Parameterized Complexity

or "what I do for a living"

Michał Pilipczuk

Institute of Informatics, University of Warsaw, Poland

Cor Baayen award ceremony,
Budapest, October 26$^{\text{th}}$, 2016

Thanks!

Tatjana Abramovskaya
Anna Adamaszek
Florian Barbero
Ivan Bliznets
Hans L. Bodlaender
Mikołaj Bojańczyk
Marthe Bonamy
Dmitry Chistikov
Rajesh Chitnis
Marek Cygan
Wojtek Czerwiński
Claire David
Pål Grønås Drange
Markus S. Dregi
Fedor V. Fomin
Ariel Gabizon
Archontia C. Giannopoulou
Petr A. Golovach
MohammadTaghi Hajiaghayi
Pinar Heggernes
Pim van 't Hof
Piotr Hofman
Bart M. P. Jansen
Tomasz Kociumaka
Paweł Komosa
Łukasz Kowalik
Stefan Kratsch
Stephan Kreutzer
Erik Jan van Leeuwen
Daniel Lokshtanov
Lukáš Mach
Fredrik Manne
Dániel Marx
Filip Murlak
Jesper Nederlof
Charles Paperman
Christophe Paul
Daniël Paulusma
Geevarghese Philip
Marcin Pilipczuk
Roman Rabinovich
Jean-Florent Raymond
Felix Reidl
Johan van Rooij
Piotr Sankowski
Saket Saurabh
Ildi Schlotter
Sebastian Siebertz
Somnath Sikdar
Riste Škrekovski
Arek Socała
Szymon Toruńczyk
Fernando Sánchez Villaamil
Yngve Villanger
Magnus Wahlström
Michael Wehar
Jakub Onufry Wojtaszczyk
Marcin Wrochna

Thanks!

Tatjana Abramovskaya
Anna Adamaszek
Florian Barbero
Ivan Bliznets
Hans L. Bodlaender
Mikołaj Bojańczyk
Marthe Bonamy
Dmitry Chistikov
Rajesh Chitnis
Marek Cygan
Wojtek Czerwiński
Claire David
Pål Grønås Drange
Markus S. Dregi
Fedor V. Fomin
Ariel Gabizon
Archontia C. Giannopoulou
Petr A. Golovach
MohammadTaghi Hajiaghayi
Pinar Heggernes
Pim van 't Hof
Piotr Hofman
Bart M. P. Jansen
Tomasz Kociumaka
Paweł Komosa
Łukasz Kowalik
Stefan Kratsch
Stephan Kreutzer
Erik Jan van Leeuwen
Daniel Lokshtanov
Lukáš Mach
Fredrik Manne
Dániel Marx
Filip Murlak
Jesper Nederlof
Charles Paperman
Christophe Paul
Daniël Paulusma
Geevarghese Philip
Marcin Pilipczuk
Roman Rabinovich
Jean-Florent Raymond
Felix Reidl
Johan van Rooij
Piotr Sankowski
Saket Saurabh
Ildi Schlotter
Sebastian Siebertz
Somnath Sikdar
Riste Škrekovski
Arek Socała
Szymon Toruńczyk
Fernando Sánchez Villaamil
Yngve Villanger
Magnus Wahlström
Michael Wehar
Jakub Onufry Wojtaszczyk
Marcin Wrochna

Thanks!

Tatjana Abramovskaya
Anna Adamaszek
Florian Barbero
Ivan Bliznets
Hans L. Bodlaender
Mikołaj Bojańczyk
Marthe Bonamy
Dmitry Chistikov
Rajesh Chitnis
Marek Cygan
Wojtek Czerwiński
Claire David
Pål Grønås Drange
Markus S. Dregi
Fedor V. Fomin
Ariel Gabizon
Archontia C. Giannopoulou
Petr A. Golovach
MohammadTaghi Hajiaghayi
Pinar Heggernes
Pim van 't Hof
Piotr Hofman
Bart M. P. Jansen
Tomasz Kociumaka
Paweł Komosa
Łukasz Kowalik
Stefan Kratsch
Stephan Kreutzer
Erik Jan van Leeuwen
Daniel Lokshtanov
Lukáš Mach
Fredrik Manne
Dániel Marx
Filip Murlak
Jesper Nederlof
Charles Paperman
Christophe Paul
Daniël Paulusma
Geevarghese Philip
Marcin Pilipczuk
Roman Rabinovich
Jean-Florent Raymond
Felix Reidl
Johan van Rooij
Piotr Sankowski
Saket Saurabh
Ildi Schlotter
Sebastian Siebertz
Somnath Sikdar
Riste Škrekovski
Arek Socała
Szymon Toruńczyk
Fernando Sánchez Villaamil
Yngve Villanger
Magnus Wahlström
Michael Wehar
Jakub Onufry Wojtaszczyk
Marcin Wrochna

Thanks!

Tatjana Abramovskaya
Anna Adamaszek
Florian Barbero
Ivan Bliznets
Hans L. Bodlaender
Mikołaj Bojańczyk
Marthe Bonamy
Dmitry Chistikov
Rajesh Chitnis
Marek Cygan
Wojtek Czerwiński
Claire David
Pål Grønås Drange
Markus S. Dregi
Fedor V. Fomin
Ariel Gabizon
Archontia C. Giannopoulou
Petr A. Golovach
MohammadTaghi Hajiaghayi
Pinar Heggernes
Pim van 't Hof
Piotr Hofman
Bart M. P. Jansen
Tomasz Kociumaka
Paweł Komosa
Łukasz Kowalik
Stefan Kratsch
Stephan Kreutzer
Erik Jan van Leeuwen
Daniel Lokshtanov
Lukáš Mach
Fredrik Manne
Dániel Marx
Filip Murlak
Jesper Nederlof
Charles Paperman
Christophe Paul
Daniël Paulusma
Geevarghese Philip
Marcin Pilipczuk
Roman Rabinovich
Jean-Florent Raymond
Felix Reidl
Johan van Rooij
Piotr Sankowski
Saket Saurabh
Ildi Schlotter
Sebastian Siebertz
Somnath Sikdar
Riste Škrekovski
Arek Socała
Szymon Toruńczyk
Fernando Sánchez Villaamil
Yngve Villanger
Magnus Wahlström
Michael Wehar
Jakub Onufry Wojtaszczyk
Marcin Wrochna

Outline of the talk

- Short introduction to Parameterized Complexity.
- Some concepts, some problems, and some solutions.
  - Graph modification problems.
  - Treewidth, tree decompositions, and their algorithmic usage.
  - Network problems on planar graphs.
- Commercial break!

Parameterized complexity

- **Classic complexity theory**: Complexity measured in the size of input data.

Parameterized complexity

- **Classic complexity theory**: Complexity measured in the size of input data.
  - **Main distinction**: P-time solvable vs. NP-complete

Parameterized complexity

- **Classic complexity theory**: Complexity measured in the size of input data.
  - **Main distinction**: P-time solvable vs. NP-complete
- **Parameterized complexity**: Express complexity using a function of the input size and auxiliary parameters.

Parameterized complexity

- **Classic complexity theory**: Complexity measured in the size of input data.
  - **Main distinction**: P-time solvable vs. NP-complete
- **Parameterized complexity**: Express complexity using a function of the input size and auxiliary parameters.
  - Evaluation of query φ on database D.

Parameterized complexity

- **Classic complexity theory**: Complexity measured in the size of input data.
  - **Main distinction**: P-time solvable vs. NP-complete
- **Parameterized complexity**: Express complexity using a function of the input size and auxiliary parameters.
  - Evaluation of query φ on database D.
    - **Parameters**: $||\varphi||$, size of output, ...

Parameterized complexity

- **Classic complexity theory**: Complexity measured in the size of input data.
  - **Main distinction**: P-time solvable vs. NP-complete
- **Parameterized complexity**: Express complexity using a function of the input size and auxiliary parameters.
  - Evaluation of query φ on database D.
    - **Parameters**: $||\varphi||$, size of output, ...
  - Searching for pattern of size k in a large graph G.

Parameterized complexity

- **Classic complexity theory**: Complexity measured in the size of input data.
  - **Main distinction**: P-time solvable vs. NP-complete
- **Parameterized complexity**: Express complexity using a function of the input size and auxiliary parameters.
  - Evaluation of query φ on database D.
    - **Parameters**: $||\varphi||$, size of output, ...
  - Searching for pattern of size k in a large graph G.
    - **Parameters**: k, $\Delta(G)$, $\mathrm{genus}(G)$, ...

Parameterized complexity

- **Classic complexity theory**: Complexity measured in the size of input data.
  - **Main distinction**: P-time solvable vs. NP-complete
- **Parameterized complexity**: Express complexity using a function of the input size and auxiliary parameters.
  - Evaluation of query φ on database D.
    - **Parameters**: $||\varphi||$, size of output, ...
  - Searching for pattern of size k in a large graph G.
    - **Parameters**: k, $\Delta(G)$, $\mathrm{genus}(G)$, ...
  - Solving an integer linear program (ILP).

Parameterized complexity

- **Classic complexity theory**: Complexity measured in the size of input data.
  - **Main distinction**: P-time solvable vs. NP-complete
- **Parameterized complexity**: Express complexity using a function of the input size and auxiliary parameters.
  - Evaluation of query φ on database D.
    - **Parameters**: $||\varphi||$, size of output, ...
  - Searching for pattern of size k in a large graph G.
    - **Parameters**: k, $\Delta(G)$, $\mathrm{genus}(G)$, ...
  - Solving an integer linear program (ILP).
    - **Parameters**: # variables, max coefficient, structural measures of the matrix...

Parameterized complexity

- **Classic complexity theory**: Complexity measured in the size of input data.
  - **Main distinction**: P-time solvable vs. NP-complete
- **Parameterized complexity**: Express complexity using a function of the input size and auxiliary parameters.
  - Evaluation of query φ on database D.
    - **Parameters**: $||\varphi||$, size of output, ...
  - Searching for pattern of size k in a large graph G.
    - **Parameters**: k, $\Delta(G)$, $\mathrm{genus}(G)$, ...
  - Solving an integer linear program (ILP).
    - **Parameters**: # variables, max coefficient, structural measures of the matrix...
- Parameterized complexity: a way of thinking about measuring resource usage.

Parameterized complexity

- **Classic complexity theory**: Complexity measured in the size of input data.
  - **Main distinction**: P-time solvable vs. NP-complete
- **Parameterized complexity**: Express complexity using a function of the input size and auxiliary parameters.
  - Evaluation of query φ on database D.
    - **Parameters**: $||\varphi||$, size of output, ...
  - Searching for pattern of size k in a large graph G.
    - **Parameters**: k, $\Delta(G)$, $\mathrm{genus}(G)$, ...
  - Solving an integer linear program (ILP).
    - **Parameters**: # variables, max coefficient, structural measures of the matrix...
- Parameterized complexity: a way of thinking about measuring resource usage.
- Establishing formal framework $\rightsquigarrow$ Mathematical work on foundations

Notions of efficiency

- Fix a problem with one parameter k.

Notions of efficiency

- Fix a problem with one parameter k.
- **Idea**: Algorithm is efficient when the parameter is small.

Notions of efficiency

- Fix a problem with one parameter k.
- **Idea**: Algorithm is efficient when the parameter is small.
- **First try**: Problem is solvable in polynomial time whenever the parameter is fixed to a constant.

Notions of efficiency

- Fix a problem with one parameter k.
- **Idea**: Algorithm is efficient when the parameter is small.
- **First try**: Problem is solvable in polynomial time whenever the parameter is fixed to a constant.
  - **XP**: running time of the form $f(k) \cdot n^{g(k)}$ for some functions f, g.

Notions of efficiency

- Fix a problem with one parameter k.
- **Idea**: Algorithm is efficient when the parameter is small.
- **First try**: Problem is solvable in polynomial time whenever the parameter is fixed to a constant.
  - **XP**: running time of the form $f(k) \cdot n^{g(k)}$ for some functions f, g.
  - **Example**: Clique of size k can be found in time $\mathcal{O}(|G|^k)$.

Notions of efficiency

- Fix a problem with one parameter k.
- **Idea**: Algorithm is efficient when the parameter is small.
- **First try**: Problem is solvable in polynomial time whenever the parameter is fixed to a constant.
  - **XP**: running time of the form $f(k) \cdot n^{g(k)}$ for some functions f, g.
  - **Example**: Clique of size k can be found in time $\mathcal{O}(|G|^k)$.
- **Second try**: Whenever the parameter is a constant, the running time is linear/quadratic/cubic/...

Notions of efficiency

- Fix a problem with one parameter k.
- **Idea**: Algorithm is efficient when the parameter is small.
- **First try**: Problem is solvable in polynomial time whenever the parameter is fixed to a constant.
  - **XP**: running time of the form $f(k) \cdot n^{g(k)}$ for some functions f, g.
  - **Example**: Clique of size k can be found in time $\mathcal{O}(|G|^k)$.
- **Second try**: Whenever the parameter is a constant, the running time is linear/quadratic/cubic/...
  - **FPT**: running time of the form $f(k) \cdot n^c$ for some function f and constant c.

Notions of efficiency

- Fix a problem with one parameter k.
- **Idea**: Algorithm is efficient when the parameter is small.
- **First try**: Problem is solvable in polynomial time whenever the parameter is fixed to a constant.
  - **XP**: running time of the form $f(k) \cdot n^{g(k)}$ for some functions f, g.
  - **Example**: Clique of size k can be found in time $\mathcal{O}(|G|^k)$.
- **Second try**: Whenever the parameter is a constant, the running time is linear/quadratic/cubic/...
  - **FPT**: running time of the form $f(k) \cdot n^c$ for some function f and constant c.
  - **Example**: Path of length k can be found in time $\mathcal{O}(1.66^k \cdot |G| \log |G|)$.

