Karlsruher Institut für Technologie

Übung 8 – Algorithmen II

Michael Axtmann – *michael.axtmann@kit.edu*

http://algo2.iti.kit.edu/AlgorithmenII_WS16.php

Institut für Theoretische Informatik - Algorithmik II

Themenübersicht

Karlsruher Institut für Technologie

Übungsinhalt

- Approximationsalgorithmen
  (schwierige Probleme gut abschätzen)

- parametrisierte Algorithmen
  (schwierige Probleme in Spezialfällen exakt lösen)

Ergänzendes Material

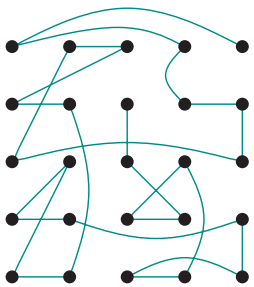
- parametrisierte Algorithmen: Rekurrenzrelationen
- parametrisierte Algorithmen: Vertex Cluster Deletion

Approximationsalgorithmen

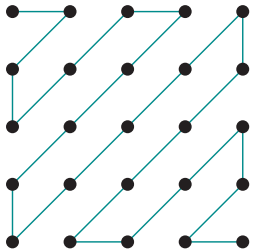
Grundlagen

Warum Lösungen abschätzen?

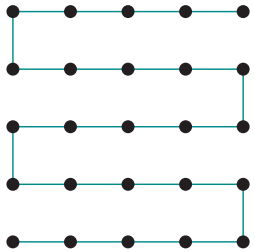
- es gibt “schwierige” Probleme (z.B. TSP mit exponentieller Laufzeit)
  - $\rightarrow$ exakte Berechnung nicht möglich zu unseren Lebzeiten
- vernünftige Näherungen effizient berechnen
  - $\rightarrow$ exakte/optimale Ergebnisse nicht immer wichtig
  - $\rightarrow$ “gute” Lösungen genügen oft
  - $\rightarrow$ aber Abstand zur korrekten Lösung wissenswert



Weglänge “lang”



Weglänge $8 + 16\sqrt{2}$



Weglänge 24

Approximationsalgorithmen

Gütemaß

- Ein Algorithmus *ALG* hat einen Approximationsfaktor ρ, wenn gilt

$$\frac{w\left(ALG(I)\right)}{w\left(OPT(I)\right)} \leq \rho$$

f.a. Probleminstanzen I, OPT optimale Lösung, w Bewertungsfunktion

(hier für Minimierungsprobleme, $\rho > 0$ ist obere Schranke)

Beispiel:

- *ALG* schätzt Distanz von Strecke x auf nächste Zweierpotenz $2^{\lceil \log |x| \rceil}$
- *OPT* bestimmt korrekte Distanz $|x|$

$$\Rightarrow \quad \frac{w(ALG)}{w(OPT)} = \frac{2^{\lceil \log |x| \rceil}}{|x|} \leq \frac{2^{\log |x|+1}}{|x|} = \frac{2|x|}{|x|} = 2 = \rho$$

(Zahlenbeispiel: $|x| = 2^{10} + 1 \quad \rightarrow \quad \frac{2^{11}}{2^{10}+1} = 2 - \frac{2}{2^{10}+1} \approx 2$)

Approximationsalgorithmen

Klassen

Approximationsprobleme klassifzierbar durch

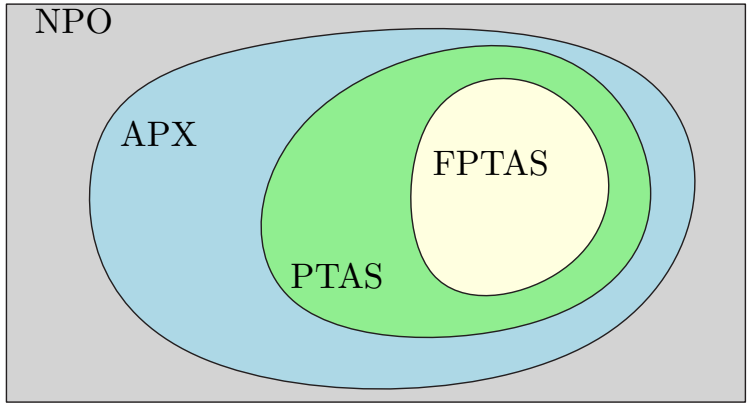
- Laufzeit $T(n, \varepsilon)$
- Approximationsfaktor $\rho(n, \varepsilon)$

Klassen an Approximationsproblemen

APX (*approximable*)
$\rho = const.$, T polynomiell in n

PTAS (*polynomial time approximation scheme*)
$\rho = 1 \pm \varepsilon$, bel. $\varepsilon \in (0, 1)$, T poly. in n

FPTAS (*fully polynomial time approximation scheme*)
$\rho = 1 \pm \varepsilon$, bel. $\varepsilon \in (0, 1)$, T poly. in n, $\frac{1}{\varepsilon}$



Beispiele:

$\Rightarrow$ $T(n, \varepsilon) = n^{\frac{1}{\varepsilon}}$, $\rho(n, \varepsilon) = 2$

$\Rightarrow$ $T(n, \varepsilon) = n^{\frac{1}{\varepsilon}}$, $\rho(n, \varepsilon) = 1 - \varepsilon$

$\Rightarrow$ $T(n, \varepsilon) = \frac{1}{\varepsilon} n$, $\rho(n, \varepsilon) = 1 + 2\varepsilon$

Approximationsalgorithmen

Klassen

Approximationsprobleme klassifzierbar durch

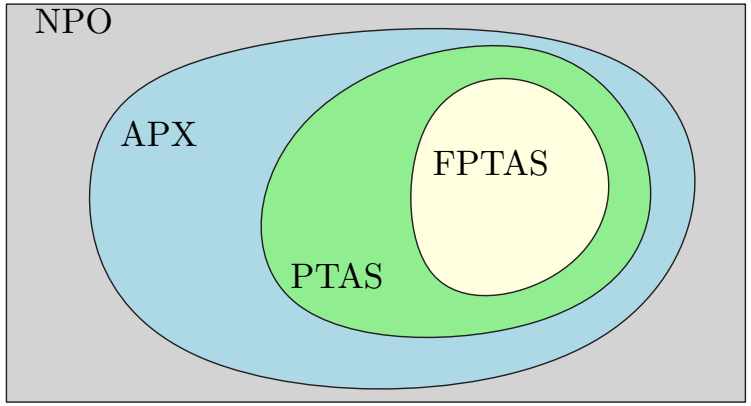
- Laufzeit $T(n, \varepsilon)$
- Approximationsfaktor $\rho(n, \varepsilon)$

Klassen an Approximationsproblemen

APX (*approximable*)
$\rho = const.$, T polynomiell in n

PTAS (*polynomial time approximation scheme*)
$\rho = 1 \pm \varepsilon$, bel. $\varepsilon \in (0, 1)$, T poly. in n

FPTAS (*fully polynomial time approximation scheme*)
$\rho = 1 \pm \varepsilon$, bel. $\varepsilon \in (0, 1)$, T poly. in n, $\frac{1}{\varepsilon}$



Beispiele:

$\Rightarrow$ $T(n, \varepsilon) = n^{\frac{1}{\varepsilon}}$, $\rho(n, \varepsilon) = 2$ (APX)

$\Rightarrow$ $T(n, \varepsilon) = n^{\frac{1}{\varepsilon}}$, $\rho(n, \varepsilon) = 1 - \varepsilon$

$\Rightarrow$ $T(n, \varepsilon) = \frac{1}{\varepsilon} n$, $\rho(n, \varepsilon) = 1 + 2\varepsilon$

