Złożoność parametryzowana

czyli czym zajmuję się na co dzień

Michał Pilipczuk

Instytut Informatyki, Uniwersytet Warszawski

Wręczenie nagrody im. W. Lipskiego,
8. października 2015

Dzięki!

Tatjana Abramovskaya
Ivan Bliznets
Hans Bodlaender
Marthe Bonamy
Rajesh Chitnis
Marek Cygan
Claire David
Pål Grønås Drange
Markus S. Dregi
Fedor Fomin
Ariel Gabizon
Archontia Giannopoulou
Petr Golovach
MohammadTaghi Hajiaghayi
Pinar Heggernes
Pim van 't Hof
Piotr Hofman
Bart Jansen
Paweł Komosa
Łukasz Kowalik
Stefan Kratsch
Stephan Kreutzer
Erik Jan van Leeuwen
Daniel Lokshtanov
Lukáš Mach
Fredrik Manne
Dániel Marx
Filip Murlak
Jesper Nederlof
Daniël Paulusma
Geevarghese Philip
Marcin Pilipczuk
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
Jakub Onufry Wojtaszczyk
Marcin Wrochna

Dzięki!

Tatjana Abramovskaya
Ivan Bliznets
Hans Bodlaender
Marthe Bonamy
Rajesh Chitnis
Marek Cygan
Claire David
Pål Grønås Drange
Markus S. Dregi
Fedor Fomin
Ariel Gabizon
Archontia Giannopoulou
Petr Golovach
MohammadTaghi Hajiaghayi
Pinar Heggernes
Pim van 't Hof
Piotr Hofman
Bart Jansen
Paweł Komosa
Łukasz Kowalik
Stefan Kratsch
Stephan Kreutzer
Erik Jan van Leeuwen
Daniel Lokshtanov
Lukáš Mach
Fredrik Manne
Dániel Marx
Filip Murlak
Jesper Nederlof
Daniël Paulusma
Geevarghese Philip
Marcin Pilipczuk
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
Jakub Onufry Wojtaszczyk
Marcin Wrochna

Dzięki!

Tatjana Abramovskaya
Ivan Bliznets
Hans Bodlaender
Marthe Bonamy
Rajesh Chitnis
Marek Cygan
Claire David
Pål Grønås Drange
Markus S. Dregi
Fedor Fomin
Ariel Gabizon
Archontia Giannopoulou
Petr Golovach
MohammadTaghi Hajiaghayi
Pinar Heggernes
Pim van 't Hof
Piotr Hofman
Bart Jansen
Paweł Komosa
Łukasz Kowalik
Stefan Kratsch
Stephan Kreutzer
Erik Jan van Leeuwen
Daniel Lokshtanov
Lukáš Mach
Fredrik Manne
Dániel Marx
Filip Murlak
Jesper Nederlof
Daniël Paulusma
Geevarghese Philip
Marcin Pilipczuk
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
Jakub Onufry Wojtaszczyk
Marcin Wrochna

Dzięki!

Tatjana Abramovskaya
Ivan Bliznets
Hans Bodlaender
Marthe Bonamy
Rajesh Chitnis
Marek Cygan
Claire David
Pål Grønås Drange
Markus S. Dregi
Fedor Fomin
Ariel Gabizon
Archontia Giannopoulou
Petr Golovach
MohammadTaghi Hajiaghayi
Pinar Heggernes
Pim van 't Hof
Piotr Hofman
Bart Jansen
Paweł Komosa
Łukasz Kowalik
Stefan Kratsch
Stephan Kreutzer
Erik Jan van Leeuwen
Daniel Lokshtanov
Lukáš Mach
Fredrik Manne
Dániel Marx
Filip Murlak
Jesper Nederlof
Daniël Paulusma
Geevarghese Philip
Marcin Pilipczuk
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
Jakub Onufry Wojtaszczyk
Marcin Wrochna

Złożoność parametryzowana

- **Klasyczna teoria złożoności**:

  Miarą trudności instancji problemu jest całkowita wielkość wejścia.

Złożoność parametryzowana

- **Klasyczna teoria złożoności**:

  Miarą trudności instancji problemu jest całkowita wielkość wejścia.

  - Problemy mogą być w L, P, NP, PSPACE, $3\text{-}\mathrm{EXPTIME}$ itd.

Złożoność parametryzowana

- **Klasyczna teoria złożoności**:

  Miarą trudności instancji problemu jest całkowita wielkość wejścia.

  - Problemy mogą być w L, P, NP, PSPACE, $3\text{-}\mathrm{EXPTIME}$ itd.

- **Złożoność parametryzowana**:

  Trudność instancji to pojęcie wielowymiarowe.

Złożoność parametryzowana

- **Klasyczna teoria złożoności**:

  Miarą trudności instancji problemu jest całkowita wielkość wejścia.

  - Problemy mogą być w L, P, NP, PSPACE, $3\text{-}\mathrm{EXPTIME}$ itd.

- **Złożoność parametryzowana**:

  Trudność instancji to pojęcie wielowymiarowe.

  - **Parametr**: Dodatkowa miara trudności k.

Złożoność parametryzowana

- **Klasyczna teoria złożoności**:

  Miarą trudności instancji problemu jest całkowita wielkość wejścia.

  - Problemy mogą być w L, P, NP, PSPACE, $3\text{-}\mathrm{EXPTIME}$ itd.

- **Złożoność parametryzowana**:

  Trudność instancji to pojęcie wielowymiarowe.

  - **Parametr**: Dodatkowa miara trudności k.
  - Złożoność czasowa algorytmu jest funkcją parametrów.

Złożoność parametryzowana

- **Klasyczna teoria złożoności**:

  Miarą trudności instancji problemu jest całkowita wielkość wejścia.

  - Problemy mogą być w L, P, NP, PSPACE, $3\text{-}\mathrm{EXPTIME}$ itd.

- **Złożoność parametryzowana**:

  Trudność instancji to pojęcie wielowymiarowe.

  - **Parametr**: Dodatkowa miara trudności k.
  - Złożoność czasowa algorytmu jest funkcją parametrów.
  - **Idea**: Opracować algorytm działający szybko, gdy instancja jest „prosta”.

Przykłady

- **Problemy grafowe**:

Przykłady

- **Problemy grafowe**:
  - Wielkość szukanego obiektu (np. problem *k-kliki*).

Przykłady

- **Problemy grafowe**:
  - Wielkość szukanego obiektu (np. problem *k-kliki*).
  - Maksymalny stopień w grafie.

Przykłady

- **Problemy grafowe**:
  - Wielkość szukanego obiektu (np. problem *k-kliki*).
  - Maksymalny stopień w grafie.
  - Odległość edycyjna od klasy, na której problem jest prosty.

Przykłady

- **Problemy grafowe**:
  - Wielkość szukanego obiektu (np. problem *k-kliki*).
  - Maksymalny stopień w grafie.
  - Odległość edycyjna od klasy, na której problem jest prosty.
  - Miary strukturalne (np. treewidth, cliquewidth).

Przykłady

- **Problemy grafowe**:
  - Wielkość szukanego obiektu (np. problem *k-kliki*).
  - Maksymalny stopień w grafie.
  - Odległość edycyjna od klasy, na której problem jest prosty.
  - Miary strukturalne (np. treewidth, cliquewidth).
- **Problemy na słowach**:

Przykłady

- **Problemy grafowe**:
  - Wielkość szukanego obiektu (np. problem *k-kliki*).
  - Maksymalny stopień w grafie.
  - Odległość edycyjna od klasy, na której problem jest prosty.
  - Miary strukturalne (np. treewidth, cliquewidth).
- **Problemy na słowach**:
  - Wielkość wzorca do wyszukania w tekście.

Przykłady

- **Problemy grafowe**:
  - Wielkość szukanego obiektu (np. problem *k-kliki*).
  - Maksymalny stopień w grafie.
  - Odległość edycyjna od klasy, na której problem jest prosty.
  - Miary strukturalne (np. treewidth, cliquewidth).
- **Problemy na słowach**:
  - Wielkość wzorca do wyszukania w tekście.
  - Liczba liter w alfabecie.

Przykłady

- **Problemy grafowe**:
  - Wielkość szukanego obiektu (np. problem *k-kliki*).
  - Maksymalny stopień w grafie.
  - Odległość edycyjna od klasy, na której problem jest prosty.
  - Miary strukturalne (np. treewidth, cliquewidth).
- **Problemy na słowach**:
  - Wielkość wzorca do wyszukania w tekście.
  - Liczba liter w alfabecie.
- **Bazy danych**:

Przykłady

- **Problemy grafowe**:
  - Wielkość szukanego obiektu (np. problem *k-kliki*).
  - Maksymalny stopień w grafie.
  - Odległość edycyjna od klasy, na której problem jest prosty.
  - Miary strukturalne (np. treewidth, cliquewidth).
- **Problemy na słowach**:
  - Wielkość wzorca do wyszukania w tekście.
  - Liczba liter w alfabecie.
- **Bazy danych**:
  - Wielkość zapytania.

Przykłady

- **Problemy grafowe**:
  - Wielkość szukanego obiektu (np. problem *k-kliki*).
  - Maksymalny stopień w grafie.
  - Odległość edycyjna od klasy, na której problem jest prosty.
  - Miary strukturalne (np. treewidth, cliquewidth).
- **Problemy na słowach**:
  - Wielkość wzorca do wyszukania w tekście.
  - Liczba liter w alfabecie.
- **Bazy danych**:
  - Wielkość zapytania.
- Złożoność parametryzowana to **sposób patrzenia** na złożoność obliczeniową.

Pojęcia wydajności

- Chcemy znaleźć algorytm, który będzie działał szybko gdy parametr jest mały.

Pojęcia wydajności

- Chcemy znaleźć algorytm, który będzie działał szybko gdy parametr jest mały.
- Problem jest w klasie $\underline{\mathrm{XP}}$ jeśli dla każdej stałej wartości parametru k, da się go rozwiązywać w czasie wielomianowym.

Pojęcia wydajności

- Chcemy znaleźć algorytm, który będzie działał szybko gdy parametr jest mały.
- Problem jest w klasie $\underline{\mathrm{XP}}$ jeśli dla każdej stałej wartości parametru k, da się go rozwiązywać w czasie wielomianowym.
  - **Formalnie**: Istnieje algorytm działający w czasie $f(k) \cdot n^{f(k)}$, dla pewnej obliczalnej funkcji f.

Pojęcia wydajności

- Chcemy znaleźć algorytm, który będzie działał szybko gdy parametr jest mały.
- Problem jest w klasie $\underline{\mathrm{XP}}$ jeśli dla każdej stałej wartości parametru k, da się go rozwiązywać w czasie wielomianowym.
  - **Formalnie**: Istnieje algorytm działający w czasie $f(k) \cdot n^{f(k)}$, dla pewnej obliczalnej funkcji f.
- **Przykład**: Klikę wielkości 50 da się znajdować w czasie $\mathcal{O}(n^{50})$.

Pojęcia wydajności

- Chcemy znaleźć algorytm, który będzie działał szybko gdy parametr jest mały.
- Problem jest w klasie $\underline{\mathrm{XP}}$ jeśli dla każdej stałej wartości parametru k, da się go rozwiązywać w czasie wielomianowym.
  - **Formalnie**: Istnieje algorytm działający w czasie $f(k) \cdot n^{f(k)}$, dla pewnej obliczalnej funkcji f.
- **Przykład**: Klikę wielkości 50 da się znajdować w czasie $\mathcal{O}(n^{50})$.
- Problem jest w klasie $\underline{\mathrm{FPT}}$ jeśli dla każdej stałej wartości parametru k, da się go rozwiązywać w czasie liniowym/kwadratowym/sześciennym...

Pojęcia wydajności

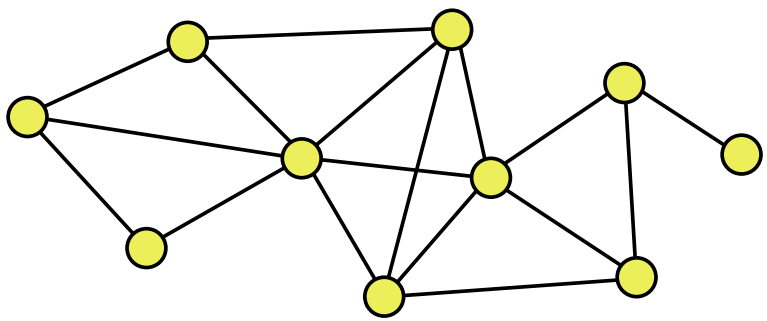
- Chcemy znaleźć algorytm, który będzie działał szybko gdy parametr jest mały.
- Problem jest w klasie $\underline{\mathrm{XP}}$ jeśli dla każdej stałej wartości parametru k, da się go rozwiązywać w czasie wielomianowym.
  - **Formalnie**: Istnieje algorytm działający w czasie $f(k) \cdot n^{f(k)}$, dla pewnej obliczalnej funkcji f.
- **Przykład**: Klikę wielkości 50 da się znajdować w czasie $\mathcal{O}(n^{50})$.
- Problem jest w klasie $\underline{\mathrm{FPT}}$ jeśli dla każdej stałej wartości parametru k, da się go rozwiązywać w czasie liniowym/kwadratowym/sześciennym...
  - **Formalnie**: Istnieje algorytm działający w czasie $f(k) \cdot n^c$, dla pewnej obliczalnej funkcji f i stałej c.

Pojęcia wydajności

- Chcemy znaleźć algorytm, który będzie działał szybko gdy parametr jest mały.
- Problem jest w klasie $\underline{\mathrm{XP}}$ jeśli dla każdej stałej wartości parametru k, da się go rozwiązywać w czasie wielomianowym.
  - **Formalnie**: Istnieje algorytm działający w czasie $f(k) \cdot n^{f(k)}$, dla pewnej obliczalnej funkcji f.
- **Przykład**: Klikę wielkości 50 da się znajdować w czasie $\mathcal{O}(n^{50})$.
- Problem jest w klasie $\underline{\mathrm{FPT}}$ jeśli dla każdej stałej wartości parametru k, da się go rozwiązywać w czasie liniowym/kwadratowym/sześciennym...
  - **Formalnie**: Istnieje algorytm działający w czasie $f(k) \cdot n^c$, dla pewnej obliczalnej funkcji f i stałej c.
- **Przykład**: Pokrycie wierzchołkowe wielkości k da się znajdować w czasie $\mathcal{O}(2^k \cdot (n+m))$.

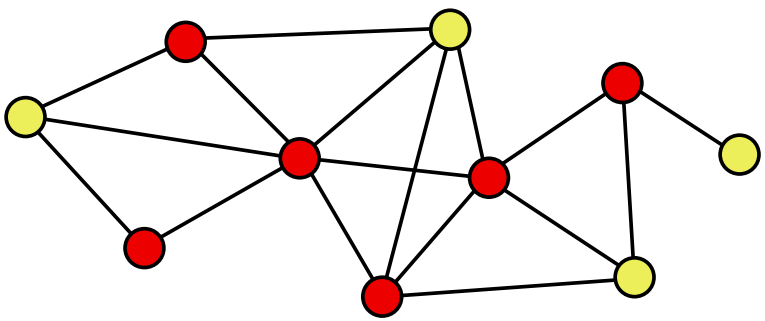
Kilka przykładów

Problem	Parametr	Złożoność



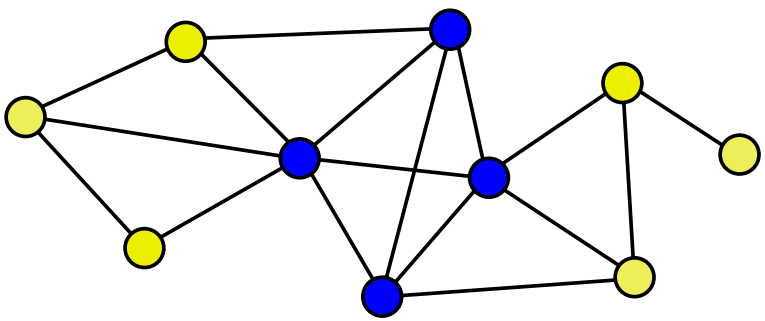
Kilka przykładów

Problem	Parametr	Złożoność
Vertex Cover	wielkość rozwiązania	**FPT**



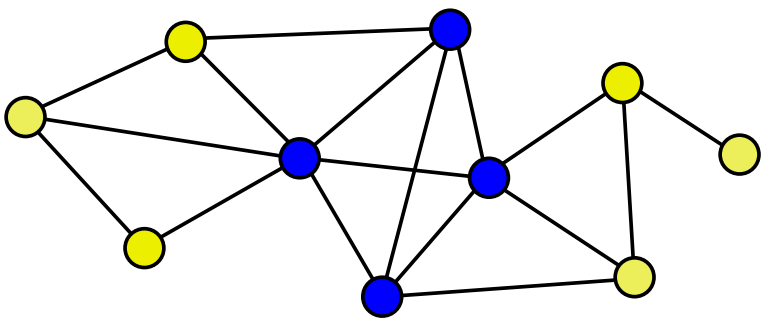
Kilka przykładów

Problem	Parametr	Złożoność
Vertex Cover	wielkość rozwiązania	**FPT**
k-Klika	wielkość rozwiązania	



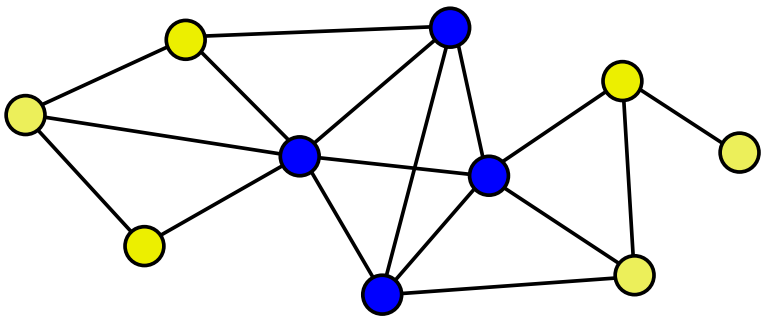
Kilka przykładów

Problem	Parametr	Złożoność
Vertex Cover	wielkość rozwiązania	**FPT**
k-Klika	wielkość rozwiązania	**W[1]-zupełny**



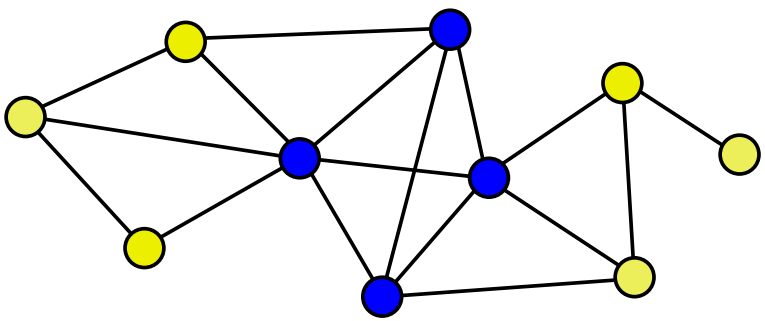
Kilka przykładów

Problem	Parametr	Złożoność
Vertex Cover	wielkość rozwiązania	**FPT**
k-Klika	wielkość rozwiązania	**W[1]-zupełny**
k-Klika	maksymalny stopień	