Notions of efficiency

- Fix a problem with one parameter k.
- **Idea**: Algorithm is efficient when the parameter is small.
- **First try**: Problem is solvable in polynomial time whenever the parameter is fixed to a constant.
  - **XP**: running time of the form $f(k) \cdot n^{g(k)}$ for some functions f, g.
  - **Example**: Clique of size k can be found in time $\mathcal{O}(|G|^k)$.
- **Second try**: Whenever the parameter is a constant, the running time is linear/quadratic/cubic/...
  - **FPT**: running time of the form $f(k) \cdot n^c$ for some function f and constant c.
  - **Example**: Path of length k can be found in time $\mathcal{O}(1.66^k \cdot |G| \log |G|)$.
- **FPT vs. W-hard**: Clique is unlikely to be FPT.

Notions of efficiency

- Fix a problem with one parameter k.
- **Idea**: Algorithm is efficient when the parameter is small.
- **First try**: Problem is solvable in polynomial time whenever the parameter is fixed to a constant.
  - **XP**: running time of the form $f(k) \cdot n^{g(k)}$ for some functions f, g.
  - **Example**: Clique of size k can be found in time $\mathcal{O}(|G|^k)$.
- **Second try**: Whenever the parameter is a constant, the running time is linear/quadratic/cubic/...
  - **FPT**: running time of the form $f(k) \cdot n^c$ for some function f and constant c.
  - **Example**: Path of length k can be found in time $\mathcal{O}(1.66^k \cdot |G| \log |G|)$.
- **FPT vs. W-hard**: Clique is unlikely to be FPT.
- **Directions of research**:

Notions of efficiency

- Fix a problem with one parameter k.
- **Idea**: Algorithm is efficient when the parameter is small.
- **First try**: Problem is solvable in polynomial time whenever the parameter is fixed to a constant.
  - **XP**: running time of the form $f(k) \cdot n^{g(k)}$ for some functions f, g.
  - **Example**: Clique of size k can be found in time $\mathcal{O}(|G|^k)$.
- **Second try**: Whenever the parameter is a constant, the running time is linear/quadratic/cubic/...
  - **FPT**: running time of the form $f(k) \cdot n^c$ for some function f and constant c.
  - **Example**: Path of length k can be found in time $\mathcal{O}(1.66^k \cdot |G| \log |G|)$.
- **FPT vs. W-hard**: Clique is unlikely to be FPT.
- **Directions of research**:
  - Classification FPT vs. W-hard.

Notions of efficiency

- Fix a problem with one parameter k.
- **Idea**: Algorithm is efficient when the parameter is small.
- **First try**: Problem is solvable in polynomial time whenever the parameter is fixed to a constant.
  - **XP**: running time of the form $f(k) \cdot n^{g(k)}$ for some functions f, g.
  - **Example**: Clique of size k can be found in time $\mathcal{O}(|G|^k)$.
- **Second try**: Whenever the parameter is a constant, the running time is linear/quadratic/cubic/...
  - **FPT**: running time of the form $f(k) \cdot n^c$ for some function f and constant c.
  - **Example**: Path of length k can be found in time $\mathcal{O}(1.66^k \cdot |G| \log |G|)$.
- **FPT vs. W-hard**: Clique is unlikely to be FPT.
- **Directions of research**:
  - Classification FPT vs. W-hard.
  - Algorithms with as low parametric dependence as possible.

Notions of efficiency

- Fix a problem with one parameter k.
- **Idea**: Algorithm is efficient when the parameter is small.
- **First try**: Problem is solvable in polynomial time whenever the parameter is fixed to a constant.
  - **XP**: running time of the form $f(k) \cdot n^{g(k)}$ for some functions f, g.
  - **Example**: Clique of size k can be found in time $\mathcal{O}(|G|^k)$.
- **Second try**: Whenever the parameter is a constant, the running time is linear/quadratic/cubic/...
  - **FPT**: running time of the form $f(k) \cdot n^c$ for some function f and constant c.
  - **Example**: Path of length k can be found in time $\mathcal{O}(1.66^k \cdot |G| \log |G|)$.
- **FPT vs. W-hard**: Clique is unlikely to be FPT.
- **Directions of research**:
  - Classification FPT vs. W-hard.
  - Algorithms with as low parametric dependence as possible.
  - Algorithms with as low input size dependence as possible.

Significance and community

- A boom in the significance of Parameterized Complexity in recent years.

Significance and community

- A boom in the significance of Parameterized Complexity in recent years.
  - 13 out of 182 papers at SODA 2017 related to Parameterized Complexity.

Significance and community

- A boom in the significance of Parameterized Complexity in recent years.
  - 13 out of 182 papers at SODA 2017 related to Parameterized Complexity.
  - At least 6 ERC grants in progress or starting soon.

Significance and community

- A boom in the significance of Parameterized Complexity in recent years.
  - 13 out of 182 papers at SODA 2017 related to Parameterized Complexity.
  - At least 6 ERC grants in progress or starting soon.
  - By now, a mainstream research direction in TCS.

Significance and community

- A boom in the significance of Parameterized Complexity in recent years.
  - 13 out of 182 papers at SODA 2017 related to Parameterized Complexity.
  - At least 6 ERC grants in progress or starting soon.
  - By now, a mainstream research direction in TCS.
  - Predominantly a European topic.

Significance and community

- A boom in the significance of Parameterized Complexity in recent years.
  - 13 out of 182 papers at SODA 2017 related to Parameterized Complexity.
  - At least 6 ERC grants in progress or starting soon.
  - By now, a mainstream research direction in TCS.
  - Predominantly a European topic.
- Well-organized, closely collaborating community.

Significance and community

- A boom in the significance of Parameterized Complexity in recent years.
  - 13 out of 182 papers at SODA 2017 related to Parameterized Complexity.
  - At least 6 ERC grants in progress or starting soon.
  - By now, a mainstream research direction in TCS.
  - Predominantly a European topic.
- Well-organized, closely collaborating community.
  - http://fpt.wikidot.com/

Significance and community

- A boom in the significance of Parameterized Complexity in recent years.
  - 13 out of 182 papers at SODA 2017 related to Parameterized Complexity.
  - At least 6 ERC grants in progress or starting soon.
  - By now, a mainstream research direction in TCS.
  - Predominantly a European topic.
- Well-organized, closely collaborating community.
  - `http://fpt.wikidot.com/`
  - 4 established textbooks and monographs, focusing on different aspects.

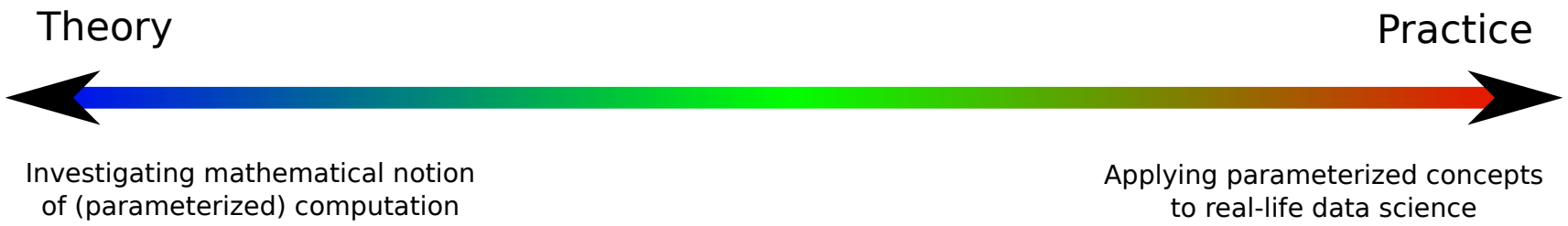
Significance and community

- A boom in the significance of Parameterized Complexity in recent years.
  - 13 out of 182 papers at SODA 2017 related to Parameterized Complexity.
  - At least 6 ERC grants in progress or starting soon.
  - By now, a mainstream research direction in TCS.
  - Predominantly a European topic.
- Well-organized, closely collaborating community.
  - `http://fpt.wikidot.com/`
  - 4 established textbooks and monographs, focusing on different aspects.
  - IPEC: International Symposium on Parameterized and Exact Computation

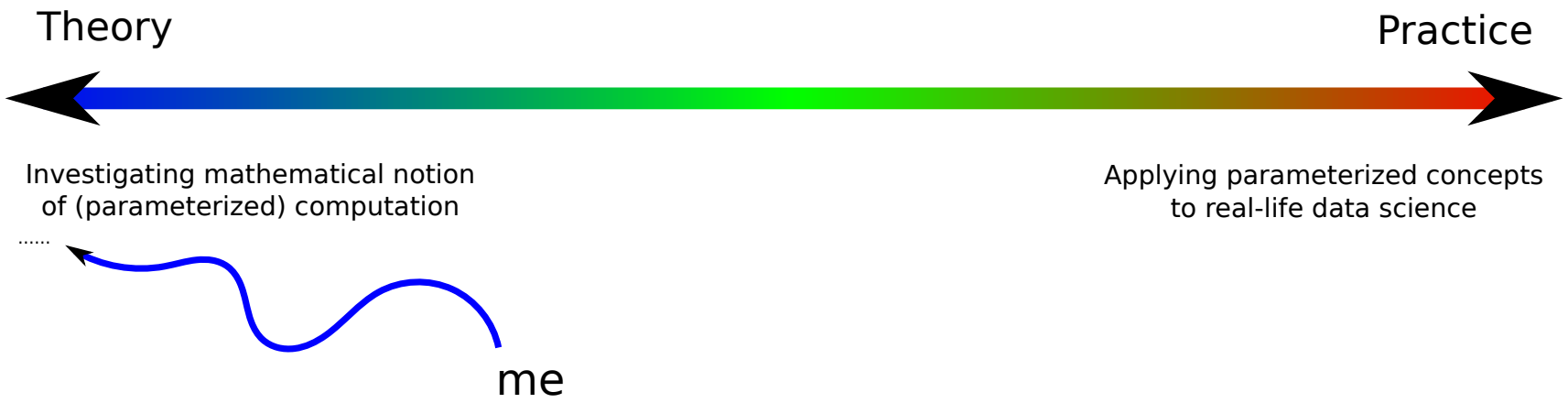
Significance and community

- A boom in the significance of Parameterized Complexity in recent years.
  - 13 out of 182 papers at SODA 2017 related to Parameterized Complexity.
  - At least 6 ERC grants in progress or starting soon.
  - By now, a mainstream research direction in TCS.
  - Predominantly a European topic.
- Well-organized, closely collaborating community.
  - `http://fpt.wikidot.com/`
  - 4 established textbooks and monographs, focusing on different aspects.
  - IPEC: International Symposium on Parameterized and Exact Computation
  - PACE: Parameterized Algorithms and Computational Experiments Challenge

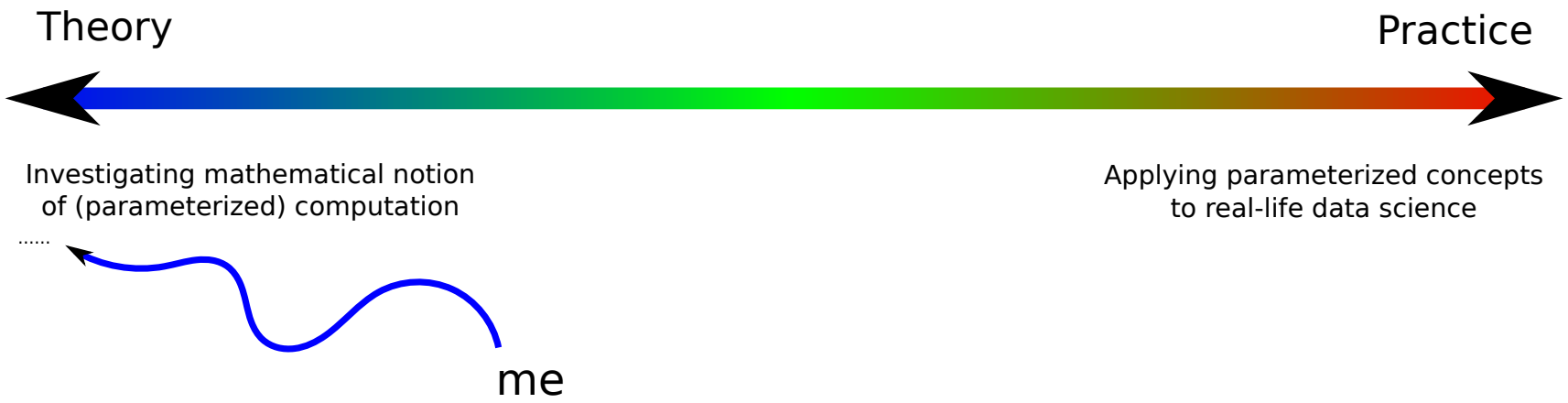
Plan for now


Theory
Practice
Investigating mathematical notion
of (parameterized) computation
Applying parameterized concepts
to real-life data science

Plan for now


Theory
Practice
Investigating mathematical notion
of (parameterized) computation
.....
me
Applying parameterized concepts
to real-life data science

Plan for now



- **Plan for now**: A few example concepts and results.

Graph modification problems

- **Setting**:

Graph modification problems

- **Setting**:
  - We are given some graph where we expect high structure.

Graph modification problems

- **Setting**:
  - We are given some graph where we expect high structure.
  - Data is noisy, so some adjacencies may be incorrect.

Graph modification problems

- **Setting**:
  - We are given some graph where we expect high structure.
  - Data is noisy, so some adjacencies may be incorrect.
    - The number of errors is expected to be small

Graph modification problems

- **Setting**:
  - We are given some graph where we expect high structure.
  - Data is noisy, so some adjacencies may be incorrect.
    - The number of errors is expected to be small
  - **Goal**: Find the smallest possible edit to correctly structured data.

