The basic problem

**Minor/Topological Subgraph Containment**

*Input*: Undirected graphs $H$ and $G$.

*Question*: Is $H$ contained in $G$ as a minor/topological subgraph?
The algorithms

- **First goal [XP]:**
  Polynomial time algorithm for every fixed $H$, e.g., $O(|G|^{|H|})$. 

For **Minor Containment**, $f(|H|)|V(G)|^3$ algorithm [Robertson and Seymour].

For **Topological Subgraph Containment**, $f(|H|)|V(G)|^3$ algorithm [Grohe et al.].
The algorithms

- **First goal [XP]:**
  Polynomial time algorithm for every fixed $H$, e.g., $O(|G|^{|H|})$.

- **Second goal [FPT]:**
  Polynomial time algorithm for every fixed $H$, with exponent independent of $H$, i.e., $f(|H|)|G|^c$ for some (small) constant $c$. 
The algorithms

- **First goal** [XP]:
  Polynomial time algorithm for every fixed $H$, e.g., $O(|G|^{|H|})$.

- **Second goal** [FPT]:
  Polynomial time algorithm for every fixed $H$, with exponent independent of $H$, i.e., $f(|H|)|G|^c$ for some (small) constant $c$.

- For **Minor Containment**, $f(|H|)|V(G)|^3$ algorithm [Robertson and Seymour].
The algorithms

- **First goal [XP]:** Polynomial time algorithm for every fixed \( H \), e.g., \( O(|G|^{|H|}) \).

- **Second goal [FPT]:** Polynomial time algorithm for every fixed \( H \), with exponent independent of \( H \), i.e., \( f(|H|)|G|^c \) for some (small) constant \( c \).

- **For Minor Containment,** \( f(|H|)|V(G)|^3 \) algorithm [Robertson and Seymour].

- **For Topological Subgraph Containment,** \( f(|H|)|V(G)|^3 \) algorithm [Grohe et al.].
Directed world

- We consider topological subgraph and immersion relations.
Directed world

We consider topological subgraph and immersion relations.

- **Topological subgraph**: vertices of $H$ map to different vertices in $G$, and arcs in $H$ map to vertex-disjoint directed paths between corresponding images in $G$.
- **Immersion**: vertices of $H$ map to different vertices in $G$, and arcs in $H$ map to edge-disjoint directed paths between corresponding images in $G$. 

NP-hard in general setting even for small, fixed subgraphs \cite{Fortune}. For acyclic digraphs there is an XP algorithm, but FPT is unlikely \cite{Slivkins}. 

Fedor Fomin and Michał Pilipczuk

Jungles, bundles, and fixed parameter tractability

4/21
Directed world

- We consider topological subgraph and immersion relations.
  - **Topological subgraph:** vertices of $H$ map to different vertices in $G$, and arcs in $H$ map to *vertex-disjoint* directed paths between corresponding images in $G$.
  - **Immersion:** vertices of $H$ map to different vertices in $G$, and arcs in $H$ map to *edge-disjoint* directed paths between corresponding images in $G$.
Directed world

- We consider topological subgraph and immersion relations.
  - **Topological subgraph**: vertices of $H$ map to different vertices in $G$, and arcs in $H$ map to *vertex-disjoint* directed paths between corresponding images in $G$.
  - **Immersion**: vertices of $H$ map to different vertices in $G$, and arcs in $H$ map to *edge-disjoint* directed paths between corresponding images in $G$.

- **NP-hard** in general setting even for small, fixed subgraphs $H$ [Fortune et al.].
Directed world

- We consider topological subgraph and immersion relations.
  - **Topological subgraph**: vertices of $H$ map to different vertices in $G$, and arcs in $H$ map to *vertex-disjoint* directed paths between corresponding images in $G$.
  - **Immersion**: vertices of $H$ map to different vertices in $G$, and arcs in $H$ map to *edge-disjoint* directed paths between corresponding images in $G$.