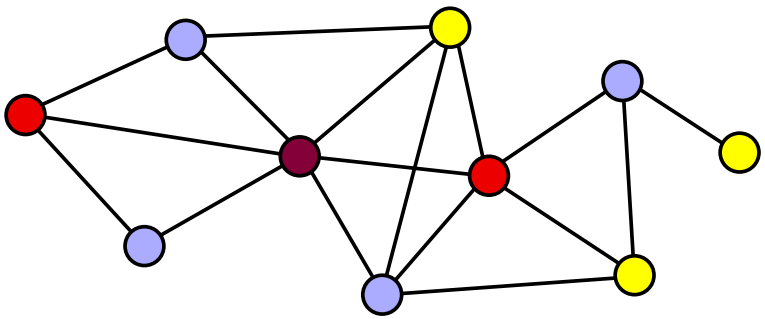
Kilka przykładów

Problem	Parametr	Złożoność
Vertex Cover	wielkość rozwiązania	**FPT**
k-Klika	wielkość rozwiązania	**W[1]-zupełny**
k-Klika	maksymalny stopień	**FPT**



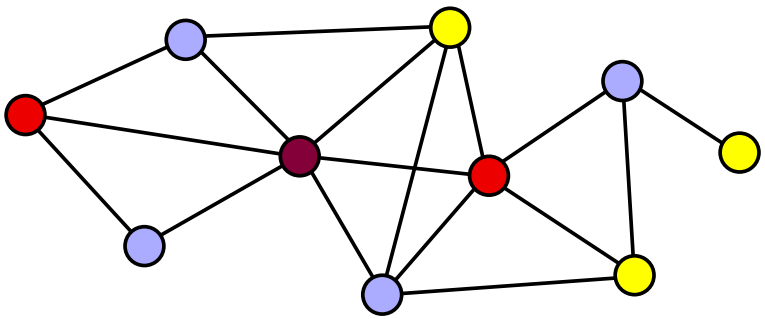
Kilka przykładów

Problem	Parametr	Złożoność
Vertex Cover	wielkość rozwiązania	**FPT**
k-Klika	wielkość rozwiązania	**W[1]-zupełny**
k-Klika	maksymalny stopień	**FPT**
k-Kolorowanie	liczba kolorów	



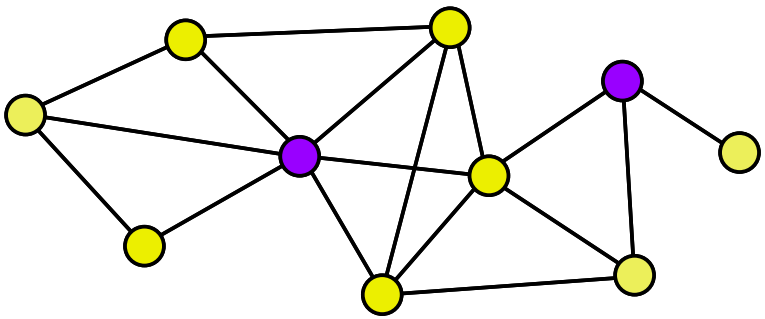
Kilka przykładów

Problem	Parametr	Złożoność
Vertex Cover	wielkość rozwiązania	**FPT**
k-Klika	wielkość rozwiązania	**W[1]-zupełny**
k-Klika	maksymalny stopień	**FPT**
k-Kolorowanie	liczba kolorów	**NP-trudny dla $k = 3$**



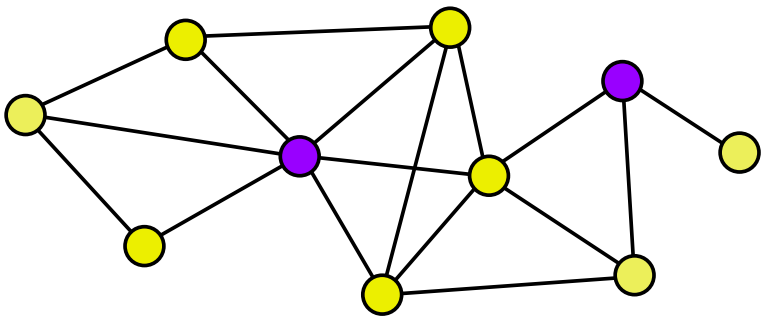
Kilka przykładów

Problem	Parametr	Złożoność
Vertex Cover	wielkość rozwiązania	**FPT**
k-Klika	wielkość rozwiązania	**W[1]-zupełny**
k-Klika	maksymalny stopień	**FPT**
k-Kolorowanie	liczba kolorów	**NP-trudny dla $k = 3$**
Zbiór Dominujący	wielkość rozwiązania	



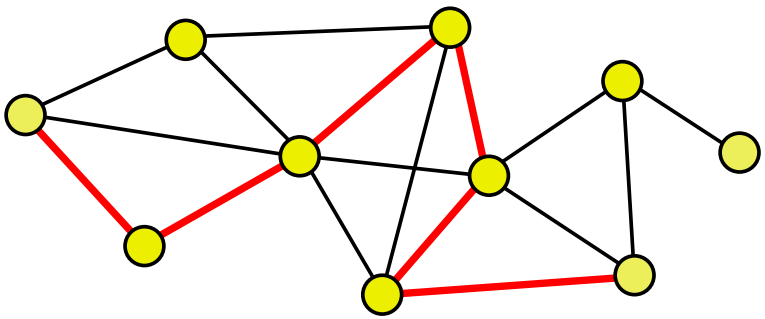
Kilka przykładów

Problem	Parametr	Złożoność
Vertex Cover	wielkość rozwiązania	**FPT**
k-Klika	wielkość rozwiązania	**W[1]-zupełny**
k-Klika	maksymalny stopień	**FPT**
k-Kolorowanie	liczba kolorów	**NP-trudny dla $k = 3$**
Zbiór Dominujący	wielkość rozwiązania	**W[2]-zupełny**



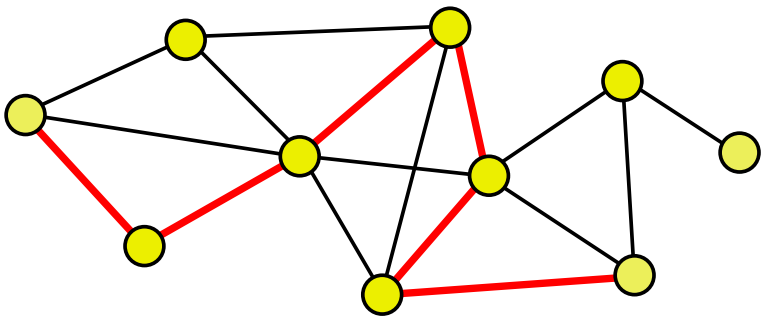
Kilka przykładów

Problem	Parametr	Złożoność
Vertex Cover	wielkość rozwiązania	**FPT**
k-Klika	wielkość rozwiązania	**W[1]-zupełny**
k-Klika	maksymalny stopień	**FPT**
k-Kolorowanie	liczba kolorów	**NP-trudny dla $k = 3$**
Zbiór Dominujący	wielkość rozwiązania	**W[2]-zupełny**
k-Ścieżka	wielkość rozwiązania	



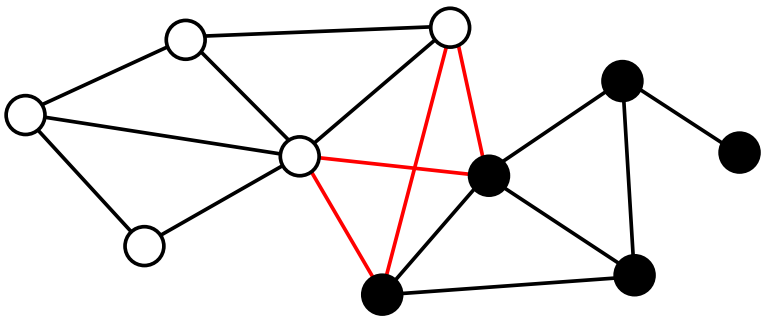
Kilka przykładów

Problem	Parametr	Złożoność
Vertex Cover	wielkość rozwiązania	**FPT**
k-Klika	wielkość rozwiązania	**W[1]-zupełny**
k-Klika	maksymalny stopień	**FPT**
k-Kolorowanie	liczba kolorów	**NP-trudny dla $k = 3$**
Zbiór Dominujący	wielkość rozwiązania	**W[2]-zupełny**
k-Ścieżka	wielkość rozwiązania	**FPT** [AYZ'95]



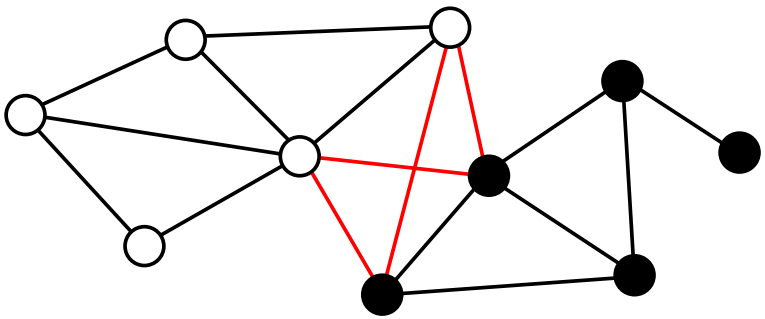
Kilka przykładów

Problem	Parametr	Złożoność
Vertex Cover	wielkość rozwiązania	**FPT**
k-Klika	wielkość rozwiązania	**W[1]-zupełny**
k-Klika	maksymalny stopień	**FPT**
k-Kolorowanie	liczba kolorów	**NP-trudny dla $k = 3$**
Zbiór Dominujący	wielkość rozwiązania	**W[2]-zupełny**
k-Ścieżka	wielkość rozwiązania	**FPT** [AYZ'95]
Bisekcja	wielkość cięcia	

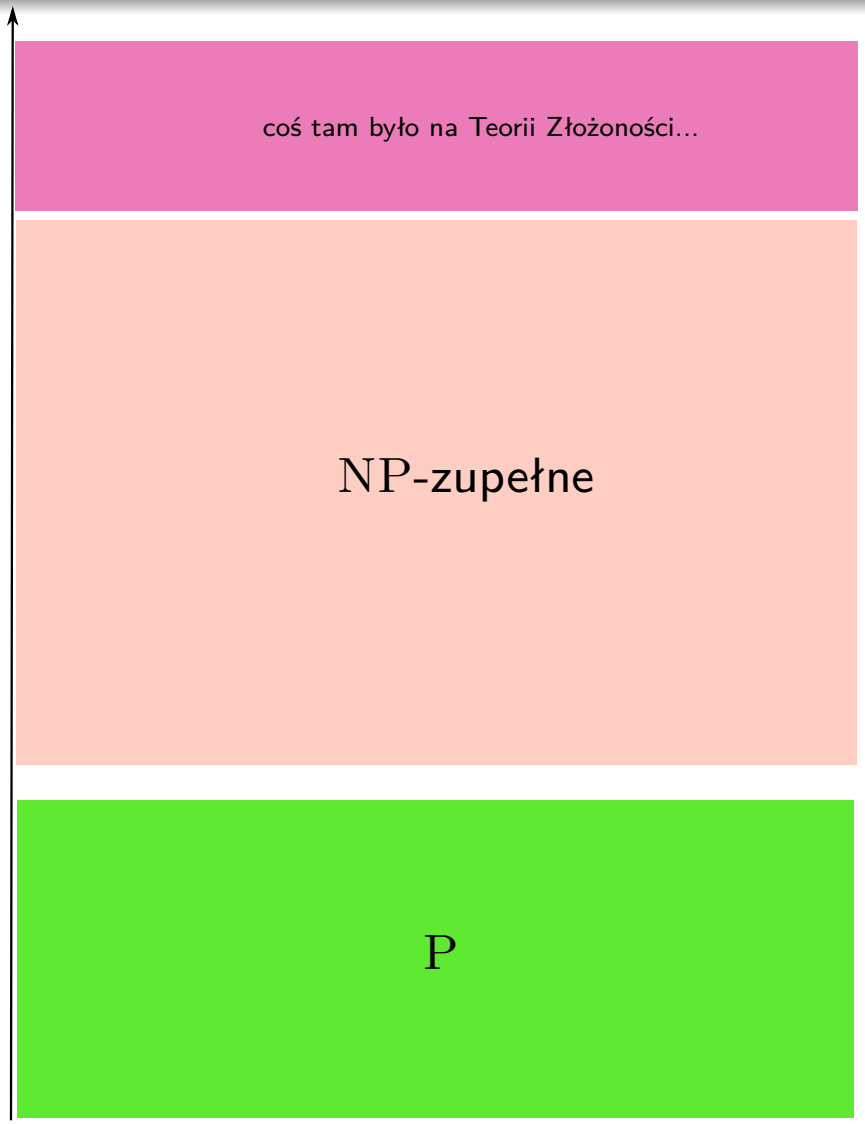


Kilka przykładów

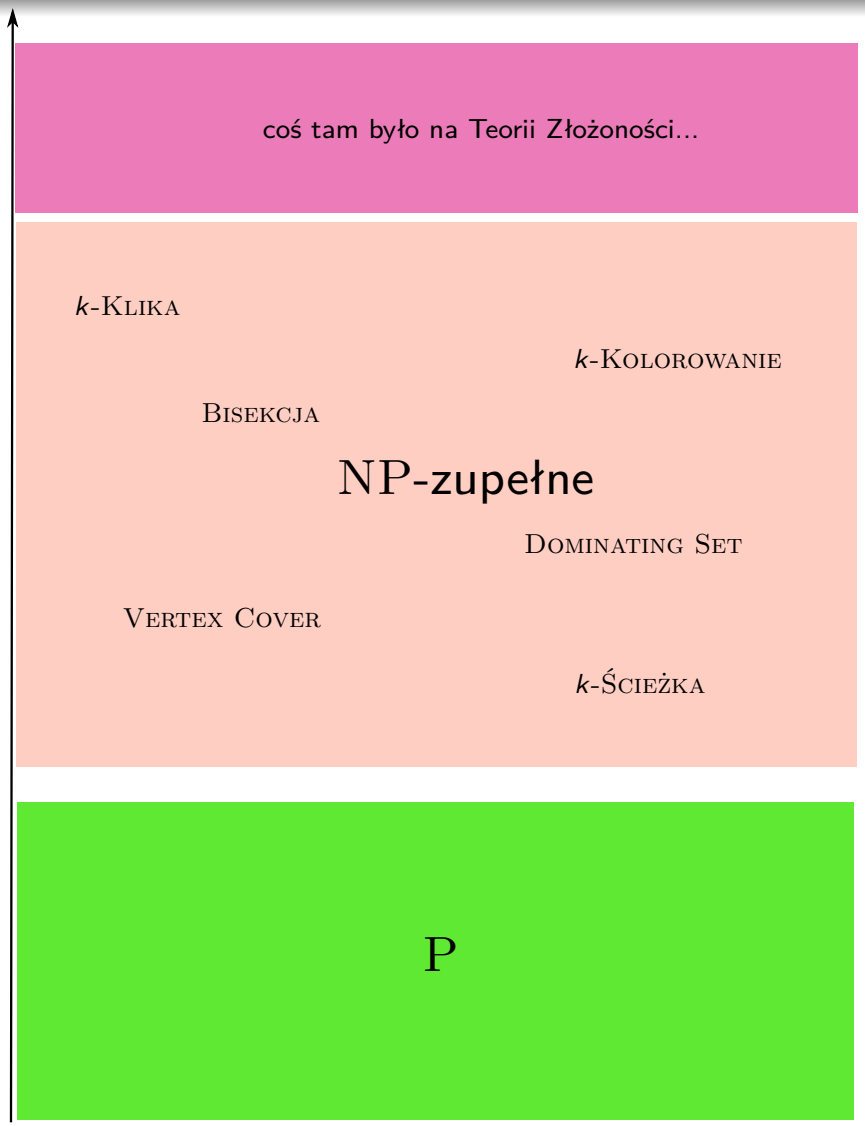
Problem	Parametr	Złożoność
Vertex Cover	wielkość rozwiązania	**FPT**
k-Klika	wielkość rozwiązania	**W[1]-zupełny**
k-Klika	maksymalny stopień	**FPT**
k-Kolorowanie	liczba kolorów	**NP-trudny dla $k = 3$**
Zbiór Dominujący	wielkość rozwiązania	**W[2]-zupełny**
k-Ścieżka	wielkość rozwiązania	**FPT** [AYZ'95]
Bisekcja	wielkość cięcia	**FPT** [CLPPS'14]



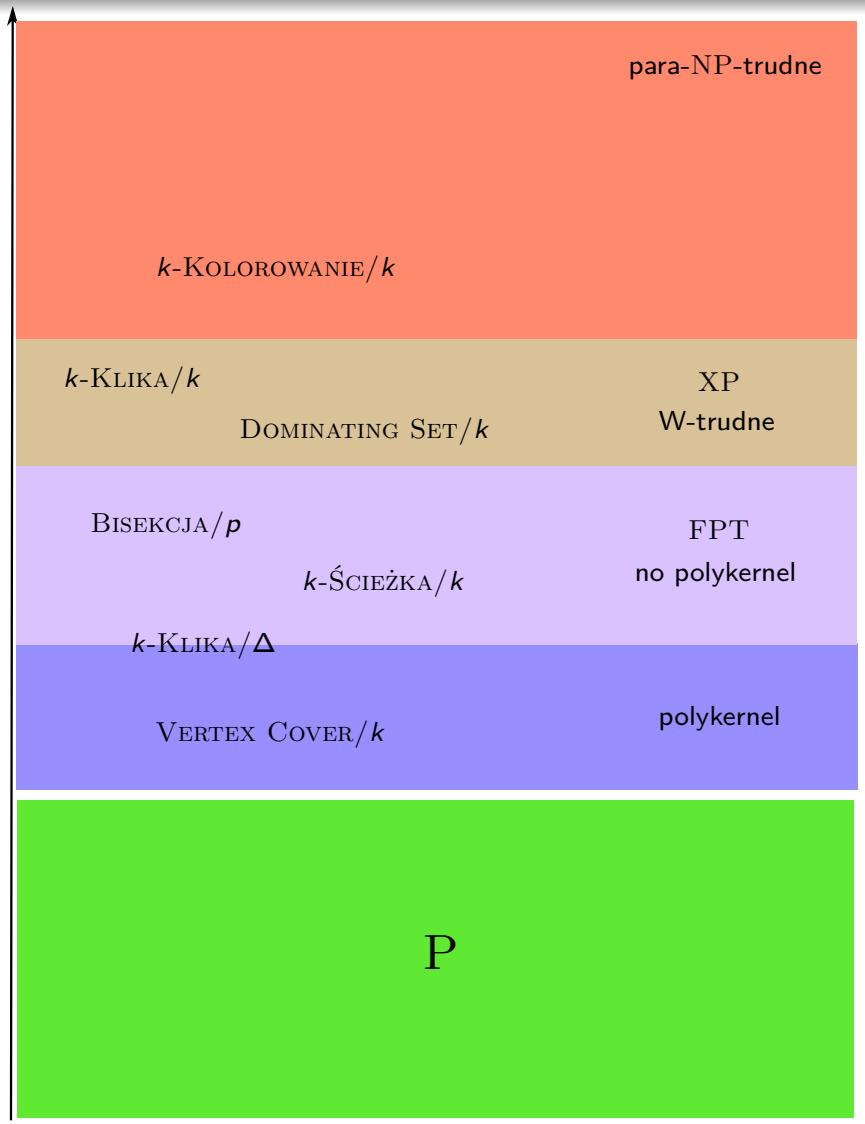
Teoria złożoności oczami algorytmika



Teoria złożoności oczami algorytmika



Teoria złożoności oczami algorytmika



Sztuka parametryzacji

- Problem obliczeniowy może mieć diametralnie różne złożoności przy różnych parametryzacjach.