Approximationsalgorithmen

Klassen

Approximationsprobleme klassifzierbar durch

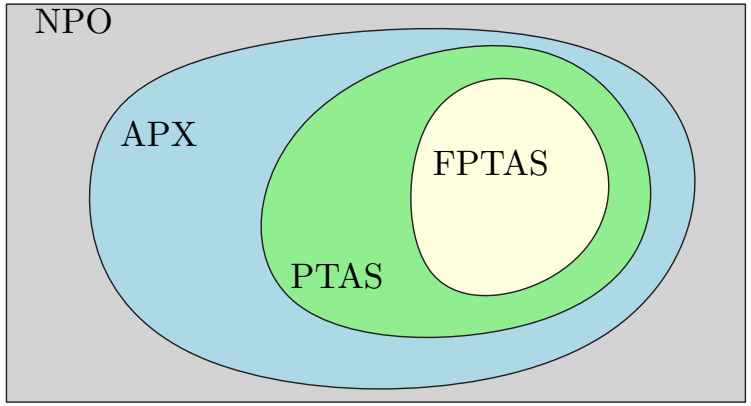
- Laufzeit $T(n, \varepsilon)$
- Approximationsfaktor $\rho(n, \varepsilon)$

Klassen an Approximationsproblemen

APX (*approximable*)
$\rho = const.$, T polynomiell in n

PTAS (*polynomial time approximation scheme*)
$\rho = 1 \pm \varepsilon$, bel. $\varepsilon \in (0, 1)$, T poly. in n

FPTAS (*fully polynomial time approximation scheme*)
$\rho = 1 \pm \varepsilon$, bel. $\varepsilon \in (0, 1)$, T poly. in n, $\frac{1}{\varepsilon}$



Beispiele:

$\Rightarrow$ $T(n, \varepsilon) = n^{\frac{1}{\varepsilon}}$, $\rho(n, \varepsilon) = 2$ (APX)

$\Rightarrow$ $T(n, \varepsilon) = n^{\frac{1}{\varepsilon}}$, $\rho(n, \varepsilon) = 1 - \varepsilon$ (PTAS)

$\Rightarrow$ $T(n, \varepsilon) = \frac{1}{\varepsilon} n$, $\rho(n, \varepsilon) = 1 + 2\varepsilon$

Approximationsalgorithmen

Klassen

Approximationsprobleme klassifzierbar durch

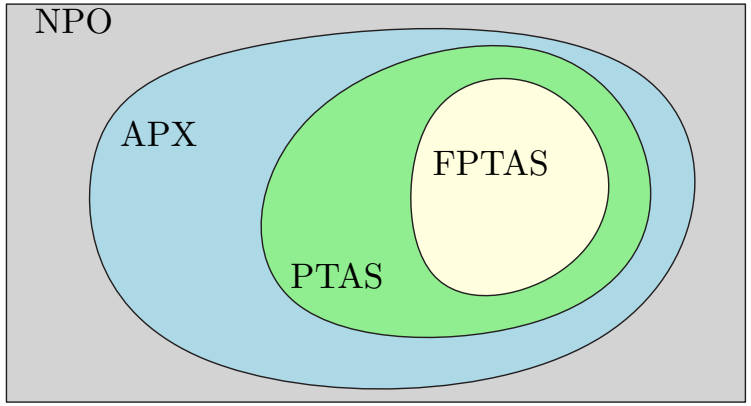
- Laufzeit $T(n, \varepsilon)$
- Approximationsfaktor $\rho(n, \varepsilon)$

Klassen an Approximationsproblemen

APX (*approximable*)
$\rho = const.$, T polynomiell in n

PTAS (*polynomial time approximation scheme*)
$\rho = 1 \pm \varepsilon$, bel. $\varepsilon \in (0, 1)$, T poly. in n

FPTAS (*fully polynomial time approximation scheme*)
$\rho = 1 \pm \varepsilon$, bel. $\varepsilon \in (0, 1)$, T poly. in n, $\frac{1}{\varepsilon}$



Beispiele:

$\Rightarrow$ $T(n, \varepsilon) = n^{\frac{1}{\varepsilon}}$, $\rho(n, \varepsilon) = 2$ (APX)

$\Rightarrow$ $T(n, \varepsilon) = n^{\frac{1}{\varepsilon}}$, $\rho(n, \varepsilon) = 1 - \varepsilon$ (PTAS)

$\Rightarrow$ $T(n, \varepsilon) = \frac{1}{\varepsilon} n$, $\rho(n, \varepsilon) = 1 + 2\varepsilon$ (FPTAS)

Approximationsalgorithmen

Minimum Metric TSP (NP-hart)

Problemstellung

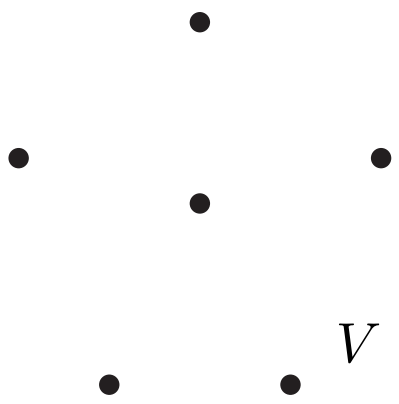
- Gegeben eine Menge an Punkten V in der Ebene
- Vollständiger Graph
- Kantengewichte erfüllen Dreiecks-Ungleichung
- Finde Kreis minimaler Länge der alle Punkte abläuft

Approximationsalgorithmen

Minimum Metric TSP

$3/2$-Approximation (Algorithmus) (Christofides-Heuristik)

- Graph $G = (V, E)$ (vollständig, ungerichtet)
- bestimme MST T
- bestimme Knoten U mit ungeradem Grad in T
- finde *minimales perfektes Matching* M auf (U, E)
  (alle Knoten gematcht, Summe der Gewichte der Matchingkanten minimal)
- füge Kanten M zu $T \to T'$
- bestimme Eulerkreis EK auf T'
  (alle Knoten haben geraden Grad)
- wandle EK zu Hamiltonkreis HK



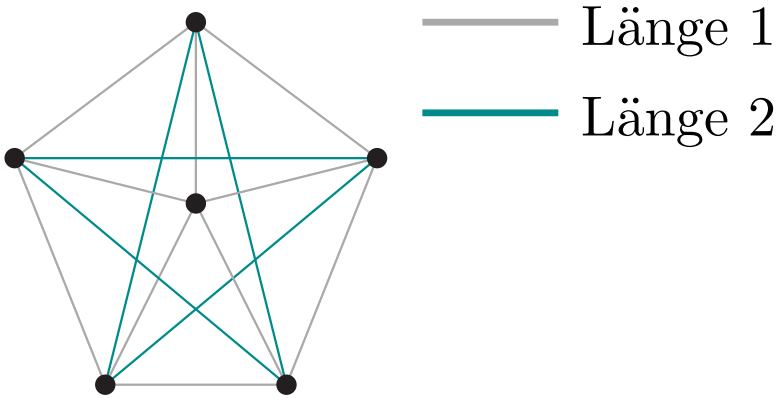
(Laufzeit durch Matching dominiert, $\mathcal{O}(n^3)$)

Approximationsalgorithmen

Minimum Metric TSP

$3/2$-Approximation (Algorithmus) (Christofides-Heuristik)

- Graph $G = (V, E)$ (vollständig, ungerichtet)
- bestimme MST T
- bestimme Knoten U mit ungeradem Grad in T
- finde *minimales perfektes Matching* M auf (U, E)
  (alle Knoten gematcht, Summe der Gewichte der Matchingkanten minimal)
- füge Kanten M zu $T \to T'$
- bestimme Eulerkreis EK auf T'
  (alle Knoten haben geraden Grad)
- wandle EK zu Hamiltonkreis HK



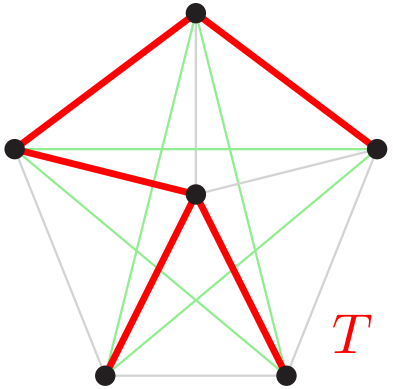
(Laufzeit durch Matching dominiert, $\mathcal{O}(n^3)$)