- NP-hard in general setting even for small, fixed subgraphs $H$ [Fortune et al.].
- For acyclic digraphs there is an XP algorithm, but FPT is unlikely [Slivkins].
Tournament world

- Tournaments identified as a class of digraphs where a sound containment theory can be constructed [Chudnovsky, Fradkin, Kim, Scott, and Seymour].
Tournament world

- Tournaments identified as a class of digraphs where a sound containment theory can be constructed [Chudnovsky, Fradkin, Kim, Scott, and Seymour].
- In fact all the results hold for more general semi-complete digraphs, but for simplicity assume tournaments in this talk.
Tournament world

- Tournaments identified as a class of digraphs where a sound containment theory can be constructed [Chudnovsky, Fradkin, Kim, Scott, and Seymour].
- In fact all the results hold for more general semi-complete digraphs, but for simplicity assume tournaments in this talk.
- A number of FPT algorithms (immersion) and XP algorithms (topological containment).
Tournament world

- Tournaments identified as a class of digraphs where a sound containment theory can be constructed [Chudnovsky, Fradkin, Kim, Scott, and Seymour].
- In fact all the results hold for more general semi-complete digraphs, but for simplicity assume tournaments in this talk.
- A number of FPT algorithms (immersion) and XP algorithms (topological containment).

**Our goal:** Refine the running time of algorithms around the topological subgraph problem from XP to FPT.
XP algorithm of Fradkin and Seymour

$H$, $T$

Path decomposition of width $O(f(|H|)^2)$

Run dynamic programming $f(|H|)$-jungle $|H|$-triple

Answer YES
Is pathwidth of $T$ larger than $f(|H|)$?
XP algorithm of Fradkin and Seymour

Is pathwidth of $T$ larger than $f(|H|)$?

Path decomposition of width $O(f(|H|)^2)$
Is pathwidth of $T$ larger than $f(|H|)$?

Path decomposition of width $O(f(|H|)^2)$

Run dynamic programming
The XP algorithm of Fradkin and Seymour

Is pathwidth of $T$ larger than $f(|H|)$?

- Path decomposition of width $O(f(|H|)^2)$
- $f(|H|)$-jungle

Run dynamic programming

$H$, $T$
Is pathwidth of $T$ larger than $f(|H|)$?

- Path decomposition of width $O(f(|H|)^2)$
- $f(|H|)$-jungle

- Run dynamic programming
- $|H|$-triple
Is pathwidth of $T$ larger than $f(|H|)$?

- Path decomposition of width $O(f(|H|)^2)$
  - Run dynamic programming
    - Answer YES
- $f(|H|)$-jungle
  - $|H|$-triple
    - Answer YES
Is pathwidth of $T$ larger than $f(|H|)$?

- Path decomposition of width $O(f(|H|)^2)$
  - Run dynamic programming
  - $|H|$-triple
  - Answer YES

- $f(|H|)$-jungle

$H, T$
Is pathwidth of $T$ larger than $f(|H|)$?

- Path decomposition of width $O(f(|H|)^2)$
- $f(|H|)$-jungle
  - Run dynamic programming
  - $|H|$-triple
  - Answer YES
XP algorithm of Fradkin and Seymour

- Is pathwidth of $T$ larger than $f(|H|)$?
- Path decomposition of width $O(f(|H|)^2)$
- Run dynamic programming
  - $f(|H|)$-jungle
  - $|H|$-triple
  - Answer YES

FPT
The main result

FPT approximation of pathwidth of a tournament

There exists an algorithm, which given a tournament $T$ on $n$ vertices and an integer $k$, outputs either a path decomposition of $T$ of width at most $4k^2 + 7k$, or a $k$-jungle in $T$, in time complexity $2^{O(k \log k)} \cdot n^3 \log n$. 
Separation

- \((A, B)\) is a separation of order \(k\) if

\(A \cup B = V(T), |A \setminus B| = k;\) and there are no edges from \(A \setminus B\) to \(B \setminus A\).

