The Complexity Landscape of Fixed-Parameter Directed Steiner Network Problems

Dániel Marx

Institute for Computer Science and Control,
Hungarian Academy of Sciences (MTA SZTAKI)
Budapest, Hungary

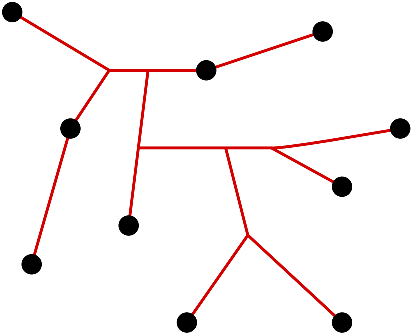
(Joint work with Andreas Feldmann)

SWAT 2016
Reykjavik, Iceland
June 23, 2016

Steiner Tree

Steiner Tree

Given an edge-weighted graph G and set $T \subseteq V(G)$ of terminals, find a minimum-weight tree in G containing every vertex of T.



This talk

I will talk about two topics:

1. A classification result for directed Steiner problems.
2. How this fits into the general theme of systematically classifying easy and hard graph problems.

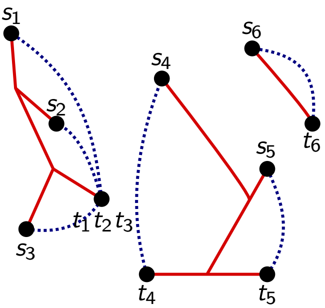
Steiner Tree

Some known results:

- NP-hard
- Easy 2-approximation: use a minimum spanning tree.
- 1.386-approximation [Byrka et al. 2013].
- $3^k \cdot n^{O(1)}$ time algorithm for k terminals using dynamic programming (i.e., fixed-parameter tractable parameterized by the number of terminals)
- Can be improved to $2^k \cdot n^{O(1)}$ time using fast subset convolution [Björklund et al. 2006].

Steiner Forest

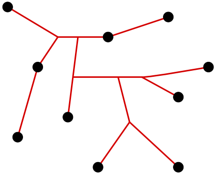
Steiner Forest
Given an edge-weighted graph G and a list $(s_1, t_1), \ldots, (s_k, t_k)$ of pairs of terminals, find a minimum-weight forest in G that connects s_i and t_i for every $1 \le i \le k$.



Fixed-parameter tractable parameterized by k: Guess a partition of the $2k$ terminals ($k^{O(k)} = 2^{O(k \log k)}$ possibilities) and solve a Steiner Tree for each class of the partition.

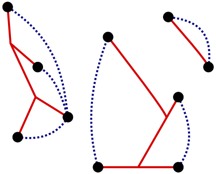
Variants of Steiner Tree

Steiner Tree



Connect all the terminals

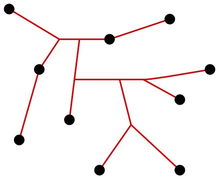
Steiner Forest



Create connections
satisying every request

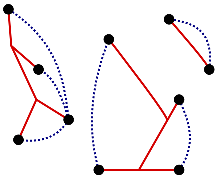
Variants of Steiner Tree

Steiner Tree



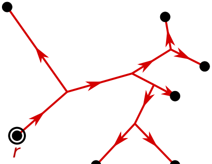
Connect all the terminals

Steiner Forest



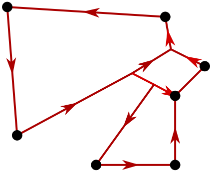
Create connections
satisying every request

Steiner Tree



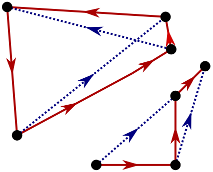
Make every terminal
reachable from the root

Strongly Connected Steiner Subgraph (SCSS)



Make all the terminals
reachable from each other

Directed Steiner Network (DSN)



Create connections
satisying every request

Directed Steiner vs. SCSS

The DP for Steiner Tree generalizes to the directed version:

Directed Steiner Tree with k terminals can be solved in time $2^k \cdot n^{O(1)}$.

Directed Steiner vs. SCSS

The DP for Steiner Tree generalizes to the directed version:

Directed Steiner Tree with k terminals can be solved in time $2^k \cdot n^{O(1)}$.

SCSS seems to be much harder:

Theorem [Feldman and Ruhl 2006]

Strongly Connected Steiner Subgraph with k terminals can be solved in time $n^{O(k)}$.

Theorem [Chitnis, Hajiaghayi, and M. 2014]

Assuming ETH, Strongly Connected Steiner Subgraph is W[1]-hard and has no $f(k)n^{o(k/\log k)}$ time algorithm for any function f.

Directed Steiner Network

Theorem [Feldman and Ruhl 2006]

Directed Steiner Network with k requests can be solved in time $n^{O(k)}$.

Corollary: Strongly Connected Steiner Subgraph with k terminals can be solved in time $n^{O(k)}$.

Proof is based on a "pebble game": $O(k)$ pebbles need to reach their destinations using certain allowed moves, tracing the solution.

Directed Steiner Network

A new combinatorial result:

Theorem [Feldmann and M. 2016]

[The underlying undireced graph of] every minimum cost solution of Directed Steiner Network with k requests has cutwidth and treewidth $O(k)$.

Directed Steiner Network

A new combinatorial result:

Theorem [Feldmann and M. 2016]

[The underlying undireced graph of] every minimum cost solution of Directed Steiner Network with k requests has cutwidth and treewidth $O(k)$.

A new algorithmic result:

Theorem [Feldmann and M. 2016]

If a Directed Steiner Network instance with k requests has a minimum cost solution with treewidth w [of the underlying undirected graph], then it can be solved in time $f(k, w) \cdot n^{O(w)}$.

Corollary: A new proof that DSN and SCSS can be solved in time $f(k)n^{O(k)}$.

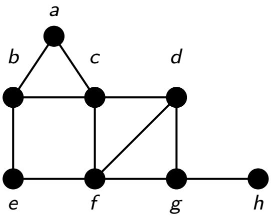
Treewidth — a measure of “tree-likeness”

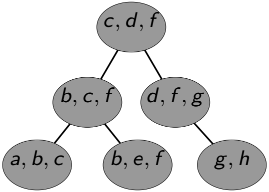
Tree decomposition: Vertices are arranged in a tree structure satisfying the following properties:

1. If u and v are neighbors, then there is a bag containing both of them.
2. For every v, the bags containing v form a connected subtree.