Approximationsalgorithmen

Minimum Metric TSP

$3/2$-Approximation (Algorithmus) (Christofides-Heuristik)

- Graph $G = (V, E)$ (vollständig, ungerichtet)
- bestimme MST T
- bestimme Knoten U mit ungeradem Grad in T
- finde *minimales perfektes Matching* M auf (U, E)
  (alle Knoten gematcht, Summe der Gewichte der Matchingkanten minimal)
- füge Kanten M zu $T \to T'$
- bestimme Eulerkreis EK auf T'
  (alle Knoten haben geraden Grad)
- wandle EK zu Hamiltonkreis HK



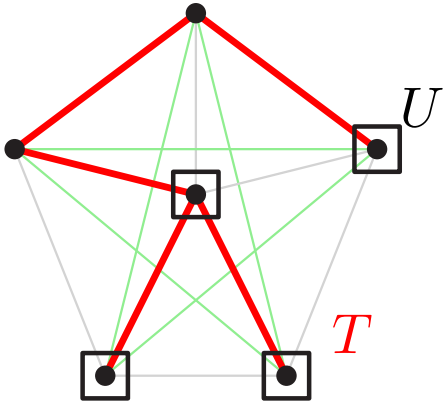
(Laufzeit durch Matching dominiert, $\mathcal{O}(n^3)$)

Approximationsalgorithmen

Minimum Metric TSP

$3/2$-Approximation (Algorithmus) (Christofides-Heuristik)

- Graph $G = (V, E)$ (vollständig, ungerichtet)
- bestimme MST T
- bestimme Knoten U mit ungeradem Grad in T
- finde *minimales perfektes Matching* M auf (U, E)
  (alle Knoten gematcht, Summe der Gewichte der Matchingkanten minimal)
- füge Kanten M zu $T \to T'$
- bestimme Eulerkreis EK auf T'
  (alle Knoten haben geraden Grad)
- wandle EK zu Hamiltonkreis HK



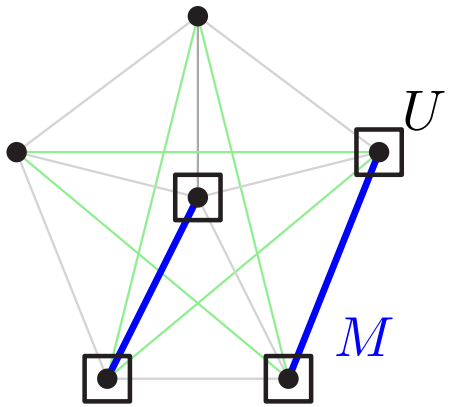
(Laufzeit durch Matching dominiert, $\mathcal{O}(n^3)$)

Approximationsalgorithmen

Minimum Metric TSP

$3/2$-Approximation (Algorithmus) (Christofides-Heuristik)

- Graph $G = (V, E)$ (vollständig, ungerichtet)
- bestimme MST T
- bestimme Knoten U mit ungeradem Grad in T
- finde *minimales perfektes Matching* M auf (U, E)
  (alle Knoten gematcht, Summe der Gewichte der Matchingkanten minimal)
- füge Kanten M zu $T \to T'$
- bestimme Eulerkreis EK auf T'
  (alle Knoten haben geraden Grad)
- wandle EK zu Hamiltonkreis HK



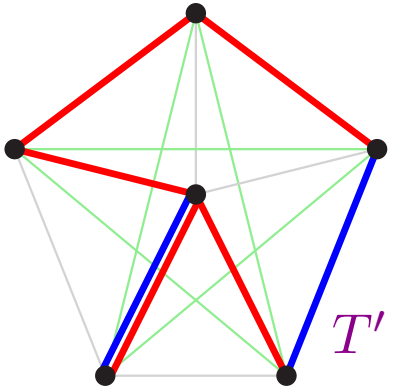
(Laufzeit durch Matching dominiert, $\mathcal{O}(n^3)$)

Approximationsalgorithmen

Minimum Metric TSP

$3/2$-Approximation (Algorithmus) (Christofides-Heuristik)

- Graph $G = (V, E)$ (vollständig, ungerichtet)
- bestimme MST T
- bestimme Knoten U mit ungeradem Grad in T
- finde *minimales perfektes Matching* M auf (U, E)
  (alle Knoten gematcht, Summe der Gewichte der Matchingkanten minimal)
- füge Kanten M zu $T \to T'$
- bestimme Eulerkreis EK auf T'
  (alle Knoten haben geraden Grad)
- wandle EK zu Hamiltonkreis HK



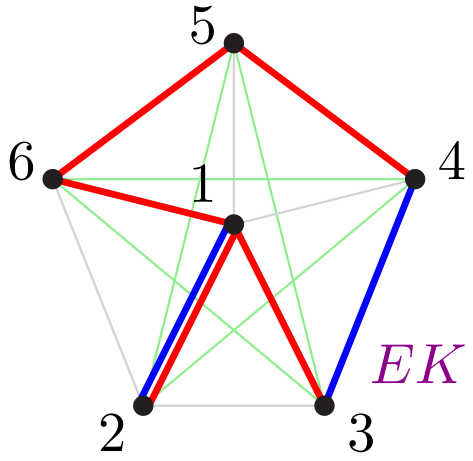
(Laufzeit durch Matching dominiert, $\mathcal{O}(n^3)$)

Approximationsalgorithmen

Minimum Metric TSP

$3/2$-Approximation (Algorithmus) (Christofides-Heuristik)

- Graph $G = (V, E)$ (vollständig, ungerichtet)
- bestimme MST T
- bestimme Knoten U mit ungeradem Grad in T
- finde *minimales perfektes Matching* M auf (U, E)
  (alle Knoten gematcht, Summe der Gewichte der Matchingkanten minimal)
- füge Kanten M zu $T \to T'$
- bestimme Eulerkreis EK auf T'
  (alle Knoten haben geraden Grad)
- wandle EK zu Hamiltonkreis HK



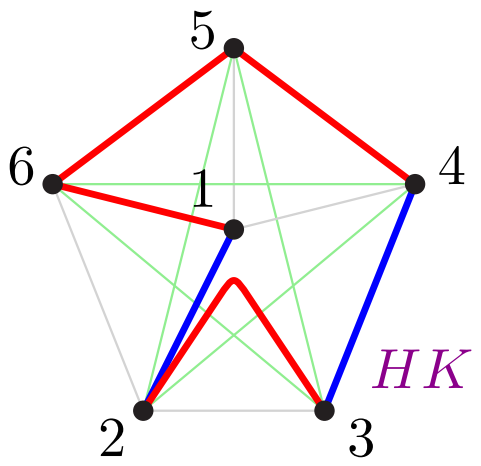
(Laufzeit durch Matching dominiert, $\mathcal{O}(n^3)$)

Approximationsalgorithmen

Minimum Metric TSP

$3/2$-Approximation (Algorithmus) (Christofides-Heuristik)

- Graph $G = (V, E)$ (vollständig, ungerichtet)
- bestimme MST T
- bestimme Knoten U mit ungeradem Grad in T
- finde *minimales perfektes Matching* M auf (U, E)
  (alle Knoten gematcht, Summe der Gewichte der Matchingkanten minimal)
- füge Kanten M zu $T \to T'$
- bestimme Eulerkreis EK auf T'
  (alle Knoten haben geraden Grad)
- wandle EK zu Hamiltonkreis HK