Sztuka parametryzacji

- Problem obliczeniowy może mieć diametralnie różne złożoności przy różnych parametryzacjach.
- Każdy parametr odpowiada **klasie** instancji, na której podejrzewamy, że problem jest prosty.

Sztuka parametryzacji

- Problem obliczeniowy może mieć diametralnie różne złożoności przy różnych parametryzacjach.
- Każdy parametr odpowiada **klasie** instancji, na której podejrzewamy, że problem jest prosty.
  - Dobór parametru może być umotywowany zastosowaniami.

Sztuka parametryzacji

- Problem obliczeniowy może mieć diametralnie różne złożoności przy różnych parametryzacjach.
- Każdy parametr odpowiada **klasie** instancji, na której podejrzewamy, że problem jest prosty.
  - Dobór parametru może być umotywowany zastosowaniami.
- Algorytmy dla różnych parametryzacji odpowiadają różnym metodom ataku na trudny problem.

Sztuka parametryzacji

- Problem obliczeniowy może mieć diametralnie różne złożoności przy różnych parametryzacjach.
- Każdy parametr odpowiada **klasie** instancji, na której podejrzewamy, że problem jest prosty.
  - Dobór parametru może być umotywowany zastosowaniami.
- Algorytmy dla różnych parametryzacji odpowiadają różnym metodom ataku na trudny problem.
- **Parameter ecology**: Znajdowanie dokładnej złożoności danego problemu względem różnych parametryzacji.

Subgraph Isomorphism

- Subgraph Isomorphism: Czy dany graf H jest podgrafem grafu G?

Subgraph Isomorphism

- Subgraph Isomorphism: Czy dany graf H jest podgrafem grafu G?
- $+\infty$ parametrów: liczba składowych, max. stopień, treewidth, cliquewidth, genus, liczba Hadwigera...

Subgraph Isomorphism

- Subgraph Isomorphism: Czy dany graf H jest podgrafem grafu G?
- $+\infty$ parametrów: liczba składowych, max. stopień, treewidth, cliquewidth, genus, liczba Hadwigera...
- Każdy z nich można rozpatrywać zarówno dla H jak i dla G.

Subgraph Isomorphism

- Subgraph Isomorphism: Czy dany graf H jest podgrafem grafu G?
- $+\infty$ parametrów: liczba składowych, max. stopień, treewidth, cliquewidth, genus, liczba Hadwigera...
- Każdy z nich można rozpatrywać zarówno dla H jak i dla G.
- To jest tematem pracy:

Everything you always wanted to know about the parameterized complexity of Subgraph Isomorphism *(but were afraid to ask)*;

Marx, Pilipczuk; STACS 2014

Subgraph Isomorphism

- Subgraph Isomorphism: Czy dany graf H jest podgrafem grafu G?
- $+\infty$ parametrów: liczba składowych, max. stopień, treewidth, cliquewidth, genus, liczba Hadwigera...
- Każdy z nich można rozpatrywać zarówno dla H jak i dla G.
- To jest tematem pracy:

  Everything you always wanted to know about the parameterized complexity of Subgraph Isomorphism *(but were afraid to ask)*;

  Marx, Pilipczuk; STACS 2014

- 10 parametrów, dla każdej kombinacji pytamy się o algorytm o złożoności

$$f(p_1, p_2, \ldots, p_\ell) \cdot n^{g(p_{\ell+1}, p_{\ell+2}, \ldots, p_k)}.$$

Subgraph Isomorphism

- Subgraph Isomorphism: Czy dany graf H jest podgrafem grafu G?
- $+\infty$ parametrów: liczba składowych, max. stopień, treewidth, cliquewidth, genus, liczba Hadwigera...
- Każdy z nich można rozpatrywać zarówno dla H jak i dla G.
- To jest tematem pracy:

  Everything you always wanted to know about the parameterized complexity of Subgraph Isomorphism *(but were afraid to ask)*;

  Marx, Pilipczuk; STACS 2014

- 10 parametrów, dla każdej kombinacji pytamy się o algorytm o złożoności

$$f(p_1, p_2, \ldots, p_\ell) \cdot n^{g(p_{\ell+1}, p_{\ell+2}, \ldots, p_k)}.$$

- **Wynik**: Wszystkie kombinacje są wyjaśnione przez 11 wyników pozytywnych i 17 negatywnych.

Subgraph Isomorphism

Short Description	Thm	H										G								
		$\lvert V(\cdot)\rvert$	cc	Δ	fvs	pw	tw	cw	genus	hadw	hadw_{T}	cc	Δ	fvs	pw	tw	cw	genus	hadw	hadw_{T}
FO model checking	Thm P.1 (page 17)	M															M			
	Thm P.2 (page 17)	M																		M
Color coding	Thm P.3 (page 17)	M					E													
Matoušek-Thomas	Thm P.4 (page 18)		M	M												E				
Paths&Cycles → Paths&Cycles	Thm P.5 (page 18)											E	2							
Dynamic Programming	Thm P.6 (page 19)		E	2												M				
	Thm P.7 (page 21)		E	2													E			
	Thm P.8⋆ (page 23)		M													1				
FVS and CSPs	Thm P.9⋆ (page 28)		M	2									M	M						
	Thm P.10⋆ (page 36)		E										M	M				E		
	Thm P.11⋆ (page 46)		E	E									M	M					E	
Bin Packing	Thm N.1 (page 46)											M	2			1				
	Thm N.2 (page 47)		1				1							E	E			0		
	Thm N.3 (page 47)			2								1	3		E	1				
Planar cubic HamPath	Thm N.4 (page 48)		1	2			1						3					0		
Clique	Thm N.5 (page 48)	M	1					E												
HamPath in bounded **cw**	Thm N.6 (page 48)		1	2			1										M			
Grid Tiling, 1-in-n gadgets	Thm N.7⋆ (page 57)		M			E	1					1	3	M	M			0		
	Thm N.8⋆ (page 61)		1			E	1						M	M	M			M	E	
	Thm N.9⋆ (page 61)		1			E	1						3	M	M			M		
	Thm N.10⋆ (page 63)		1	3		E	1						M	M	M		E	M		
Grid Tiling, moustache gadgets	Thm N.11⋆ (page 64)		1	3		E	1							M	M			0		
	Thm N.12⋆ (page 66)		1			E	1						3		M			0		
Small planar graph	Thm N.13⋆ (page 67)	M	1	3					0											
Exact Planar Arc Supply	Thm N.14⋆ (page 74)		M	2			1					1		M	M			0		
	Thm N.15⋆ (page 77)		M	2			1					1	3		M			0		
	Thm N.16⋆ (page 79)		M	2			1					1		M	M		E	M		
	Thm N.17⋆ (page 80)		M	2			1					1	M		M		E	M		

Subgraph Isomorphism

	A	B	C
X	boring	boring	boring
Y	boring	PTime	makes no sense
Z	NP-complete	boring	boring

Subgraph Isomorphism

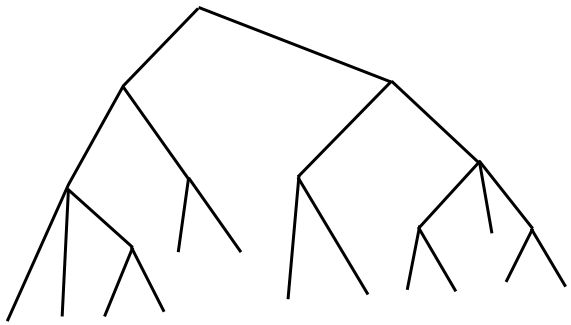
Short Description	Thm	H										G								
		$\lvert V(\cdot)\rvert$	cc	Δ	fvs	pw	tw	cw	genus	hadw	hadw_{T}	cc	Δ	fvs	pw	tw	cw	genus	hadw	hadw_{T}
FO model checking	Thm P.1 (page 17)	M															M			
	Thm P.2 (page 17)	M																		M
Color coding	Thm P.3 (page 17)	M					E													
Matoušek-Thomas	Thm P.4 (page 18)		M	M												E				
Paths&Cycles → Paths&Cycles	Thm P.5 (page 18)											E	2							
Dynamic Programming	Thm P.6 (page 19)		E	2												M				
	Thm P.7 (page 21)		E	2													E			
	Thm P.8⋆ (page 23)		M													1				
FVS and CSPs	Thm P.9⋆ (page 28)		M	2									M	M						
	Thm P.10⋆ (page 36)		E										M	M				E		
	Thm P.11⋆ (page 46)		E	E									M	M					E	
Bin Packing	Thm N.1 (page 46)											M	2			1				
	Thm N.2 (page 47)		1				1							E	E			0		
	Thm N.3 (page 47)			2								1	3		E	1				
Planar cubic HamPath	Thm N.4 (page 48)		1	2			1						3					0		
Clique	Thm N.5 (page 48)	M	1					E												
HamPath in bounded **cw**	Thm N.6 (page 48)		1	2			1										M			
Grid Tiling, 1-in-n gadgets	Thm N.7⋆ (page 57)		M			E	1					1	3	M	M			0		
	Thm N.8⋆ (page 61)		1			E	1						M	M	M			M	E	
	Thm N.9⋆ (page 61)		1			E	1						3	M	M			M		
	Thm N.10⋆ (page 63)		1	3		E	1						M	M	M		E	M		
Grid Tiling, moustache gadgets	Thm N.11⋆ (page 64)		1	3		E	1							M	M			0		
	Thm N.12⋆ (page 66)		1			E	1						3		M			0		
Small planar graph	Thm N.13⋆ (page 67)	M	1	3					0											
Exact Planar Arc Supply	Thm N.14⋆ (page 74)		M	2			1					1		M	M			0		
	Thm N.15⋆ (page 77)		M	2			1					1	3		M			0		
	Thm N.16⋆ (page 79)		M	2			1					1		M	M		E	M		
	Thm N.17⋆ (page 80)		M	2			1					1	M		M		E	M		

Parametryzacje strukturalne

- **Pomysł**: Miarą trudności jest stopień skomplikowania wejściowego grafu.

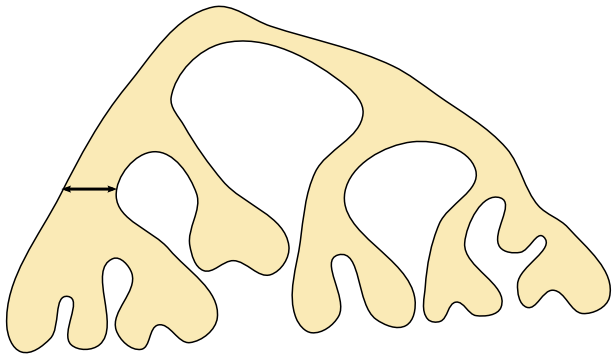
Parametryzacje strukturalne

- **Pomysł**: Miarą trudności jest stopień skomplikowania wejściowego grafu.
- Wiele NP-trudnych problemów da się rozwiązywać szybko na drzewach.



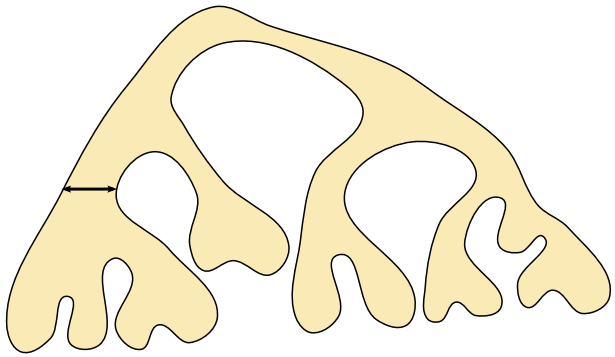
Parametryzacje strukturalne

- **Pomysł**: Miarą trudności jest stopień skomplikowania wejściowego grafu.
- Wiele NP-trudnych problemów da się rozwiązywać szybko na drzewach.
- **Idea**: Graf ma *szerokość drzewiastą* (*treewidth*) t jeśli wygląda jak drzewo o szerokości pnia t.



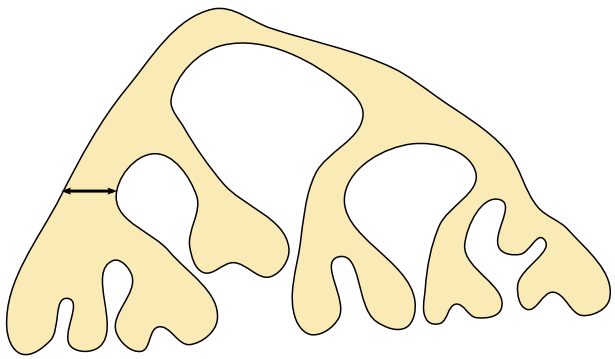
Parametryzacje strukturalne

- **Pomysł**: Miarą trudności jest stopień skomplikowania wejściowego grafu.
- Wiele NP-trudnych problemów da się rozwiązywać szybko na drzewach.
- **Idea**: Graf ma *szerokość drzewiastą* (*treewidth*) t jeśli wygląda jak drzewo o szerokości pnia t.
  - Bardziej formalnie: Istnieje *dekompozycja drzewiasta* o szerokości t.



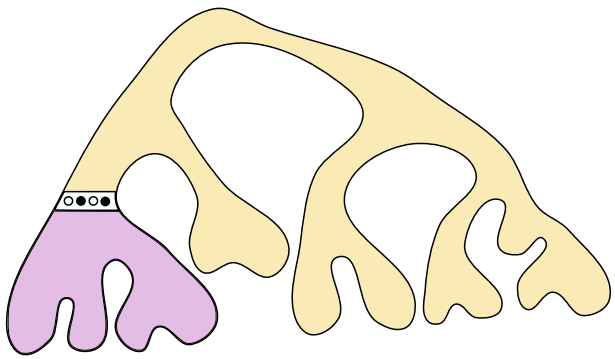
Parametryzacje strukturalne

- **Pomysł**: Miarą trudności jest stopień skomplikowania wejściowego grafu.
- Wiele NP-trudnych problemów da się rozwiązywać szybko na drzewach.
- **Idea**: Graf ma *szerokość drzewiastą* (*treewidth*) t jeśli wygląda jak drzewo o szerokości pnia t.
  - Bardziej formalnie: Istnieje *dekompozycja drzewiasta* o szerokości t.
- Algorytmy dla drzew da się uogólniać do grafów o małym treewidthie.



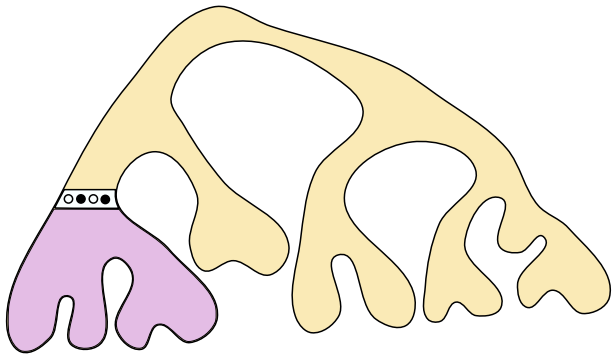
Parametryzacje strukturalne

- **Pomysł**: Miarą trudności jest stopień skomplikowania wejściowego grafu.
- Wiele NP-trudnych problemów da się rozwiązywać szybko na drzewach.
- **Idea**: Graf ma *szerokość drzewiastą* (*treewidth*) t jeśli wygląda jak drzewo o szerokości pnia t.
  - Bardziej formalnie: Istnieje *dekompozycja drzewiasta* o szerokości t.
- Algorytmy dla drzew da się uogólniać do grafów o małym treewidthie.
  - **Przykład**: Vertex Cover da się rozwiązać w czasie $\mathcal{O}(2^t \cdot n)$.



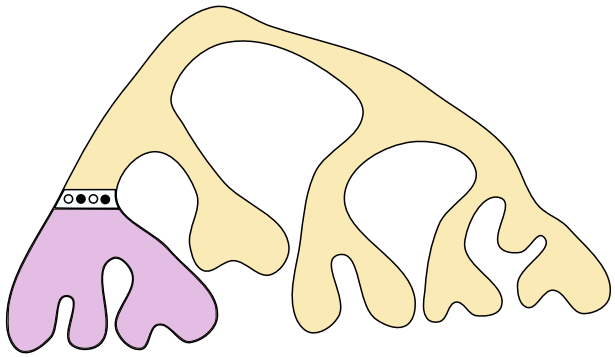
Parametryzacje strukturalne

- **Pomysł**: Miarą trudności jest stopień skomplikowania wejściowego grafu.
- Wiele NP-trudnych problemów da się rozwiązywać szybko na drzewach.
- **Idea**: Graf ma *szerokość drzewiastą* (*treewidth*) t jeśli wygląda jak drzewo o szerokości pnia t.
  - Bardziej formalnie: Istnieje *dekompozycja drzewiasta* o szerokości t.
- Algorytmy dla drzew da się uogólniać do grafów o małym treewidthie.
  - **Przykład**: Vertex Cover da się rozwiązać w czasie $\mathcal{O}(2^t \cdot n)$.
  - **Przykład**: Każdy problem definiowalny w MSO da się rozwiązać w czasie $f(t) \cdot n$, dla jakiejś f.