Width of the decomposition: largest bag size -1.

treewidth: width of the best decomposition.





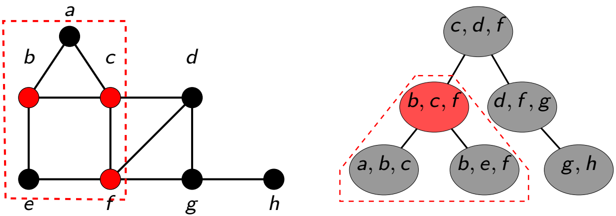
Treewidth — a measure of "tree-likeness"

Tree decomposition: Vertices are arranged in a tree structure satisfying the following properties:

1. If u and v are neighbors, then there is a bag containing both of them.
2. For every v, the bags containing v form a connected subtree.

Width of the decomposition: largest bag size -1.

treewidth: width of the best decomposition.



A subtree communicates with the outside world only via the root of the subtree.

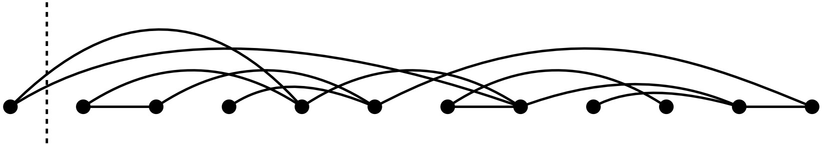
Cutwidth

A graph *G* has **cutwidth** at most *t* if there is a layout (an ordering of the vertices) where every "gap" is crossed by at most *t* edges.

Fact

Treewidth of *G* is at most the cutwidth of *G*

(So an upper bound on cutwidth is stronger than an upper bound on treewidth!)



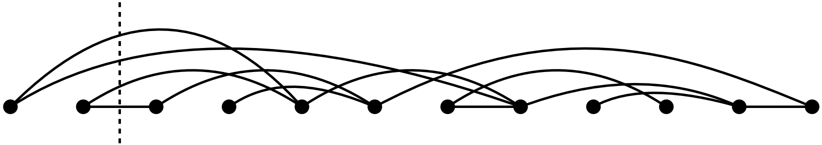
Cutwidth

A graph G has **cutwidth** at most t if there is a layout (an ordering of the vertices) where every "gap" is crossed by at most t edges.

Fact

Treewidth of G is at most the cutwidth of G

(So an upper bound on cutwidth is stronger than an upper bound on treewidth!)



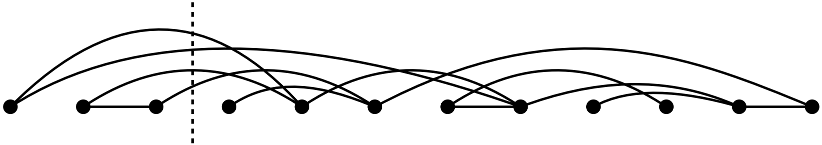
Cutwidth

A graph G has **cutwidth** at most t if there is a layout (an ordering of the vertices) where every "gap" is crossed by at most t edges.

Fact

Treewidth of G is at most the cutwidth of G

(So an upper bound on cutwidth is stronger than an upper bound on treewidth!)



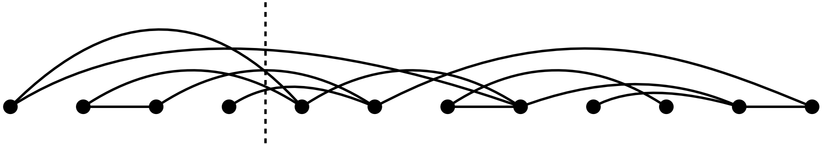
Cutwidth

A graph G has **cutwidth** at most t if there is a layout (an ordering of the vertices) where every "gap" is crossed by at most t edges.

Fact

Treewidth of G is at most the cutwidth of G

(So an upper bound on cutwidth is stronger than an upper bound on treewidth!)



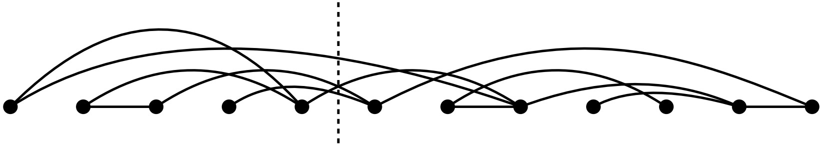
Cutwidth

A graph G has **cutwidth** at most t if there is a layout (an ordering of the vertices) where every "gap" is crossed by at most t edges.

Fact

Treewidth of G is at most the cutwidth of G

(So an upper bound on cutwidth is stronger than an upper bound on treewidth!)



Directed Steiner Network

A new combinatorial result:

Theorem [Feldmann and M. 2016]

[The underlying undireced graph of] every minimum cost solution of Directed Steiner Network with k requests has cutwidth and treewidth $O(k)$.

A new algorithmic result:

Theorem [Feldmann and M. 2016]

If a Directed Steiner Network instance with k requests has a minimum cost solution with treewidth w [of the underlying undirected graph], then it can be solved in time $f(k,w) \cdot n^{O(w)}$.

Corollary: A new proof that DSN and SCSS can be solved in time $f(k)n^{O(k)}$.

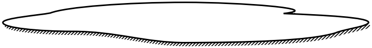
Side trip: planar graphs

$$\Bigg($$

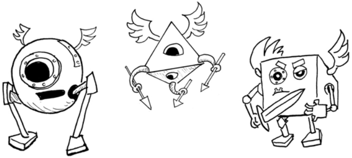
Square root phenomenon

NP-hard problems become easier on planar graphs and geometric objects, and usually exactly by a square root factor.

Planar graphs



Geometric objects



Better exponential algorithms

Most NP-hard problems (e.g., 3-Coloring, Independent Set, Hamiltonian Cycle, Steiner Tree, etc.) remain NP-hard on planar graphs,[1] so what do we mean by "easier"?

[1] Notable exception: Max Cut is in P for planar graphs.

Better exponential algorithms

Most NP-hard problems (e.g., 3-Coloring, Independent Set, Hamiltonian Cycle, Steiner Tree, etc.) remain NP-hard on planar graphs,[1] so what do we mean by "easier"?