Graph modification problems

- **Setting**:
  - We are given some graph where we expect high structure.
  - Data is noisy, so some adjacencies may be incorrect.
    - The number of errors is expected to be small
  - **Goal**: Find the smallest possible edit to correctly structured data.
- **Parameter**: k, the number of allowed edits.

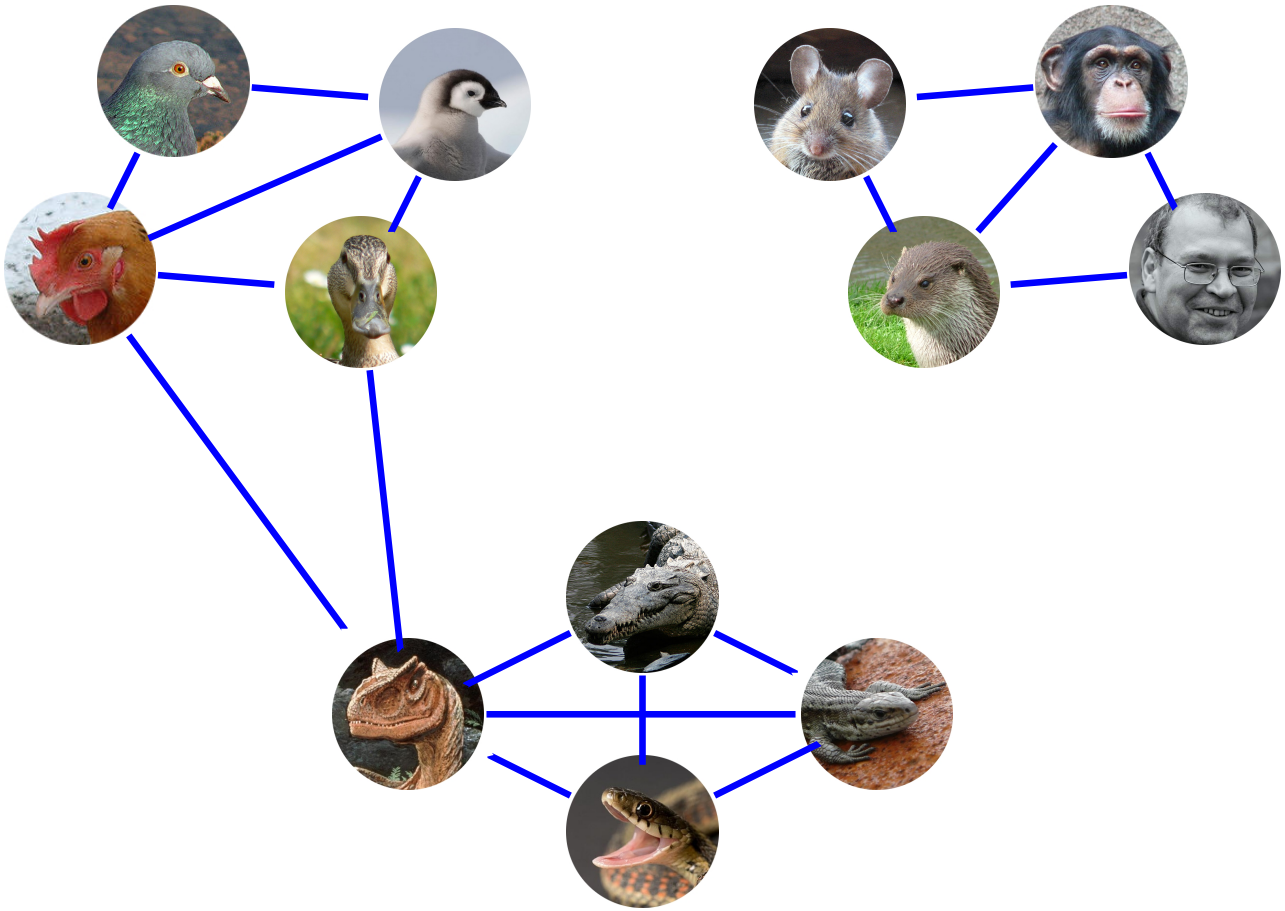
Graph modification problems

- **Setting**:
  - We are given some graph where we expect high structure.
  - Data is noisy, so some adjacencies may be incorrect.
    - The number of errors is expected to be small
  - **Goal**: Find the smallest possible edit to correctly structured data.
- **Parameter**: k, the number of allowed edits.
- **Variants**: Target class of structured graphs, allowed edit operations...