Parametryzacje strukturalne

- **Pomysł**: Miarą trudności jest stopień skomplikowania wejściowego grafu.
- Wiele NP-trudnych problemów da się rozwiązywać szybko na drzewach.
- **Idea**: Graf ma *szerokość drzewiastą* (*treewidth*) t jeśli wygląda jak drzewo o szerokości pnia t.
  - Bardziej formalnie: Istnieje *dekompozycja drzewiasta* o szerokości t.
- Algorytmy dla drzew da się uogólniać do grafów o małym treewidthie.
  - **Przykład**: Vertex Cover da się rozwiązać w czasie $\mathcal{O}(2^t \cdot n)$.
  - **Przykład**: Każdy problem definiowalny w MSO da się rozwiązać w czasie $f(t) \cdot n$, dla jakiejś f.
- **Zastosowanie**: Graf planarny o n wierzchołkach ma treewidth $\mathcal{O}(\sqrt{n})$.



Parametryzacje strukturalne: wyzwania

- Mamy dwie podstawowe rodziny problemów:

Parametryzacje strukturalne: wyzwania

- Mamy dwie podstawowe rodziny problemów:
  - **Algorytmy na dekompozycjach**: Jak szybko można rozwiązać problem, jeśli dana jest dekompozycja szerokości t?

Parametryzacje strukturalne: wyzwania

- Mamy dwie podstawowe rodziny problemów:
  - **Algorytmy na dekompozycjach**: Jak szybko można rozwiązać problem, jeśli dana jest dekompozycja szerokości t?
  - **Znajdowanie dekompozycji**: Jak znaleźć optymalną dekompozycję, lub bliską optymalnej?

Parametryzacje strukturalne: wyzwania

- Mamy dwie podstawowe rodziny problemów:
  - **Algorytmy na dekompozycjach**: Jak szybko można rozwiązać problem, jeśli dana jest dekompozycja szerokości t?
  - **Znajdowanie dekompozycji**: Jak znaleźć optymalną dekompozycję, lub bliską optymalnej?
- **Problem**: Znajdowanie dekompozycji drzewiastej jest NP-trudne.

Parametryzacje strukturalne: wyzwania

- Mamy dwie podstawowe rodziny problemów:
  - **Algorytmy na dekompozycjach**: Jak szybko można rozwiązać problem, jeśli dana jest dekompozycja szerokości t?
  - **Znajdowanie dekompozycji**: Jak znaleźć optymalną dekompozycję, lub bliską optymalnej?
- **Problem**: Znajdowanie dekompozycji drzewiastej jest NP-trudne.
- **Pomysł**: Przecież możemy użyć do tego czasu FPT od docelowej szerokości!

Znajdowanie dekompozycji

- **(Bodlaender'96)**: Algorytm o złożoności $t^{\mathcal{O}(t^3)} \cdot n$.

Znajdowanie dekompozycji

- **(Bodlaender'96)**: Algorytm o złożoności $t^{\mathcal{O}(t^3)} \cdot n$.
- **(RS, GM XIII)**: 4-aproksymacja w czasie $\mathcal{O}(8^t t^2 \cdot n^2)$

Znajdowanie dekompozycji

- **(Bodlaender'96)**: Algorytm o złożoności $t^{\mathcal{O}(t^3)} \cdot n$.
- **(RS, GM XIII)**: 4-aproksymacja w czasie $\mathcal{O}(8^t t^2 \cdot n^2)$
  - Albo daje dekompozycję szerokości $\leq 4t + 4$, albo stwierdza, że $\mathbf{tw}(G) > t$.

Znajdowanie dekompozycji

- **(Bodlaender'96)**: Algorytm o złożoności $t^{\mathcal{O}(t^3)} \cdot n$.
- **(RS, GM XIII)**: 4-aproksymacja w czasie $\mathcal{O}(8^t t^2 \cdot n^2)$
  - Albo daje dekompozycję szerokości $\leq 4t + 4$, albo stwierdza, że $\mathbf{tw}(G) > t$.
- **Przypomnienie**: Vertex Cover da się rozwiązywać w czasie $\mathcal{O}(2^t \cdot n)$.

Znajdowanie dekompozycji

- **(Bodlaender'96)**: Algorytm o złożoności $t^{\mathcal{O}(t^3)} \cdot n$.
- **(RS, GM XIII)**: 4-aproksymacja w czasie $\mathcal{O}(8^t t^2 \cdot n^2)$
  - Albo daje dekompozycję szerokości $\leq 4t + 4$, albo stwierdza, że $\mathbf{tw}(G) > t$.
- **Przypomnienie**: Vertex Cover da się rozwiązywać w czasie $\mathcal{O}(2^t \cdot n)$.
  - Użycie każdego z algorytmów pogarsza złożoność!

Znajdowanie dekompozycji

- **(Bodlaender'96)**: Algorytm o złożoności $t^{\mathcal{O}(t^3)} \cdot n$.
- **(RS, GM XIII)**: 4-aproksymacja w czasie $\mathcal{O}(8^t t^2 \cdot n^2)$
  - Albo daje dekompozycję szerokości $\leq 4t + 4$, albo stwierdza, że $\mathbf{tw}(G) > t$.
- **Przypomnienie**: Vertex Cover da się rozwiązywać w czasie $\mathcal{O}(2^t \cdot n)$.
  - Użycie każdego z algorytmów pogarsza złożoność!
- Dlatego napisaliśmy pracę:

An $\mathcal{O}(c^k \cdot n)$ 5-approximation algorithm for treewidth;

Bodlaender, Drange, Dregi, Fomin, Lokshtanov, Pilipczuk; FOCS 2013

Znajdowanie dekompozycji

- **(Bodlaender'96)**: Algorytm o złożoności $t^{\mathcal{O}(t^3)} \cdot n$.
- **(RS, GM XIII)**: 4-aproksymacja w czasie $\mathcal{O}(8^t t^2 \cdot n^2)$
  - Albo daje dekompozycję szerokości $\leq 4t + 4$, albo stwierdza, że $\mathbf{tw}(G) > t$.
- **Przypomnienie**: Vertex Cover da się rozwiązywać w czasie $\mathcal{O}(2^t \cdot n)$.
  - Użycie każdego z algorytmów pogarsza złożoność!
- Dlatego napisaliśmy pracę:

An $\mathcal{O}(c^k \cdot n)$ 5-approximation algorithm for treewidth;

Bodlaender, Drange, Dregi, Fomin, Lokshtanov, Pilipczuk; FOCS 2013

- Niestety, $c = 15^{30}$.

Znajdowanie dekompozycji

- **(Bodlaender'96)**: Algorytm o złożoności $t^{\mathcal{O}(t^3)} \cdot n$.
- **(RS, GM XIII)**: 4-aproksymacja w czasie $\mathcal{O}(8^t t^2 \cdot n^2)$
  - Albo daje dekompozycję szerokości $\leq 4t + 4$, albo stwierdza, że $\mathbf{tw}(G) > t$.
- **Przypomnienie**: Vertex Cover da się rozwiązywać w czasie $\mathcal{O}(2^t \cdot n)$.
  - Użycie każdego z algorytmów pogarsza złożoność!
- Dlatego napisaliśmy pracę:

An $\mathcal{O}(c^k \cdot n)$ 5-approximation algorithm for treewidth;

Bodlaender, Drange, Dregi, Fomin, Lokshtanov, Pilipczuk; FOCS 2013

- Niestety, $c = 15^{30}$.
- **To appear soon**: $\mathcal{O}(t)$-aproksymacja w czasie $\mathrm{poly}(t) \cdot n \log n$.

Izomorfizm grafów o małym treewidthie

- **Izomorfizm grafów**: Dane grafy G i H, sprawdzić czy G i H są izomorficzne.

Izomorfizm grafów o małym treewidthie

- **Izomorfizm grafów**: Dane grafy G i H, sprawdzić czy G i H są izomorficzne.
- Klasyczna złożoność tego problemu jest wielkim problemem otwartym.

Izomorfizm grafów o małym treewidthie

- **Izomorfizm grafów**: Dane grafy G i H, sprawdzić czy G i H są izomorficzne.
- Klasyczna złożoność tego problemu jest wielkim problemem otwartym.
- **(Bodlaender'90)**: Złożoność $n^{\mathcal{O}(t)}$, jeśli $\mathbf{tw}(G), \mathbf{tw}(H) \leq t$.

Izomorfizm grafów o małym treewidthie

- **Izomorfizm grafów**: Dane grafy G i H, sprawdzić czy G i H są izomorficzne.
- Klasyczna złożoność tego problemu jest wielkim problemem otwartym.
- **(Bodlaender'90)**: Złożoność $n^{\mathcal{O}(t)}$, jeśli $\mathbf{tw}(G), \mathbf{tw}(H) \leq t$.
- Na pytanie, czy FPT, odpowiadamy w pracy:

Fixed-parameter tractable canonization and isomorphism test for graphs of bounded treewidth;

Lokshtanov, Pilipczuk, Pilipczuk, Saurabh; FOCS 2014

Izomorfizm grafów o małym treewidthie

- **Izomorfizm grafów**: Dane grafy G i H, sprawdzić czy G i H są izomorficzne.
- Klasyczna złożoność tego problemu jest wielkim problemem otwartym.
- **(Bodlaender'90)**: Złożoność $n^{\mathcal{O}(t)}$, jeśli $\mathbf{tw}(G), \mathbf{tw}(H) \leq t$.
- Na pytanie, czy FPT, odpowiadamy w pracy:

  Fixed-parameter tractable canonization and isomorphism test for graphs of bounded treewidth;

  Lokshtanov, Pilipczuk, Pilipczuk, Saurabh; FOCS 2014

- **Czas działania**: $t^{\mathcal{O}(t^5)} \cdot n^5$.

Izomorfizm grafów o małym treewidthie

- **Izomorfizm grafów**: Dane grafy G i H, sprawdzić czy G i H są izomorficzne.
- Klasyczna złożoność tego problemu jest wielkim problemem otwartym.
- **(Bodlaender'90)**: Złożoność $n^{\mathcal{O}(t)}$, jeśli $\mathbf{tw}(G), \mathbf{tw}(H) \leq t$.
- Na pytanie, czy FPT, odpowiadamy w pracy:

Fixed-parameter tractable canonization and isomorphism test for graphs of bounded treewidth;

Lokshtanov, Pilipczuk, Pilipczuk, Saurabh; FOCS 2014

- **Czas działania**: $t^{\mathcal{O}(t^5)} \cdot n^5$.
- **Idea**: Użyć aproksymacyjnego podejścia Robertsona i Seymoura do znalezienia kanonicznej dekompozycji drzewiastej.

Optymalizacja czasu działania

- **Pytania złożonościowe**: Czy problem jest w klasie FPT, czy jest W-trudny?

Optymalizacja czasu działania

- **Pytania złożonościowe**: Czy problem jest w klasie FPT, czy jest W-trudny?
- **Pytania algorytmiczne**: W złożoności postaci $f(k) \cdot n^c$, jakie jest najlepsze możliwe f? A jakie najlepsze możliwe c?

Optymalizacja czasu działania

- **Pytania złożonościowe**: Czy problem jest w klasie FPT, czy jest W-trudny?
- **Pytania algorytmiczne**: W złożoności postaci $f(k) \cdot n^c$, jakie jest najlepsze możliwe f? A jakie najlepsze możliwe c?
- Dla wielu problemów na ogólnych grafach, f musi być co najmniej pojedynczo-wykładnicze przy założeniu ETH.

Optymalizacja czasu działania

- **Pytania złożonościowe**: Czy problem jest w klasie FPT, czy jest W-trudny?
- **Pytania algorytmiczne**: W złożoności postaci $f(k) \cdot n^c$, jakie jest najlepsze możliwe f? A jakie najlepsze możliwe c?
- Dla wielu problemów na ogólnych grafach, f musi być co najmniej pojedynczo-wykładnicze przy założeniu ETH.
  - *Exponential-Time Hypothesis* $\simeq \mathrm{3SAT}$ nie da się rozwiązać w czasie $2^{o(n)}$.

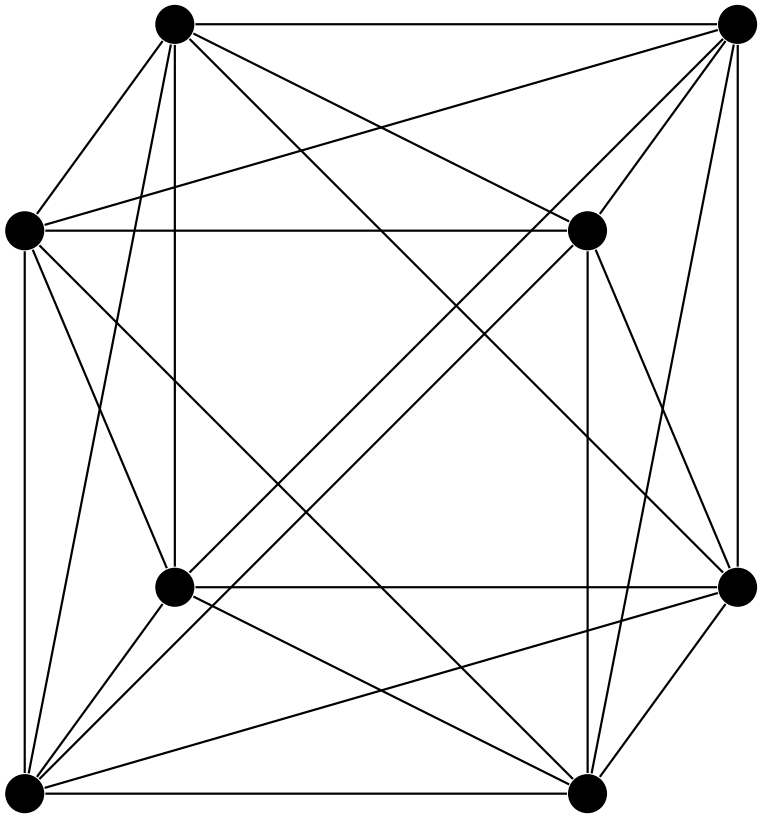
Optymalizacja czasu działania

- **Pytania złożonościowe**: Czy problem jest w klasie FPT, czy jest W-trudny?
- **Pytania algorytmiczne**: W złożoności postaci $f(k) \cdot n^c$, jakie jest najlepsze możliwe f? A jakie najlepsze możliwe c?
- Dla wielu problemów na ogólnych grafach, f musi być co najmniej pojedynczo-wykładnicze przy założeniu ETH.
  - *Exponential-Time Hypothesis* $\simeq \mathrm{3SAT}$ nie da się rozwiązać w czasie $2^{o(n)}$.
- Ale są bardziej egzotyczne ograniczenia dolne.

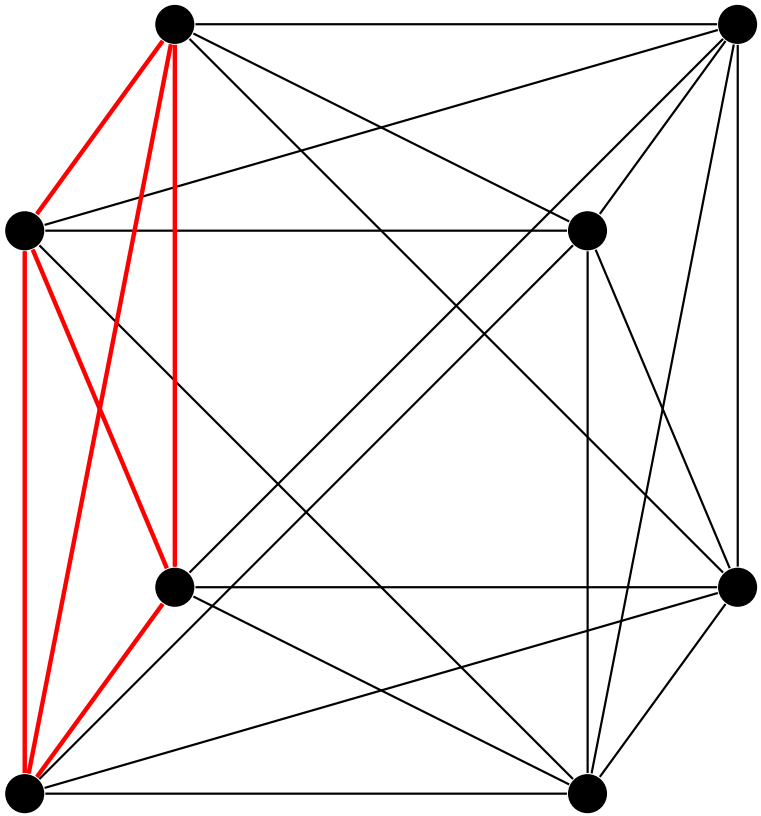
Optymalizacja czasu działania

- **Pytania złożonościowe**: Czy problem jest w klasie FPT, czy jest W-trudny?
- **Pytania algorytmiczne**: W złożoności postaci $f(k) \cdot n^c$, jakie jest najlepsze możliwe f? A jakie najlepsze możliwe c?
- Dla wielu problemów na ogólnych grafach, f musi być co najmniej pojedynczo-wykładnicze przy założeniu ETH.
  - *Exponential-Time Hypothesis* $\simeq \mathrm{3SAT}$ nie da się rozwiązać w czasie $2^{o(n)}$.
- Ale są bardziej egzotyczne ograniczenia dolne.
- Edge Clique Cover: Mając dany graf G i liczbę k, czy krawędzie G da się pokryć przy pomocy k klik w G?