Separations \((A, B)\) and \((C, D)\) do not cross if \(A \subseteq C\) and \(D \subseteq B\) or vice versa.
Separation

- $(A, B)$ is a separation of order $k$ if
  - $A \cup B = V(T)$, $|A \cap B| = k$;

- Separations $(A, B)$ and $(C, D)$ do not cross if $A \subseteq C$ and $D \subseteq B$ or vice versa.
Separation

\((A, B)\) is a separation of order \(k\) if

- \(A \cup B = V(T), |A \cap B| = k;\)
- and there are no edges from \(A \setminus B\) to \(B \setminus A\).
Separation

- \((A, B)\) is a separation of order \(k\) if
  - \(A \cup B = V(T), |A \cap B| = k\);
  - and there are no edges from \(A \setminus B\) to \(B \setminus A\).

- Separations \((A, B)\) and \((C, D)\) do not cross if \(A \subseteq C\) and \(D \subseteq B\) or vice versa.
Pathwidth

- **Path decomposition** of $T$ is a sequence of bags $[W_1, W_2, \ldots, W_h]$ such that

  $\bigcup W_i = V(T)$;
  
  $W_i \setminus W_k \subseteq W_j$ for $i < j < k$;
  
  for every edge $(u, v)$, either $u, v \in W_i$ for some $i$, or $u \in W_i$ and $v \in W_j$ for some $i > j$.

Width of $[W_1, W_2, \ldots, W_h]$ is $\max |W_i| - 1$.

$pw(T)$ is the minimum possible width of a decomposition.
Path decomposition of $T$ is a sequence of bags $[W_1, W_2, \ldots, W_h]$ such that

- $\bigcup W_i = V(T)$;
Path decomposition of $T$ is a sequence of bags $[W_1, W_2, \ldots, W_h]$ such that

- $\bigcup W_i = V(T);$ 
- $W_i \cap W_k \subseteq W_j$ for $i < j < k;$

Path decomposition of $T$ is a sequence of bags $[W_1, W_2, \ldots, W_h]$ such that

- $\bigcup W_i = V(T);$ 
- $W_i \cap W_k \subseteq W_j$ for $i < j < k;$ 
- for every edge $(u, v),$ either $u, v \in W_i$ for some $i,$ or $u \in W_i, v \in W_j$ for some $i > j.$
Path width

**Path decomposition** of $T$ is a sequence of bags $[W_1, W_2, \ldots, W_h]$ such that

- $\bigcup W_i = V(T)$;
- $W_i \cap W_k \subseteq W_j$ for $i < j < k$;
- for every edge $(u, v)$, either $u, v \in W_i$ for some $i$, or $u \in W_i, v \in W_j$ for some $i > j$.

**Width** of $[W_1, W_2, \ldots, W_h]$ is $\max |W_i| - 1$. 

$pw(T)$ is the minimum possible width of a decomposition.
Pathwidth

- **Path decomposition** of $T$ is a sequence of bags $[W_1, W_2, \ldots, W_h]$ such that
  - $\bigcup W_i = V(T)$;
  - $W_i \cap W_k \subseteq W_j$ for $i < j < k$;
  - for every edge $(u, v)$, either $u, v \in W_i$ for some $i$, or $u \in W_i, v \in W_j$ for some $i > j$.

- **Width** of $[W_1, W_2, \ldots, W_h]$ is $\max |W_i| - 1$.

- $\text{pw}(T)$ is the minimum possible width of a decomposition.
Approximation algorithm of Fradkin and Seymour

- We greedily incorporate separations of larger and larger order up to order $k$, constructing a cross-free family of separations called a bundle.
Approximation algorithm of Fradkin and Seymour

- We greedily incorporate separations of larger and larger order up to order $k$, constructing a cross-free family of separations called a bundle.
- Each new separation has to satisfy certain technical conditions.
Approximation algorithm of Fradkin and Seymour

• We greedily incorporate separations of larger and larger order up to order $k$, constructing a cross-free family of separations called a bundle.

• Each new separation has to satisfy certain technical conditions.