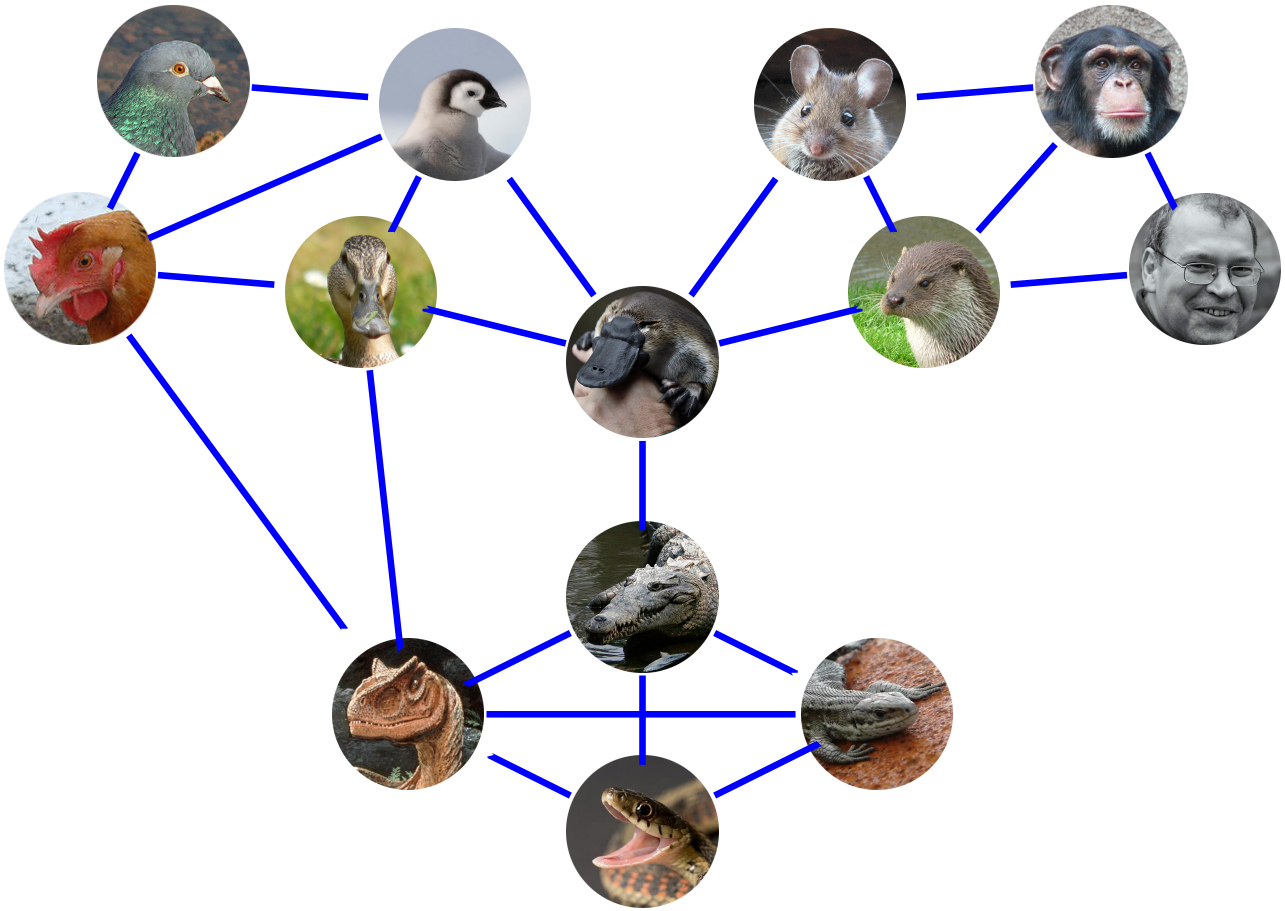
Animal classification

Animal classification

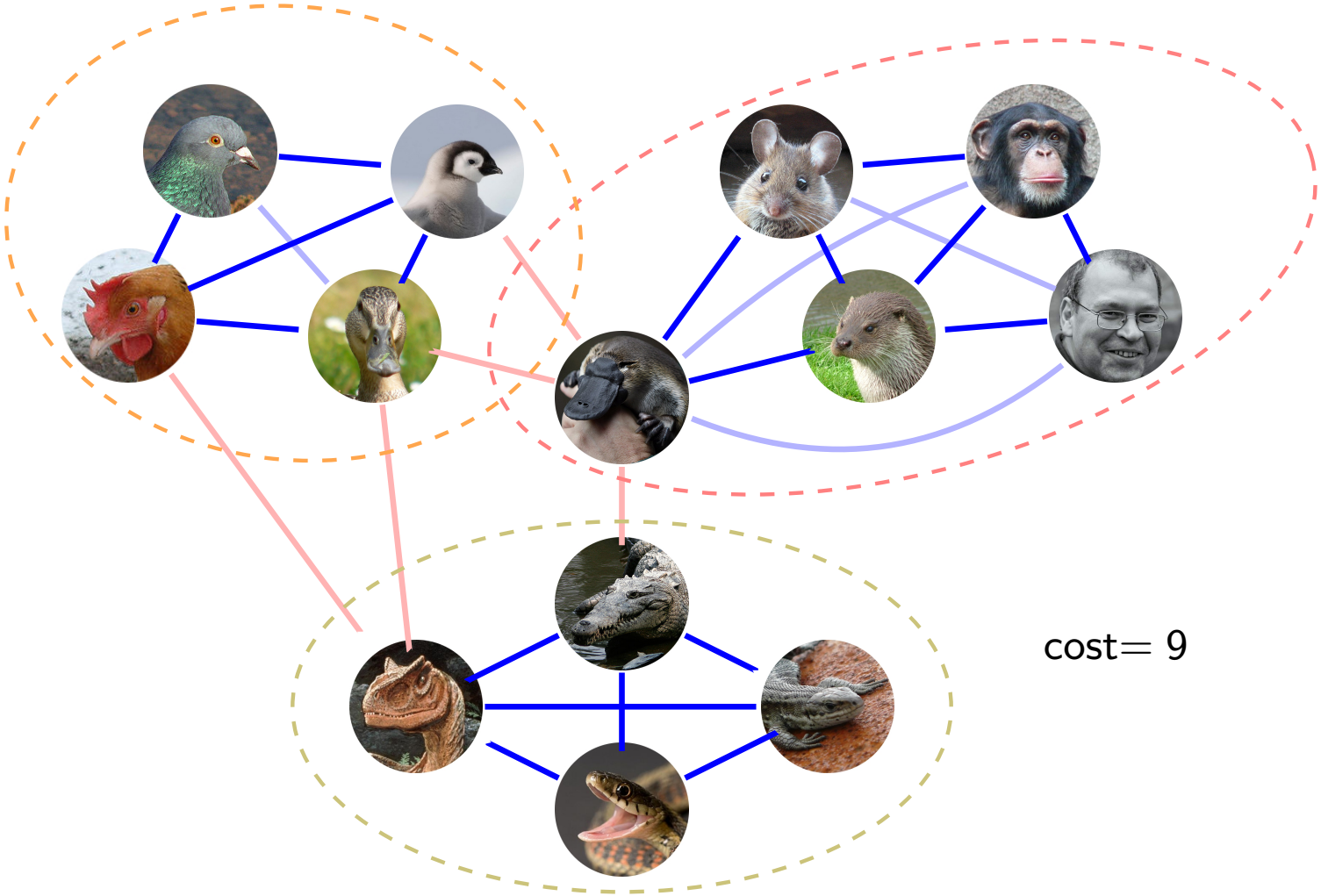
Animal classification



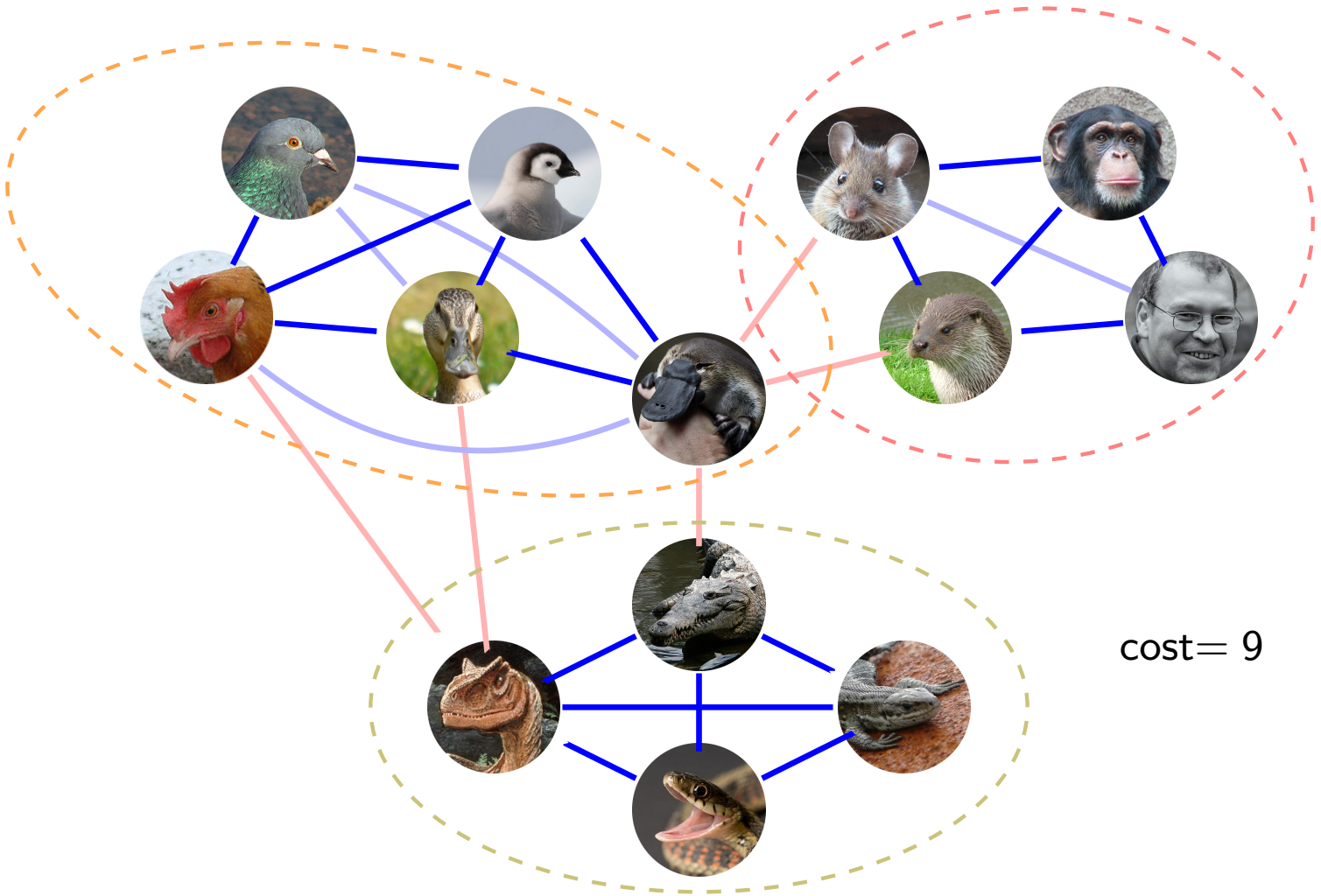
Animal classification



Animal classification



Animal classification



Cluster Editing

- **Cluster graph**: disjoint union of cliques

Cluster Editing

- **Cluster graph**: disjoint union of cliques
- Cluster Editing:
  Given a graph, add or remove at most k edges to obtain a cluster graph.

Cluster Editing

- **Cluster graph**: disjoint union of cliques
- Cluster Editing:
  Given a graph, add or remove at most k edges to obtain a cluster graph.
  - There is an FPT algorithm with running time $\mathcal{O}(1.62^k + n + m)$ [Böcker].

Cluster Editing

- **Cluster graph**: disjoint union of cliques
- Cluster Editing:
  Given a graph, add or remove at most k edges to obtain a cluster graph.
  - There is an FPT algorithm with running time $\mathcal{O}(1.62^k + n + m)$ [Böcker].
  - An algorithm with running time $2^{o(k)} \cdot n^c$ is unlikely [Komusiewicz, Uhlmann].

Cluster Editing

- **Cluster graph**: disjoint union of cliques
- Cluster Editing:
  Given a graph, add or remove at most k edges to obtain a cluster graph.
  - There is an FPT algorithm with running time $\mathcal{O}(1.62^k + n + m)$ [Böcker].
  - An algorithm with running time $2^{o(k)} \cdot n^c$ is unlikely [Komusiewicz, Uhlmann].
- What if the target number of clusters p is small compared to k?

Cluster Editing

- **Cluster graph**: disjoint union of cliques
- Cluster Editing:
  Given a graph, add or remove at most k edges to obtain a cluster graph.
  - There is an FPT algorithm with running time $\mathcal{O}(1.62^k + n + m)$ [Böcker].
  - An algorithm with running time $2^{o(k)} \cdot n^c$ is unlikely [Komusiewicz, Uhlmann].
- What if the target number of clusters p is small compared to k?

Theorem Fomin, Kratsch, Pilipczuk, Pilipczuk, Villanger; 2013

Cluster Editing can be solved in time $2^{\mathcal{O}(\sqrt{pk})} + \mathcal{O}(n+m)$, where p is the target number of clusters.

Moreover, this running time is likely to be tight for every p as a function of k.

Graph modification problems: complexity

- **Current understanding**:

Graph modification problems: complexity

- **Current understanding**:
- The vast majority of "reasonable" graph modification problems admit FPT algorithms with running times of the form $2^{\mathcal{O}(k)} \cdot n^c$...

Graph modification problems: complexity

- **Current understanding**:
- The vast majority of "reasonable" graph modification problems admit FPT algorithms with running times of the form $2^{\mathcal{O}(k)} \cdot n^c$...
- which are likely to be essentially tight: there is no $2^{o(k)} \cdot n^c$ algorithm.

Graph modification problems: complexity

- **Current understanding**:
- The vast majority of "reasonable" graph modification problems admit FPT algorithms with running times of the form $2^{\mathcal{O}(k)} \cdot n^c$...
- which are likely to be essentially tight: there is no $2^{o(k)} \cdot n^c$ algorithm.
- There are isolated examples where running time $2^{\tilde{\mathcal{O}}(\sqrt{k})} \cdot n^c$ is possible.

Graph modification problems: complexity

- **Current understanding**:
- The vast majority of "reasonable" graph modification problems admit FPT algorithms with running times of the form $2^{\mathcal{O}(k)} \cdot n^c$...
- which are likely to be essentially tight: there is no $2^{o(k)} \cdot n^c$ algorithm.
- There are isolated examples where running time $2^{\tilde{\mathcal{O}}(\sqrt{k})} \cdot n^c$ is possible.
  - Cluster Editing with a constant number of clusters,

Graph modification problems: complexity

- **Current understanding**:
- The vast majority of "reasonable" graph modification problems admit FPT algorithms with running times of the form $2^{\mathcal{O}(k)} \cdot n^c$...
- which are likely to be essentially tight: there is no $2^{o(k)} \cdot n^c$ algorithm.
- There are isolated examples where running time $2^{\tilde{\mathcal{O}}(\sqrt{k})} \cdot n^c$ is possible.
  - Cluster Editing with a constant number of clusters,
  - Chordal Completion,

Graph modification problems: complexity

- **Current understanding**:
- The vast majority of "reasonable" graph modification problems admit FPT algorithms with running times of the form $2^{\mathcal{O}(k)} \cdot n^c$...
- which are likely to be essentially tight: there is no $2^{o(k)} \cdot n^c$ algorithm.
- There are isolated examples where running time $2^{\tilde{\mathcal{O}}(\sqrt{k})} \cdot n^c$ is possible.
  - Cluster Editing with a constant number of clusters,
  - Chordal Completion,
  - Interval Completion,

Graph modification problems: complexity

- **Current understanding**:
- The vast majority of "reasonable" graph modification problems admit FPT algorithms with running times of the form $2^{\mathcal{O}(k)} \cdot n^c$...
- which are likely to be essentially tight: there is no $2^{o(k)} \cdot n^c$ algorithm.
- There are isolated examples where running time $2^{\tilde{\mathcal{O}}(\sqrt{k})} \cdot n^c$ is possible.
  - Cluster Editing with a constant number of clusters,
  - Chordal Completion,
  - Interval Completion,
  - Trivially Perfect Completion,

Graph modification problems: complexity

- **Current understanding**:
- The vast majority of "reasonable" graph modification problems admit FPT algorithms with running times of the form $2^{\mathcal{O}(k)} \cdot n^c$...
- which are likely to be essentially tight: there is no $2^{o(k)} \cdot n^c$ algorithm.
- There are isolated examples where running time $2^{\tilde{\mathcal{O}}(\sqrt{k})} \cdot n^c$ is possible.
  - Cluster Editing with a constant number of clusters,
  - Chordal Completion,
  - Interval Completion,
  - Trivially Perfect Completion,
  - and a few others connected to chordal graphs.

Graph modification problems: complexity

- **Current understanding**:
- The vast majority of "reasonable" graph modification problems admit FPT algorithms with running times of the form $2^{\mathcal{O}(k)} \cdot n^c$...
- which are likely to be essentially tight: there is no $2^{o(k)} \cdot n^c$ algorithm.
- There are isolated examples where running time $2^{\tilde{\mathcal{O}}(\sqrt{k})} \cdot n^c$ is possible.
  - Cluster Editing with a constant number of clusters,
  - Chordal Completion,
  - Interval Completion,
  - Trivially Perfect Completion,
  - and a few others connected to chordal graphs.
- This running time is again essentially tight.

Treewidth and tree decompositions

- **Idea**: Many hard problems become easy when the input structure is a tree.

Treewidth and tree decompositions

- **Idea**: Many hard problems become easy when the input structure is a tree.
- Maybe if the input is "tree-like", then we can still design some efficient algorithms using the same principles?

Treewidth and tree decompositions

- **Idea**: Many hard problems become easy when the input structure is a tree.
- Maybe if the input is "tree-like", then we can still design some efficient algorithms using the same principles?
- **Parameterized Complexity**:

Treewidth and tree decompositions

- **Idea**: Many hard problems become easy when the input structure is a tree.
- Maybe if the input is "tree-like", then we can still design some efficient algorithms using the same principles?
- **Parameterized Complexity**:
  - Define a graph parameter **treelikeness**(G) resemblance to a tree.

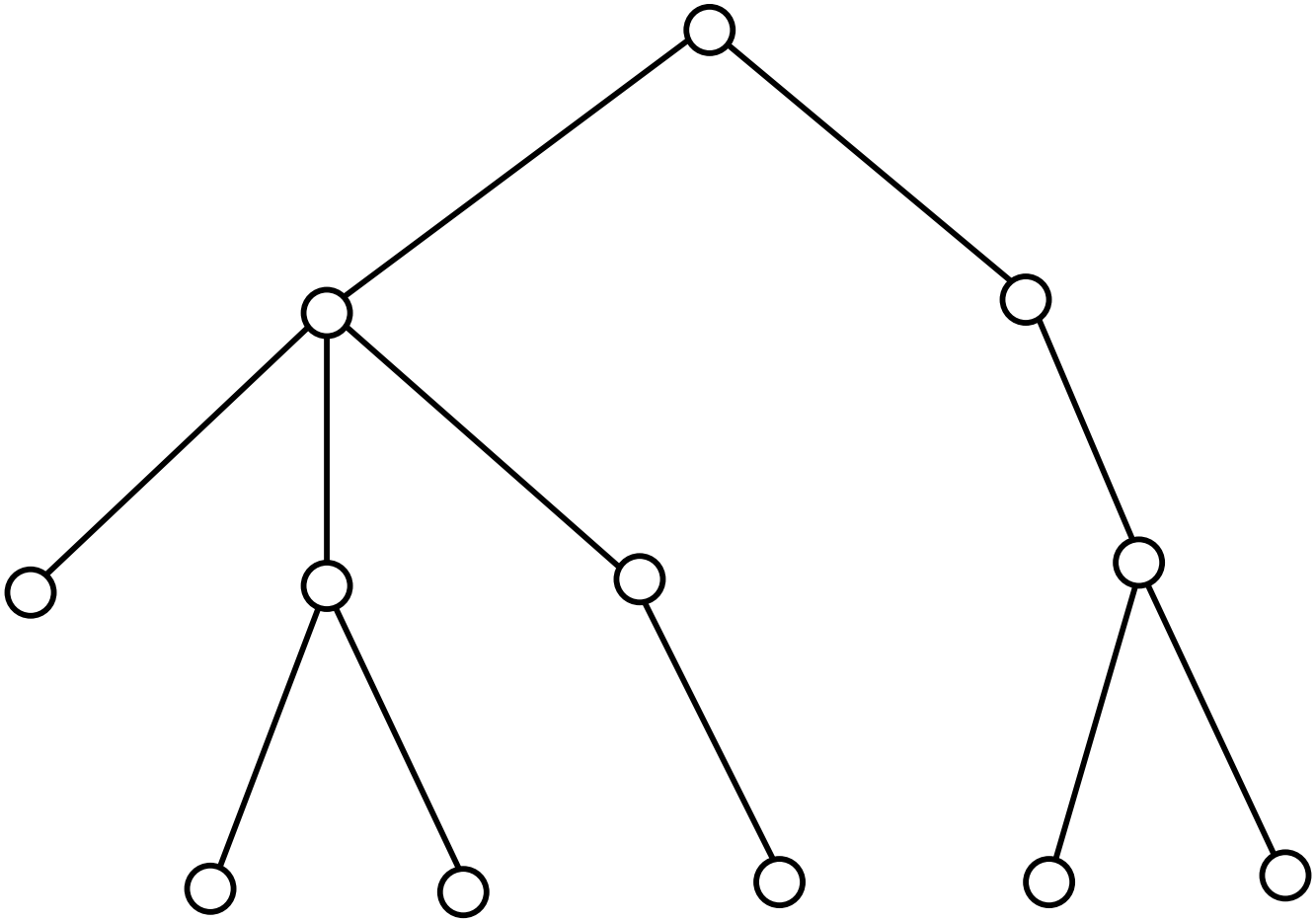
Treewidth and tree decompositions

- **Idea**: Many hard problems become easy when the input structure is a tree.
- Maybe if the input is "tree-like", then we can still design some efficient algorithms using the same principles?
- **Parameterized Complexity**:
  - Define a graph parameter **treelikeness**(G) resemblance to a tree.
  - Express the running time in terms of the size of the input and its tree-likeness.