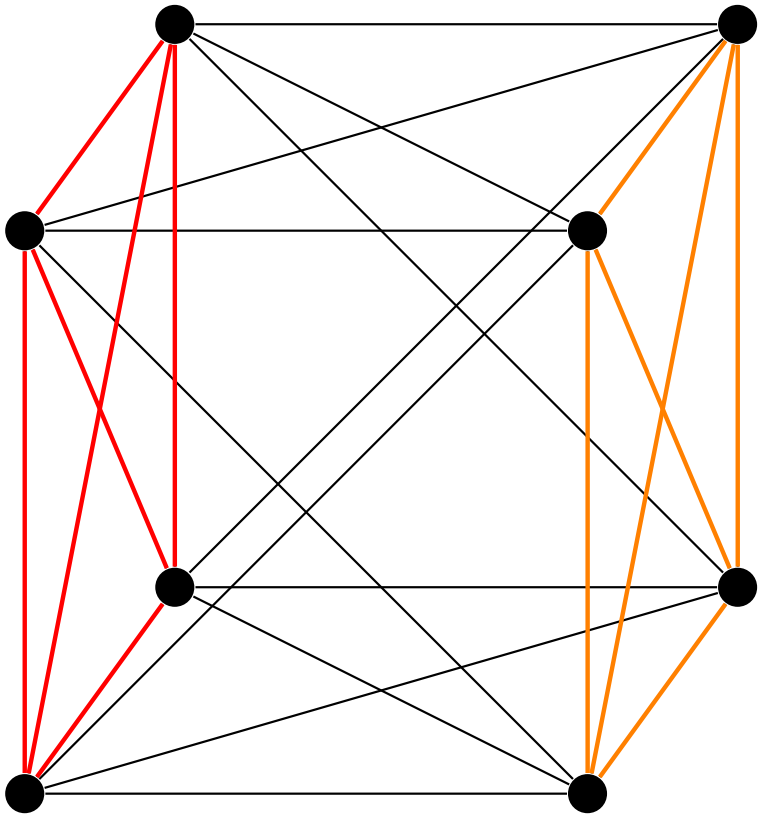
Edge Clique Cover



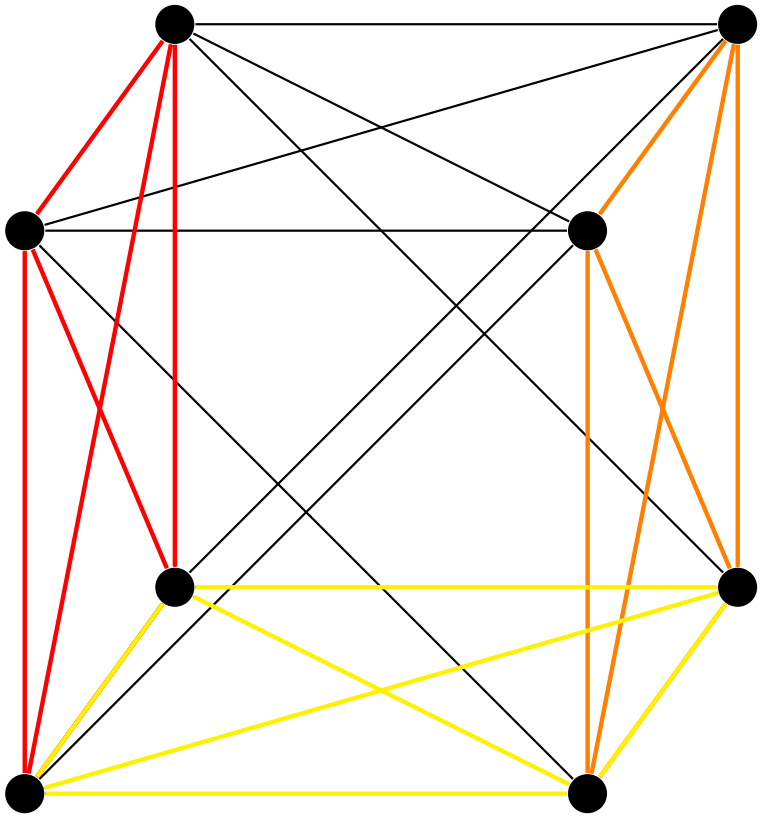
Edge Clique Cover



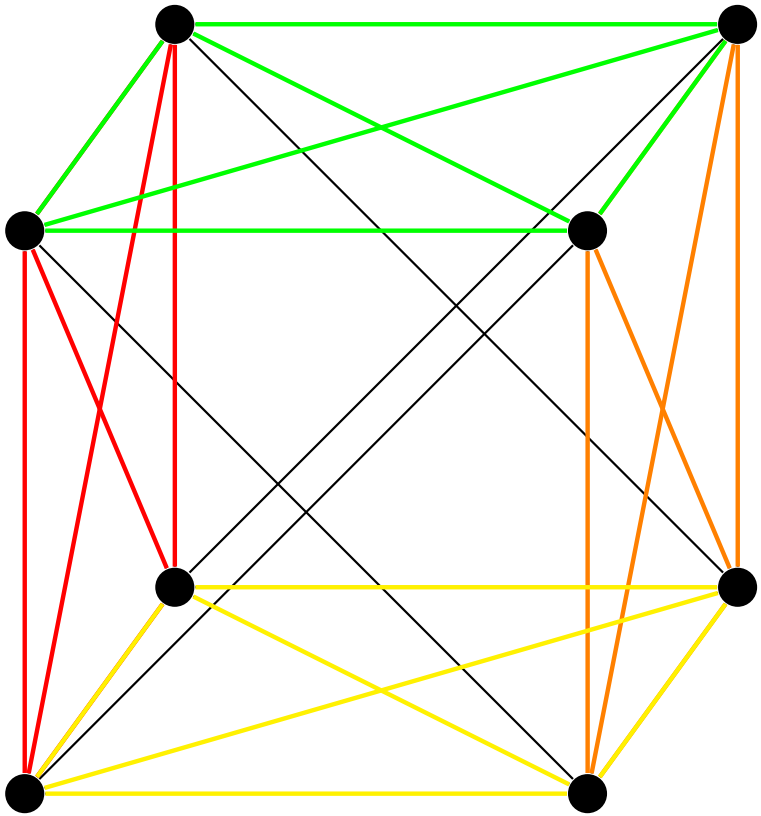
Edge Clique Cover



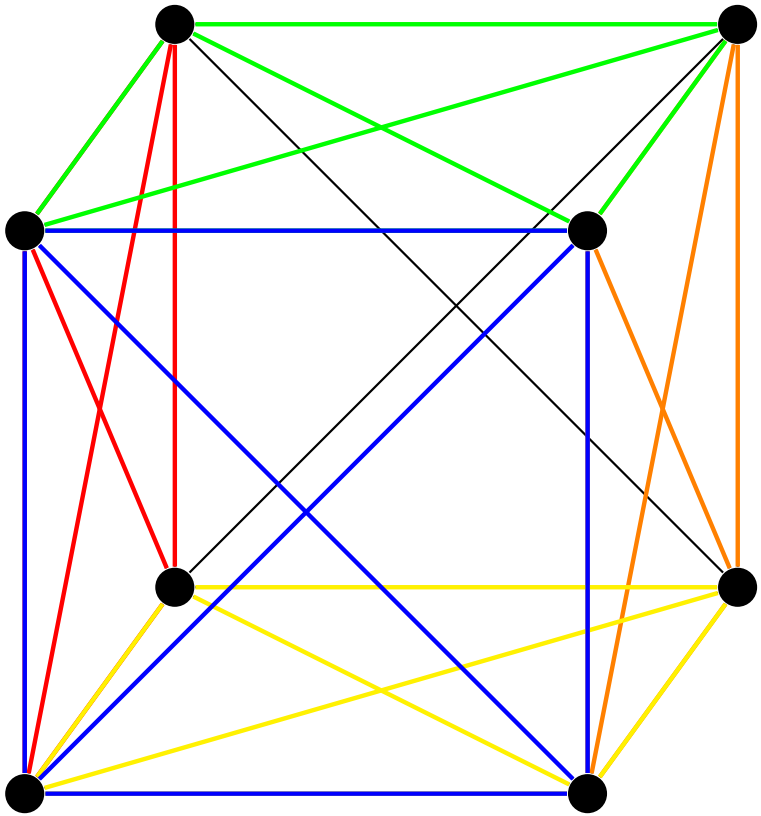
Edge Clique Cover



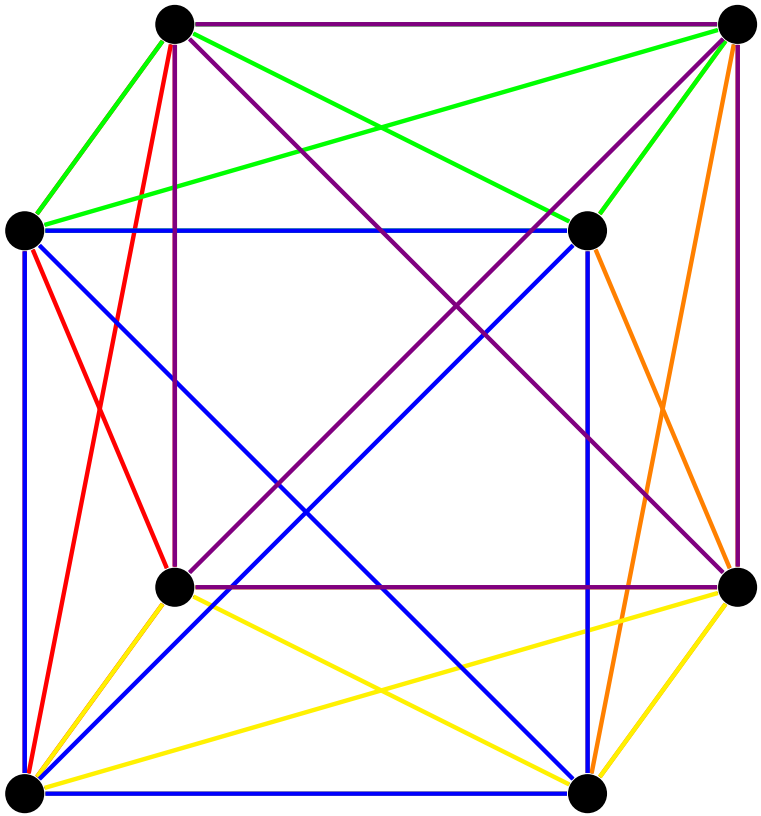
Edge Clique Cover



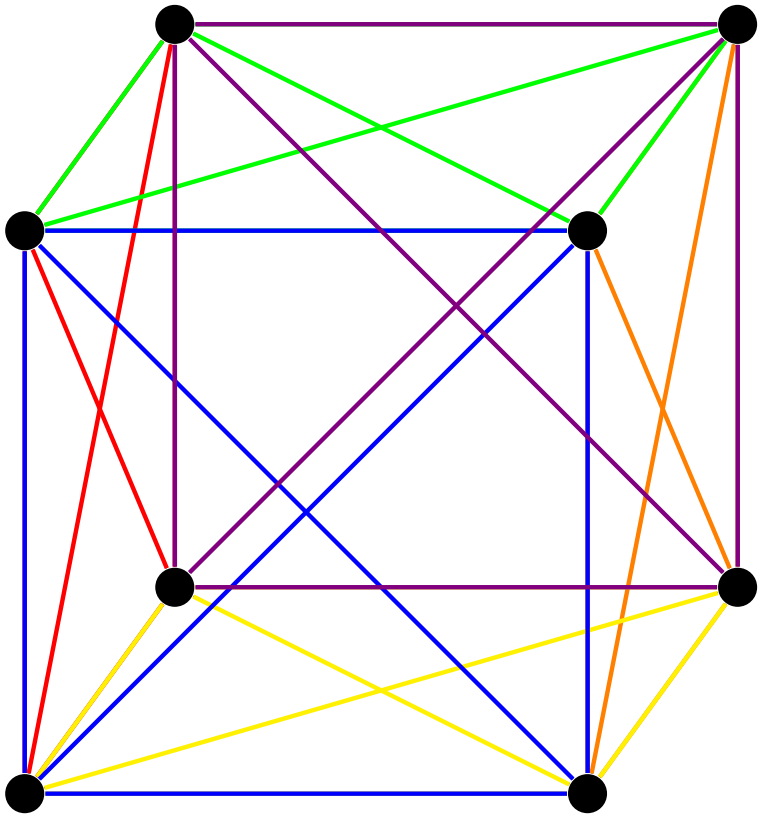
Edge Clique Cover



Edge Clique Cover

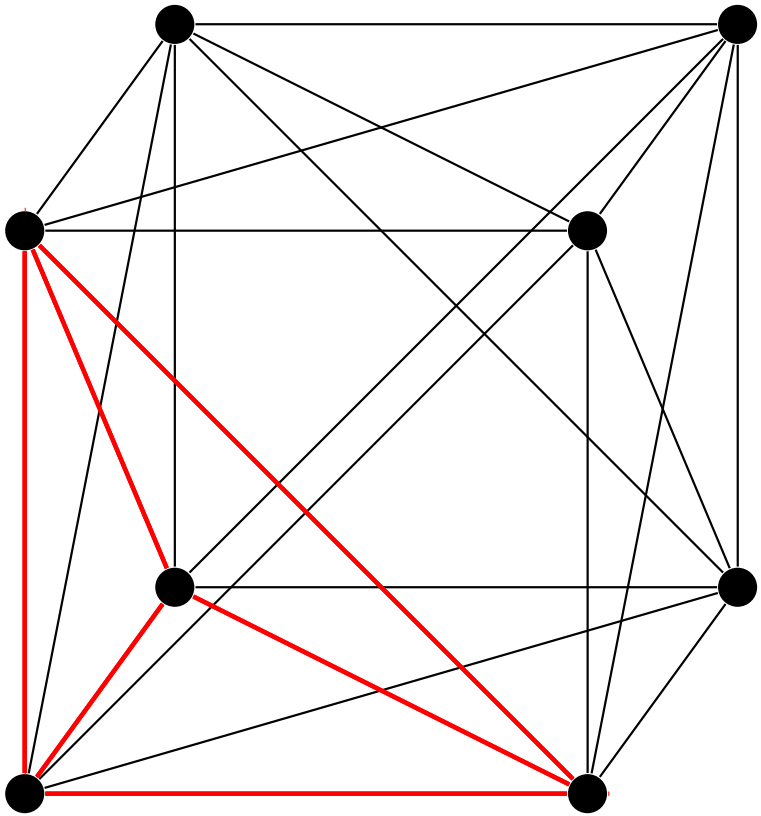


Edge Clique Cover

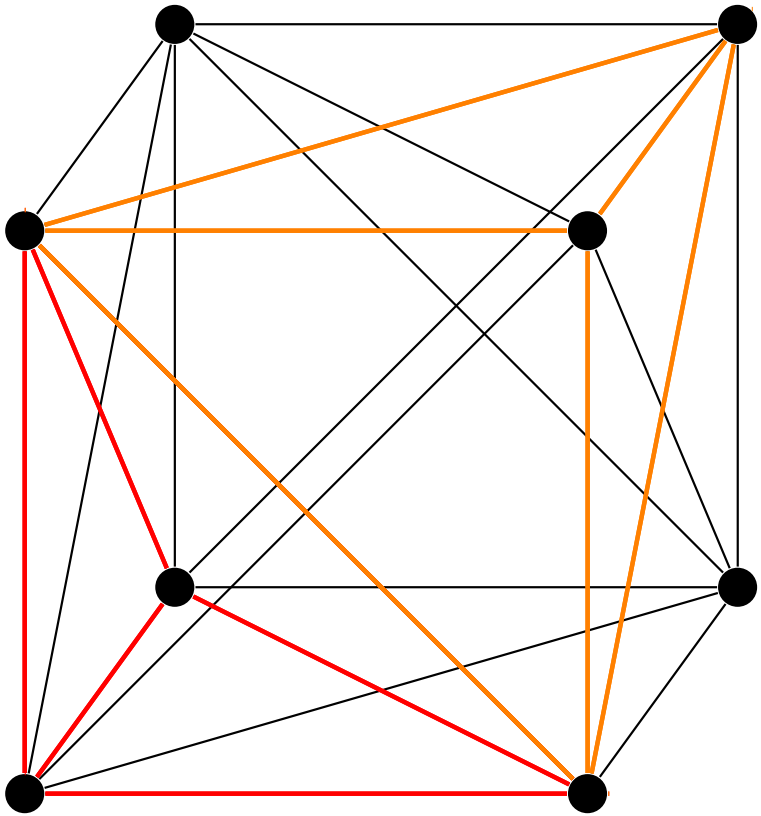


Rozwiązanie z 6 klikami

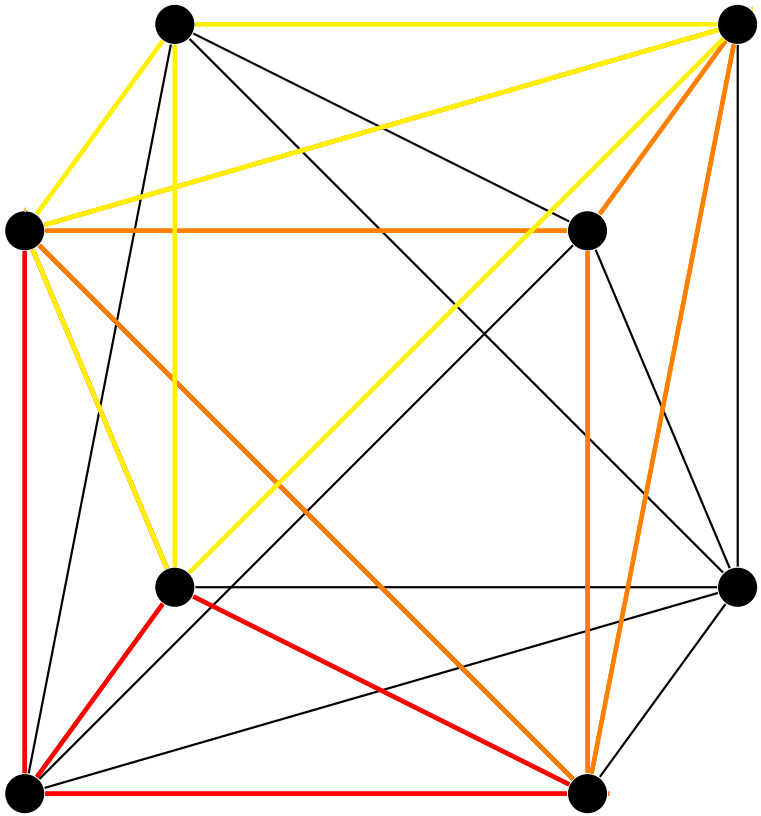
Edge Clique Cover



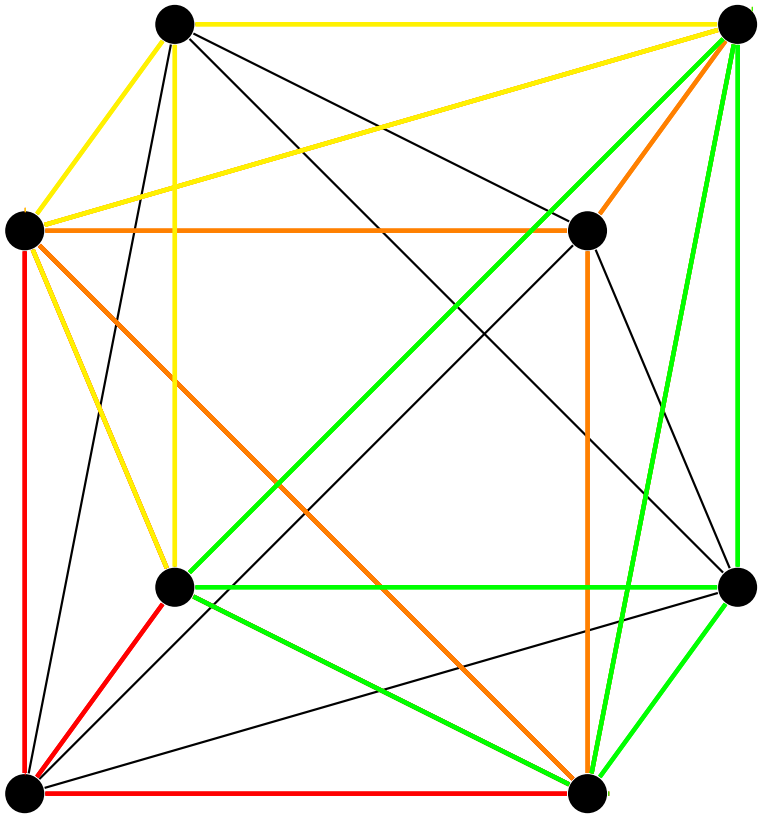
Edge Clique Cover



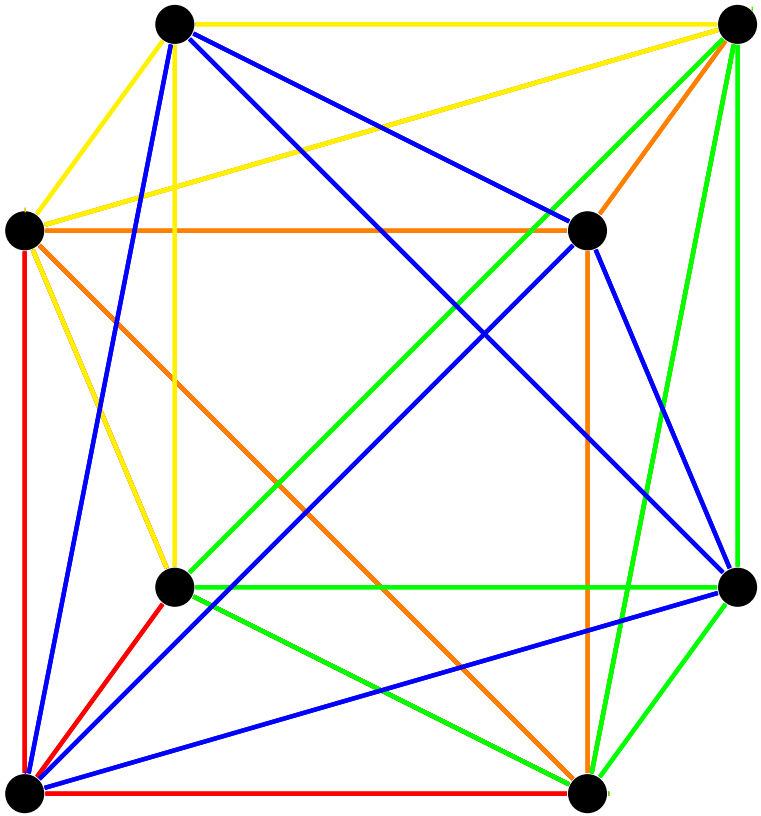
Edge Clique Cover



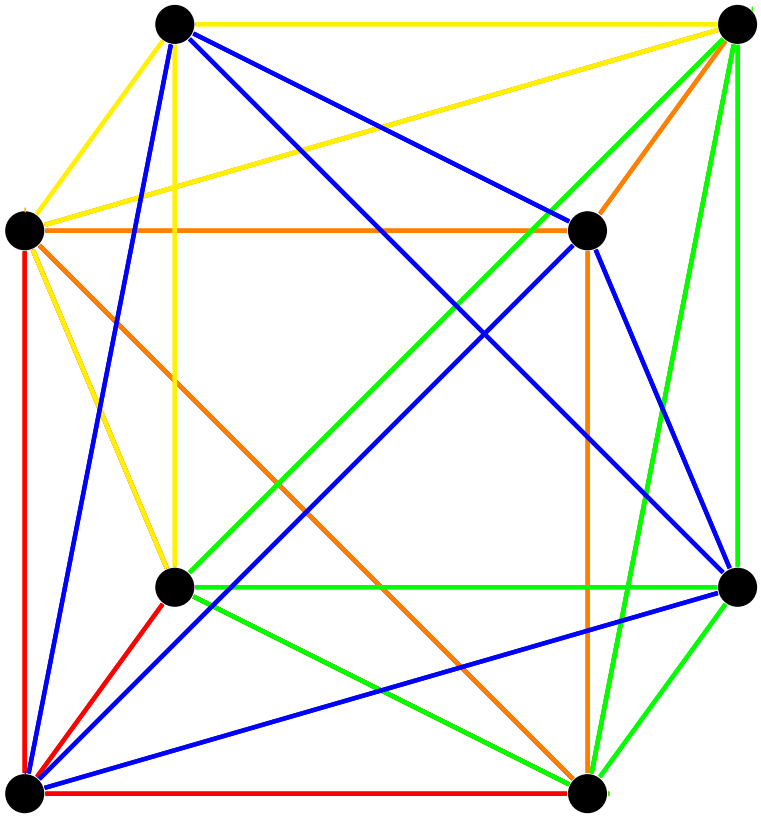
Edge Clique Cover



Edge Clique Cover



Edge Clique Cover



Rozwiązanie z 5 klikami

Edge Clique Cover: wyniki

- Praca:

Known algorithms for Edge Clique Cover *are probably optimal*;

Cygan, Pilipczuk, Pilipczuk; SODA 2013

Edge Clique Cover: wyniki

- Praca:

Known algorithms for Edge Clique Cover *are probably optimal*;

Cygan, Pilipczuk, Pilipczuk; SODA 2013

- Proste redukcje $\Rightarrow$ Zmniejszenie liczby wierzchołków do co najwyżej 2^k

Edge Clique Cover: wyniki

- Praca:

Known algorithms for Edge Clique Cover *are probably optimal*;

Cygan, Pilipczuk, Pilipczuk; SODA 2013

- Proste redukcje $\Rightarrow$ Zmniejszenie liczby wierzchołków do co najwyżej 2^k
- Algorytm brutalny na tym **jądrze** $\Rightarrow$ Złożoność $2^{2^{\mathcal{O}(k)}} + \mathrm{poly}(n)$.

Edge Clique Cover: wyniki

- Praca:

Known algorithms for Edge Clique Cover *are probably optimal*;

Cygan, Pilipczuk, Pilipczuk; SODA 2013

- Proste redukcje $\Rightarrow$ Zmniejszenie liczby wierzchołków do co najwyżej 2^k
