Minor Containment and Disjoint Paths in almost-linear time

Tuukka Korhonen[1], Michał Pilipczuk[2], Giannos Stamoulis[2]

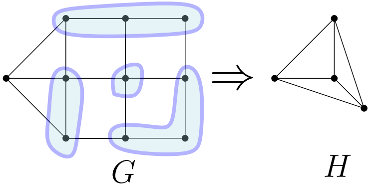
[1]University of Copenhagen [2]University of Warsaw

FOCS 2024

28 October 2024

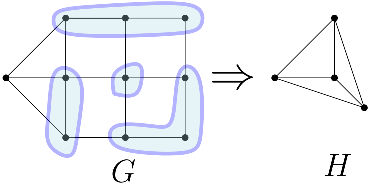
Minors of graphs

- A graph H is a minor of a graph G if H can be obtained from G by
  - Vertex deletions
  - Edge deletions
  - Edge contractions



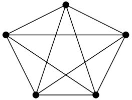
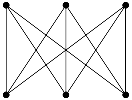
Minors of graphs

- A graph H is a minor of a graph G if H can be obtained from G by
  - Vertex deletions
  - Edge deletions
  - Edge contractions



Theorem (Kuratowski-Wagner, 1930, 1937)

A graph is planar if and only if it does not contain K_5 or $K_{3,3}$ as a minor.

Planar $\Leftrightarrow$ no  or 

The Graph Minors Series

Theorem (Robertson & Seymour, 1984-2004)

Let $\mathcal{C}$ be a minor-closed graph class. There exists a finite set of graphs $\mathcal{H}$, s.t. a graph G is in $\mathcal{C}$ if and only if G does not contain a graph from $\mathcal{H}$ as a minor.

The Graph Minors Series

Theorem (Robertson & Seymour, 1984-2004)

Let $\mathcal{C}$ be a minor-closed graph class. There exists a finite set of graphs $\mathcal{H}$, s.t. a graph G is in $\mathcal{C}$ if and only if G does not contain a graph from $\mathcal{H}$ as a minor.

Theorem (Robertson & Seymour, 1984-2012)

There exists an $f(H) \cdot n^3$ time algorithm to test if a given graph H is a minor of a given n-vertex graph G.

The Graph Minors Series

Theorem (Robertson & Seymour, 1984-2004)

Let $\mathcal{C}$ be a minor-closed graph class. There exists a finite set of graphs $\mathcal{H}$, s.t. a graph G is in $\mathcal{C}$ if and only if G does not contain a graph from $\mathcal{H}$ as a minor.

Theorem (Robertson & Seymour, 1984-2012)

There exists an $f(H) \cdot n^3$ time algorithm to test if a given graph H is a minor of a given n-vertex graph G.

Corollary

For every minor-closed graph class $\mathcal{C}$, there exists an $\mathcal{O}(n^3)$ time algorithm to test if a given n-vertex graph is in $\mathcal{C}$.

The Graph Minors Series

Theorem (Robertson & Seymour, 1984-2004)

Let $\mathcal{C}$ be a minor-closed graph class. There exists a finite set of graphs $\mathcal{H}$, s.t. a graph G is in $\mathcal{C}$ if and only if G does not contain a graph from $\mathcal{H}$ as a minor.

Theorem (Robertson & Seymour, 1984-2012)

There exists an $f(H) \cdot n^3$ time algorithm to test if a given graph H is a minor of a given n-vertex graph G.

Corollary

For every minor-closed graph class $\mathcal{C}$, there exists an $\mathcal{O}(n^3)$ time algorithm to test if a given n-vertex graph is in $\mathcal{C}$.

- More generally, an $f(H) \cdot n^3$ time algorithm for Rooted Minor Containment

The Graph Minors Series

Theorem (Robertson & Seymour, 1984-2004)

Let $\mathcal{C}$ be a minor-closed graph class. There exists a finite set of graphs $\mathcal{H}$, s.t. a graph G is in $\mathcal{C}$ if and only if G does not contain a graph from $\mathcal{H}$ as a minor.

Theorem (Robertson & Seymour, 1984-2012)

There exists an $f(H) \cdot n^3$ time algorithm to test if a given graph H is a minor of a given n-vertex graph G.

Corollary

For every minor-closed graph class $\mathcal{C}$, there exists an $\mathcal{O}(n^3)$ time algorithm to test if a given n-vertex graph is in $\mathcal{C}$.