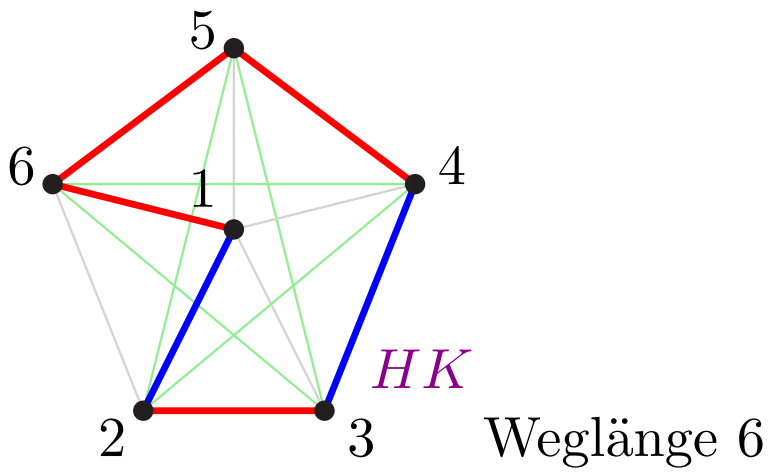
(Laufzeit durch Matching dominiert, $\mathcal{O}(n^3)$)

Approximationsalgorithmen

Minimum Metric TSP

$3/2$-Approximation (Algorithmus) (Christofides-Heuristik)

- Graph $G = (V, E)$ (vollständig, ungerichtet)
- bestimme MST T
- bestimme Knoten U mit ungeradem Grad in T
- finde *minimales perfektes Matching* M auf (U, E)
  (alle Knoten gematcht, Summe der Gewichte der Matchingkanten minimal)
- füge Kanten M zu $T \to T'$
- bestimme Eulerkreis EK auf T'
  (alle Knoten haben geraden Grad)
- wandle EK zu Hamiltonkreis HK



(Laufzeit durch Matching dominiert, $\mathcal{O}(n^3)$)

Approximationsalgorithmen

Minimum Metric TSP

3/2-Approximation (Beweis) (Christofides-Heuristik)

- Abschätzungen: (analog zu vorherigem Beweis)

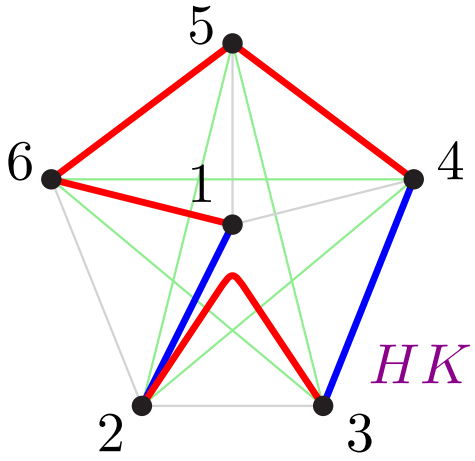
  $\Rightarrow \; w(HK) \leq w(EK) = w(T') = w(T) + w(M) \leq w(OPT) + w(M)$

- Bestimmung von $w(M)$:
  - sei HK' Hamiltonkreis auf U (U: Knoten mit ungeradem Grad in T)
    (erzeugt durch Überspringen aller Knoten $V \setminus U$ in OPT)
  - definiere alternierende perfekte Matchings M_1, M_2 auf HK'
    (existiert, da $|U|$ gerade, $HK' = M_1 \cup M_2$)

  $\Rightarrow \; w(M) \leq w(M_1), w(M) \leq w(M_2)$ (M min. Matching!)

  $\Rightarrow \; 2 \cdot w(M) \leq w(M_1) + w(M_2) = w(HK') \leq w(OPT)$
  (Überspringen gerader Knoten und Dreiecks-Ungleichung)

$\Rightarrow \; w(HK) \leq \frac{3}{2} w(OPT)$



Approximationsalgorithmen

Minimum Metric TSP

3/2-Approximation (Beweis) (Christofides-Heuristik)

- Abschätzungen: (analog zu vorherigem Beweis)

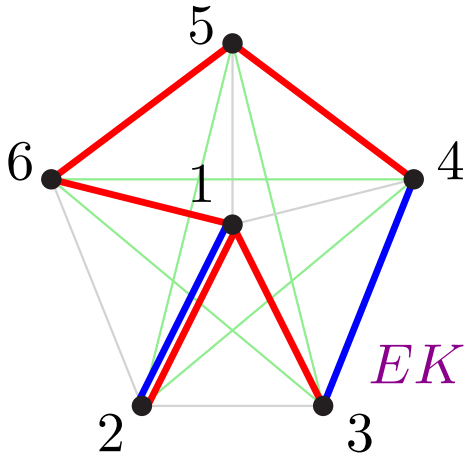
  $\Rightarrow \; w(HK) \leq w(EK) = w(T') = w(T) + w(M) \leq w(OPT) + w(M)$

- Bestimmung von $w(M)$:
  - sei HK' Hamiltonkreis auf U (U: Knoten mit ungeradem Grad in T)
    (erzeugt durch Überspringen aller Knoten $V \setminus U$ in OPT)
  - definiere alternierende perfekte Matchings M_1, M_2 auf HK'
    (existiert, da $|U|$ gerade, $HK' = M_1 \cup M_2$)

  $\Rightarrow \; w(M) \leq w(M_1), w(M) \leq w(M_2)$ (M min. Matching!)

  $\Rightarrow \; 2 \cdot w(M) \leq w(M_1) + w(M_2) = w(HK') \leq w(OPT)$
  (Überspringen gerader Knoten und Dreiecks-Ungleichung)

$\Rightarrow \; w(HK) \leq \frac{3}{2} w(OPT)$



Approximationsalgorithmen

Minimum Metric TSP

$3/2$-Approximation (Beweis) (Christofides-Heuristik)

- Abschätzungen: (analog zu vorherigem Beweis)

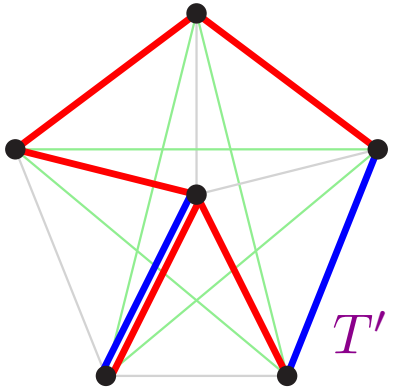
  $\Rightarrow \; w(HK) \leq w(EK) = w(T') = w(T) + w(M) \leq w(OPT) + w(M)$

- Bestimmung von $w(M)$:
  - sei HK' Hamiltonkreis auf U (U: Knoten mit ungeradem Grad in T)
    (erzeugt durch Überspringen aller Knoten $V \setminus U$ in OPT)
  - definiere alternierende perfekte Matchings M_1, M_2 auf HK'
    (existiert, da $|U|$ gerade, $HK' = M_1 \cup M_2$)

  $\Rightarrow \; w(M) \leq w(M_1), w(M) \leq w(M_2)$ (M min. Matching!)

  $\Rightarrow \; 2 \cdot w(M) \leq w(M_1) + w(M_2) = w(HK') \leq w(OPT)$
  (Überspringen gerader Knoten und Dreiecks-Ungleichung)

$\Rightarrow \; w(HK) \leq \frac{3}{2} w(OPT)$



Approximationsalgorithmen

Minimum Metric TSP

3/2-Approximation (Beweis) (Christofides-Heuristik)

- Abschätzungen: (analog zu vorherigem Beweis)

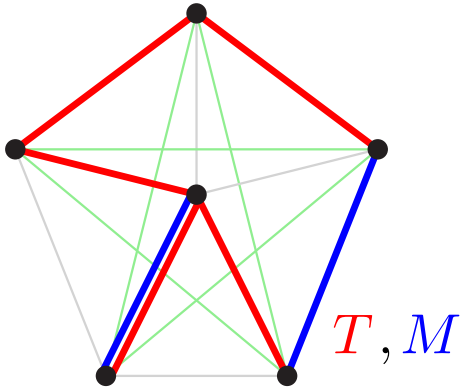
  $\Rightarrow \; w(HK) \leq w(EK) = w(T') = w(T) + w(M) \leq w(OPT) + w(M)$

- Bestimmung von $w(M)$:
  - sei HK' Hamiltonkreis auf U (U: Knoten mit ungeradem Grad in T)
    (erzeugt durch Überspringen aller Knoten $V \setminus U$ in OPT)
  - definiere alternierende perfekte Matchings M_1, M_2 auf HK'
    (existiert, da $|U|$ gerade, $HK' = M_1 \cup M_2$)

  $\Rightarrow \; w(M) \leq w(M_1), w(M) \leq w(M_2)$ (M min. Matching!)