- Algorytm brutalny na tym **jądrze** $\Rightarrow$ Złożoność $2^{2^{\mathcal{O}(k)}} + \mathrm{poly}(n)$.
- **Nasze wyniki**: To jest optymalne!

Edge Clique Cover: wyniki

- Praca:

Known algorithms for Edge Clique Cover *are probably optimal*;

Cygan, Pilipczuk, Pilipczuk; SODA 2013

- Proste redukcje $\Rightarrow$ Zmniejszenie liczby wierzchołków do co najwyżej 2^k
- Algorytm brutalny na tym **jądrze** $\Rightarrow$ Złożoność $2^{2^{\mathcal{O}(k)}} + \mathrm{poly}(n)$.
- **Nasze wyniki**: To jest optymalne!
  - Istnienie algorytmu o złożoności $2^{2^{o(k)}} \cdot \mathrm{poly}(n)$ przeczyłoby ETH.

Edge Clique Cover: wyniki

- Praca:

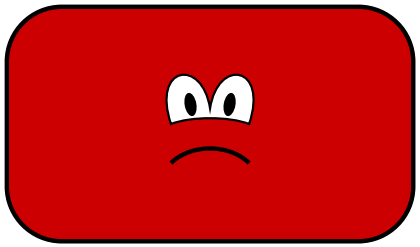
  Known algorithms for Edge Clique Cover *are probably optimal*;

  Cygan, Pilipczuk, Pilipczuk; SODA 2013

- Proste redukcje $\Rightarrow$ Zmniejszenie liczby wierzchołków do co najwyżej 2^k
- Algorytm brutalny na tym **jądrze** $\Rightarrow$ Złożoność $2^{2^{\mathcal{O}(k)}} + \mathrm{poly}(n)$.
- **Nasze wyniki**: To jest optymalne!
  - Istnienie algorytmu o złożoności $2^{2^{o(k)}} \cdot \mathrm{poly}(n)$ przeczyłoby ETH.
  - Istnienie jądra o wielkości $2^{o(k)}$ implikowałoby $\mathrm{P}{=}\mathrm{NP}$.

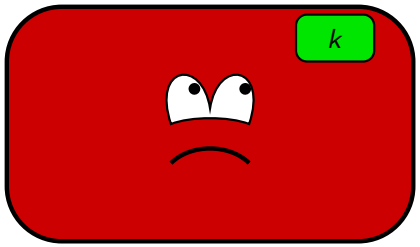
Kernelizacja

Kernelizacja



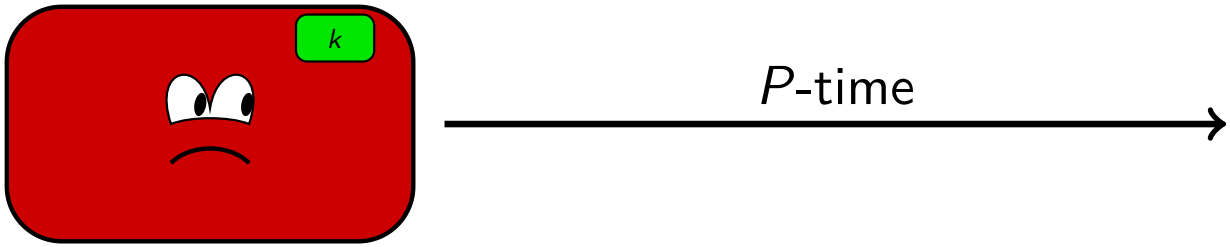
instancja L

Kernelizacja

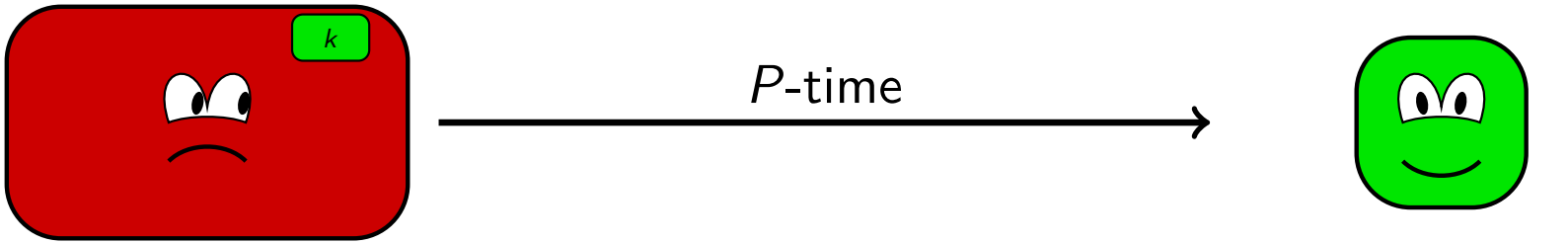


instancja L

Kernelizacja



Kernelizacja

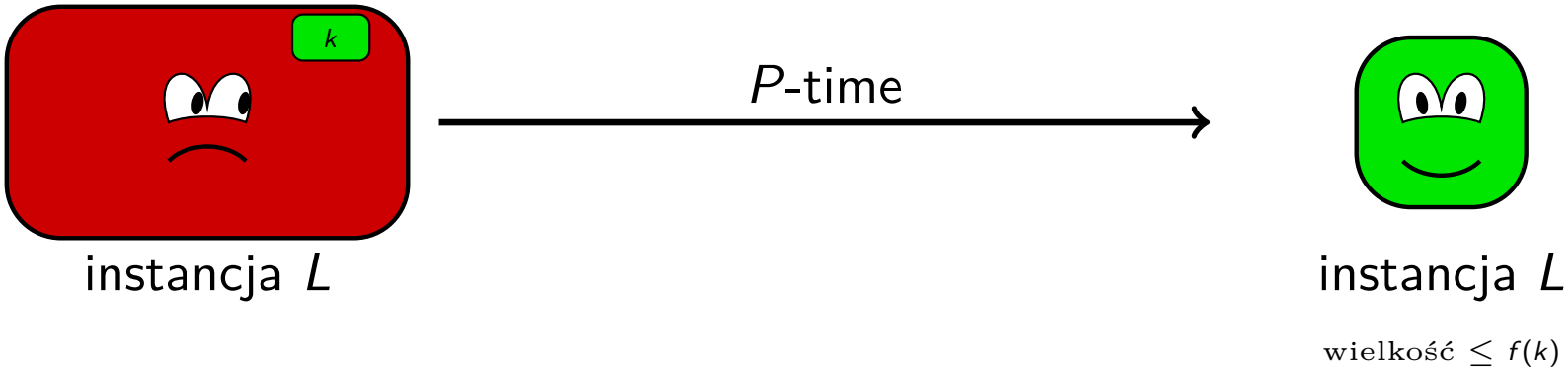


instancja L

instancja L

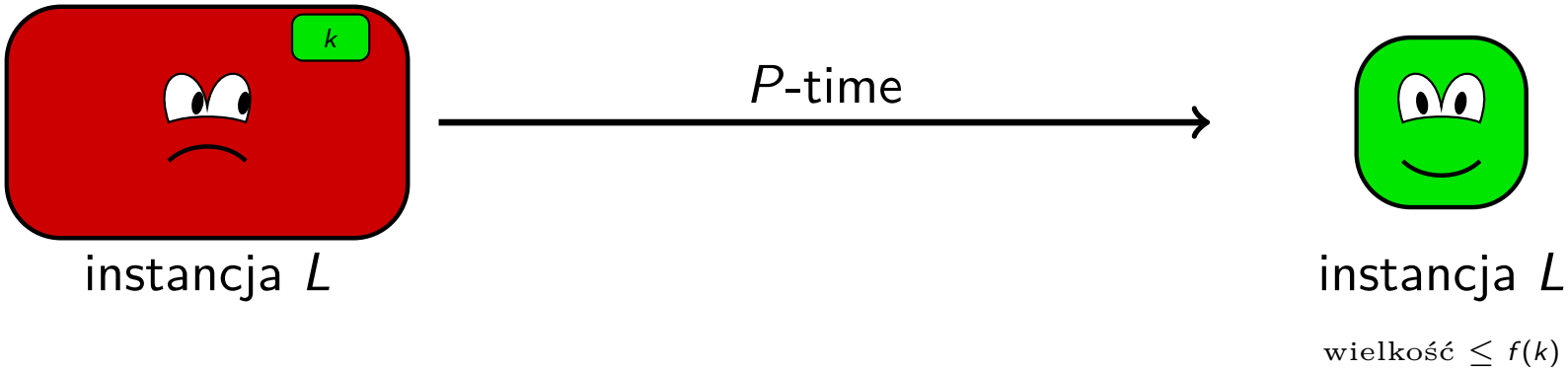
$\text{wielkość} \leq f(k)$

Kernelizacja



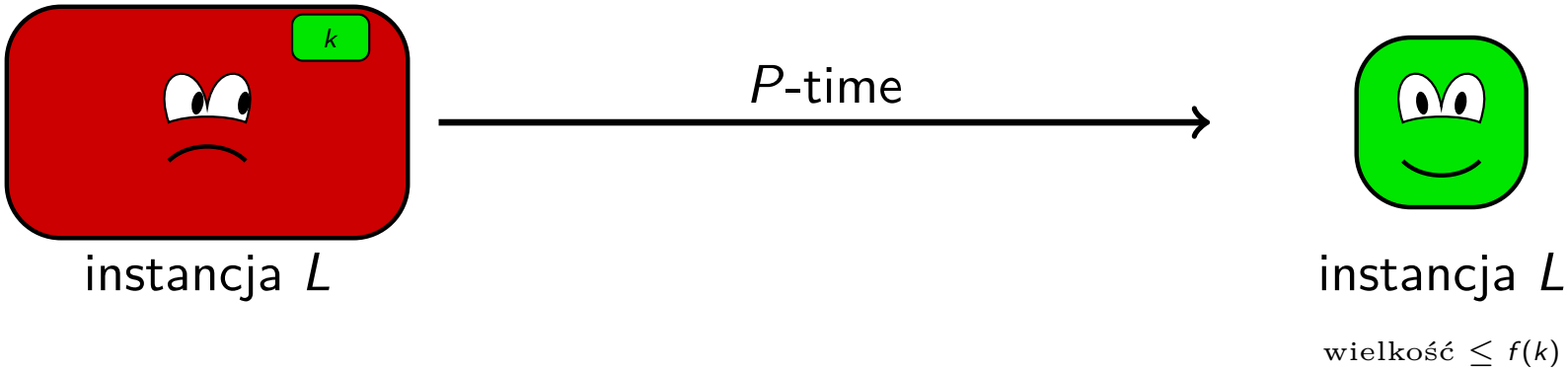
- Jeśli f jest wielomianowe, to problem ma *wielomianowe jądro*.

Kernelizacja



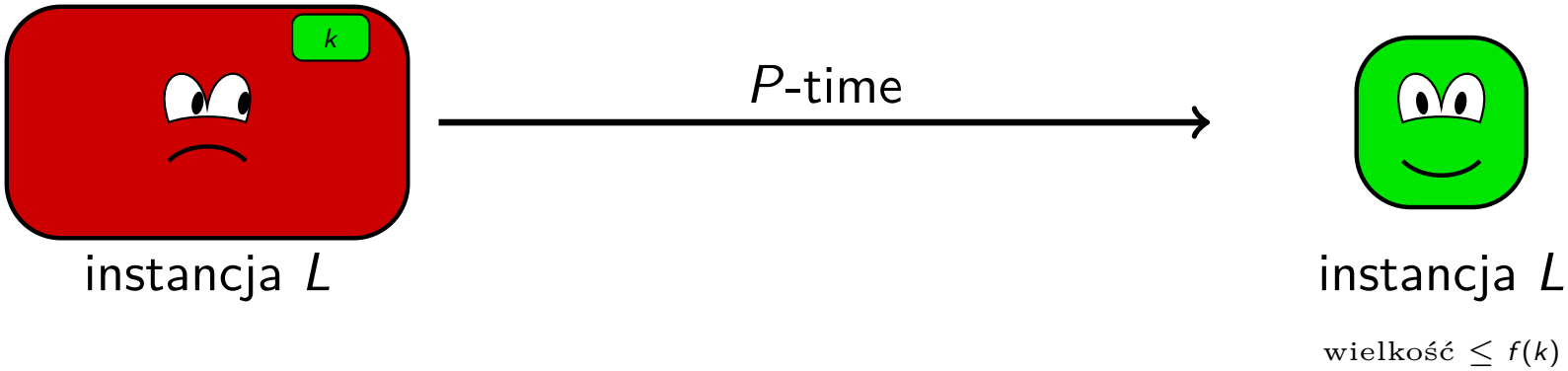
- Jeśli f jest wielomianowe, to problem ma *wielomianowe jądro*.
- Problem jest FPT wtw. ma jakieś jądro.