- More generally, an $f(H) \cdot n^3$ time algorithm for Rooted Minor Containment
  - $\Rightarrow$ $f(k) \cdot n^3$ time algorithm for the k-Disjoint Paths problem

Improvements to the Robertson-Seymour algorithm

Improvements to the Robertson-Seymour algorithm

- The algorithm of Robertson & Seymour was improved to $f(H) \cdot n^2$ by [Kawarabayashi, Kobayashi & Reed, 2012]

Improvements to the Robertson-Seymour algorithm

- The algorithm of Robertson & Seymour was improved to $f(H) \cdot n^2$ by [Kawarabayashi, Kobayashi & Reed, 2012]
- Linear-time algorithms for planar graphs by [Bodlaender, 1993] and [Reed, Robertson, Schrijver & Seymour, 1993]

Improvements to the Robertson-Seymour algorithm

- The algorithm of Robertson & Seymour was improved to $f(H) \cdot n^2$ by [Kawarabayashi, Kobayashi & Reed, 2012]
- Linear-time algorithms for planar graphs by [Bodlaender, 1993] and [Reed, Robertson, Schrijver & Seymour, 1993]

Theorem (This work)

There is an $f(H) \cdot m^{1+o(1)}$ time algorithm for Rooted Minor Containment

Improvements to the Robertson-Seymour algorithm

- The algorithm of Robertson & Seymour was improved to $f(H) \cdot n^2$ by [Kawarabayashi, Kobayashi & Reed, 2012]
- Linear-time algorithms for planar graphs by [Bodlaender, 1993] and [Reed, Robertson, Schrijver & Seymour, 1993]

Theorem (This work)

There is an $f(H) \cdot m^{1+o(1)}$ time algorithm for Rooted Minor Containment

Corollary

Every minor-closed graph class has an $n^{1+o(1)}$ time recognition algorithm

Improvements to the Robertson-Seymour algorithm

- The algorithm of Robertson & Seymour was improved to $f(H) \cdot n^2$ by [Kawarabayashi, Kobayashi & Reed, 2012]
- Linear-time algorithms for planar graphs by [Bodlaender, 1993] and [Reed, Robertson, Schrijver & Seymour, 1993]

Theorem (This work)

There is an $f(H) \cdot m^{1+o(1)}$ time algorithm for Rooted Minor Containment

Corollary

Every minor-closed graph class has an $n^{1+o(1)}$ time recognition algorithm

Corollary

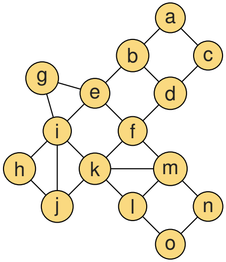
There is an $f(k) \cdot m^{1+o(1)}$ time algorithm for the k-Disjoint Paths problem

Our Algorithm

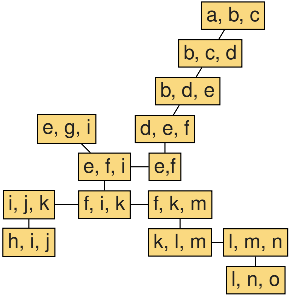
The Irrelevant Vertex technique

The Irrelevant Vertex technique

- Treewidth of a graph: Parameter between 0 and $n-1$ measuring how tree-like the graph is



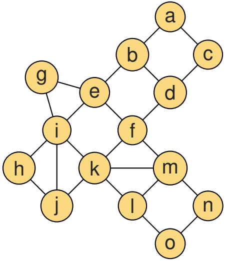
Graph G
Treewidth 2



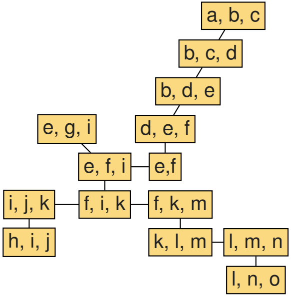
A tree decomposition of G
Width = 2

The Irrelevant Vertex technique

- Treewidth of a graph: Parameter between 0 and $n-1$ measuring how tree-like the graph is
- If treewidth of G is $\leq f(H)$, solve the problem by dynamic programming in $f(H) \cdot n$ time