The running time is still exponential, but significantly smaller:

$$2^{O(n)} \quad \Rightarrow \quad 2^{O(\sqrt{n})}$$
$$n^{O(k)} \quad \Rightarrow \quad n^{O(\sqrt{k})}$$
$$2^{O(k)} \cdot n^{O(1)} \quad \Rightarrow \quad 2^{O(\sqrt{k})} \cdot n^{O(1)}$$

[1] Notable exception: Max Cut is in P for planar graphs.

Planar Steiner Problems

Square root phenomenon for SCSS:

Theorem [Chitnis, Hajiaghayi, M. 2014]

Strongly Connected Steiner Subgraph with k terminals can be solved in time $f(k)n^{O(\sqrt{k})}$ on planar graphs.

Proof by a complicated generalization of the Feldman-Ruhl pebble game.

Lower bound:

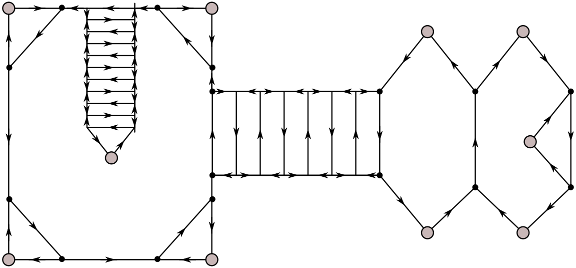
Theorem [Chitnis, Hajiaghayi, M. 2014]

Assuming ETH, Strongly Connected Steiner Subgraph with k terminals cannot be solved in time $f(k)n^{o(\sqrt{k})}$ on planar graphs.

Planar Strongly Connected Steiner Subgraph

Theorem [Feldmann and M. 2016]

Every minimum cost solution of SCSS with k terminals has "distance $O(k)$ from treewidth 2."



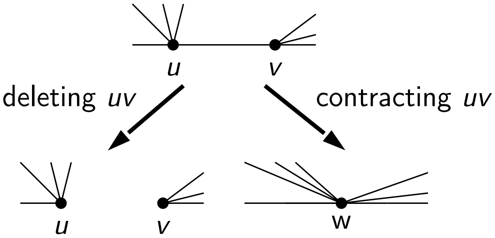
Corollary

Every minimum cost solution of SCSS with k terminals has treewidth $O(\sqrt{k})$ on planar graphs.

Minors

Definition

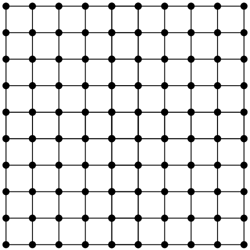
Graph H is a **minor** of G ($H \leq G$) if H can be obtained from G by deleting edges, deleting vertices, and contracting edges.



Planar Excluded Grid Theorem

Theorem [Robertson, Seymour, Thomas 1994]

Every planar graph with treewidth at least $5k$ has a $k \times k$ grid minor.

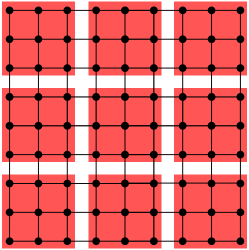


Note: for general graphs, treewidth at least $k^{19} \cdot \text{polylog}(k)$ guarantees a $k \times k$ grid minor (Julia's talk yesterday).

Planar Strongly Connected Steiner Subgraph

Theorem [Feldmann and M. 2016]

Every minimum cost solution of SCSS with k terminals has "distance $O(k)$ from treewidth 2."



Observation: In a $3\sqrt{k} \times 3\sqrt{k}$, each of the k small 3×3 grids have to be hit to make it treewidth 2.

Corollary

Every minimum cost solution of SCSS with k terminals has treewidth $O(\sqrt{k})$ on planar graphs.

Planar Directed Steiner Network

No square root phenomenon for DSN:

Theorem [Chitnis, Hajiaghayi, M. 2014]

Directed Steiner Network with k requests is W[1]-hard on planar graphs and (assuming ETH) cannot be solved in time $f(k)n^{o(k)}$.

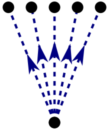
Perhaps because the problem description is not fully planar?
(Requests do not respect planarity.)

Side trip: planar graphs

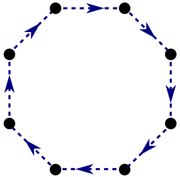
)

Special cases of Directed Steiner Network

Directed Steiner Tree and Strongly Connected Steiner Subgraph are both restrictions of Directed Steiner Network to certain type of patterns:



Directed Steiner Tree

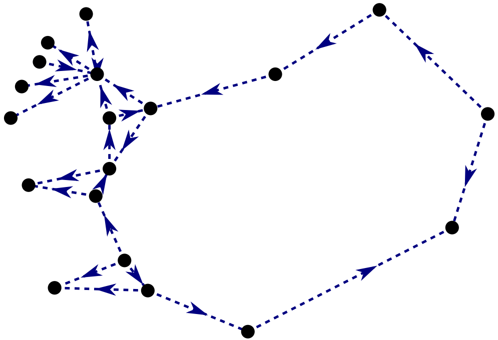


SCSS

Goal: characterize the patterns that give rise to FPT/W[1]-hard problems.

Patterns for DIRECTED STEINER NETWORK

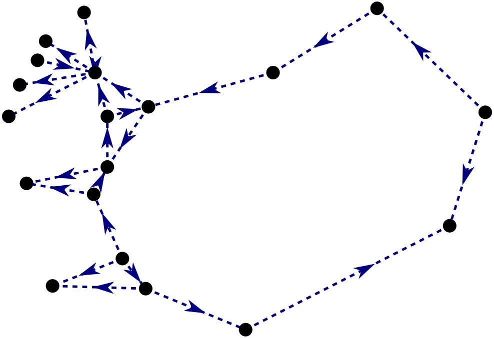
Question:
What is the complexity of DIRECTED STEINER NETWORK for this pattern?



Patterns for Directed Steiner Network

Question:

What is the complexity of Directed Steiner Network for this pattern?



Answer:

Directed Steiner Network has an $n^{O(k)}$ algorithm for k requests, so it is polynomial-time solvable for every fixed pattern.

Patterns for Directed Steiner Network

Goal: For every class of $\mathcal{H}$ of directed patterns, characterize the complexity of Directed Steiner Network *when restricted to demand patterns from* $\mathcal{H}$.

