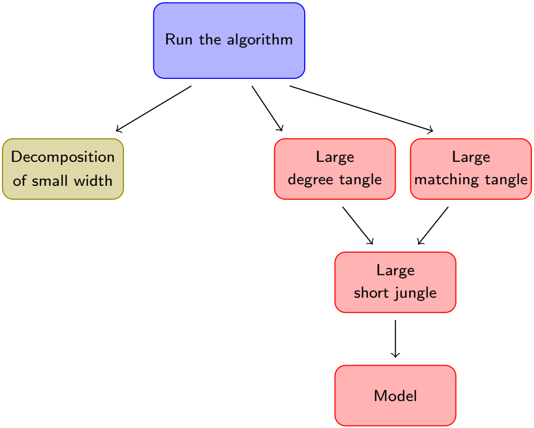
Obstacles: degree ordering



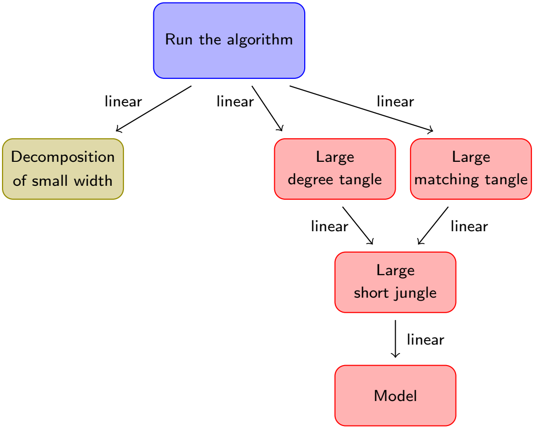
Obstacles: degree ordering



Obstacles: degree ordering



Obstacles: degree ordering



Open problem #1

- We would like to be able to perform an irrelevant vertex rule on the obtained combinatorial obstacle.

Open problem #1

- We would like to be able to perform an irrelevant vertex rule on the obtained combinatorial obstacle.
  - **Example**: ROOTED IMMERSION

Open problem #1

- We would like to be able to perform an irrelevant vertex rule on the obtained combinatorial obstacle.
  - **Example**: ROOTED IMMERSION
- A triple is very robust, however you need to dig deeply to get it.

Open problem #1

- We would like to be able to perform an irrelevant vertex rule on the obtained combinatorial obstacle.
  - **Example**: ROOTED IMMERSION
- A triple is very robust, however you need to dig deeply to get it.
- Short jungles or degree/matching tangles: not clear how to use.

Open problem #1

- We would like to be able to perform an irrelevant vertex rule on the obtained combinatorial obstacle.
  - **Example**: Rooted Immersion
- A triple is very robust, however you need to dig deeply to get it.
- Short jungles or degree/matching tangles: not clear how to use.
- **Open problem**: find better obstacles, on which irrelevant vertex rules can be employed.

Open problem #2

- We have containment theory for tournaments...

Open problem #2

- We have containment theory for tournaments...
- and a number of subexponential-time FPT algorithms (e.g., FAST).

Open problem #2

- We have containment theory for tournaments...
- and a number of subexponential-time FPT algorithms (e.g., FAST).
- **Open problem**: Is it possible to develop a meta-explanation of subexponential-time FPT algorithms on tournaments, similarly to bidimensionality?

Open problem #3

- Fixed-parameter tractability of VERTEX-DISJOINT PATHS on tournaments is still open!

Open problem #3

- Fixed-parameter tractability of VERTEX-DISJOINT PATHS on tournaments is still open!

VERTEX-DISJOINT PATHS

Input: a tournament T with k pairs of terminals $(s_1, t_1), (s_2, t_2), \ldots, (s_k, t_k)$

Question: Is there a family of vertex-disjoint paths $P_1, P_2, \ldots, P_k$, s.t. P_i leads from s_i to t_i?

Open problem #3

- Fixed-parameter tractability of VERTEX-DISJOINT PATHS on tournaments is still open!

VERTEX-DISJOINT PATHS

Input: a tournament T with k pairs of terminals $(s_1, t_1), (s_2, t_2), \ldots, (s_k, t_k)$

Question: Is there a family of vertex-disjoint paths $P_1, P_2, \ldots, P_k$, s.t. P_i leads from s_i to t_i?

- Problem with the current approach: irrelevant vertex rule on a triple fails.

Open problems

- Finding better obstacles for irrelevant vertex rules.
- Bidimensionality for tournaments?
- FPT algorithm for vertex-disjoint paths.