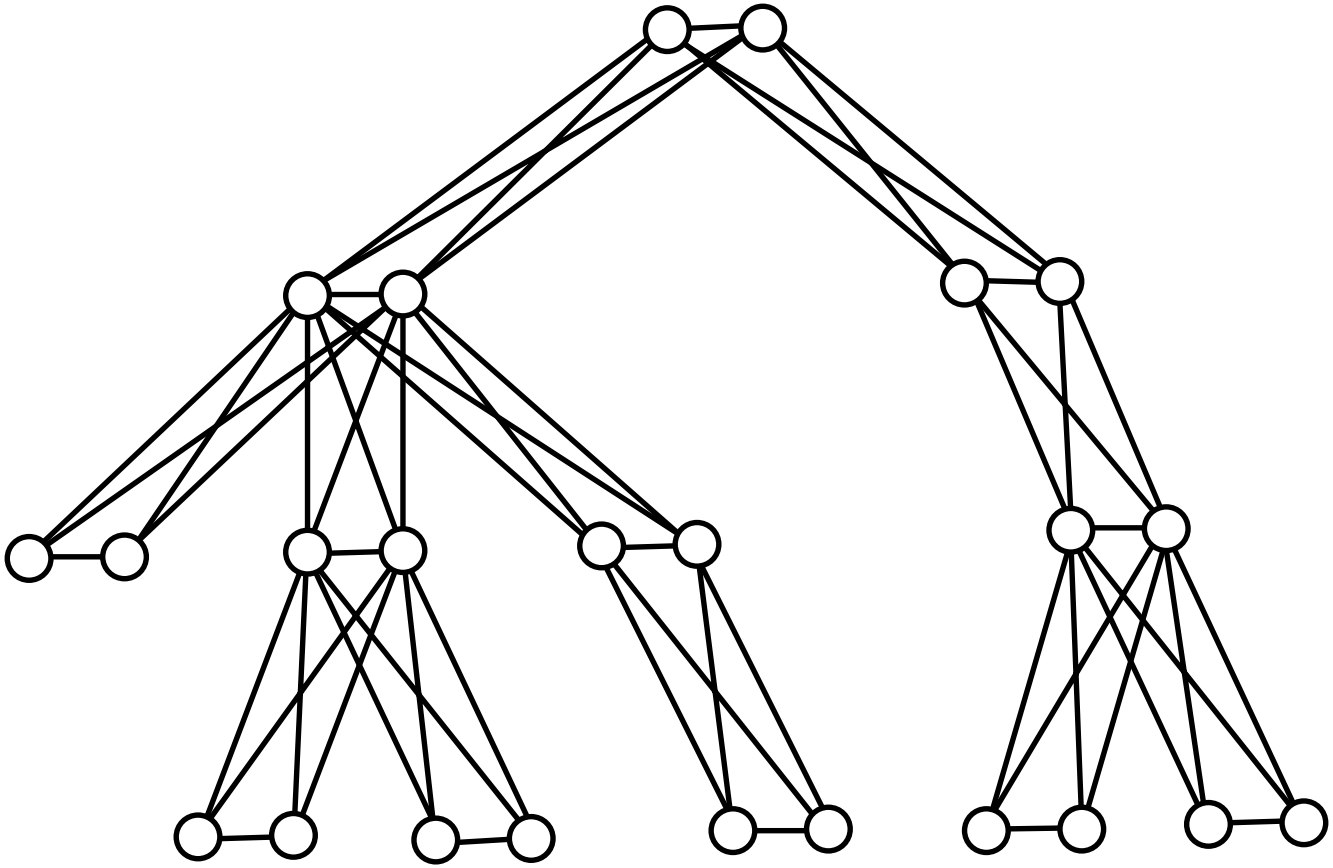
Treewidth and tree decompositions

- **Idea**: Many hard problems become easy when the input structure is a tree.
- Maybe if the input is "tree-like", then we can still design some efficient algorithms using the same principles?
- **Parameterized Complexity**:
  - Define a graph parameter **treelikeness**(G) resemblance to a tree.
  - Express the running time in terms of the size of the input and its tree-likeness.
- Standard measure for tree-likeness is the **treewidth** of a graph.

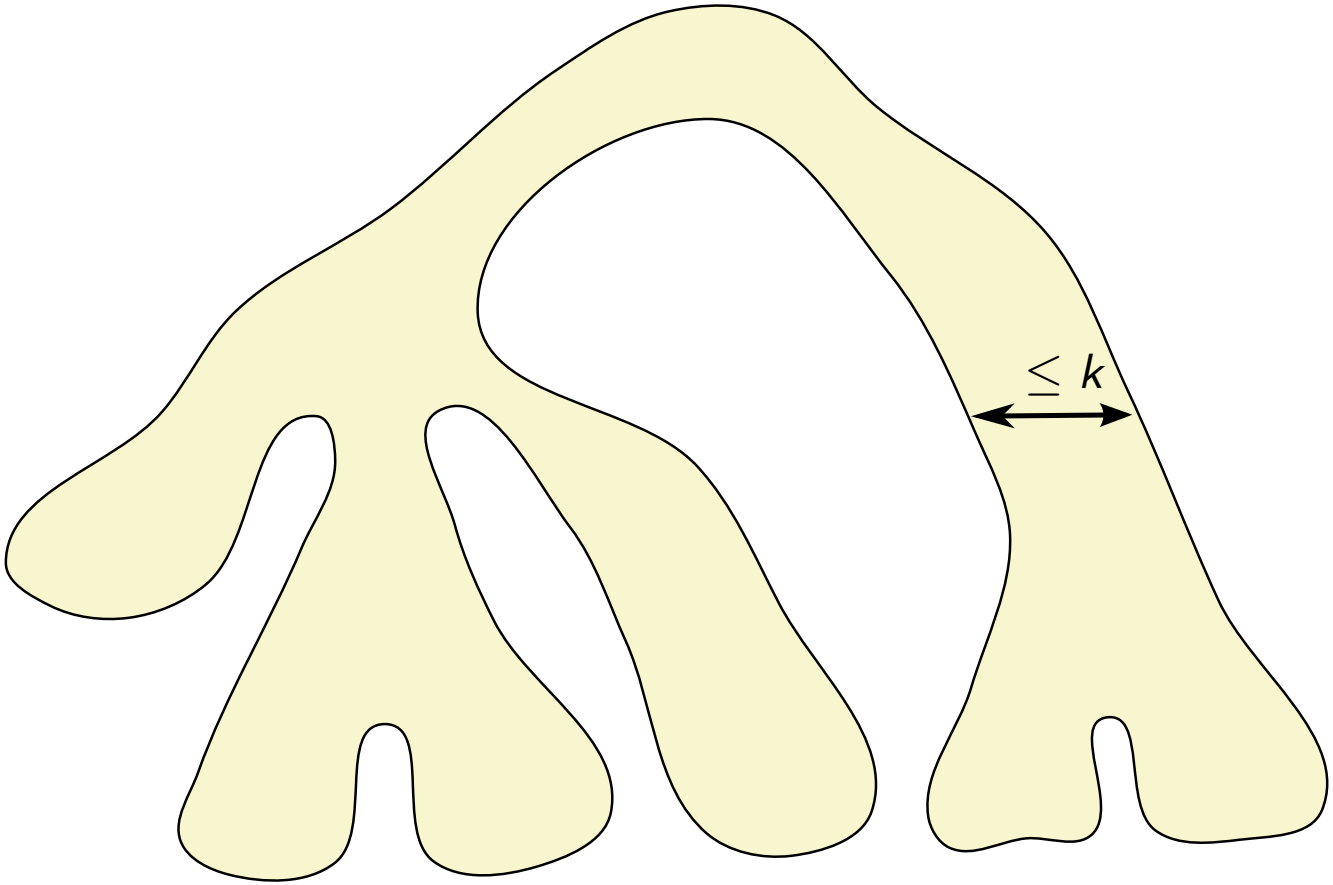
Treelike graphs



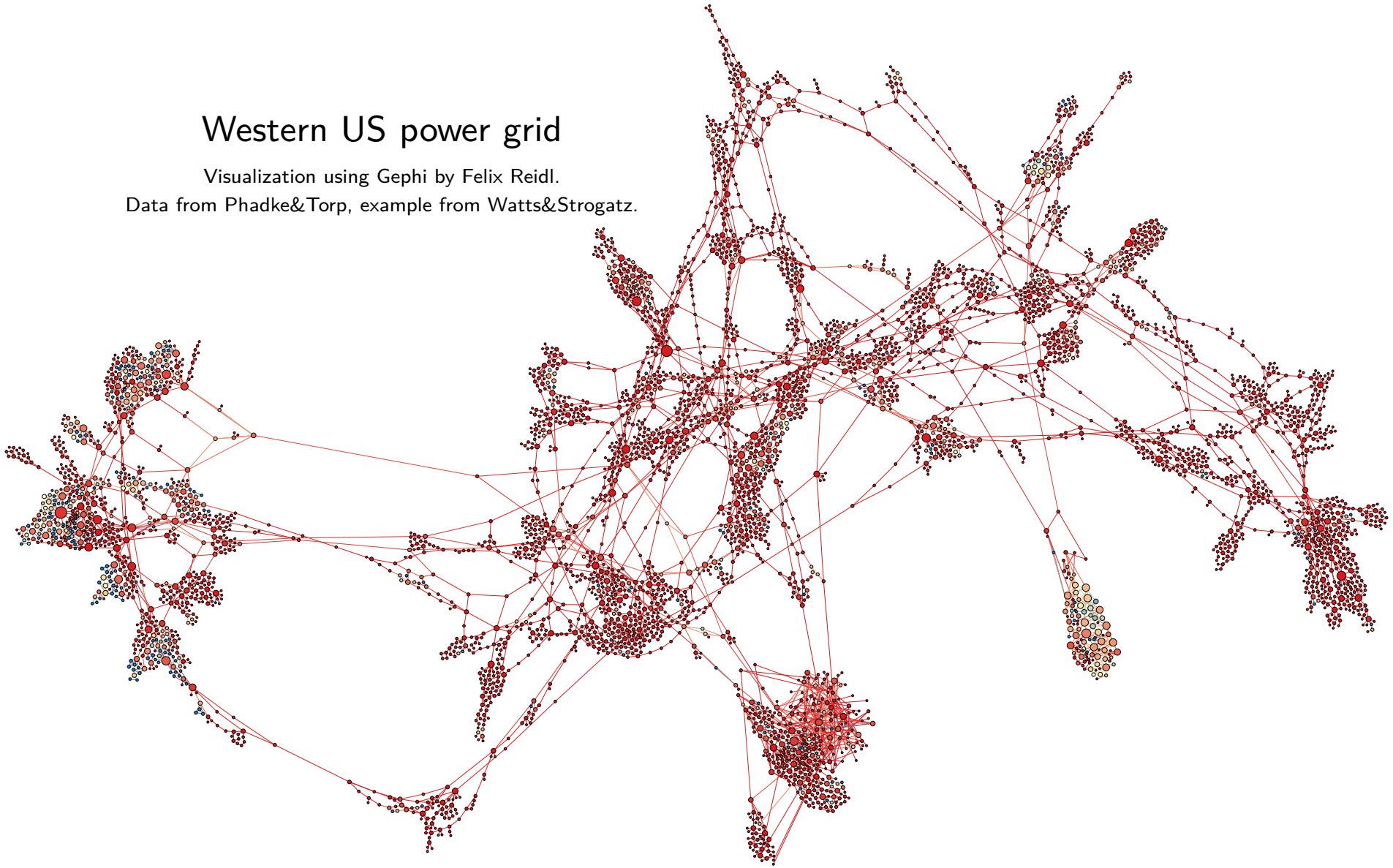
Treelike graphs



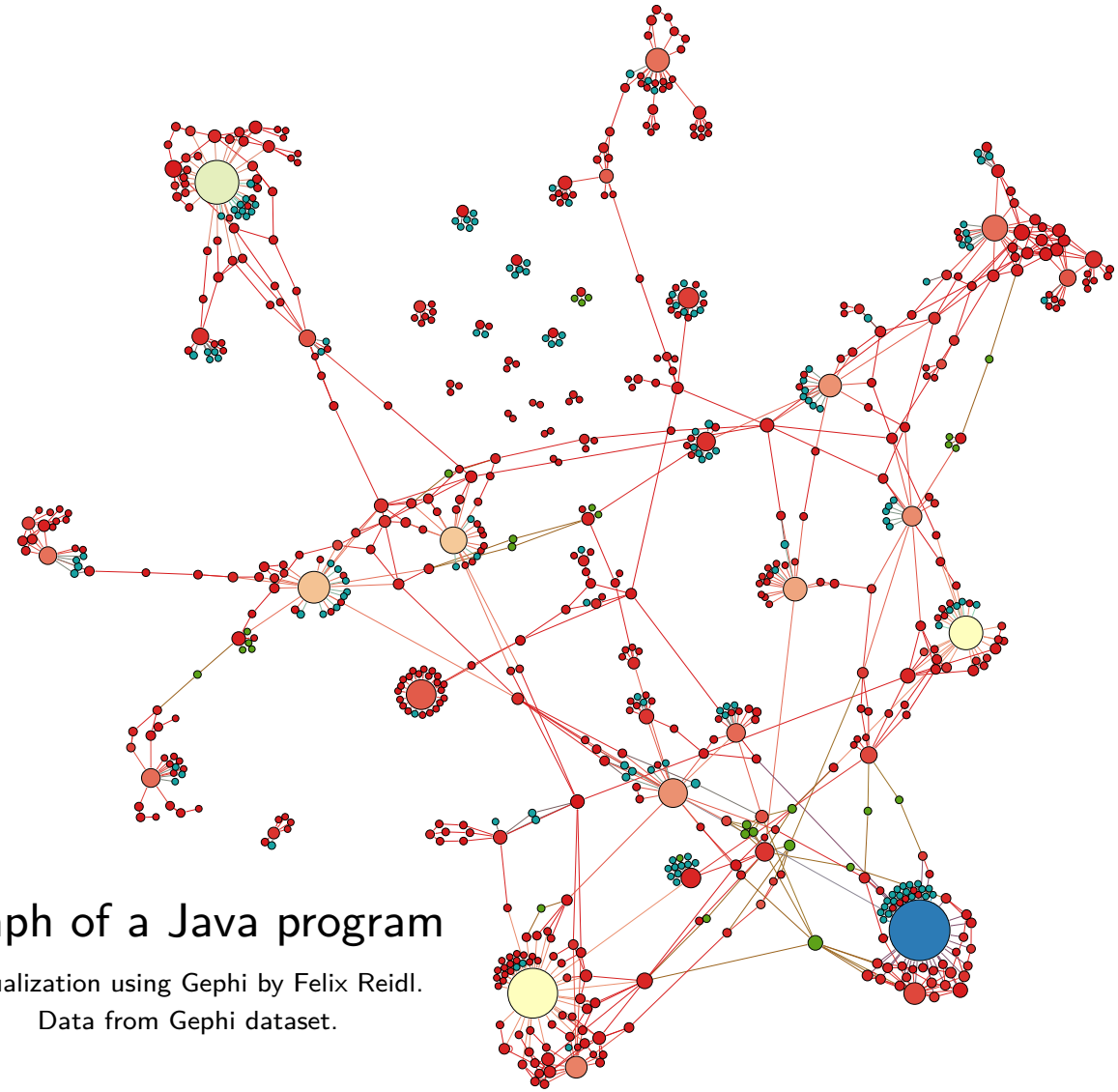
Treelike graphs



Treelike networks



Treelike networks



Graph of a Java program

Visualization using Gephi by Felix Reidl.
Data from Gephi dataset.