Kernelizacja



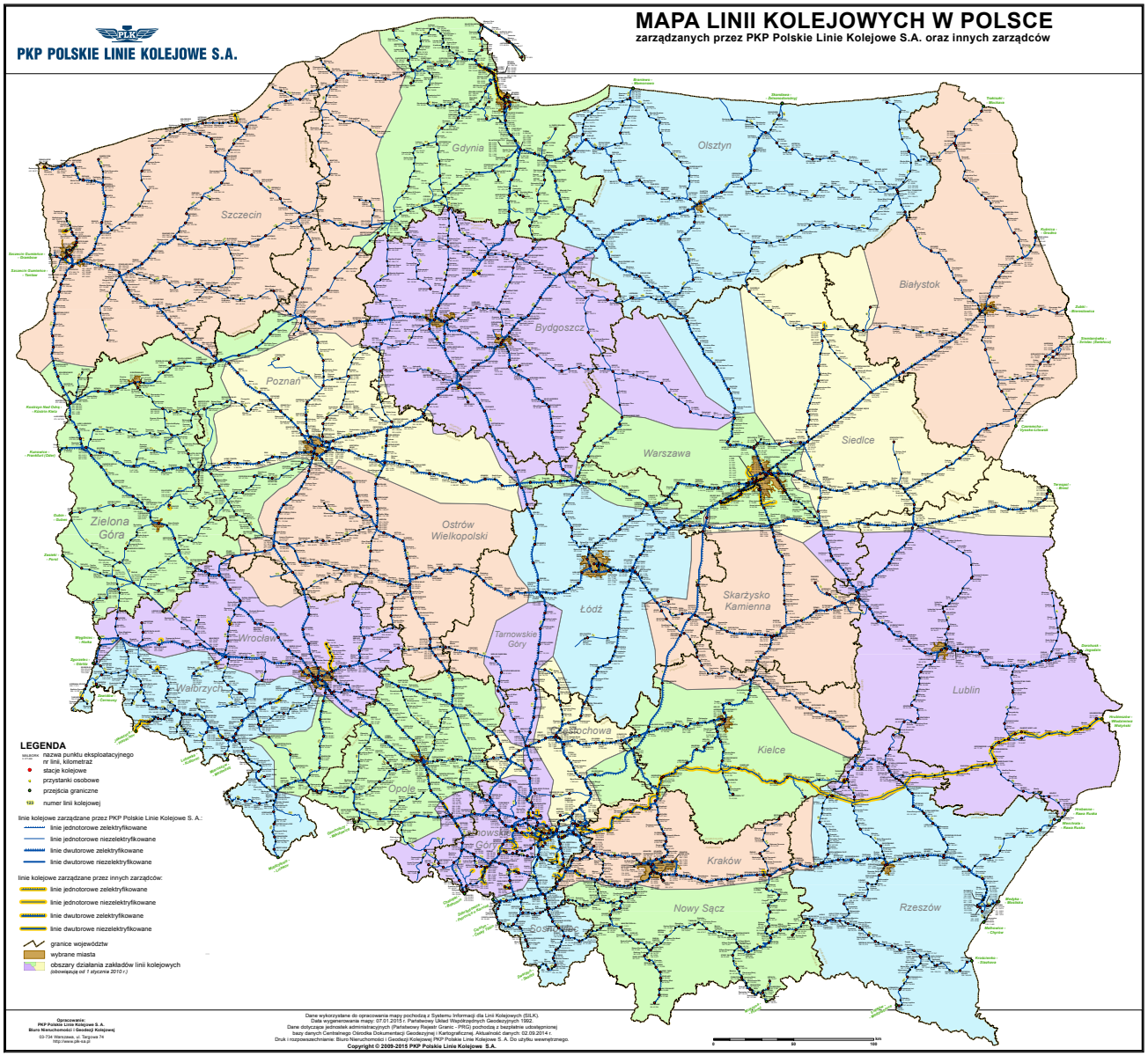
- Jeśli f jest wielomianowe, to problem ma *wielomianowe jądro*.
- Problem jest FPT wtw. ma jakieś jądro.
- Nie każdy problem FPT ma wielomianowe jądro.

Kernelizacja

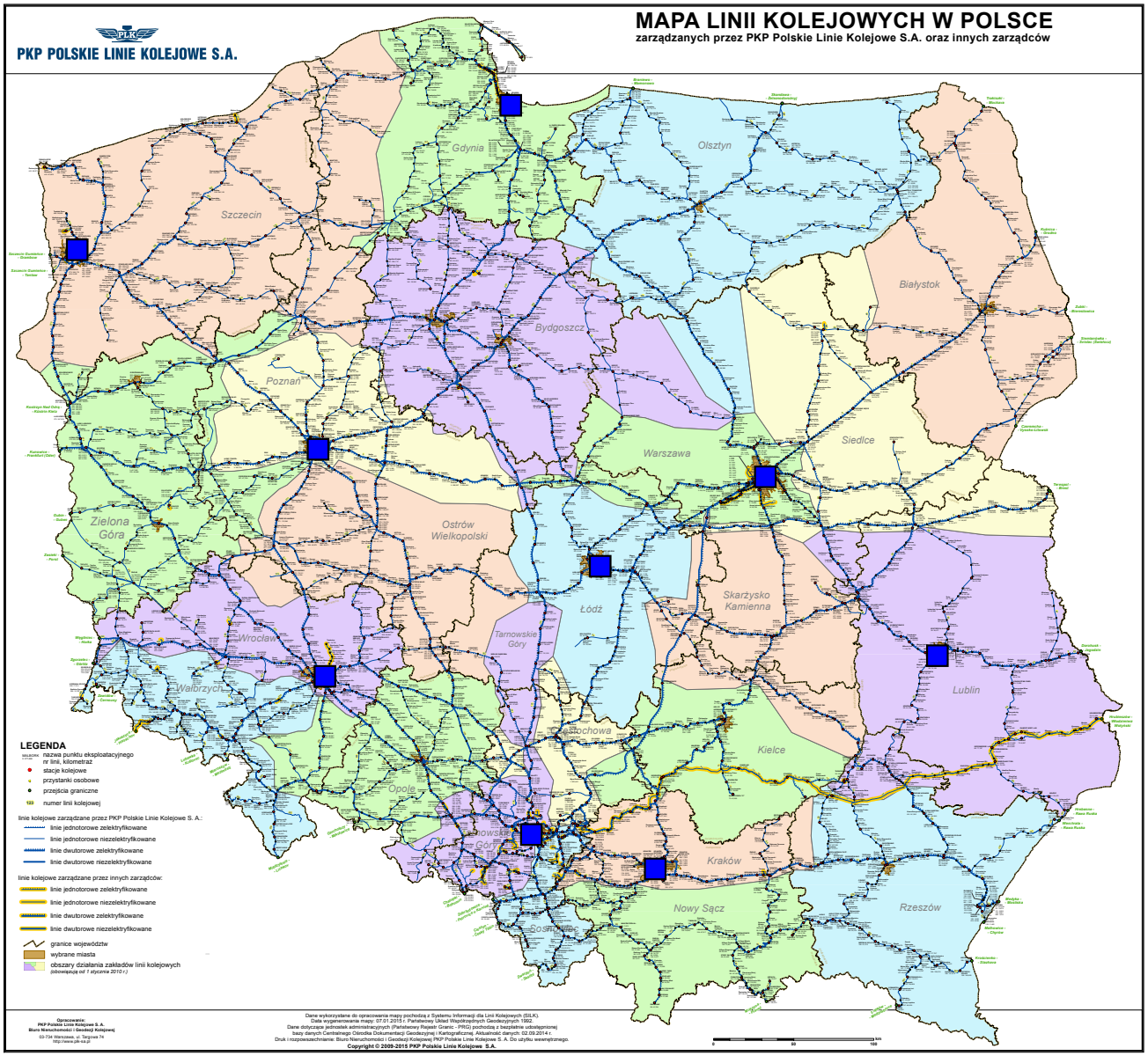


- Jeśli f jest wielomianowe, to problem ma *wielomianowe jądro*.
- Problem jest FPT wtw. ma jakieś jądro.
- Nie każdy problem FPT ma wielomianowe jądro.
  - Istnieje metodologia pozwalająca to odróżniać (pod $\mathrm{NP} \not\subseteq \mathrm{coNP/poly}$).