Example:

- If $\mathcal{H}$ is the class of all directed in-stars (or out-stars), then $\mathcal{H}$-DSN is FPT.
- If $\mathcal{H}$ is the class of all directed cycles, then $\mathcal{H}$-DSN is W[1]-hard.

Patterns for Directed Steiner Network

Goal: For every class of $\mathcal{H}$ of directed patterns, characterize the complexity of Directed Steiner Network *when restricted to demand patterns from* $\mathcal{H}$.

Example:

- If $\mathcal{H}$ is the class of all directed in-stars (or out-stars), then $\mathcal{H}$-DSN is FPT.
- If $\mathcal{H}$ is the class of all directed cycles, then $\mathcal{H}$-DSN is W[1]-hard.

Main result:

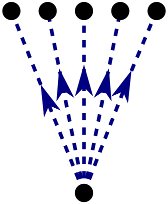
Theorem [Feldmann and M. 2016]

For any class $\mathcal{H}$ of directed patterns,

- if $\mathcal{H}$ has combinatorial property X, then $\mathcal{H}$-DSN and
- $\mathcal{H}$-DSN is W[1]-hard otherwise.

FPT special cases

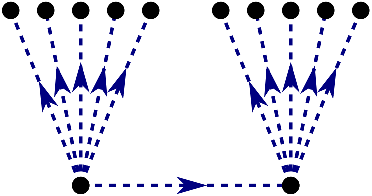
What classes $\mathcal{H}$ give FPT cases of $\mathcal{H}$-DSN?



We know that out-stars are FPT.

FPT special cases

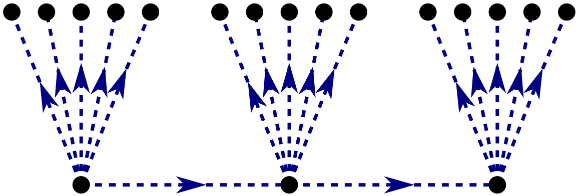
What classes $\mathcal{H}$ give FPT cases of $\mathcal{H}$-DSN?



This is also FPT: minimal solutions have bounded treewidth.

FPT special cases

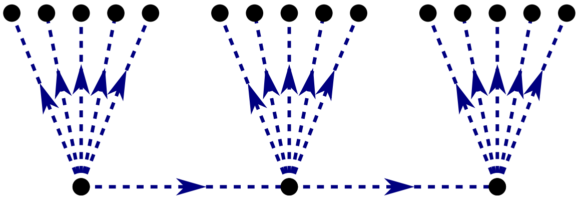
What classes $\mathcal{H}$ give FPT cases of $\mathcal{H}$-DSN?



This is also FPT: minimal solutions have bounded treewidth.

FPT special cases

What classes $\mathcal{H}$ give FPT cases of $\mathcal{H}$-DSN?



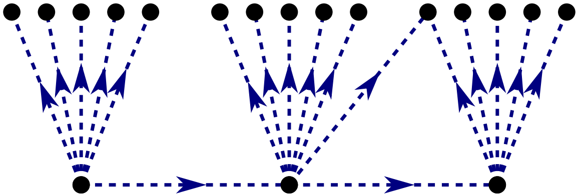
$\mathcal{C}_\lambda$: in- or out-caterpillar of length λ.

Lemma

If the pattern is in $\mathcal{C}_\lambda$, then every minimal solution has treewidth $O(\lambda^2)$.

FPT special cases

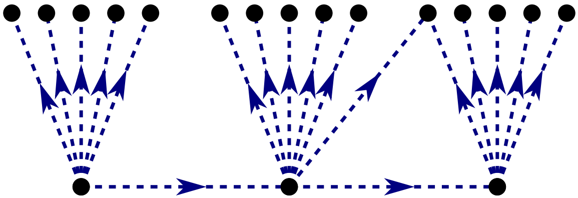
What classes $\mathcal{H}$ give FPT cases of $\mathcal{H}$-DSN?



What about this pattern?

FPT special cases

What classes $\mathcal{H}$ give FPT cases of $\mathcal{H}$-DSN?

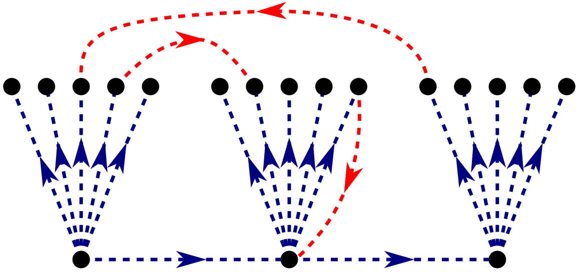


Lemma

If the pattern is **transitively equivalent** to a member of $\mathcal{C}_\lambda$, then every minimal solution has treewidth $O(\lambda^2)$.

FPT special cases

What classes $\mathcal{H}$ give FPT cases of $\mathcal{H}$-DSN?



$\mathcal{C}_{\lambda,\delta}$: in- or out-caterpillar of length λ with δ additional edges.

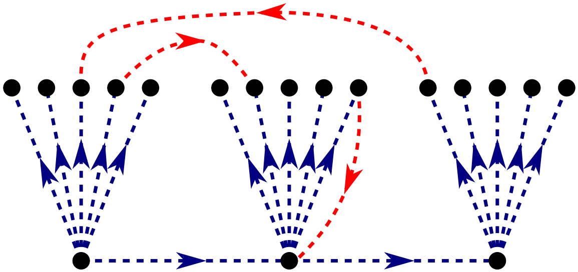
Lemma

If the pattern is **transitively equivalent** to a member of $\mathcal{C}_{\lambda,\delta}$, then every minimal solution has treewidth $O((1+\lambda)(\lambda+\delta))$.

FPT special cases

Theorem

If every $H \in \mathcal{H}$ is **transitively equivalent** to a member of $\mathcal{C}_{\lambda,\delta}$ for some constants $\lambda, \delta \geq 0$, then $\mathcal{H}$-DSN is FPT.

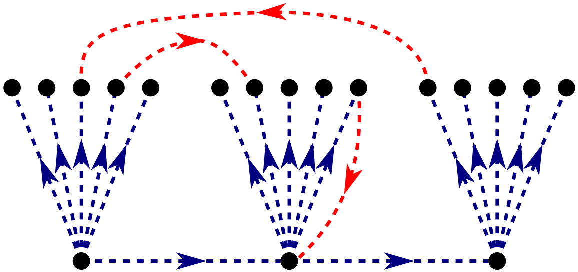


Does this cover all the FPT cases?