  $\Rightarrow \; 2 \cdot w(M) \leq w(M_1) + w(M_2) = w(HK') \leq w(OPT)$
  (Überspringen gerader Knoten und Dreiecks-Ungleichung)

$\Rightarrow \; w(HK) \leq \frac{3}{2} w(OPT)$



Approximationsalgorithmen

Minimum Metric TSP

3/2-Approximation (Beweis) (Christofides-Heuristik)

- Abschätzungen: (analog zu vorherigem Beweis)

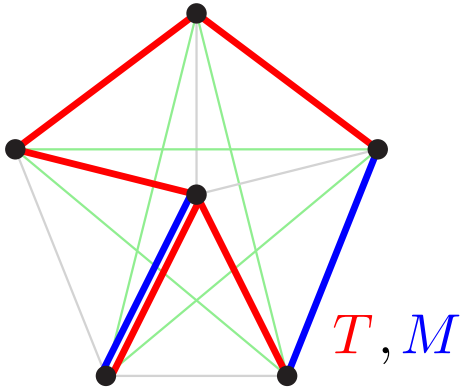
  $\Rightarrow \; w(HK) \leq w(EK) = w(T') = w(T) + w(M) \leq w(OPT) + w(M)$

- Bestimmung von $w(M)$:
  - sei HK' Hamiltonkreis auf U (U: Knoten mit ungeradem Grad in T)
    (erzeugt durch Überspringen aller Knoten $V \setminus U$ in OPT)
  - definiere alternierende perfekte Matchings M_1, M_2 auf HK'
    (existiert, da $|U|$ gerade, $HK' = M_1 \cup M_2$)

  $\Rightarrow \; w(M) \leq w(M_1), w(M) \leq w(M_2)$ (M min. Matching!)

  $\Rightarrow \; 2 \cdot w(M) \leq w(M_1) + w(M_2) = w(HK') \leq w(OPT)$
  (Überspringen gerader Knoten und Dreiecks-Ungleichung)

$\Rightarrow \; w(HK) \leq \frac{3}{2} w(OPT)$



Approximationsalgorithmen

Minimum Metric TSP

$3/2$-Approximation (Beweis) (Christofides-Heuristik)

- Abschätzungen: (analog zu vorherigem Beweis)

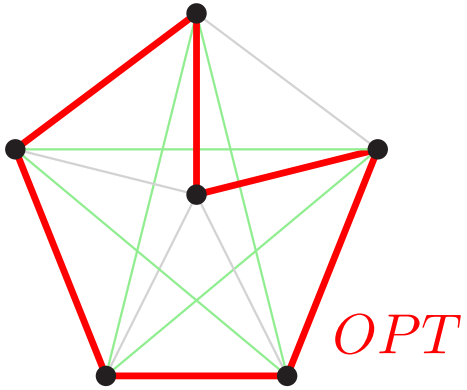
  $\Rightarrow \quad w(HK) \leq w(EK) = w(T') = w(T) + w(M) \leq w(OPT) + w(M)$

- Bestimmung von $w(M)$:
  - sei HK' Hamiltonkreis auf U (U: Knoten mit ungeradem Grad in T)
    (erzeugt durch Überspringen aller Knoten $V \setminus U$ in OPT)
  - definiere alternierende perfekte Matchings M_1, M_2 auf HK'
    (existiert, da $|U|$ gerade, $HK' = M_1 \cup M_2$)

  $\Rightarrow \quad w(M) \leq w(M_1), w(M) \leq w(M_2)$ (M min. Matching!)

  $\Rightarrow \quad 2 \cdot w(M) \leq w(M_1) + w(M_2) = w(HK') \leq w(OPT)$
  (Überspringen gerader Knoten und Dreiecks-Ungleichung)

$\Rightarrow \quad w(HK) \leq \frac{3}{2} w(OPT)$



Approximationsalgorithmen

Minimum Metric TSP

$3/2$-Approximation (Beweis) (Christofides-Heuristik)

- Abschätzungen: (analog zu vorherigem Beweis)

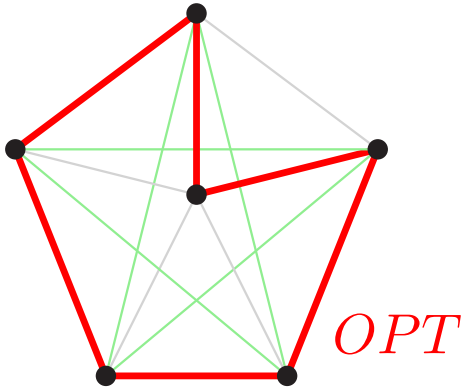
  $\Rightarrow \quad w(HK) \leq w(EK) = w(T') = w(T) + w(M) \leq w(OPT) + w(M)$

- Bestimmung von $w(M)$:
  - sei HK' Hamiltonkreis auf U (U: Knoten mit ungeradem Grad in T)
    (erzeugt durch Überspringen aller Knoten $V \setminus U$ in OPT)
  - definiere alternierende perfekte Matchings M_1, M_2 auf HK'
    (existiert, da $|U|$ gerade, $HK' = M_1 \cup M_2$)

  $\Rightarrow \quad w(M) \leq w(M_1), w(M) \leq w(M_2)$ (M min. Matching!)

  $\Rightarrow \quad 2 \cdot w(M) \leq w(M_1) + w(M_2) = w(HK') \leq w(OPT)$
  (Überspringen gerader Knoten und Dreiecks-Ungleichung)

$\Rightarrow \quad w(HK) \leq \frac{3}{2} w(OPT)$



Approximationsalgorithmen

Minimum Metric TSP

3/2-Approximation (Beweis) (Christofides-Heuristik)

- Abschätzungen: (analog zu vorherigem Beweis)

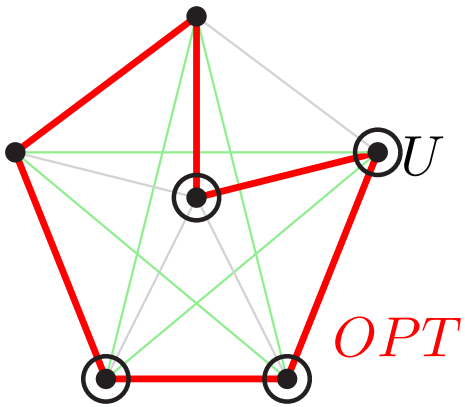
  $\Rightarrow \quad w(HK) \leq w(EK) = w(T') = w(T) + w(M) \leq w(OPT) + w(M)$

- Bestimmung von $w(M)$:
  - sei HK' Hamiltonkreis auf U (U: Knoten mit ungeradem Grad in T)
    (erzeugt durch Überspringen aller Knoten $V \setminus U$ in OPT)
  - definiere alternierende perfekte Matchings M_1, M_2 auf HK'
    (existiert, da $|U|$ gerade, $HK' = M_1 \cup M_2$)

  $\Rightarrow \quad w(M) \leq w(M_1), w(M) \leq w(M_2)$ (M min. Matching!)