Treewidth: challenges

- **Main challenges**:

Treewidth: challenges

- **Main challenges**:
  - How efficiently can we compute a tree decomposition?

Treewidth: challenges

- **Main challenges**:
  - How efficiently can we compute a tree decomposition?
    - NP-hard in general, but there are FPT and FPT-approximation algorithms.

Treewidth: challenges

- **Main challenges**:
  - How efficiently can we compute a tree decomposition?
    - NP-hard in general, but there are FPT and FPT-approximation algorithms.
  - How efficiently can we solve our favourite problem given a tree decomposition of small width?

Treewidth: challenges

- **Main challenges**:
  - How efficiently can we compute a tree decomposition?
    - NP-hard in general, but there are FPT and FPT-approximation algorithms.
  - How efficiently can we solve our favourite problem given a tree decomposition of small width?
    - Dynamic programming, algebraic approach...

Treewidth: challenges

- **Main challenges**:
  - How efficiently can we compute a tree decomposition?
    - NP-hard in general, but there are FPT and FPT-approximation algorithms.
  - How efficiently can we solve our favourite problem given a tree decomposition of small width?
    - Dynamic programming, algebraic approach...
    - Connections to logic on graphs (MSO).

Treewidth: challenges

- **Main challenges**:
  - How efficiently can we compute a tree decomposition?
    - NP-hard in general, but there are FPT and FPT-approximation algorithms.
  - How efficiently can we solve our favourite problem given a tree decomposition of small width?
    - Dynamic programming, algebraic approach...
    - Connections to logic on graphs (MSO).
- Not only for NP-hard problems!

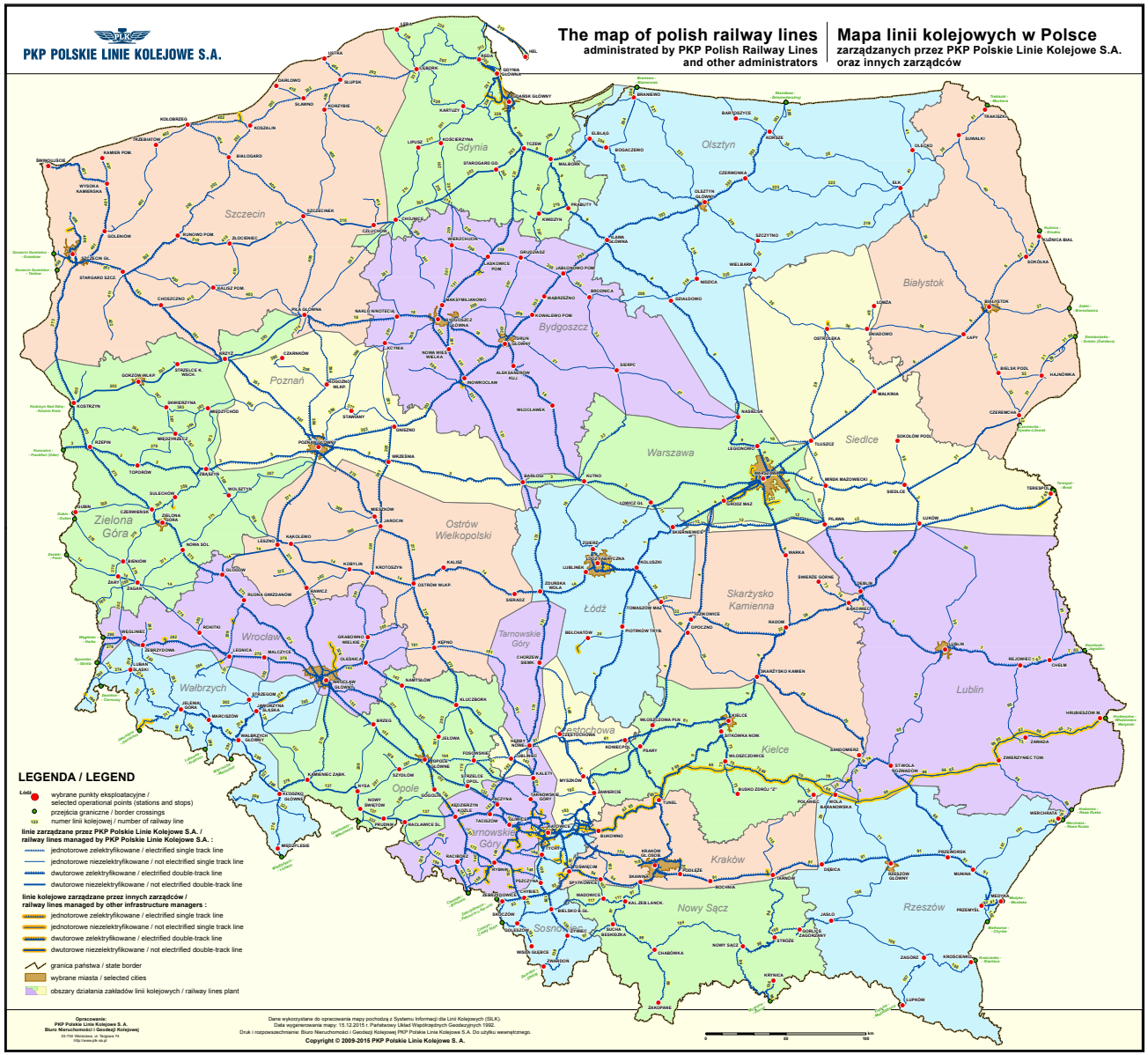
Treewidth: challenges

- **Main challenges**:
  - How efficiently can we compute a tree decomposition?
    - NP-hard in general, but there are FPT and FPT-approximation algorithms.
  - How efficiently can we solve our favourite problem given a tree decomposition of small width?
    - Dynamic programming, algebraic approach...
    - Connections to logic on graphs (MSO).
- Not only for NP-hard problems!
  - (Fomin, Lokshtanov, Pilipczuk, Saurabh, Wrochna; 2015+): Algorithms with running time $\mathrm{poly}(\mathbf{tw}) \cdot n$ or $\mathrm{poly}(\mathbf{tw}) \cdot n \log n$ for problems like maximum matching, maximum flow, solving systems of linear equations...

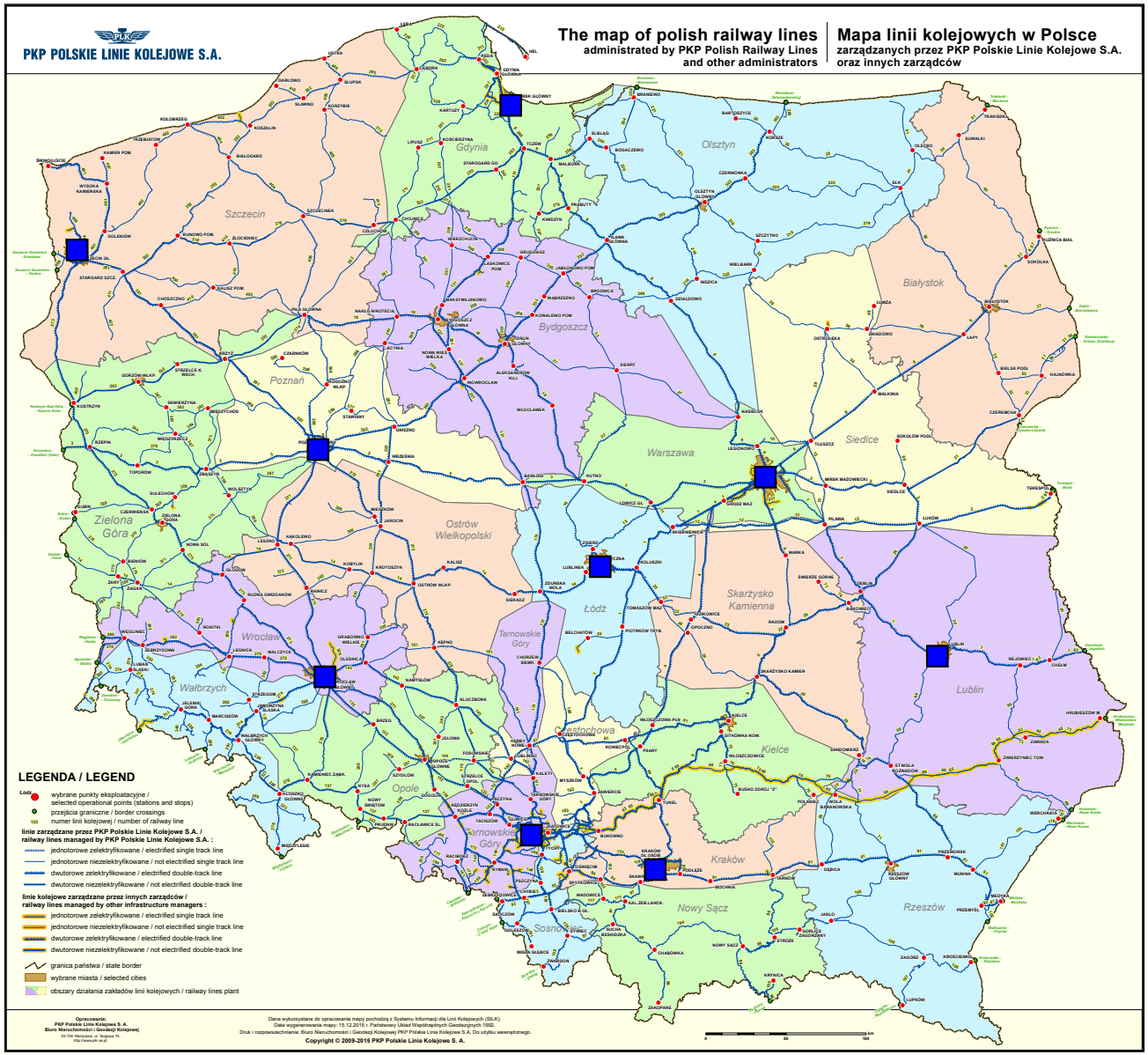
Treewidth: challenges

- **Main challenges**:
  - How efficiently can we compute a tree decomposition?
    - NP-hard in general, but there are FPT and FPT-approximation algorithms.
  - How efficiently can we solve our favourite problem given a tree decomposition of small width?
    - Dynamic programming, algebraic approach...
    - Connections to logic on graphs (MSO).
- Not only for NP-hard problems!
  - (Fomin, Lokshtanov, Pilipczuk, Saurabh, Wrochna; 2015+):
    Algorithms with running time $\mathrm{poly}(\mathbf{tw}) \cdot n$ or $\mathrm{poly}(\mathbf{tw}) \cdot n \log n$ for problems like maximum matching, maximum flow, solving systems of linear equations...
- **Practice**: Need for a reliable library for treewidth-based algorithms.