• Having a maximum bundle we obtain some path decomposition:
Approximation algorithm of Fradkin and Seymour

- We greedily incorporate separations of larger and larger order up to order $k$, constructing a cross-free family of separations called a **bundle**.
- Each new separation has to satisfy certain technical conditions.
- Having a maximum bundle we obtain some path decomposition:
  - **Small width**: we are happy.
Approximation algorithm of Fradkin and Seymour

- We greedily incorporate separations of larger and larger order up to order $k$, constructing a cross-free family of separations called a bundle.
- Each new separation has to satisfy certain technical conditions.
- Having a maximum bundle we obtain some path decomposition:
  - **Small width**: we are happy.
  - **Large width**: a $k$-jungle due to maximality of the bundle.
Algorithm: overview
Algorithm: overview
Algorithm: overview
Algorithm: overview
Algorithm: overview
Algorithm: overview
Algorithm: overview
Algorithm: overview
Algorithm: overview
Algorithm: overview
Algorithm: overview
Algorithm: overview
Algorithm: overview
The new separation cannot be 'close' to the neighbouring ones.
The new separation cannot be 'close' to the neighbouring ones.

There have to be at least \( k|a_1 - b|, k|a_2 - b| \) vertices in between, respectively.
After guessing what exactly happens on neighbouring separators \(2^{O(k)}\) guesses), we have the following problem:

**Tournament Balanced Separator**

**Input:** A tournament \(S\) on \(n\) vertices; disjoint sets \(X, Y \subseteq V(S)\); integers \(a, b, c\).

**Question:** Does there exist a separation \((C, D)\) of \(S\) such that

- \(|C \cap D| \leq b\);
- \(X \subseteq C \setminus D, Y \subseteq D \setminus C\);
- \(|(C \setminus D) \setminus X| \geq a\) and \(|(D \setminus C) \setminus Y| \geq c|\)?
Subproblem

\[(a, b, c)\]

\[\geq a \quad \geq c\]

\[\leq b\]
Finding balanced separator

- Original implementation via brute-force enumeration of all $O(n^b)$ candidate separators.
Finding balanced separator

- Original implementation via brute-force enumeration of all $O(n^b)$ candidate separators.
- We show an algorithm working in time
  
  $$2^{O(\min(a+c,b) \log(a+b+c))} \cdot n^2 \log n.$$
Finding balanced separator

- Original implementation via brute-force enumeration of all $O(n^b)$ candidate separators.

- We show an algorithm working in time

$$2^{O\left(\min(a+c,b)\log(a+b+c)\right)} \cdot n^2 \log n.$$ 

- As $a, c = O(k^2)$ and $b \leq k$, this gives $2^{O(k \log k)} \cdot n^2 \log n$ for inserting one separation.
Finding balanced separator

- Original implementation via brute-force enumeration of all $O(n^b)$ candidate separators.
- We show an algorithm working in time
  \[ 2^{O(\min(a+c,b) \log(a+b+c))} \cdot n^2 \log n. \]

- As $a, c = O(k^2)$ and $b \leq k$, this gives $2^{O(k \log k)} \cdot n^2 \log n$ for inserting one separation.
- Now we present a randomized version; derandomization via splitters.
Finding balanced separator

- Original implementation via brute-force enumeration of all $O(n^b)$ candidate separators.
- We show an algorithm working in time

$$2^{O(\min(a+c,b) \log(a+b+c))} \cdot n^2 \log n.$$ 

- As $a, c = O(k^2)$ and $b \leq k$, this gives $2^{O(k \log k)} \cdot n^2 \log n$ for inserting one separation.
- Now we present a randomized version; derandomization via splitters.
- Assume that a solution exists and fix one solution $(C, D)$. 

Color coding

- Independently at random color every nonterminal white or black, with probability $1/2$. 

Examine the event:

$C \setminus D$ get black,

at least $a$ nonterminals from $C \setminus (X \cup D)$ get white,

at least $c$ nonterminals from $D \setminus (Y \cup C)$ get white.

Probability: at least $2 - (a + b + c)$.

By tweaking $1/2$ we get $2 - O(\min(a + c, b) \log(a + b + c))$.

By repeating the experiment $2O(\min(a + c, b) \log(a + b + c))$ times, with constant probability we hit the event.

Finding a solution respecting the coloring is polynomial time solvable.
Independently at random color every nonterminal white or black, with probability 1/2.

