The background

**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 a fixed $H$, e.g., $O(|G|^{|H|})$. 
The algorithms

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

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

For **Minor Containment**, $f(|H|)|V(G)|^3$ algorithm [Robertson and Seymour, 1995].

For **Topological Subgraph Containment**, $f(|H|)|V(G)|^3$ algorithm [Grohe et al., 2011].
The algorithms

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

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

- **For Minor Containment,**
  $f(|H|)|V(G)|^3$ algorithm [Robertson and Seymour, 1995].
The algorithms

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

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

- **For Minor Containment,**
  $f(|H|)|V(G)|^3$ algorithm [Robertson and Seymour, 1995].

- **For Topological Subgraph Containment,**
  $f(|H|)|V(G)|^3$ algorithm [Grohe et al., 2011].
We consider topological subgraph containment.
Directed world

- We consider topological subgraph containment.
  - $H$ is a topological subgraph of $G$ if some its subdivision is a subgraph of $G$. 

NP-hard in general setting even for small, fixed subgraphs $H$ [Fortune et al., 1980]. In acyclic digraphs there is an XP algorithm, but FPT is unlikely [Slivkins, 2010]. There is hope, when $G$ is a tournament.
We consider topological subgraph containment.

- $H$ is a topological subgraph of $G$ if some its subdivision is a subgraph of $G$.

- $NP$-hard in general setting even for small, fixed subgraphs $H$ [Fortune et al., 1980].
We consider topological subgraph containment.

- $H$ is a topological subgraph of $G$ if some its subdivision is a subgraph of $G$.

- $NP$-hard in general setting even for small, fixed subgraphs $H$ [Fortune et al., 1980].

- In acyclic digraphs there is an XP algorithm, but FPT is unlikely [Slivkins, 2010].
We consider topological subgraph containment.

- $H$ is a topological subgraph of $G$ if some its subdivision is a subgraph of $G$.

- NP-hard in general setting even for small, fixed subgraphs $H$ [Fortune et al., 1980].

- In acyclic digraphs there is an XP algorithm, but FPT is unlikely [Slivkins, 2010].

- There is hope, when $G$ is a tournament.
Containment in tournaments was studied extensively by Chudnovsky, Fradkin, Scott and Seymour.
Tournament world

- Containment in tournaments was studied extensively by Chudnovsky, Fradkin, Scott and Seymour.
- A number of FPT algorithms (immersion) and XP algorithms (topological containment).
Containment in tournaments was studied extensively by Chudnovsky, Fradkin, Scott and Seymour.
A number of FPT algorithms (immersion) and XP algorithms (topological containment).

**Our results:** Refining recent work of Fradkin and Seymour to get FPT algorithm for topological containment ($O(|V|^5)$).
Approach of Fradkin and Seymour

\[ H, T \]

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

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

Run dynamic programming.

\[ f(|H|) - \text{jungle} \]

\[ |H| - \text{triple} \]

Answer YES

\[ \text{XP} \]

\[ \text{FPT} \]
Approach of Fradkin and Seymour

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

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

Run dynamic programming.

Answer YES

FPT
Approach of Fradkin and Seymour

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

Path decomposition of width $O(f(|H|)^2)$
Approach of Fradkin and Seymour

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

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

Run dynamic programming.
Approach 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

$H, T$
Approach of Fradkin and Seymour

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

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

$H, T$

Run dynamic programming.

$f(|H|)$-jungle

$|H|$-triple

Answer YES
Approach 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
Approach 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
Approach of Fradkin and Seymour

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

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

Run dynamic programming.

FPT

$f(|H|)$-jungle

|H|-triple

Answer YES
Is the 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
The result

FPT approximation of tournament pathwidth

There exists an algorithm, which given a tournament $T$ 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 $2^{O(k^2)}|V(T)|^5$. 
Separation

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

\[ A \cup B = V, \quad |A \setminus B| \leq k; \]

and there are no edges from $A \setminus B$ to $B \setminus A$.
Separation

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

- $A \cup B = V(T)$, $|A \cap B| \leq k$;
- 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

- \(A \cup B = V(T), |A \cap B| \leq k;\)
- and there are no edges from \(A \setminus B\) to \(B \setminus A\).
(A, B) is a separation of order \( k \) if
- \( A \cup B = V(T), |A \cap B| \leq k; \)
- 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.
Path decompositions 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 $u \in W_i, v \in W_j$ for $i > j$.

$pw(T) = \max W_i - 1$.

**k-jungle**: set $X$ of cardinality $k$, such that every two vertices of $X$ are $k$-connected.
Path decomposition of $T$ is a sequence of bags $[W_1, W_2, \ldots, W_h]$ such that

- $\bigcup W_i = V(T)$;
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 $i < j < k$;
**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 $i > j$. 

**pw** $(T) = \max_{W_i} W_i - 1$.

- **k-jungle**: set $X$ of cardinality $k$, such that every two vertices of $X$ are $k$-connected.
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 $i > j$.