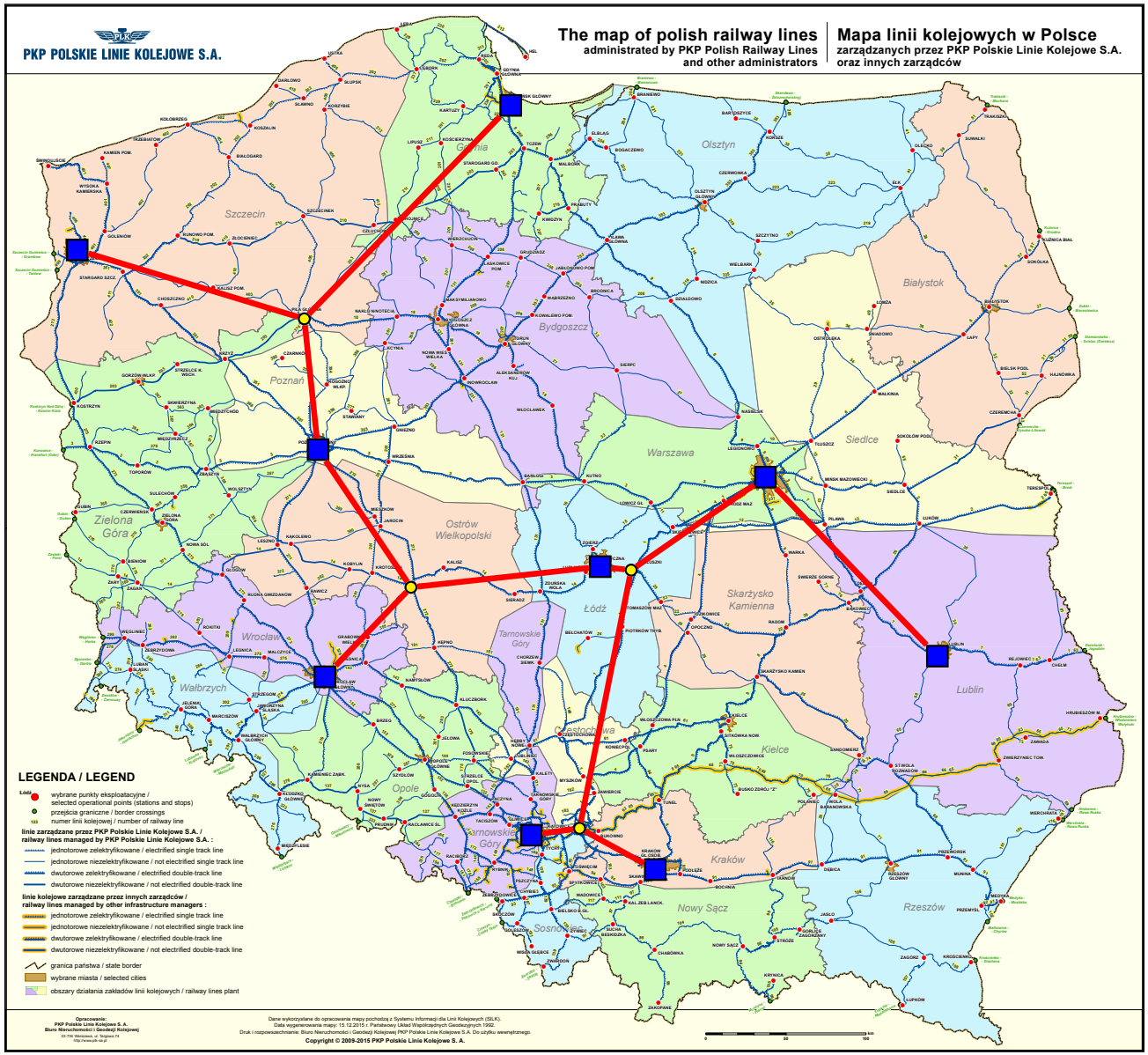
Steiner Tree



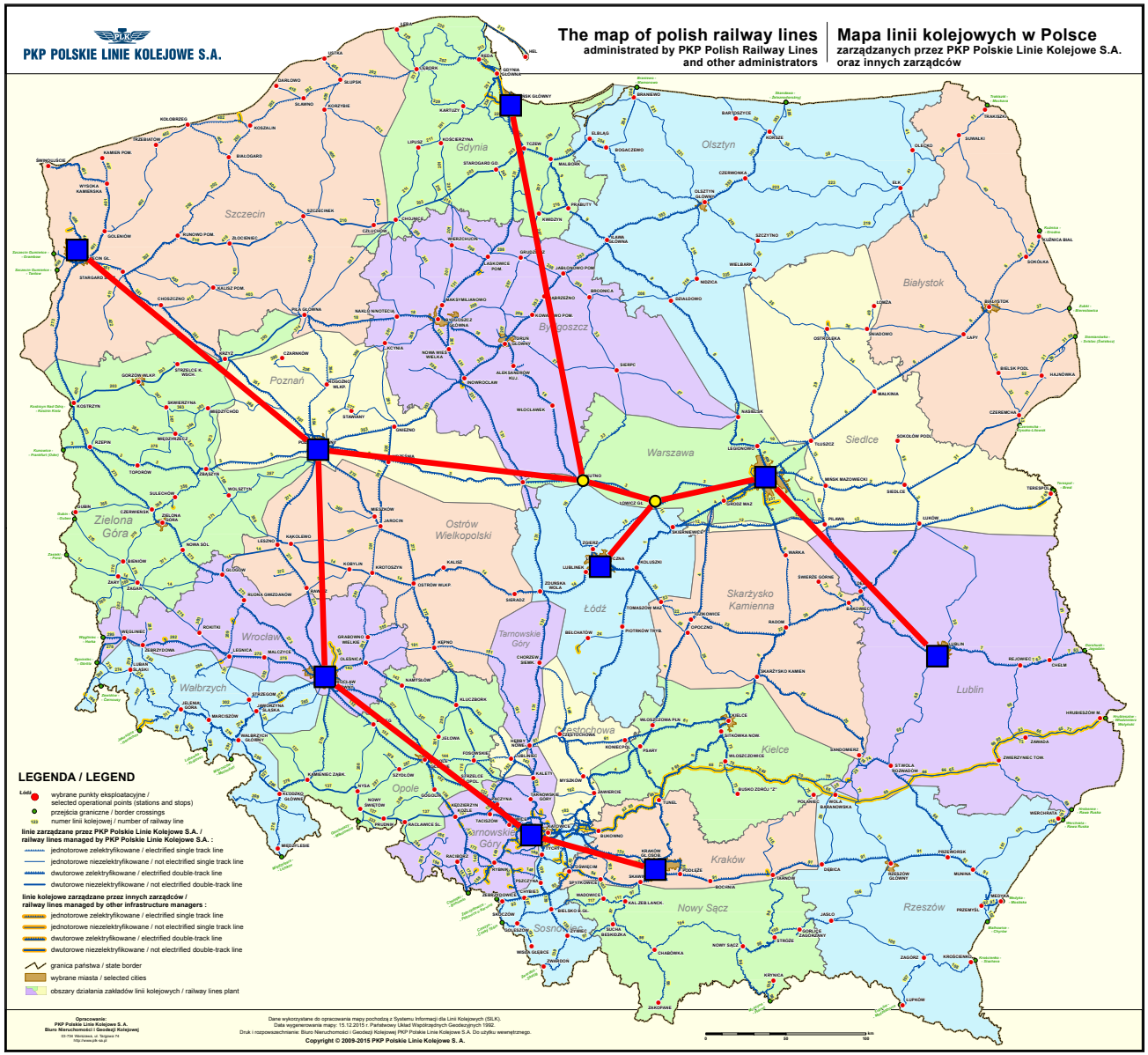
Steiner Tree



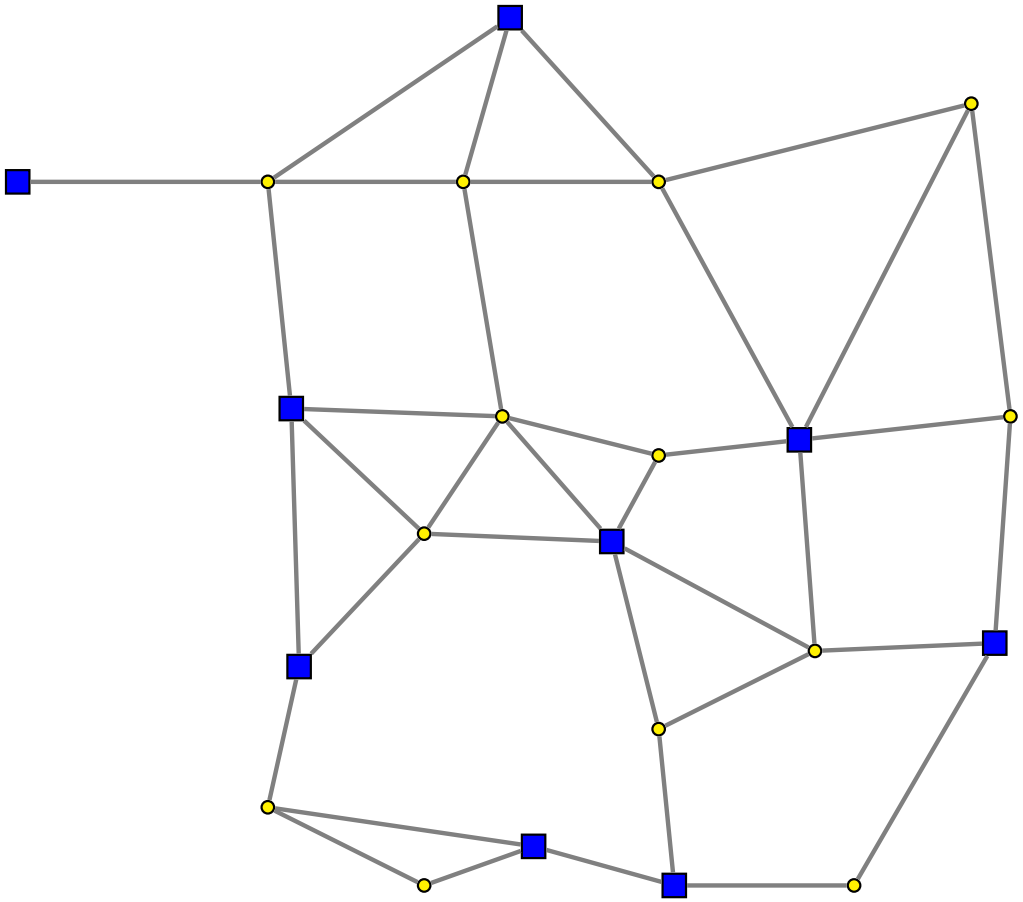
Steiner Tree



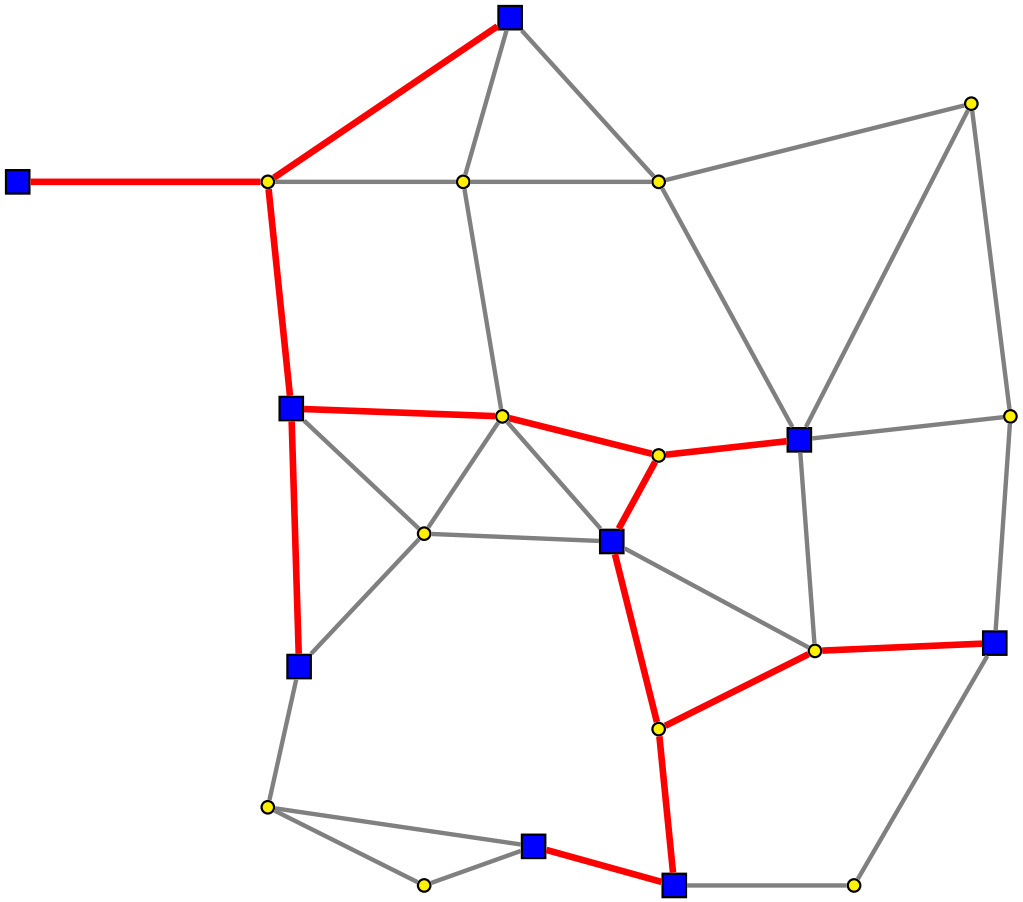
Steiner Tree



Steiner Tree



Steiner Tree



Steiner Tree: formalization

- **Statement**: Given an edge-weighted graph and a terminal set, find the shortest tree that connects the terminals.

Steiner Tree: formalization

- **Statement**: Given an edge-weighted graph and a terminal set, find the shortest tree that connects the terminals.
- Two natural parameters:

Steiner Tree: formalization

- **Statement**: Given an edge-weighted graph and a terminal set, find the shortest tree that connects the terminals.
- Two natural parameters:
  - $|T|$: the number of terminals.

Steiner Tree: formalization

- **Statement**: Given an edge-weighted graph and a terminal set, find the shortest tree that connects the terminals.
- Two natural parameters:
  - $|T|$: the number of terminals.
  - k: the target size of the tree (unit edge cost).

Steiner Tree: formalization

- **Statement**: Given an edge-weighted graph and a terminal set, find the shortest tree that connects the terminals.
- Two natural parameters:
  - $|T|$: the number of terminals.
  - k: the target size of the tree (unit edge cost).
- (Dreyfus & Wagner; 1972): Algorithm with running time $\mathcal{O}(3^{|T|} \cdot |G| \log |G|)$

Steiner Tree: formalization

- **Statement**: Given an edge-weighted graph and a terminal set, find the shortest tree that connects the terminals.
- Two natural parameters:
  - $|T|$: the number of terminals.
  - k: the target size of the tree (unit edge cost).
- (Dreyfus & Wagner; 1972): Algorithm with running time $\mathcal{O}(3^{|T|} \cdot |G| \log |G|)$
- Can we do anything better on planar graphs?