FPT special cases

Theorem

If every $H \in \mathcal{H}$ is **transitively equivalent** to a member of $\mathcal{C}_{\lambda,\delta}$ for some constants $\lambda, \delta \geq 0$, then $\mathcal{H}$-DSN is FPT.

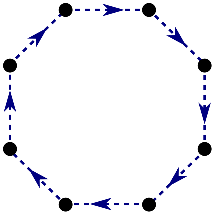


Does this cover all the FPT cases?

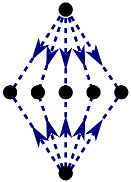
(Yes)

W[1]-hard special cases

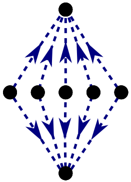
We show that the following classes $\mathcal{H}$ make $\mathcal{H}$-DSN W[1]-hard:



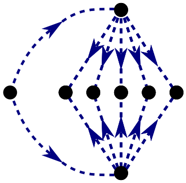
cycles (SCSS)



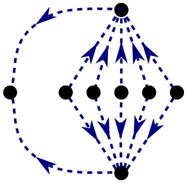
out-diamonds



in-diamonds



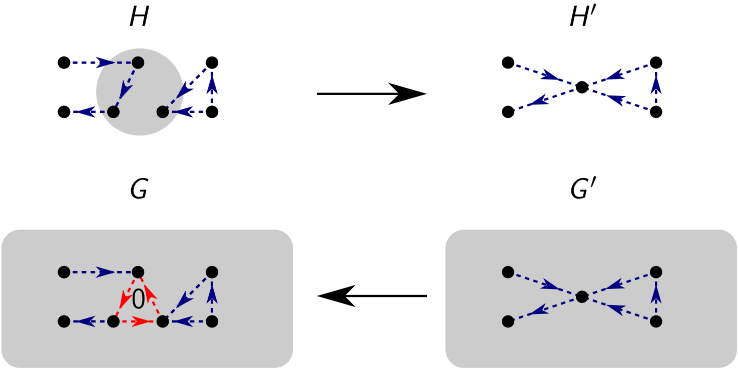
flawed out-diamonds



flawed in-diamonds

Identifying terminals

If H' is obtained from H by **identifying terminals**, then the problem cannot be harder for H' than for H:



$\Rightarrow$ We can assume that $\mathcal{H}$ is closed under identifying terminals.

Combinatorial classification

The following combinatorial result connects the algorithmic and the hardness results:

Theorem

Let $\mathcal{H}$ be a class of patterns closed under identifying terminals and transitive equivalence. Then exactly one of the following holds:

1. There are constants λ, δ such that every $H \in \mathcal{H}$ is transitively equivalent to a member of $\mathcal{C}_{\lambda,\delta}$
2. $\mathcal{H}$ contains either
    - all directed cycles,
    - all in-diamonds,
    - all out-diamonds,
    - all flawed in-diamonds, or
    - all flawed out-diamonds.

Classification result

Our main result:

Theorem [Feldmann and M. 2016]

Let $\mathcal{H}$ be a class of patterns.

1. If there are constants λ, δ such that every $H \in \mathcal{H}$ is transitively equivalent to a member of $\mathcal{C}_{\lambda,\delta}$, then $\mathcal{H}\text{-DSN}$ is FPT,
2. and it is W[1]-hard otherwise.

Dichotomy problem

- We have obtained a classification result that sharply divides the set of all special cases into "easy" and "hard" (dichotomy).

Dichotomy problem

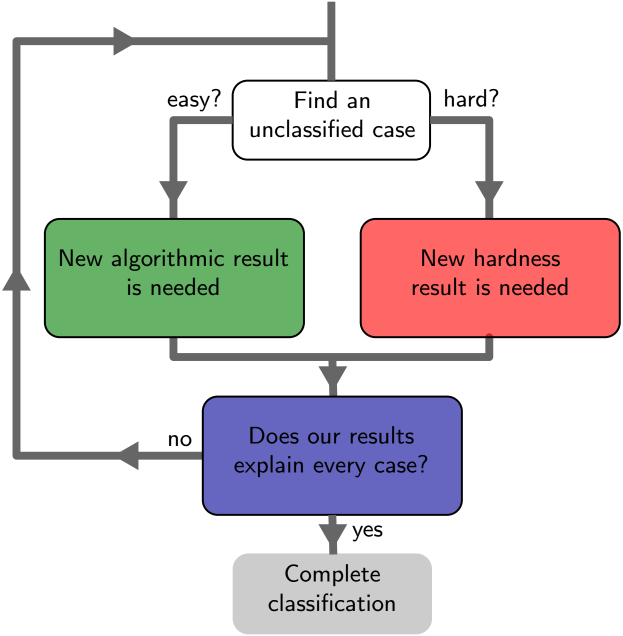
- We have obtained a classification result that sharply divides the set of all special cases into "easy" and "hard" (dichotomy).
- Such a result has to reveals all the algorithmic insights relevant for the problem, generalizing and unifying previous algorithms.

Dichotomy problem

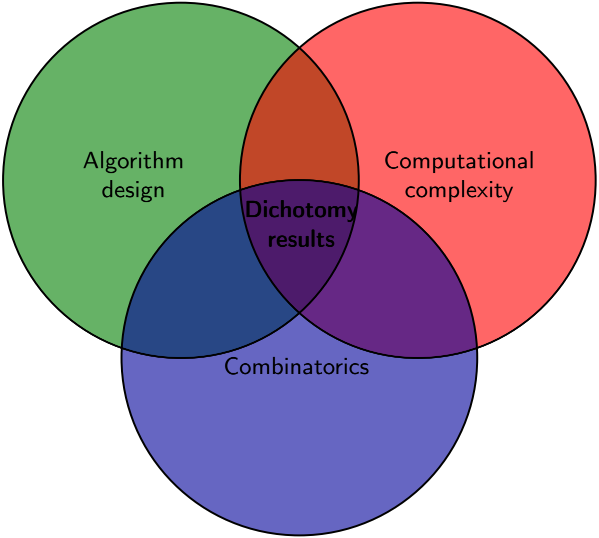
- We have obtained a classification result that sharply divides the set of all special cases into "easy" and "hard" (dichotomy).
- Such a result has to reveals all the algorithmic insights relevant for the problem, generalizing and unifying previous algorithms.
- Most algorithmic graph problems can be and should be analyzed this way!