  $\Rightarrow \quad 2 \cdot w(M) \leq w(M_1) + w(M_2) = w(HK') \leq w(OPT)$
  (Überspringen gerader Knoten und Dreiecks-Ungleichung)

$\Rightarrow \quad w(HK) \leq \frac{3}{2} w(OPT)$



Approximationsalgorithmen

Minimum Metric TSP

3/2-Approximation (Beweis) (Christofides-Heuristik)

- Abschätzungen: (analog zu vorherigem Beweis)

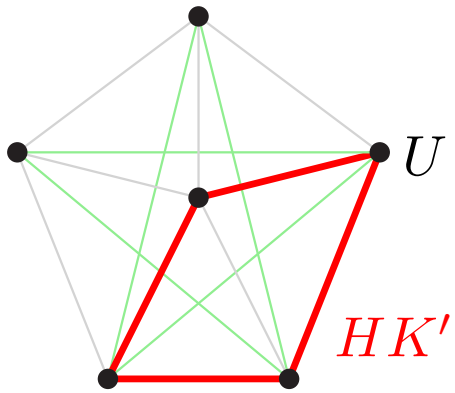
  $\Rightarrow \quad w(HK) \leq w(EK) = w(T') = w(T) + w(M) \leq w(OPT) + w(M)$

- Bestimmung von $w(M)$:
  - sei HK' Hamiltonkreis auf U (U: Knoten mit ungeradem Grad in T)
    (erzeugt durch Überspringen aller Knoten $V \setminus U$ in OPT)
  - definiere alternierende perfekte Matchings M_1, M_2 auf HK'
    (existiert, da $|U|$ gerade, $HK' = M_1 \cup M_2$)

  $\Rightarrow \quad w(M) \leq w(M_1), w(M) \leq w(M_2)$ (M min. Matching!)

  $\Rightarrow \quad 2 \cdot w(M) \leq w(M_1) + w(M_2) = w(HK') \leq w(OPT)$
  (Überspringen gerader Knoten und Dreiecks-Ungleichung)

$\Rightarrow \quad w(HK) \leq \frac{3}{2} w(OPT)$



Approximationsalgorithmen

Minimum Metric TSP

$3/2$-Approximation (Beweis) (Christofides-Heuristik)

- Abschätzungen: (analog zu vorherigem Beweis)

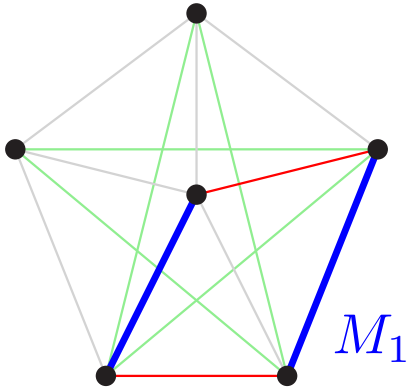
  $\Rightarrow \quad w(HK) \leq w(EK) = w(T') = w(T) + w(M) \leq w(OPT) + w(M)$

- Bestimmung von $w(M)$:
  - sei HK' Hamiltonkreis auf U (U: Knoten mit ungeradem Grad in T)
    (erzeugt durch Überspringen aller Knoten $V \setminus U$ in OPT)
  - definiere alternierende perfekte Matchings M_1, M_2 auf HK'
    (existiert, da $|U|$ gerade, $HK' = M_1 \cup M_2$)

  $\Rightarrow \quad w(M) \leq w(M_1), w(M) \leq w(M_2)$ (M min. Matching!)

  $\Rightarrow \quad 2 \cdot w(M) \leq w(M_1) + w(M_2) = w(HK') \leq w(OPT)$
  (Überspringen gerader Knoten und Dreiecks-Ungleichung)

$\Rightarrow \quad w(HK) \leq \frac{3}{2} w(OPT)$



Approximationsalgorithmen

Minimum Metric TSP

$3/2$-Approximation (Beweis) (Christofides-Heuristik)

- Abschätzungen: (analog zu vorherigem Beweis)

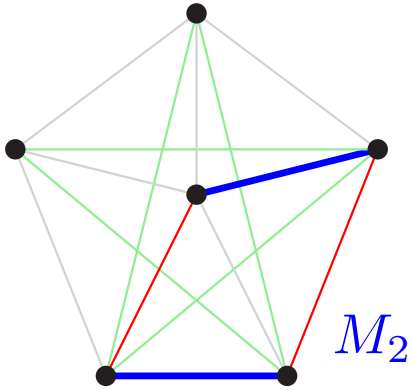
  $\Rightarrow \quad w(HK) \leq w(EK) = w(T') = w(T) + w(M) \leq w(OPT) + w(M)$

- Bestimmung von $w(M)$:
  - sei HK' Hamiltonkreis auf U (U: Knoten mit ungeradem Grad in T)
    (erzeugt durch Überspringen aller Knoten $V \setminus U$ in OPT)
  - definiere alternierende perfekte Matchings M_1, M_2 auf HK'
    (existiert, da $|U|$ gerade, $HK' = M_1 \cup M_2$)

  $\Rightarrow \quad w(M) \leq w(M_1), w(M) \leq w(M_2)$ (M min. Matching!)

  $\Rightarrow \quad 2 \cdot w(M) \leq w(M_1) + w(M_2) = w(HK') \leq w(OPT)$
  (Überspringen gerader Knoten und Dreiecks-Ungleichung)

$\Rightarrow \quad w(HK) \leq \frac{3}{2} w(OPT)$



Approximationsalgorithmen

Minimum Metric TSP

$3/2$-Approximation (Beweis) (Christofides-Heuristik)

- Abschätzungen: (analog zu vorherigem Beweis)

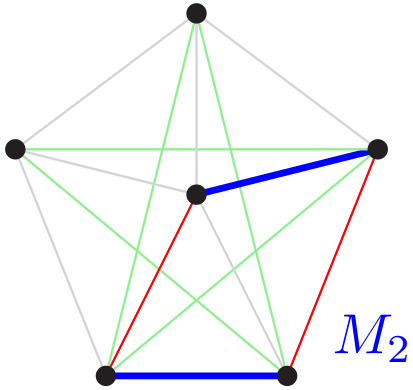
  $\Rightarrow \quad w(HK) \leq w(EK) = w(T') = w(T) + w(M) \leq w(OPT) + w(M)$

- Bestimmung von $w(M)$:
  - sei HK' Hamiltonkreis auf U (U: Knoten mit ungeradem Grad in T)
    (erzeugt durch Überspringen aller Knoten $V \setminus U$ in OPT)
  - definiere alternierende perfekte Matchings M_1, M_2 auf HK'
    (existiert, da $|U|$ gerade, $HK' = M_1 \cup M_2$)

  $\Rightarrow \quad w(M) \leq w(M_1), w(M) \leq w(M_2)$ (M min. Matching!)

  $\Rightarrow \quad 2 \cdot w(M) \leq w(M_1) + w(M_2) = w(HK') \leq w(OPT)$
  (Überspringen gerader Knoten und Dreiecks-Ungleichung)

$\Rightarrow \quad w(HK) \leq \frac{3}{2} w(OPT)$



Approximationsalgorithmen

Minimum Metric TSP

$3/2$-Approximation (Beweis) (Christofides-Heuristik)

- Abschätzungen: (analog zu vorherigem Beweis)

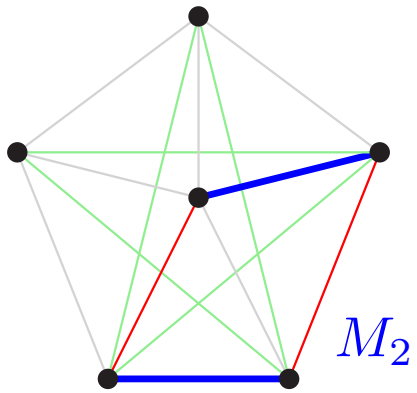
  $\Rightarrow \quad w(HK) \leq w(EK) = w(T') = w(T) + w(M) \leq w(OPT) + w(M)$

- Bestimmung von $w(M)$:
  - sei HK' Hamiltonkreis auf U (U: Knoten mit ungeradem Grad in T)
    (erzeugt durch Überspringen aller Knoten $V \setminus U$ in OPT)
  - definiere alternierende perfekte Matchings M_1, M_2 auf HK'
    (existiert, da $|U|$ gerade, $HK' = M_1 \cup M_2$)

  $\Rightarrow \quad w(M) \leq w(M_1), w(M) \leq w(M_2)$ (M min. Matching!)