Steiner Tree: formalization

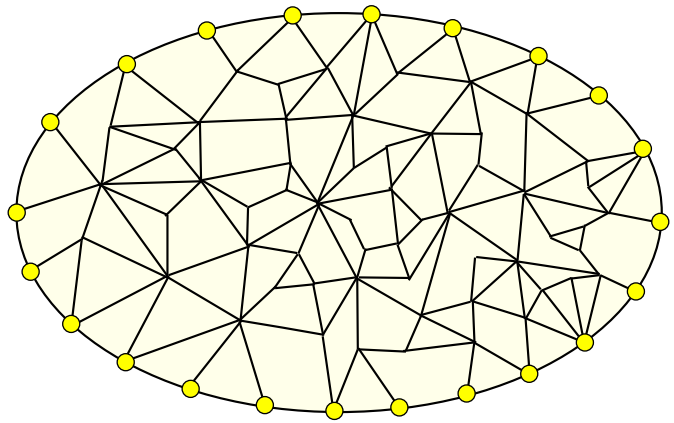
- **Statement**: Given an edge-weighted graph and a terminal set, find the shortest tree that connects the terminals.
- Two natural parameters:
  - $|T|$: the number of terminals.
  - k: the target size of the tree (unit edge cost).
- (Dreyfus & Wagner; 1972): Algorithm with running time $\mathcal{O}(3^{|T|} \cdot |G| \log |G|)$
- Can we do anything better on planar graphs?
- (Pilipczuk, Pilipczuk, Sankowski, van Leeuwen; 2014): Running time $2^{\mathcal{O}(\sqrt{k \log k})} \cdot |G|$ on unweighted planar graphs.

Steiner Tree: formalization

- **Statement**: Given an edge-weighted graph and a terminal set, find the shortest tree that connects the terminals.
- Two natural parameters:
  - $|T|$: the number of terminals.
  - k: the target size of the tree (unit edge cost).
- (Dreyfus & Wagner; 1972): Algorithm with running time $\mathcal{O}(3^{|T|} \cdot |G| \log |G|)$
- Can we do anything better on planar graphs?
- (Pilipczuk, Pilipczuk, Sankowski, van Leeuwen; 2014):
  Running time $2^{\mathcal{O}(\sqrt{k \log k})} \cdot |G|$ on unweighted planar graphs.
- Something about the parameterization by $|T|$ coming soon...

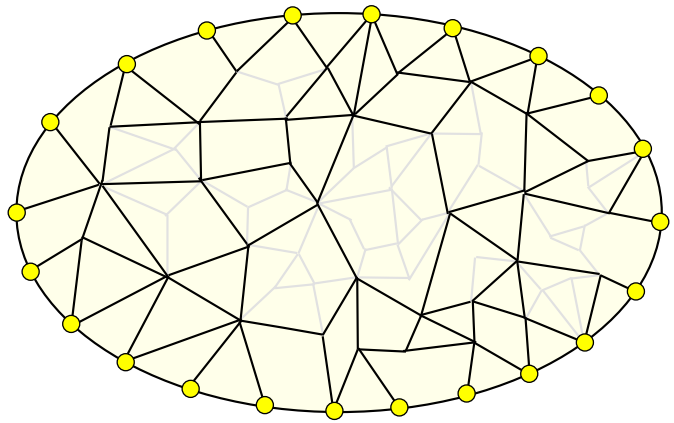
Planar Steiner Tree: sparsification

- **Given**: Planar graph G with outer face of length k.



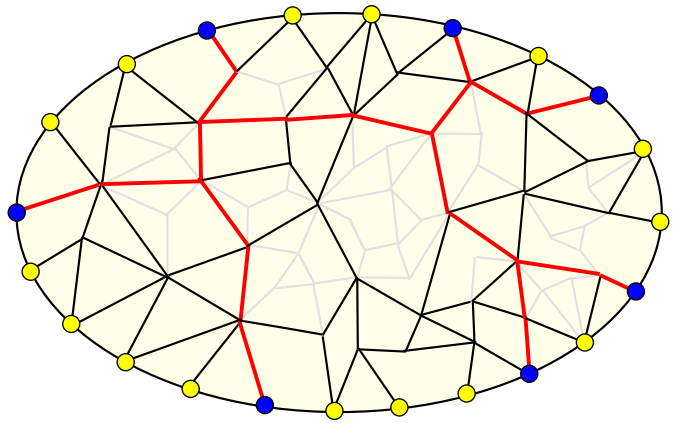
Planar Steiner Tree: sparsification

- **Given**: Planar graph G with outer face of length k.
- **Outcome**: Small subgraph $H \subseteq G$ with the following property:



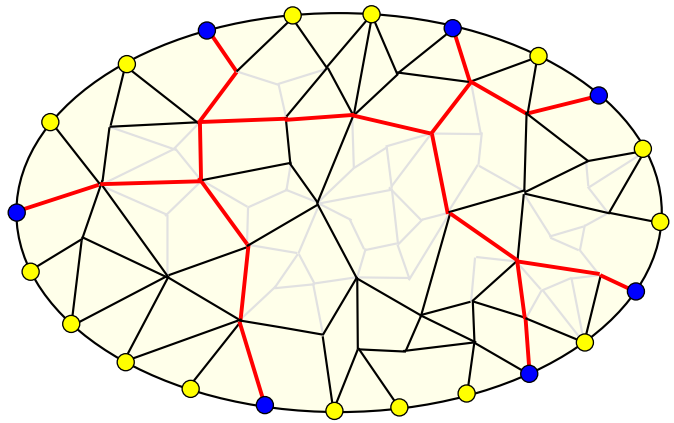
Planar Steiner Tree: sparsification

- **Given**: Planar graph G with outer face of length k.
- **Outcome**: Small subgraph $H \subseteq G$ with the following property:
  - For any choice of terminals on outer face, some opt. connection is preserved.



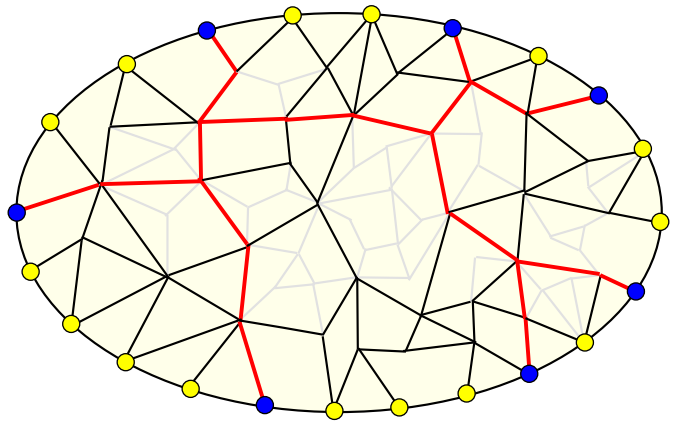
Planar Steiner Tree: sparsification

- **Given**: Planar graph G with outer face of length k.
- **Outcome**: Small subgraph $H \subseteq G$ with the following property:
  - For any choice of terminals on outer face, some opt. connection is preserved.
- **Naive**: H of size at most $2^k \cdot k$.



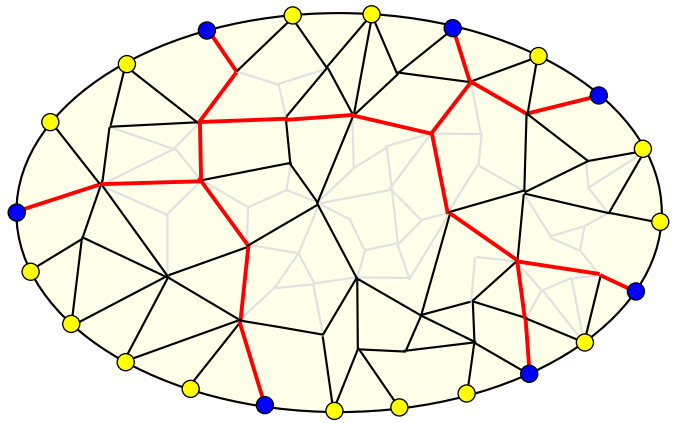
Planar Steiner Tree: sparsification

- **Given**: Planar graph G with outer face of length k.
- **Outcome**: Small subgraph $H \subseteq G$ with the following property:
  - For any choice of terminals on outer face, some opt. connection is preserved.
- **Naive**: H of size at most $2^k \cdot k$.
- **Our result**: H of size $\mathrm{poly}(k)$.



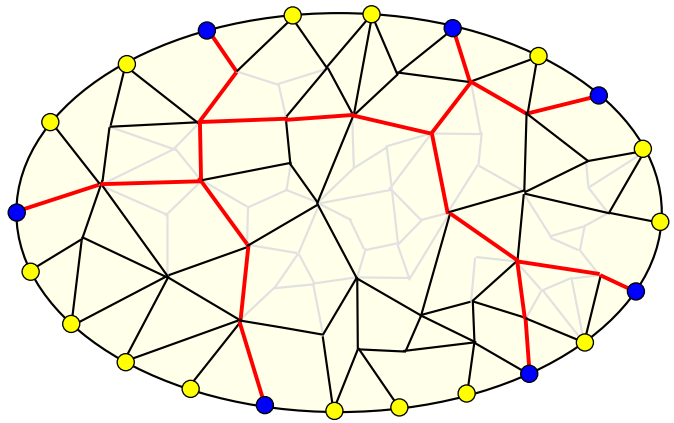
Planar Steiner Tree: sparsification

- **Given**: Planar graph G with outer face of length k.
- **Outcome**: Small subgraph $H \subseteq G$ with the following property:
  - For any choice of terminals on outer face, some opt. connection is preserved.
- **Naive**: H of size at most $2^k \cdot k$.
- **Our result**: H of size $\mathrm{poly}(k)$.
  - More precisely, of size $\mathcal{O}(k^{142})$.



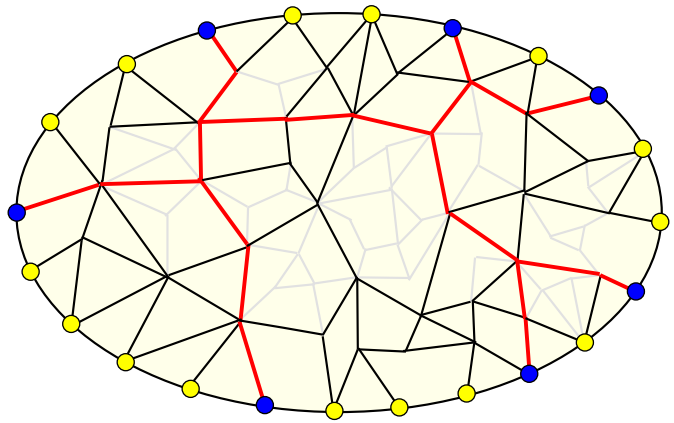
Planar Steiner Tree: sparsification

- **Given**: Planar graph G with outer face of length k.
- **Outcome**: Small subgraph $H \subseteq G$ with the following property:
  - For any choice of terminals on outer face, some opt. connection is preserved.
- **Naive**: H of size at most $2^k \cdot k$.
- **Our result**: H of size $\mathrm{poly}(k)$.
  - More precisely, of size $\mathcal{O}(k^{142})$.
  - Even more precisely, of size at most $2\,159\,872\,407\,596 \cdot k^{142}$.



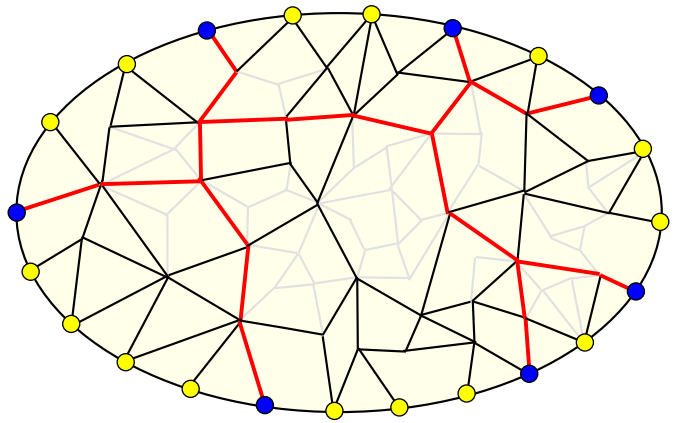
Planar Steiner Tree: sparsification

- **Given**: Planar graph G with outer face of length k.
- **Outcome**: Small subgraph $H \subseteq G$ with the following property:
  - For any choice of terminals on outer face, some opt. connection is preserved.
- **Naive**: H of size at most $2^k \cdot k$.
- **Our result**: H of size $\mathrm{poly}(k)$.
  - More precisely, of size $\mathcal{O}(k^{142})$.
  - Even more precisely, of size at most $2\,159\,872\,407\,596 \cdot k^{142}$.
  - **Conjecture**: There is a sparsifier H of size $\mathcal{O}(k^2)$.



Planar Steiner Tree: sparsification

- **Given**: Planar graph G with outer face of length k.
- **Outcome**: Small subgraph $H \subseteq G$ with the following property:
  - For any choice of terminals on outer face, some opt. connection is preserved.
- **Naive**: H of size at most $2^k \cdot k$.
- **Our result**: H of size $\mathrm{poly}(k)$.
  - More precisely, of size $\mathcal{O}(k^{142})$.
  - Even more precisely, of size at most $2\,159\,872\,407\,596 \cdot k^{142}$.
  - **Conjecture**: There is a sparsifier H of size $\mathcal{O}(k^2)$.
- **By-product**: Statement for weighted graphs, useful for the design of PTASes.



Commercial break

Some of this, and many more in the new book

Parameterized algorithms

by Cygan, Fomin, Kowalik, Lokshtanov,
Marx, Pilipczuk, Pilipczuk, and Saurabh.

Published by Springer, 2015.