Dichotomy problem

- We have obtained a classification result that sharply divides the set of all special cases into “easy” and “hard” (dichotomy).
- Such a result has to reveals all the algorithmic insights relevant for the problem, generalizing and unifying previous algorithms.
- Most algorithmic graph problems can be and should be analyzed this way!
- What is the methodology for obtaining such results?



Find an unclassified case
easy?
hard?
New algorithmic result is needed
New hardness result is needed
Does our results explain every case?
no
yes
Complete classification


Algorithm
design
Computational
complexity
Dichotomy
results
Combinatorics

In the case DIRECTED STEINER NETWORK:

- **Algorithm design:**
  Algorithm for "almost-caterpillars."
- **Computational complexity:**
  Hardness results for SCSS, diamonds, and flawed diamonds.
- **Combinatorics:**
  Either $\mathcal{H}$ contains only almost-caterpillars, or contain one of the obstructions.

In the case DIRECTED STEINER NETWORK:

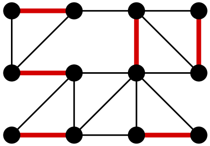
- **Algorithm design:**
  Algorithm for “almost-caterpillars.”
- **Computational complexity:**
  Hardness results for SCSS, diamonds, and flawed diamonds.
- **Combinatorics:**
  Either $\mathcal{H}$ contains only almost-caterpillars, or contain one of the obstructions.

Rest of the talk: A ~~smörgåsbord~~ hlaðborð of dichotomy results for other algorithmic graph problems.

Factor problems

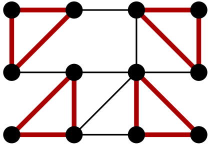
Perfect Matching **Input:** graph G. **Task:** find $\vert V(G)\vert/2$ vertex-disjoint edges.

Polynomial-time solvable [Edmonds 1961].



Triangle Factor **Input:** graph G. **Task:** find $\vert V(G)\vert/3$ vertex-disjoint triangles.

NP-complete [Karp 1975]



Factor problems

> H-FACTOR
> **Input:** graph G.
> **Task:** find $|V(G)|/|V(H)|$ vertex-disjoint copies of H in G.

Polynomial-time solvable for $H = K_2$ and NP-hard for $H = K_3$.

Which graphs H make H-FACTOR easy and which graphs make it hard?

Factor problems

> *H*-FACTOR
> **Input:** graph G.
> **Task:** find $|V(G)|/|V(H)|$ vertex-disjoint copies of H in G.

Polynomial-time solvable for $H = K_2$ and NP-hard for $H = K_3$.

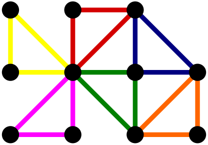
Which graphs H make H-FACTOR easy and which graphs make it hard?

Theorem [Kirkpatrick and Hell 1978]

H-FACTOR is NP-hard for every connected graph H with at least 3 vertices.

Edge-disjoint version

H-DECOMPOSITION
Input: graph G.
Task: find $|E(G)|/|E(H)|$ **edge-disjoint** copies of H in G.



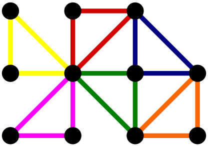
- Trivial for $H = K_2$.
- Can be solved by matching for P_3 (path on 3 vertices).

Theorem [Holyer 1981]

H-DECOMPOSITION is NP-complete if H is the clique K_r or the cycle C_r for some $r \geq 3$.

Edge-disjoint version

H-DECOMPOSITION
Input: graph G.
Task: find $|E(G)|/|E(H)|$ **edge-disjoint** copies of H in G.



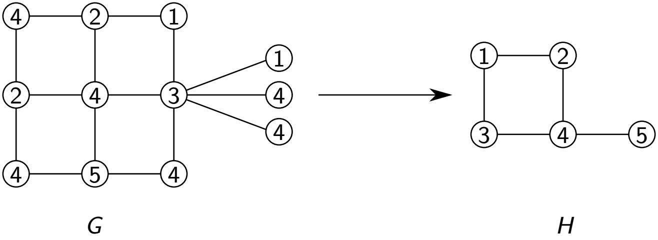
- Trivial for $H = K_2$.
- Can be solved by matching for P_3 (path on 3 vertices).

Theorem (Holyer's Conjecture) [Dor and Tarsi 1992]

H-DECOMPOSITION is NP-complete for every connected graph H with at least 3 edges.

H-coloring

A **homomorphism** from G to H is a mapping $f\colon V(G) \to V(H)$ such that if ab is an edge of G, then $f(a)f(b)$ is an edge of H.

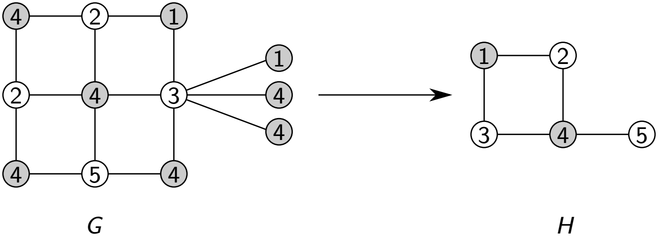


> H-COLORING
> **Input:** graph G.
> **Task:** Find a homomorphism from G to H.

- If $H = K_r$, then equivalent to r-COLORING.
- If H is bipartite, then the problem is equivalent to G being bipartite.

H-coloring

A **homomorphism** from G to H is a mapping $f\colon V(G) \to V(H)$ such that if ab is an edge of G, then $f(a)f(b)$ is an edge of H.

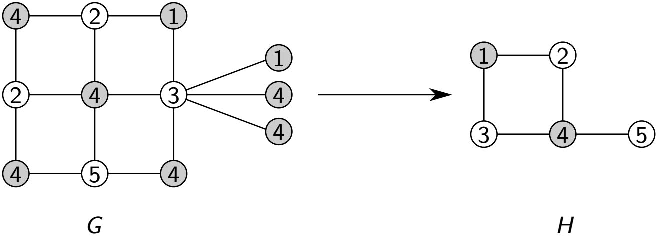


> H-COLORING
> **Input:** graph G.
> **Task:** Find a homomorphism from G to H.