  $\Rightarrow \quad 2 \cdot w(M) \leq w(M_1) + w(M_2) = w(HK') \leq w(OPT)$
  (Überspringen gerader Knoten und Dreiecks-Ungleichung)

$\Rightarrow \quad w(HK) \leq \frac{3}{2} w(OPT)$



Approximationsalgorithmen

Minimum Metric TSP

$3/2$-Approximation (Beweis) (Christofides-Heuristik)

- Abschätzungen: (analog zu vorherigem Beweis)

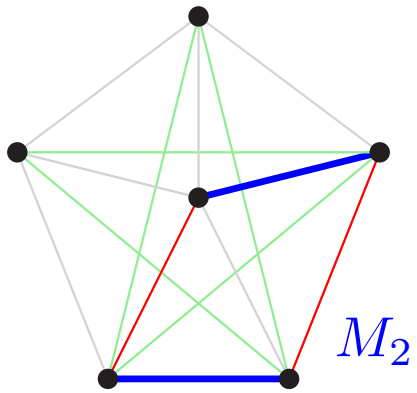
  $\Rightarrow \quad w(HK) \leq w(EK) = w(T') = w(T) + w(M) \leq w(OPT) + w(M)$

- Bestimmung von $w(M)$:
  - sei HK' Hamiltonkreis auf U (U: Knoten mit ungeradem Grad in T)
    (erzeugt durch Überspringen aller Knoten $V \setminus U$ in OPT)
  - definiere alternierende perfekte Matchings M_1, M_2 auf HK'
    (existiert, da $|U|$ gerade, $HK' = M_1 \cup M_2$)

  $\Rightarrow \quad w(M) \leq w(M_1), w(M) \leq w(M_2)$ (M min. Matching!)

  $\Rightarrow \quad 2 \cdot w(M) \leq w(M_1) + w(M_2) = w(HK') \leq w(OPT)$
  (Überspringen gerader Knoten und Dreiecks-Ungleichung)

$\Rightarrow \quad w(HK) \leq \frac{3}{2} w(OPT)$



Approximationsalgorithmen

Minimum Metric TSP

3/2-Approximation (Beweis) (Christofides-Heuristik)

- Abschätzungen: (analog zu vorherigem Beweis)

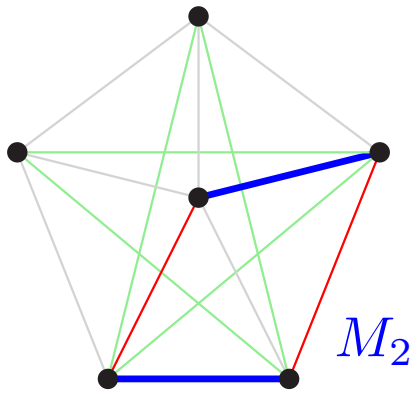
  $\Rightarrow \quad w(HK) \leq w(EK) = w(T') = w(T) + w(M) \leq w(OPT) + w(M)$

- Bestimmung von $w(M)$:
  - sei HK' Hamiltonkreis auf U (U: Knoten mit ungeradem Grad in T)
    (erzeugt durch Überspringen aller Knoten $V \setminus U$ in OPT)
  - definiere alternierende perfekte Matchings M_1, M_2 auf HK'
    (existiert, da $|U|$ gerade, $HK' = M_1 \cup M_2$)

  $\Rightarrow \quad w(M) \leq w(M_1), w(M) \leq w(M_2)$ (M min. Matching!)

  $\Rightarrow \quad 2 \cdot w(M) \leq w(M_1) + w(M_2) = w(HK') \leq w(OPT)$
  (Überspringen gerader Knoten und Dreiecks-Ungleichung)

$\Rightarrow \quad w(HK) \leq \frac{3}{2} w(OPT)$



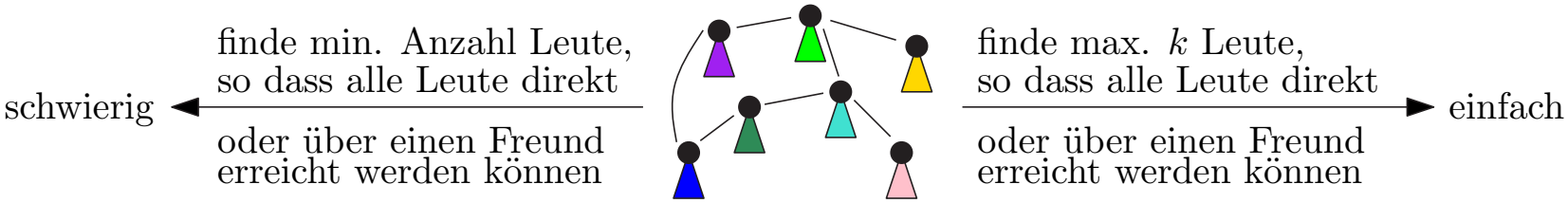
- Existiert immer perfektes Matching M?
  (Zeige, dass $|U|$ immer gerade ist!)

Parametrisierte Algorithmen

***fixed parameter tractable* (FPT)**

Warum Probleme parametrisieren?

- es gibt “schwierige” Probleme (z.B. Minimum Independent Set)
  - $\rightarrow$ allgemeine Instanzen haben zu lange Berechnungszeit
- Kann man Spezialfälle eventuell effizient berechnen?
  - $\rightarrow$ Identifizierung zusätzlicher Parameter k der Problemstellung
  - $\rightarrow$ falls “Komplexität” in diesen Parametern k steckt, effiziente Lösungen für $k = const.$!



Parametrisierte Algorithmen

Definition

- Ein Problem heisst fixed parameter tractable, wenn es eine Laufzeit

$$T(n,k) = \mathcal{O}(f(k) \cdot \mathrm{p}(n))$$

hat, mit $f(\cdot)$ berechenbar, $p(\cdot)$ Polynom.

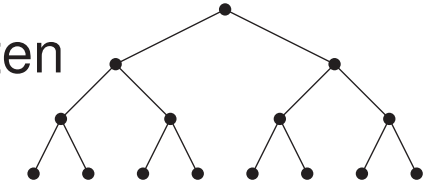
($f(\cdot)$ darf nicht von n abhängen und $p(\cdot)$ nicht von k; häufig Entscheidungsprobleme)

Parametrisierte Algorithmen

Techniken

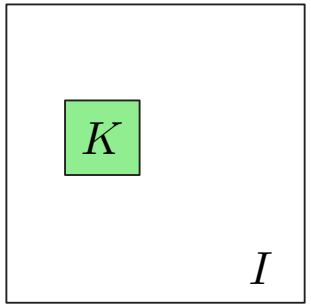
Tiefenbeschränkte Suche

- erschöpfendes Aufzählen und Testen aller Möglichkeiten
  $\rightarrow$ mit geeignetem Suchbaum beschränkter Tiefe
  (k gibt Hinweis, wie man in die Tiefe gehen muss)



Kernbildung

- Probleminstanz auf (schwierigen) Problemkern reduzieren
- Problemkern mit anderer Technik lösen



Weitere Techniken

- Farbkodierung, induktive Kompression, Baumzerlegung, ...