Examine the event:
**Color coding**

- Independently at random color every nonterminal white or black, with probability $1/2$.
- Examine the event:
  - $C \cap D$ get black,
Color coding

- Independently at random color every nonterminal white or black, with probability $1/2$.
- Examine the event:
  - $C \cap D$ get black,
  - at least $a$ nonterminals from $C \setminus (X \cup D)$ get white,
Color coding

- Independently at random color every nonterminal white or black, with probability $1/2$.
- Examine the event:
  - $C \cap D$ get black,
  - at least $a$ nonterminals from $C \setminus (X \cup D)$ get white,
  - at least $c$ nonterminals from $D \setminus (Y \cup C)$ get white.

Probability: at least $2 - (a + b + c)$. By tweaking $1/2$ we get $2 - O\left(\min(a + c, b) \log(a + b + c)\right)$. By repeating the experiment $2 O\left(\min(a + c, b) \log(a + b + c)\right)$ times, with constant probability we hit the event.

Finding a solution respecting the coloring is polynomial time solvable.
Color coding

- Independently at random color every nonterminal white or black, with probability $1/2$.
- Examine the event:
  - $C \cap D$ get black,
  - at least $a$ nonterminals from $C \setminus (X \cup D)$ get white,
  - at least $c$ nonterminals from $D \setminus (Y \cup C)$ get white.

**Probability**: at least $2^{-(a+b+c)}$. 
Color coding

- Independently at random color every nonterminal white or black, with probability $\frac{1}{2}$.
- Examine the event:
  - $C \cap D$ get black,
  - at least $a$ nonterminals from $C \setminus (X \cup D)$ get white,
  - at least $c$ nonterminals from $D \setminus (Y \cup C)$ get white.

**Probability:** at least $2^{-(a+b+c)}$.

**By tweaking** $\frac{1}{2}$ we get $2^{-O(\min(a+c, b) \log(a+b+c))}$.
Color coding

- Independently at random color every nonterminal white or black, with probability 1/2.
- Examine the event:
  - $C \cap D$ get black,
  - at least $a$ nonterminals from $C \setminus (X \cup D)$ get white,
  - at least $c$ nonterminals from $D \setminus (Y \cup C)$ get white.

**Probability:** at least $2^{-(a+b+c)}$.

By tweaking 1/2 we get $2^{-O(\min(a+c,b) \log(a+b+c))}$.

By repeating the experiment $2^{O(\min(a+c,b) \log(a+b+c))}$ times, with constant probability we hit the event.
Independently at random color every nonterminal white or black, with probability $1/2$.

Examine the event:
- $C \cap D$ get black,
- at least $a$ nonterminals from $C \setminus (X \cup D)$ get white,
- at least $c$ nonterminals from $D \setminus (Y \cup C)$ get white.

**Probability**: at least $2^{- (a+b+c)}$.

By tweaking $1/2$ we get $2^{-O(\min(a+c,b) \log(a+b+c))}$.

By repeating the experiment $2^{O(\min(a+c,b) \log(a+b+c))}$ times, with constant probability we hit the event.

**Finding a solution respecting the coloring is polynomial time solvable.**
Corollaries of FPT approximation of pathwidth of a tournament:
Corollaries

- Corollaries of FPT approximation of pathwidth of a tournament:
  - Testing topological subgraph containment is FPT.
Corollaries of FPT approximation of pathwidth of a tournament:

- Testing topological subgraph containment is FPT.
- Computing vertex deletion distance to any immersion-closed class of tournaments is FPT.
Corollaries of FPT approximation of pathwidth of a tournament:

- Testing topological subgraph containment is FPT.
- Computing vertex deletion distance to any immersion-closed class of tournaments is FPT.
- Follows from the fact that immersion relation is a well-quasi order on tournaments [Chudnovsky, Seymour].
More corollaries

- Testing rooted immersion in tournaments is FPT.
More corollaries

- Testing rooted immersion in tournaments is FPT.
- **Rooted**: some vertices of $H$ may have prescribed images.
More corollaries

- Testing rooted immersion in tournaments is FPT.
- **Rooted**: some vertices of $H$ may have prescribed images.
- Generalizes **Edge-disjoint Paths**.
More corollaries