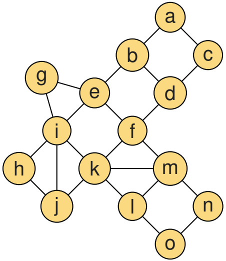
Graph G
Treewidth 2



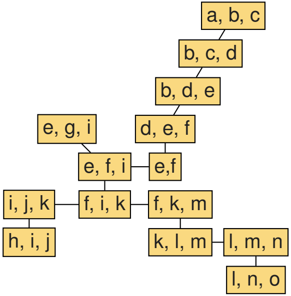
A tree decomposition of G
Width = 2

The Irrelevant Vertex technique

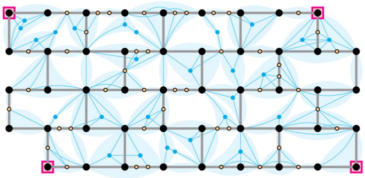
- Treewidth of a graph: Parameter between 0 and $n-1$ measuring how tree-like the graph is
- If treewidth of G is $\leq f(H)$, solve the problem by dynamic programming in $f(H) \cdot n$ time
- If treewidth is $> f(H)$, detect and remove an Irrelevant Vertex from G



Graph G
Treewidth 2

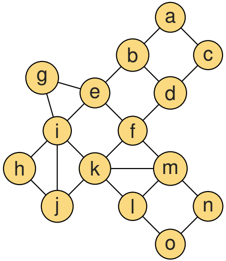


A tree decomposition of G
Width = 2

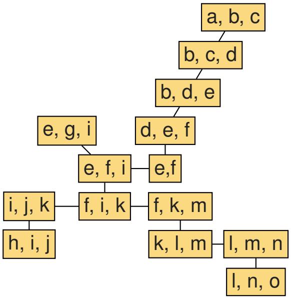


The Irrelevant Vertex technique

- Treewidth of a graph: Parameter between 0 and $n-1$ measuring how tree-like the graph is
- If treewidth of G is $\leq f(H)$, solve the problem by dynamic programming in $f(H) \cdot n$ time
- If treewidth is $> f(H)$, detect and remove an Irrelevant Vertex from G

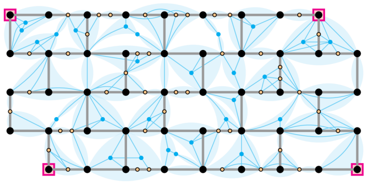


Graph G
Treewidth 2



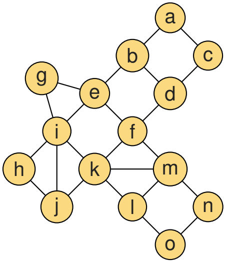
A tree decomposition of G
Width = 2

- Robertson & Seymour: Detect irrelevant vertex in $f(H) \cdot n^2$ time $\Rightarrow$ $f(H) \cdot n^3$ time algorithm

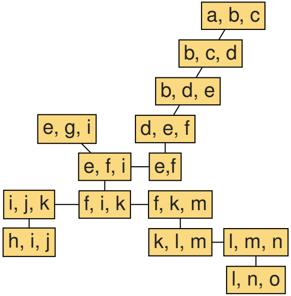


The Irrelevant Vertex technique

- Treewidth of a graph: Parameter between 0 and $n-1$ measuring how tree-like the graph is
- If treewidth of G is $\leq f(H)$, solve the problem by dynamic programming in $f(H) \cdot n$ time
- If treewidth is $> f(H)$, detect and remove an Irrelevant Vertex from G

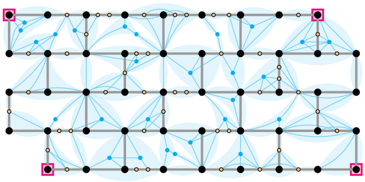


Graph G
Treewidth 2



A tree decomposition of G
Width = 2

- Robertson & Seymour: Detect irrelevant vertex in $f(H) \cdot n^2$ time $\Rightarrow$ $f(H) \cdot n^3$ time algorithm
- Kawarabayashi, Kobayashi & Reed: Detect irrelevant vertex in $f(H) \cdot n$ time $\Rightarrow$ $f(H) \cdot n^2$ time algorithm



Outline of our algorithm

Outline of our algorithm

1. Fast implementation of the irrelevant vertex technique on apex-minor-free graphs

Outline of our algorithm

1. Fast implementation of the irrelevant vertex technique on apex-minor-free graphs
   - Using dynamic treewidth data structure of [K., Majewski, Nadara, Pilipczuk & Sokołowski, FOCS 2023]