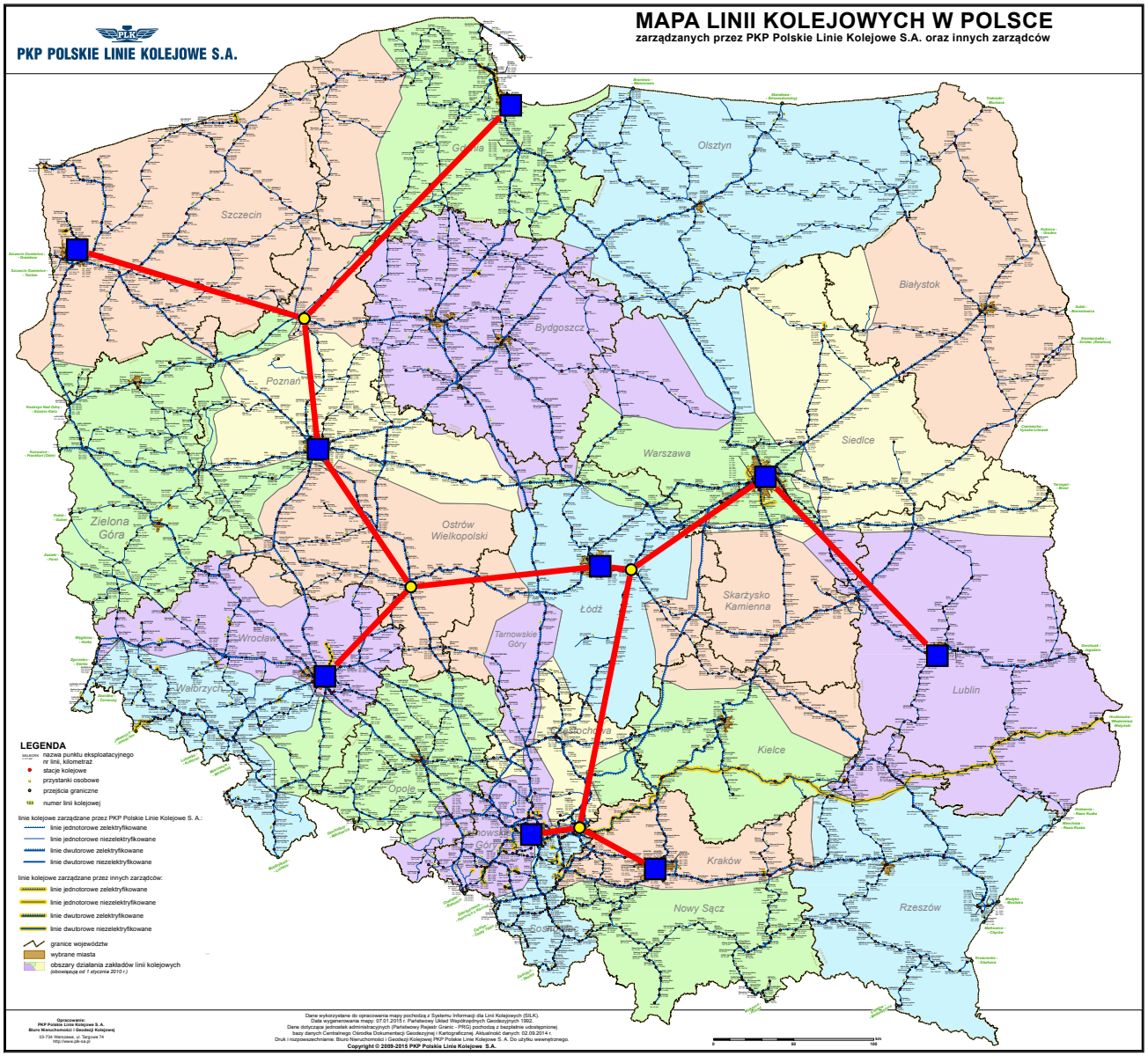
Drzewo Steinera



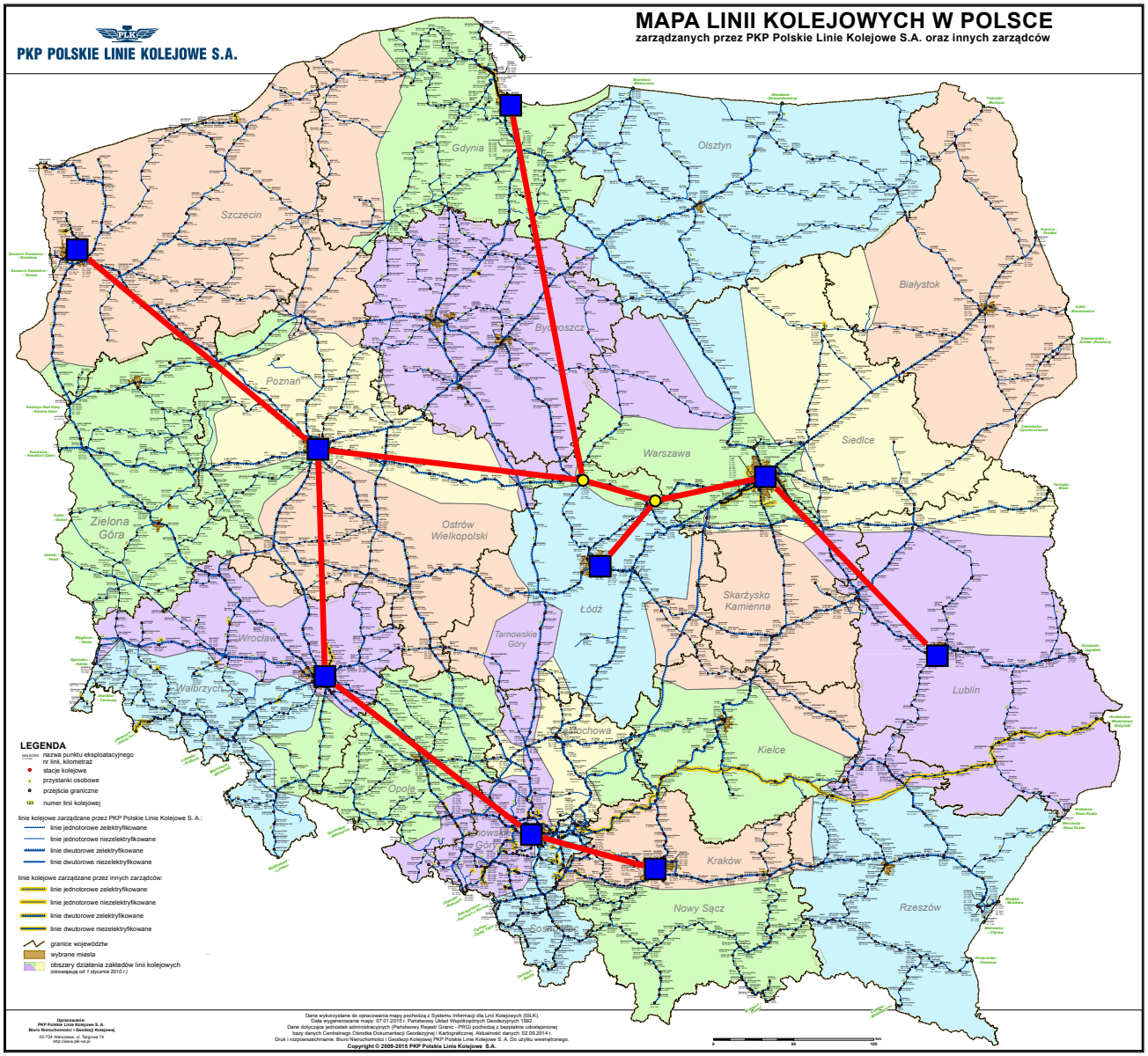
Drzewo Steinera



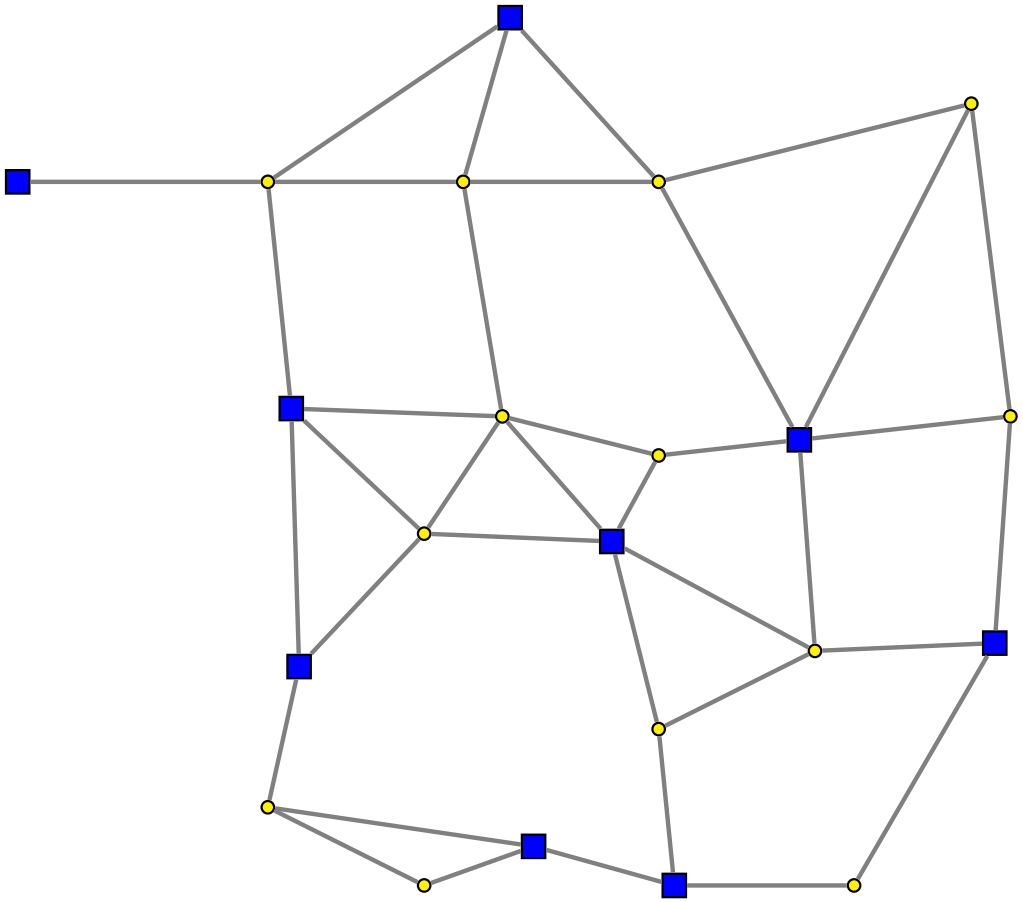
Drzewo Steinera



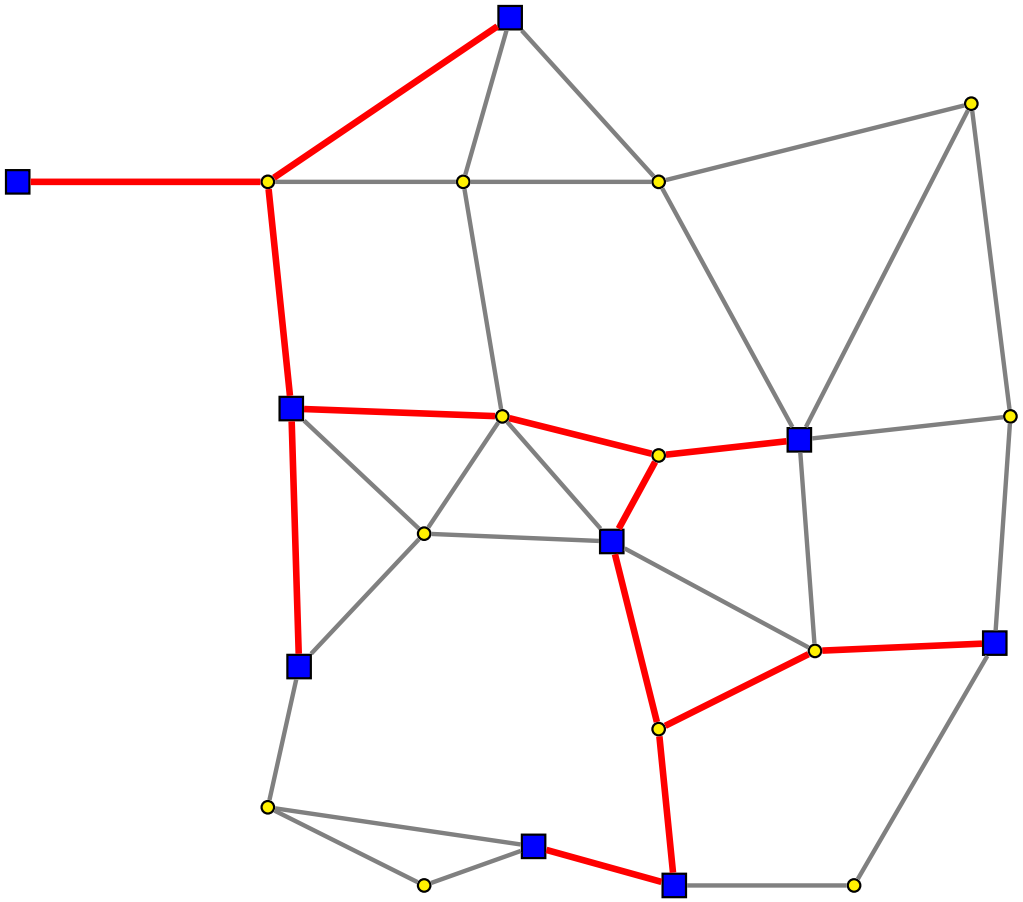
Drzewo Steinera



Drzewo Steinera



Drzewo Steinera



Złożoność parametryzowana Drzewa Steinera

- Dwa naturalne parametry:

Złożoność parametryzowana Drzewa Steinera

- Dwa naturalne parametry:
  - $|T|$, liczba terminali;

Złożoność parametryzowana Drzewa Steinera

- Dwa naturalne parametry:
  - $|T|$, liczba terminali;
  - k, docelowa wielkość drzewa.

Złożoność parametryzowana Drzewa Steinera

- Dwa naturalne parametry:
  - $|T|$, liczba terminali;
  - k, docelowa wielkość drzewa.
- **Ogólne grafy**:

Złożoność parametryzowana Drzewa Steinera

- Dwa naturalne parametry:
  - $|T|$, liczba terminali;
  - k, docelowa wielkość drzewa.
- **Ogólne grafy**:
  - Algorytm $2^{|T|} \cdot \mathrm{poly}(n)$ (prawdopodobnie optymalny).

Złożoność parametryzowana Drzewa Steinera

- Dwa naturalne parametry:
  - $|T|$, liczba terminali;
  - k, docelowa wielkość drzewa.
- **Ogólne grafy**:
  - Algorytm $2^{|T|} \cdot \mathrm{poly}(n)$ (prawdopodobnie optymalny).
  - Brak wielomianowego jądra nawet przy parametryzacji k.

Złożoność parametryzowana Drzewa Steinera

- Dwa naturalne parametry:
  - $|T|$, liczba terminali;
  - k, docelowa wielkość drzewa.
- **Ogólne grafy**:
  - Algorytm $2^{|T|} \cdot \mathrm{poly}(n)$ (prawdopodobnie optymalny).
  - Brak wielomianowego jądra nawet przy parametryzacji k.
- A co z grafami planarnymi?

Planarne Drzewo Steinera

- Praca:

Network sparsification for Steiner problems on planar and bounded-genus graphs;

Pilipczuk, Pilipczuk, Sankowski, van Leeuwen; FOCS 2014

Planarne Drzewo Steinera

- Praca:

  Network sparsification for Steiner problems on planar and bounded-genus graphs;

  Pilipczuk, Pilipczuk, Sankowski, van Leeuwen; FOCS 2014

- Jądro wielkości $\mathrm{poly}(k)$ dla Drzewa Steinera na grafach planarnych.

Planarne Drzewo Steinera

- Praca:

  Network sparsification for Steiner problems on planar and bounded-genus graphs;

  Pilipczuk, Pilipczuk, Sankowski, van Leeuwen; FOCS 2014

- Jądro wielkości $\mathrm{poly}(k)$ dla Drzewa Steinera na grafach planarnych.
- Daje algorytm o złożoności $2^{\mathcal{O}(\sqrt{k \log k})} \cdot \mathrm{poly}(n)$.

Planarne Drzewo Steinera

- Praca:

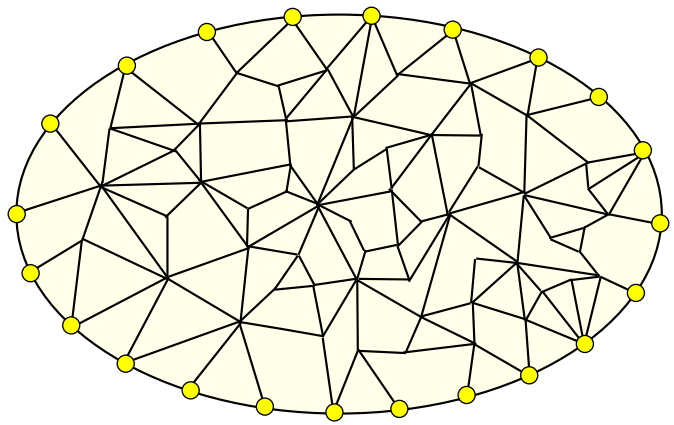
Network sparsification for Steiner problems on planar and bounded-genus graphs;

Pilipczuk, Pilipczuk, Sankowski, van Leeuwen; FOCS 2014

- Jądro wielkości $\mathrm{poly}(k)$ dla Drzewa Steinera na grafach planarnych.
- Daje algorytm o złożoności $2^{\mathcal{O}(\sqrt{k \log k})} \cdot \mathrm{poly}(n)$.
- Nadal nie umiemy w żaden sposób wykorzystać planarności dla parametryzacji przez $|T|$.

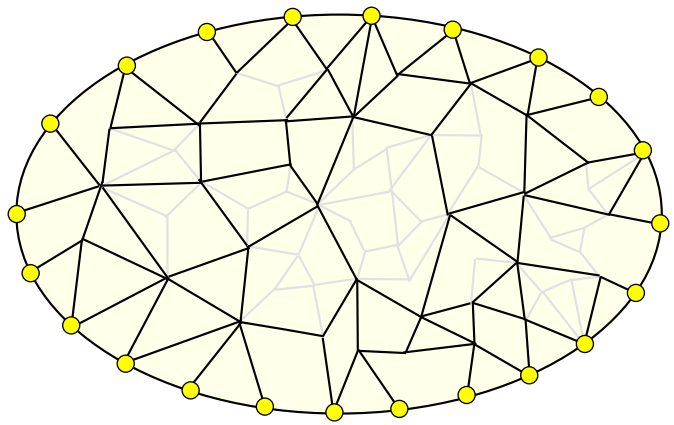
Główne twierdzenie

- **Dane**: Graf planarny G o zewnętrznej ścianie długości k.



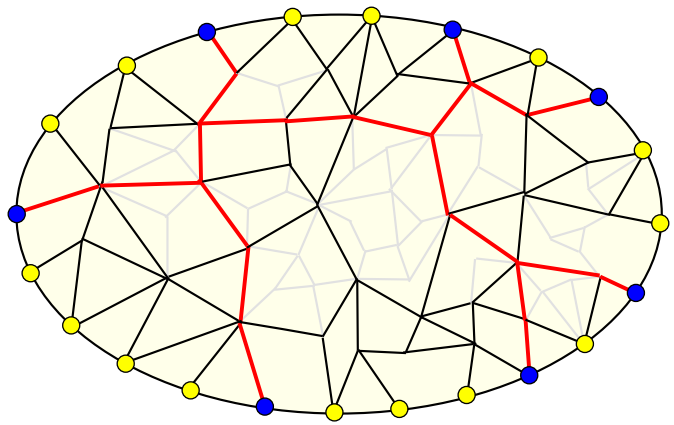
Główne twierdzenie

- **Dane**: Graf planarny G o zewnętrznej ścianie długości k.
- **Chcemy**: Możliwie mały podgraf $H \subseteq G$ taki, że:



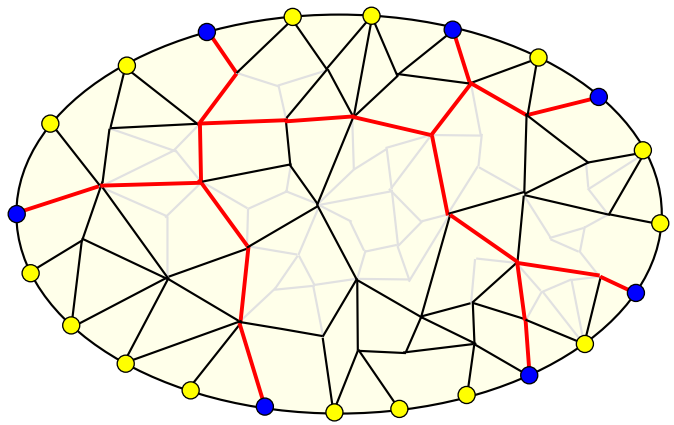
Główne twierdzenie

- **Dane**: Graf planarny G o zewnętrznej ścianie długości k.
- **Chcemy**: Możliwie mały podgraf $H \subseteq G$ taki, że:
  - Dla każdego doboru terminali z zewnętrznej ściany, wewnątrz H jest jakieś ich optymalne połączenie.



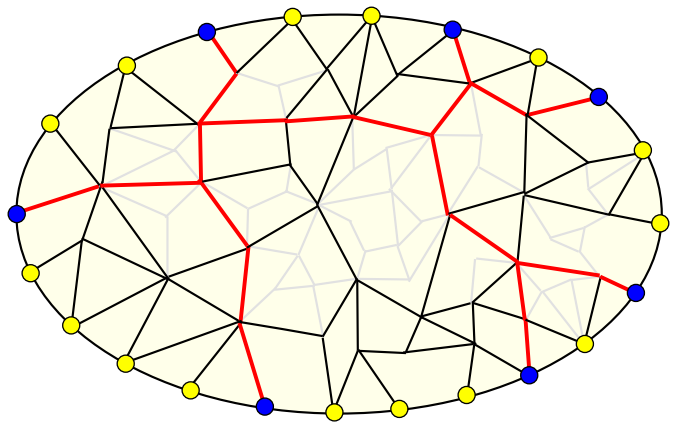
Główne twierdzenie

- **Dane**: Graf planarny G o zewnętrznej ścianie długości k.
- **Chcemy**: Możliwie mały podgraf $H \subseteq G$ taki, że:
  - Dla każdego doboru terminali z zewnętrznej ściany, wewnątrz H jest jakieś ich optymalne połączenie.
- **Naiwnie**: H posiadający $2^k \cdot k$ krawędzi.



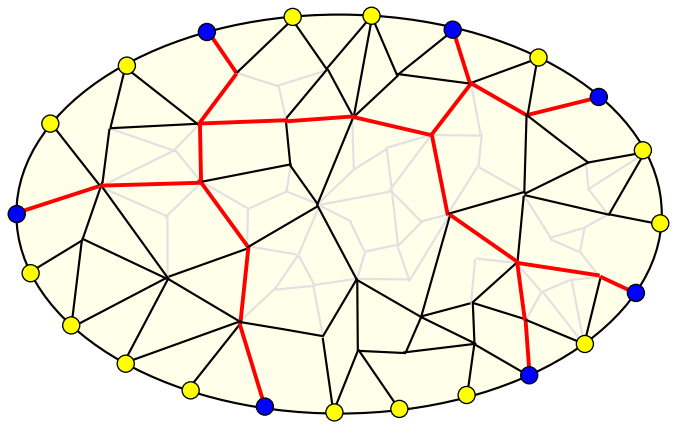
Główne twierdzenie

- **Dane**: Graf planarny G o zewnętrznej ścianie długości k.
- **Chcemy**: Możliwie mały podgraf $H \subseteq G$ taki, że:
  - Dla każdego doboru terminali z zewnętrznej ściany, wewnątrz H jest jakieś ich optymalne połączenie.
- **Naiwnie**: H posiadający $2^k \cdot k$ krawędzi.
- **Nasz wynik**: H posiadający $\mathrm{poly}(k)$ krawędzi.



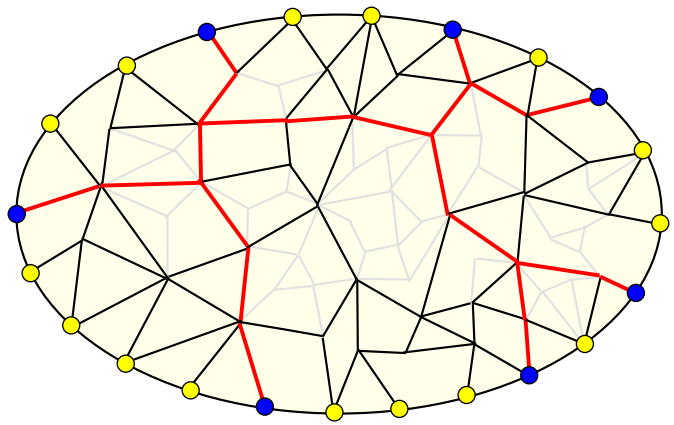
Główne twierdzenie

- **Dane**: Graf planarny G o zewnętrznej ścianie długości k.
- **Chcemy**: Możliwie mały podgraf $H \subseteq G$ taki, że:
  - Dla każdego doboru terminali z zewnętrznej ściany, wewnątrz H jest jakieś ich optymalne połączenie.
- **Naiwnie**: H posiadający $2^k \cdot k$ krawędzi.
- **Nasz wynik**: H posiadający $\mathrm{poly}(k)$ krawędzi.
- Dokładniej $\mathcal{O}(k^{142})$ krawędzi.



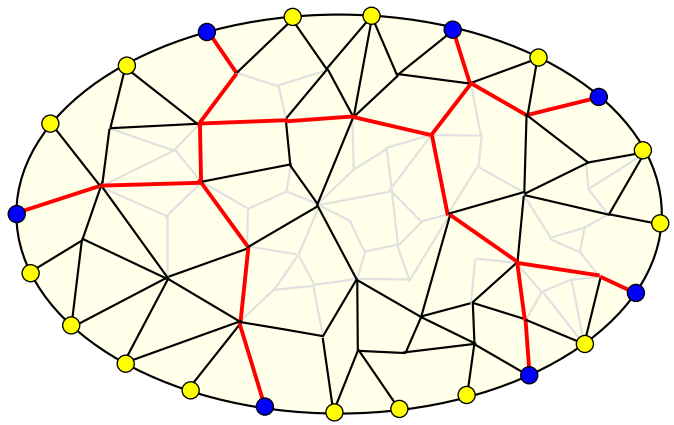
Główne twierdzenie

- **Dane**: Graf planarny G o zewnętrznej ścianie długości k.
- **Chcemy**: Możliwie mały podgraf $H \subseteq G$ taki, że:
  - Dla każdego doboru terminali z zewnętrznej ściany, wewnątrz H jest jakieś ich optymalne połączenie.
- **Naiwnie**: H posiadający $2^k \cdot k$ krawędzi.
- **Nasz wynik**: H posiadający $\mathrm{poly}(k)$ krawędzi.
- Dokładniej $\mathcal{O}(k^{142})$ krawędzi.
- Jeszcze dokładniej, $2\,159\,872\,407\,596 \cdot k^{142}$ krawędzi.



Główne twierdzenie

- **Dane**: Graf planarny G o zewnętrznej ścianie długości k.
- **Chcemy**: Możliwie mały podgraf $H \subseteq G$ taki, że:
  - Dla każdego doboru terminali z zewnętrznej ściany, wewnątrz H jest jakieś ich optymalne połączenie.
- **Naiwnie**: H posiadający $2^k \cdot k$ krawędzi.
- **Nasz wynik**: H posiadający $\mathrm{poly}(k)$ krawędzi.
- Dokładniej $\mathcal{O}(k^{142})$ krawędzi.
- Jeszcze dokładniej, $2\,159\,872\,407\,596 \cdot k^{142}$ krawędzi.
- **Hipoteza**: Istnieje H posiadający $\mathcal{O}(k^2)$ krawędzi.

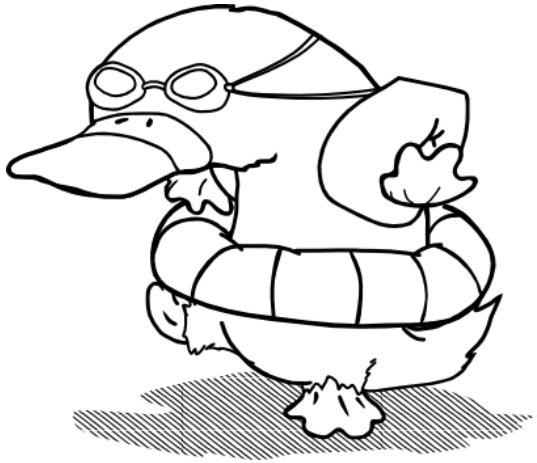


Książka

Parameterized algorithms

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

Springer, 2015.



Tikz faces based on a code by Raoul Kessels, `http://www.texample.net/tikz/examples/emoticons/`,

under Creative Commons Attribution 2.5 license (CC BY 2.5)