- $pw(T) = \max W_i - 1$. 

**k-jungle**: set $X$ of cardinality $k$, such that every two vertices of $X$ are $k$-connected.
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 $i > j$.

$\text{pw}(T) = \max W_i - 1$.

- **$k$-jungle**: set $X$ of cardinality $k$, such that every two vertices of $X$ are $k$-connected.
We greedily incorporate bigger and bigger separations of the tournament up to order \( k \), constructing a cross-free family of separations called a bundle.
Approximation algorithm of Fradkin and Seymour

- We greedily incorporate bigger and bigger separations of the tournament up to order $k$, constructing a cross-free family of separations called a bundle.
- Each new separation has to satisfy certain technical conditions.
We greedily incorporate bigger and bigger separations of the tournament up to order $k$, constructing a cross-free family of separations called a \textit{bundle}.

- Each new separation has to satisfy certain technical conditions.
- \textbf{Having a maximum bundle we obtain some path decomposition:}
Approximation algorithm of Fradkin and Seymour

- We greedily incorporate bigger and bigger separations of the tournament 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.
We greedily incorporate bigger and bigger separations of the tournament 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

[Image of an algorithm diagram]
Algorithm: overview
Algorithm: overview
Algorithm: overview
Algorithm: overview
Algorithm: overview
The new separation cannot be 'close' to the neighbouring ones.

\[ \geq k|a_1 - b| \quad \geq k|a_2 - b| \]
Inserting new separation

- 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.
Tournament Balanced Separator

**Input:** A tournament $T$; disjoint sets $X, Y \subseteq V(T)$; integers $a, b, c$.

**Question:** Does there exist a separation $(A, B)$ of $T$ such that
- $|A \cap B| \leq k$;
- $X \subseteq A \setminus B$, $Y \subseteq B \setminus A$;
- $|A \setminus (X \cup B)| \geq a$ and $|B \setminus (Y \cup A)| \geq c$?

We show an algorithm working in time $2^{O(a+b+c)}|V(T)|^4$.
Subproblem

\((a, b, c)\)

\[ \geq a \quad \quad \geq c \quad \quad \leq b \]
Branching

Take any vertex and branch on it:

\[(a, b, c)\]
Branching

Take any vertex and branch on it:

Goes to $A \setminus B$.

$(a, b, c)$
Branching

Take any vertex and branch on it:

Goes to $B \setminus A$.

$(a, b, c)$
Branching

Take any vertex and branch on it:

Goes to $A \cap B$.

$$(a, b, c)$$
Branching

Take any vertex and branch on it:
If we run out of vertices: OK!

\((a, b, c)\)
Branching

Take any vertex and branch on it:
If we run out of $b$: OK!

$$(a, b, c)$$
Branching

Take any vertex and branch on it:

By symmetry, we run out of $a$.

$$(a, b, c)$$
Branching

Take any vertex with outneighbour in $Y$ and branch on it:

$$(0, b, c)$$
Branching

Take any vertex with outneighbour in $Y$ and branch on it:

Goes to $B \setminus A$.

$(0, b, c)$
Branching

Take any vertex with outneighbour in $Y$ and branch on it:

Goes to $A \cap B$.

$(0, b, c)$
Branching

Take any vertex with outneighbour in $Y$ and branch on it:

If we run out of $b$: OK!

$(0, b, c)$
Branching

Take any vertex with outneighbour in $Y$ and branch on it:
If we run out of $c$: OK!

$$(0, b, c)$$
Branching

Take any vertex with outneighbour in \( Y \) and branch on it:
Then we run out of vertices...

\((0, b, c)\)
Reduced problem

\[(0, b, c)\]
Reduced problem

\[(0, b, c)\]

- \(b \leq c\)
- \(c \geq b\)
**Tournament Subset Separation**

**Input:** A tournament $T$ with set of terminals $X$; integers $b$, $c$.

**Question:** Does there exist a separation $(A, B)$ of $T$ of order $b$, such that $X \subseteq A \setminus B$ and $|B \setminus A| \geq c$?
Now comes the tricky part.
Now comes the tricky part.

**We consider two cases:**
Now comes the tricky part.
We consider two cases:
- $|B \setminus A| \geq 2b + 2c;$
Subsubproblem: solution

Now comes the tricky part.

We consider two cases:

1. $|B \setminus A| \geq 2b + 2c$;
2. $|B \setminus A| \leq 2b + 2c - 1$. 
Consider subtournament $T[B \setminus A]$; there must be a vertex $v$ with indegree at least $b + c - 1$. 

$|B \setminus A| \geq 2b + 2c$
Consider subtournament $T[B \setminus A]$; there must be a vertex $v$ with indegree at least $b + c - 1$.

Branch into $|V(T) \setminus X|$ subcases, in each taking different nonterminal as $v$. 