Outline of our algorithm

1. Fast implementation of the irrelevant vertex technique on apex-minor-free graphs
   - Using dynamic treewidth data structure of [K., Majewski, Nadara, Pilipczuk & Sokołowski, FOCS 2023]

2. Reducing general graphs to unbreakable graphs

Outline of our algorithm

1. Fast implementation of the irrelevant vertex technique on apex-minor-free graphs
   - Using dynamic treewidth data structure of [K., Majewski, Nadara, Pilipczuk & Sokołowski, FOCS 2023]
2. Reducing general graphs to unbreakable graphs
   - Fast implementation of the recursive understanding technique

Outline of our algorithm

1. Fast implementation of the irrelevant vertex technique on apex-minor-free graphs
   - Using dynamic treewidth data structure of [K., Majewski, Nadara, Pilipczuk & Sokołowski, FOCS 2023]
2. Reducing general graphs to unbreakable graphs
   - Fast implementation of the recursive understanding technique
   - Using recent breakthroughs in almost-linear time graph algorithms:
     - ⋆ Isolating cuts [Li & Panigrahi, 2020]
     - ⋆ Almost-linear time (deterministic) max-flow [van den Brand, Chen, Kyng, Liu, Peng, Probst Gutenberg, Sachdeva & Sidford, 2023]
     - ⋆ Mimicking networks of [Saranurak & Yingchareonthawornchai, 2022]

Outline of our algorithm

1. Fast implementation of the irrelevant vertex technique on apex-minor-free graphs
   - Using dynamic treewidth data structure of [K., Majewski, Nadara, Pilipczuk & Sokołowski, FOCS 2023]
2. Reducing general graphs to unbreakable graphs
   - Fast implementation of the recursive understanding technique
   - Using recent breakthroughs in almost-linear time graph algorithms:
     - ⋆ Isolating cuts [Li & Panigrahi, 2020]
     - ⋆ Almost-linear time (deterministic) max-flow [van den Brand, Chen, Kyng, Liu, Peng, Probst Gutenberg, Sachdeva & Sidford, 2023]
     - ⋆ Mimicking networks of [Saranurak & Yingchareonthawornchai, 2022]
3. Reducing unbreakable graphs to apex-minor-free graphs
   - Using relatively well-known graph-theoretic techniques

Conclusion

- $f(H) \cdot m^{1+o(1)}$ time algorithm for Rooted Minor Containment

Conclusion

- $f(H) \cdot m^{1+o(1)}$ time algorithm for Rooted Minor Containment
- Fast irrelevant vertex technique for apex-minor-free graphs using dynamic treewidth

Conclusion

- $f(H) \cdot m^{1+o(1)}$ time algorithm for Rooted Minor Containment
- Fast irrelevant vertex technique for apex-minor-free graphs using dynamic treewidth
- Reduction to apex-minor-free using fast recursive understanding

Conclusion

- $f(H) \cdot m^{1+o(1)}$ time algorithm for Rooted Minor Containment
- Fast irrelevant vertex technique for apex-minor-free graphs using dynamic treewidth
- Reduction to apex-minor-free using fast recursive understanding

Future work:

Conclusion

- $f(H) \cdot m^{1+o(1)}$ time algorithm for Rooted Minor Containment
- Fast irrelevant vertex technique for apex-minor-free graphs using dynamic treewidth
- Reduction to apex-minor-free using fast recursive understanding

Future work:

- Computing the Robertson-Seymour decomposition, topological minor containment

Conclusion

- $f(H) \cdot m^{1+o(1)}$ time algorithm for Rooted Minor Containment
- Fast irrelevant vertex technique for apex-minor-free graphs using dynamic treewidth
- Reduction to apex-minor-free using fast recursive understanding

Future work:

- Computing the Robertson-Seymour decomposition, topological minor containment
- Optimization to $f(H) \cdot m \text{ polylog } n$

Conclusion

- $f(H) \cdot m^{1+o(1)}$ time algorithm for Rooted Minor Containment
- Fast irrelevant vertex technique for apex-minor-free graphs using dynamic treewidth
- Reduction to apex-minor-free using fast recursive understanding

Future work:

- Computing the Robertson-Seymour decomposition, topological minor containment
- Optimization to $f(H) \cdot m \text{ polylog } n$

Thank you!