- Testing rooted immersion in tournaments is FPT.
- **Rooted**: some vertices of $H$ may have prescribed images.
- Generalizes **Edge-disjoint Paths**.
- We need additional irrelevant vertex technique in a triple.
More corollaries

- Testing rooted immersion in tournaments is FPT.
- **Rooted**: some vertices of $H$ may have prescribed images.
- Generalizes **Edge-disjoint Paths**.
- We need additional irrelevant vertex technique in a triple.
  - Quite technical.
More corollaries

- Testing rooted immersion in tournaments is FPT.
- **Rooted**: some vertices of $H$ may have prescribed images.
- Generalizes **Edge-disjoint Paths**.
- We need additional irrelevant vertex technique in a triple.
  - Quite technical.
- **FPT** was already known for closely related **Rooted Infusion**.
Later results

- P., *Computing cutwidth and pathwidth of semi-complete digraphs via degree orderings*, STACS 2013
Later results

- P., *Computing cutwidth and pathwidth of semi-complete digraphs via degree orderings*, STACS 2013
- Completely new approach.
Later results

- P., *Computing cutwidth and pathwidth of semi-complete digraphs via degree orderings*, STACS 2013
- Completely new approach.
- 7-approximation of pathwidth in $O(kn^2)$ time, instead of $O(OPT)$-approximation in $2^{O(k \log k)} \cdot n^3 \log n$ time.
Later results

- P., *Computing cutwidth and pathwidth of semi-complete digraphs via degree orderings*, STACS 2013
- Completely new approach.
- 7-approximation of pathwidth in $O(kn^2)$ time, instead of $O(OPT)$-approximation in $2^{O(k \log k)} \cdot n^3 \log n$ time.
- Running time of topological subgraph containment testing trimmed to $2^{O(|H| \log |H|)} \cdot n^2$. 
FPT approximation of pathwidth opens possibilities for new FPT results on tournaments.
Conclusions

- **FPT approximation of pathwidth opens possibilities for new FPT results on tournaments.**
- **Open problem:** VERTEX-DISJOINT PATHS
Conclusions

- FPT approximation of pathwidth opens possibilities for new FPT results on tournaments.

**Open problem:** *Vertex-disjoint Paths*
- $k$ terminal pairs: $(s_1, t_1), (s_2, t_2), \ldots, (s_k, t_k)$. 

Can one find vertex-disjoint paths $P_1, P_2, \ldots, P_k$ connecting corresponding terminals? The problem is known to be in \(\text{XP}\) by a different approach [Chudnovsky, Scott, Seymour]. FPT is not known. The current technique fails because of the irrelevant vertex rule.
Conclusions

- FPT approximation of pathwidth opens possibilities for new FPT results on tournaments.

**Open problem:** **Vertex-disjoint Paths**
- \( k \) terminal pairs: \((s_1, t_1), (s_2, t_2), \ldots, (s_k, t_k)\).
- Can one find vertex-disjoint paths \( P_1, P_2, \ldots, P_k \) connecting corresponding terminals?
Conclusions

- **FPT approximation of pathwidth** opens possibilities for new FPT results on tournaments.

- **Open problem**: *Vertex-disjoint Paths*
  - *k* terminal pairs: \((s_1, t_1), (s_2, t_2), \ldots, (s_k, t_k)\).
  - Can one find vertex-disjoint paths \(P_1, P_2, \ldots, P_k\) connecting corresponding terminals?

- The problem is known to be in XP by a different approach [Chudnovsky, Scott, Seymour]. FPT is not known.
Conclusions

- FPT approximation of pathwidth opens possibilities for new FPT results on tournaments.

**Open problem:** **Vertex-disjoint Paths**
- $k$ terminal pairs: $(s_1, t_1), (s_2, t_2), \ldots, (s_k, t_k)$.
- Can one find vertex-disjoint paths $P_1, P_2, \ldots, P_k$ connecting corresponding terminals?

The problem is known to be in XP by a different approach [Chudnovsky, Scott, Seymour]. FPT is not known.

- The current technique fails because of the irrelevant vertex rule.
Questions?