- If $H = K_r$, then equivalent to r-COLORING.
- If H is bipartite, then the problem is equivalent to G being bipartite.

H-coloring

A **homomorphism** from G to H is a mapping $f\colon V(G) \to V(H)$ such that if ab is an edge of G, then $f(a)f(b)$ is an edge of H.



> H-COLORING
> **Input:** graph G.
> **Task:** Find a homomorphism from G to H.

Theorem [Hell and Nešetřil 1990]

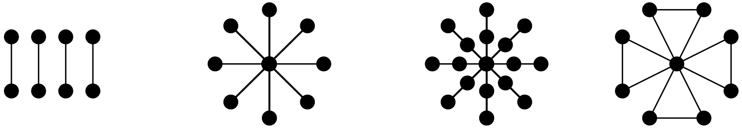
For every simple graph H, H-COLORING is polynomial-time solvable if H is bipartite and NP-complete if H is not bipartite.

Finding subgraphs

Sub($\mathcal{H}$)
Input: a graph $H \in \mathcal{H}$ and an arbitrary graph G.
Task: decide if H is a subgraph of G.

Some classes for which Sub($\mathcal{H}$) is polynomial-time solvable:

- $\mathcal{H}$ is the class of all matchings
- $\mathcal{H}$ is the class of all stars
- $\mathcal{H}$ is the class of all stars, each edge subdivided once
- $\mathcal{H}$ is the class of all windmills

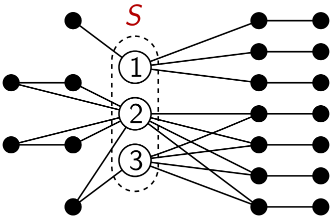


matching star subdivided star windmill

Finding subgraphs

Definition

Class $\mathcal{H}$ is **matching splittable** if there is a constant c such that every $H \in \mathcal{H}$ has a set S of at most c vertices such that every component of $H - S$ has size at most 2.



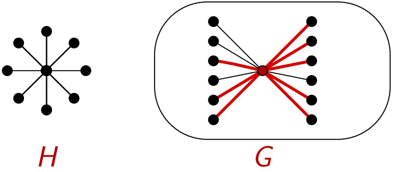
Theorem [Jansen and M. 2015]

Let $\mathcal{H}$ be a hereditary class of graphs. If $\mathcal{H}$ is matching splittable, then $\textsc{Sub}(\mathcal{H})$ is randomized polynomial-time solvable and NP-hard otherwise.

Counting subgraphs

> $\#\text{SUB}(\mathcal{H})$
> **Input:** a graph $H \in \mathcal{H}$ and an arbitrary graph G.
> **Task:** calculate the number of copies of H in G.

If $\mathcal{H}$ is the class of all stars, then $\#\text{SUB}(\mathcal{H})$ is easy: for each placement of the center of the star, calculate the number of possible different assignments of the leaves.

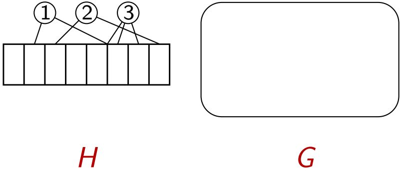


Counting subgraphs

$\#\textsc{Sub}(\mathcal{H})$
Input: a graph $H \in \mathcal{H}$ and an arbitrary graph G.
Task: calculate the number of copies of H in G.

Theorem

If every graph in $\mathcal{H}$ has vertex cover number at most c, then $\#\textsc{Sub}(\mathcal{H})$ is polynomial-time solvable.



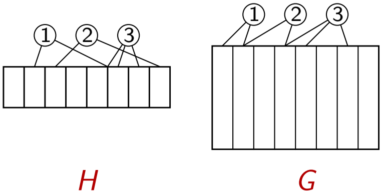
Running time is $n^{2^{O(c)}}$, better algorithms known [Vassilevska Williams and Williams], [Kowaluk, Lingas, and Lundell].

Counting subgraphs

#Sub($\mathcal{H}$)
Input: a graph $H \in \mathcal{H}$ and an arbitrary graph G.
Task: calculate the number of copies of H in G.

Theorem

If every graph in $\mathcal{H}$ has vertex cover number at most c, then #Sub($\mathcal{H}$) is polynomial-time solvable.



Running time is $n^{2^{O(c)}}$, better algorithms known [Vassilevska Williams and Williams], [Kowaluk, Lingas, and Lundell].

Counting subgraphs

Who are the bad guys now?

Theorem [Flum and Grohe 2002]

If $\mathcal{H}$ is the set of all paths, then $\#\textsc{Sub}(\mathcal{H})$ is #W[1]-hard.

Theorem [Curticapean 2013]

If $\mathcal{H}$ is the set of all matchings, then $\#\textsc{Sub}(\mathcal{H})$ is #W[1]-hard.

Counting subgraphs

Who are the bad guys now?

Theorem [Flum and Grohe 2002]

If $\mathcal{H}$ is the set of all paths, then $\#\textsc{Sub}(\mathcal{H})$ is #W[1]-hard.

Theorem [Curticapean 2013]

If $\mathcal{H}$ is the set of all matchings, then $\#\textsc{Sub}(\mathcal{H})$ is #W[1]-hard.

Dichotomy theorem:

Theorem [Curticapean and M. 2014]

Let $\mathcal{H}$ be a recursively enumerable class of graphs. If $\mathcal{H}$ has unbounded vertex cover number, then $\#\textsc{Sub}(\mathcal{H})$ is #W[1]-hard.

($\nu(G) \leq \tau(G) \leq 2\nu(G)$, hence "unbounded vertex cover number" and "unbounded matching number" are the same.)

Counting subgraphs

Theorem [Curticapean and M. 2014]

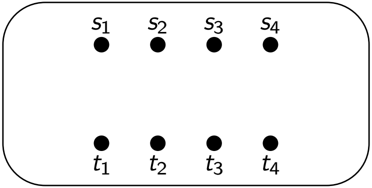
Let $\mathcal{H}$ be a recursively enumerable class of graphs.

- If $\mathcal{H}$ has bounded vertex cover number, then $\#\textsc{Sub}(\mathcal{H})$ is polynomial-time solvable.
- If $\mathcal{H}$ has unbounded vertex cover number, then $\#\textsc{Sub}(\mathcal{H})$ is #W[1]-hard (parameterized by $|V(H)|$).