$|B \setminus A| \geq 2b + 2c$
Consider subtournament \( T[B \setminus A] \); there must be a vertex \( v \) with indegree at least \( b + c - 1 \).

Branch into \( |V(T) \setminus X| \) subcases, in each taking different nonterminal as \( v \).

We compute the minimum cut between \( X \) and \( v \):
Consider subtournament $T[B \setminus A]$; there must be a vertex $v$ with indegree at least $b + c - 1$.

Branch into $|V(T) \setminus X|$ subcases, in each taking different nonterminal as $v$.

We compute the minimum cut between $X$ and $v$:

- in the correct branch it has to be at most $b$, 

\[ |B \setminus A| \geq 2b + 2c \]

- Consider subtournament \( T[B \setminus A] \); there must be a vertex \( v \) with indegree at least \( b + c - 1 \).
- Branch into \( |V(T) \setminus X| \) subcases, in each taking different nonterminal as \( v \).
- We compute the minimum cut between \( X \) and \( v \):
  - in the correct branch it has to be at most \( b \),
  - which means that it separates at least \( c \) vertices: \( v \) and \( c - 1 \) his inneighbours.
Consider subtournament $T[B \setminus A]$; there must be a vertex $v$ with indegree at least $b + c - 1$.

Branch into $|V(T) \setminus X|$ subcases, in each taking different nonterminal as $v$.

We compute the minimum cut between $X$ and $v$:

- in the correct branch it has to be at most $b$,
- which means that it separates at least $c$ vertices: $v$ and $c - 1$ his inneighbours.

The separation we have found can be different than $(A, B)$, but it suffices to our needs.
\[ |B \setminus A| \leq 2b + 2c - 1 \]

- Every vertex of \( B \setminus A \) has indegree at most \( 3b + 2c - 1 \).
\( |B \setminus A| \leq 2b + 2c - 1 \)

- Every vertex of \( B \setminus A \) has indegree at most \( 3b + 2c - 1 \).
- There are at most \( 2(3b + 2c - 1) + 1 \) such vertices in \( T \), as otherwise a higher indegree would occur inside the subtournament induced.
\[ |B \setminus A| \leq 2b + 2c - 1 \]

- Every vertex of \( B \setminus A \) has indegree at most \( 3b + 2c - 1 \).
- There are at most \( 2(3b + 2c - 1) + 1 \) such vertices in \( T \), as otherwise a higher indegree would occur inside the subtournament induced.
- **We do brute-force:** iterate through all the subsets of these vertices of small indegree.
Overview of other results

- Together with dynamic programming on path decomposition:
  - Algorithm for Topological Containment.
  - Algorithm for Edge Disjoint Paths on a $k$-triple.
  - Algorithm for Rooted Immersion, based on pathwidth.
  - Algorithms for related problems by Fradkin and Seymour, based on cutwidth.
Overview of other results

- Together with dynamic programming on path decomposition:
  - $f(|H|)|V(T)|^5$ algorithm for Topological Containment.

- We show an irrelevant vertex procedure for Edge Disjoint Paths on a $k$-triple.

- Algorithm for Rooted Immersion, based on pathwidth.

- Algorithms for related problems by Fradkin and Seymour, based on cutwidth.
Overview of other results

- Together with dynamic programming on path decomposition:
  - $f(|H||V(T)|^5$ algorithm for Topological Containment.
- We show an irrelevant vertex procedure for Edge Disjoint Paths on a $k$-triple.
Overview of other results

- Together with dynamic programming on path decomposition:
  - \( f(|H|)|V(T)|^5 \) algorithm for **Topological Containment**.
- We show an irrelevant vertex procedure for **Edge Disjoint Paths** on a \( k \)-triple.
  - \( f(|H|)|V(T)|^6 \) algorithm for **Rooted Immersion**, based on pathwidth.
Overview of other results

- Together with dynamic programming on path decomposition:
  - $f(|H|)|V(T)|^5$ algorithm for Topological Containment.
- We show an irrelevant vertex procedure for Edge Disjoint Paths on a $k$-triple.
  - $f(|H|)|V(T)|^6$ algorithm for Rooted Immersion, based on pathwidth.
  - $f(|H|)|V(T)|^5$ algorithms for related problems by Fradkin and Seymour, based on cutwidth.
Open problem

- **Vertex Disjoint Paths** in FPT time?
Open problem

- **Vertex Disjoint Paths** in FPT time?
- XP algorithm by Chudnovsky, Scott and Seymour.
Open problem

- **Vertex Disjoint Paths** in FPT time?
- XP algorithm by Chudnovsky, Scott and Seymour.
- Irrelevant vertex technique does not work: there are tournaments of large pathwidth with all the vertices relevant.
Open problem

- **Vertex Disjoint Paths** in FPT time?
- XP algorithm by Chudnovsky, Scott and Seymour.
- Irrelevant vertex technique does not work: there are tournaments of large pathwidth with all the vertices relevant.
- **This suggests change of the width parameter.**
Thank you

Questions?