Fixed-parameter tractability does not give us any extra power here!

Disjoint paths

k-Disjoint Paths
Input: graph G and pairs of vertices $(s_1, t_1), \ldots, (s_k, t_k)$.
Task: find pairwise vertex-disjoint paths $P_1, \ldots, P_k$ such that P_i connects s_i and t_i.



NP-hard, but FPT parameterized by k:

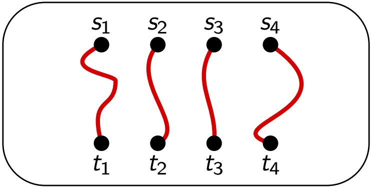
Theorem [Robertson and Seymour]

The k-Disjoint Paths problem can be solved in time $f(k)n^3$.

We consider now a maximization version of the problem.

Disjoint paths

k-Disjoint Paths
Input: graph G and pairs of vertices $(s_1, t_1), \dots, (s_k, t_k)$.
Task: find pairwise vertex-disjoint paths $P_1, \dots, P_k$ such that P_i connects s_i and t_i.



NP-hard, but FPT parameterized by k:

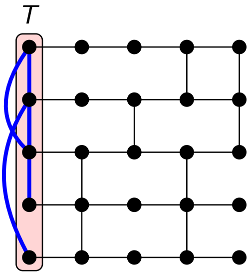
Theorem [Robertson and Seymour]

The k-Disjoint Paths problem can be solved in time $f(k)n^3$.

We consider now a maximization version of the problem.

Disjoint paths

MAXIMUM DISJOINT PATHS
Input: supply graph G, set $T \subseteq V(G)$ of terminals and a demand graph H on T.
Task: find k pairwise vertex-disjoint paths such that the two endpoints of each path are adjacent in H.

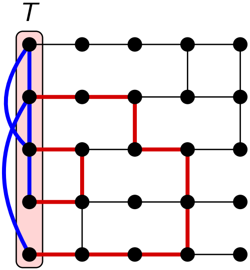


Can be solved in time $n^{O(k)}$, but W[1]-hard in general.

MAXIMUM DISJOINT $\mathcal{H}$-PATHS: special case when H restricted to be a member of $\mathcal{H}$.

Disjoint paths

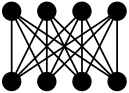
Maximum Disjoint Paths **Input:** supply graph G, set $T \subseteq V(G)$ of terminals and a demand graph H on T. **Task:** find k pairwise vertex-disjoint paths such that the two endpoints of each path are adjacent in H.



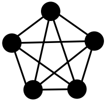
Can be solved in time $n^{O(k)}$, but W[1]-hard in general.

Maximum Disjoint $\mathcal{H}$-Paths: special case when H restricted to be a member of $\mathcal{H}$.

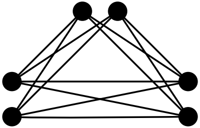
Maximum Disjoint $\mathcal{H}$-Paths



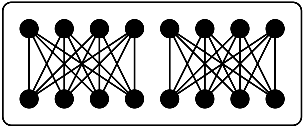
bicliques:
in P



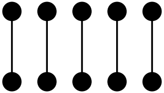
cliques:
in P



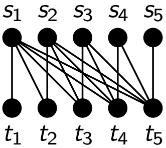
complete multipartite graphs:
in P



two disjoint bicliques:
FPT



matchings:
W[1]-hard



skew bicliques:
W[1]-hard

Maximum Disjoint $\mathcal{H}$-Paths

Questions:

- Algorithmic: FPT vs. W[1]-hard.
- Combinatorial (Erdős-Pósa): is there a function f such that there is either a set of k vertex-disjoint good paths or a set of $f(k)$ vertices covering every good path?

Maximum Disjoint $\mathcal{H}$-Paths

Questions:

- Algorithmic: FPT vs. W[1]-hard.
- Combinatorial (Erdős-Pósa): is there a function f such that there is either a set of k vertex-disjoint good paths or a set of $f(k)$ vertices covering every good path?

Theorem [M. and Wollan]

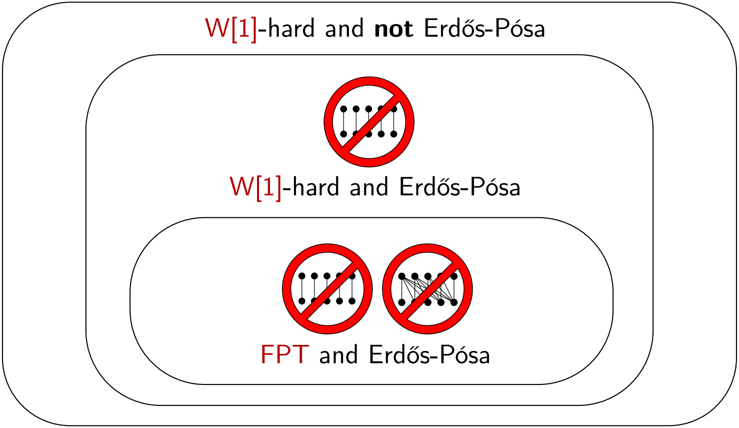
Let $\mathcal{H}$ be a hereditary class of graphs.

1. If $\mathcal{H}$ does not contain every matching and every skew biclique, then Maximum Disjoint $\mathcal{H}$-Paths is FPT and has the Erdős-Pósa Property.
2. If $\mathcal{H}$ does not contain every matching, but contains every skew biclique, then Maximum Disjoint $\mathcal{H}$-Paths is W[1]-hard, but has the Erdős-Pósa Property.
3. If $\mathcal{H}$ contains every matching, then Maximum Disjoint $\mathcal{H}$-Paths is W[1]-hard, and does not have the Erdős-Pósa Property.

Maximum Disjoint $\mathcal{H}$-Paths

Questions:

- Algorithmic: FPT vs. W[1]-hard.
- Combinatorial (Erdős-Pósa): is there a function f such that there is either a set of k vertex-disjoint good paths or a set of $f(k)$ vertices covering every good path?



Summary

- Dichotomy result for Directed Steiner Network: almost-caterpillars is FPT, everything else is W[1]-hard.
- Systematic research program to reveal all the algorithmic results that can appear in a certain framework.
- Some results for other problems, probably many more to come.