Parametrisierte Algorithmen

Schiebepuzzle

Problemstellung

- gegeben $n \times n$ Schiebepuzzle, $k \in \mathbb{N}$
- entscheide, ob das Puzzle in $\leq k$ Zügen gelöst werden kann
  - das Puzzle ist gelöst, wenn die Teile sortiert sind
  - Loch wird pro Zug eine Position horizontal oder vertikal verschoben

Algorithmus A

- es gibt ≤ 4 Möglichkeiten in jedem Zug, k Züge
  - baue Suchbaum (Höhe k, Verzweigungsgrad ≤ 4)
    $\rightarrow$ Baumgröße $\mathcal{O}(4^k)$
  - teste jeden Knoten auf korrekte Lösung
    $\rightarrow$ Aufwand $\mathcal{O}(n^2) \in \mathcal{O}(poly(n))$

24	8	13	12	20
11	2		17	21
7	15	14	19	5
6	10	3	9	1
4	23	11	18	22

$\Rightarrow$ Gesamtaufwand: $\mathcal{O}(4^k n^2) \Rightarrow$ FPT

$(T(n,k) = 4T(n,k-1) + poly(n))$

Parametrisierte Algorithmen

Schiebepuzzle

Problemstellung

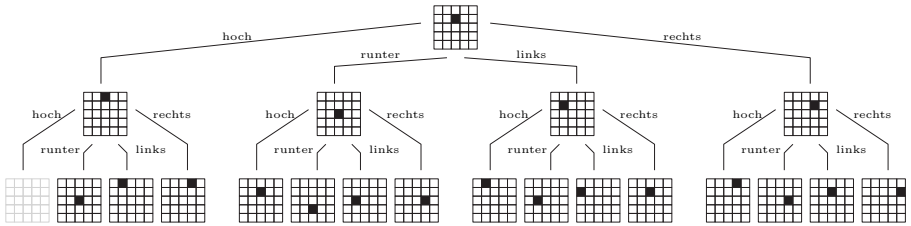
- gegeben $n \times n$ Schiebepuzzle, $k \in \mathbb{N}$
- entscheide, ob das Puzzle in $\leq k$ Zügen gelöst werden kann
  - das Puzzle ist gelöst, wenn die Teile sortiert sind
  - Loch wird pro Zug eine Position horizontal oder vertikal verschoben

Algorithmus A

- es gibt ≤ 4 Möglichkeiten in jedem Zug, k Züge
  - baue Suchbaum (Höhe k, Verzweigungsgrad ≤ 4)
    $\rightarrow$ Baumgröße $\mathcal{O}(4^k)$
  - teste jeden Knoten auf korrekte Lösung
    $\rightarrow$ Aufwand $\mathcal{O}(n^2) \in \mathcal{O}(poly(n))$

$\Rightarrow$ Gesamtaufwand: $\mathcal{O}(4^k n^2) \Rightarrow$ FPT

$(T(n,k) = 4T(n,k-1) + poly(n))$



Ende!

Feierabend!

Parametrisierte Algorithmen

Rekurrenzrelationen – Zusammenfassung

$$T(n,k) = a_1 T(n, k-k_1) + a_2 T(n, k-k_2) + \ldots + a_j T(n, k-k_j) \; \{+f(n)\}$$

$(k_i \leq k_j, i < j)$

- Verzweigungsvektor ($a_i = 1$, ohne $f(n)$)
  - $\langle k_1, k_2, \ldots, k_j \rangle$

  $\Rightarrow$ tabelliert, Basis α aus Tabelle ablesen

  $\Rightarrow$ Lösung $T(n,k) = \mathcal{O}(\alpha^k)$ ($\hat{=}$ Größe des Ausführungsbaums)

- erzeugendes Polynom (a_i beliebig, ohne $f(n)$)
  - Ansatz: $T(n,k) = x^k$

  $\Rightarrow$ $x^k - a_1 x^{k-k_1} - a_2 x^{k-k_2} - \ldots - a_j x^{k-k_j} = 0$ (eingesetzt)

  $\Rightarrow$ bestimme *betragsmäßig* größte Nullstelle α (mit Vielfachheit m)

  $\Rightarrow$ Lösung $T(n,k) = \mathcal{O}(k^{m-1}\alpha^k)$

- inhomogene Rekurrenzrelation ($f(n) \hat{=}$ Arbeit in jedem Knoten)
  - Löse homogene Rekurrenz (ohne $f(n)$) $\rightarrow T'(n,k) = r(k)$

  $\Rightarrow$ Lösung $T(n,k) = \mathcal{O}(r(k) \cdot f(n))$

Parametrisierte Algorithmen

Rekurrenzrelationen – Zusammenfassung

$$T(n,k) = a_1 T(n, k-k_1) + a_2 T(n, k-k_2) + \ldots + a_j T(n, k-k_j) \; \{+f(n)\}$$

$(k_i \leq k_j, i < j)$

- Verzweigungsvektor ($a_i = 1$, ohne $f(n)$)
  - $\langle k_1, k_2, \ldots, k_j \rangle$

  $\Rightarrow$ tabelliert, Basis α aus Tabelle ablesen

  $\Rightarrow$ Lösung $T(n,k) = \mathcal{O}(\alpha^k)$ ($\hat{=}$ Größe des Ausführungsbaums)

- erzeugendes Polynom (a_i beliebig, ohne $f(n)$)
  - Ansatz: $T(n,k) = x^k$

  $\Rightarrow$ $x^k - a_1 x^{k-k_1} - a_2 x^{k-k_2} - \ldots - a_j x^{k-k_j} = 0$ (eingesetzt)

  $\Rightarrow$ bestimme *betragsmäßig* größte Nullstelle α (mit Vielfachheit m)

  $\Rightarrow$ Lösung $T(n,k) = \mathcal{O}(k^{m-1}\alpha^k)$

- inhomogene Rekurrenzrelation ($f(n) \hat{=}$ Arbeit in jedem Knoten)
  - Löse homogene Rekurrenz (ohne $f(n)$) $\rightarrow T'(n,k) = r(k)$

  $\Rightarrow$ Lösung $T(n,k) = \mathcal{O}(r(k) \cdot f(n))$

Parametrisierte Algorithmen

Rekurrenzrelationen – Verzweigungsvektor

$$T(n,k) = a_1 T(n, k-k_1) + a_2 T(n, k-k_2) + \ldots + a_j T(n, k-k_j)$$

$(k_i \leq k_j, i < j)$

Verzweigungsvektor $(a_i = 1)$

- $\langle k_1, k_2, \ldots, k_j \rangle$

$\Rightarrow$ tabelliert, Basis α aus Tabelle ablesen

$\Rightarrow$ Lösung $T(n,k) = \mathcal{O}(\alpha^k)$ ($\hat{=}$ Größe des Ausführungsbaums)

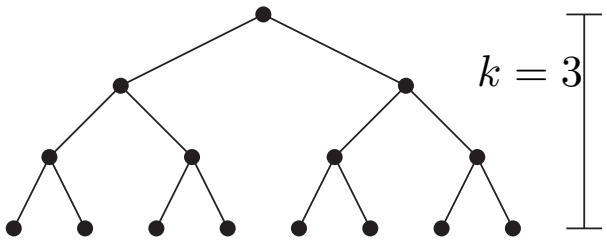
Beispiel

- $T(n,k) = T(n,k-1) + T(n,k-1)$

$\Rightarrow$ Verzweigungsvektor $\langle 1, 1 \rangle$

$\Rightarrow$ nach Tabelle $\alpha = 2$

$\Rightarrow$ $T(n,k) = \mathcal{O}(2^k)$



Verzweigungsgrad $\alpha = 2$

Parametrisierte Algorithmen

Rekurrenzrelationen – erzeugendes Polynom

$$T(n,k) = a_1 T(n, k-k_1) + a_2 T(n, k-k_2) + \ldots + a_j T(n, k-k_j)$$

$(k_i \leq k_j, i < j)$

erzeugendes Polynom (a_i beliebig)

- Ansatz: $T(n,k) = x^k$

$\Rightarrow$ $x^k - a_1 x^{k-k_1} - a_2 x^{k-k_2} - \ldots - a_j x^{k-k_j} = 0$ (eingesetzt)

$\Rightarrow$ bestimme *betragsmäßig* größte Nullstelle α (mit Vielfachheit m)

$\Rightarrow$ Lösung $T(n,k) = \mathcal{O}(k^{m-1}\alpha^k)$

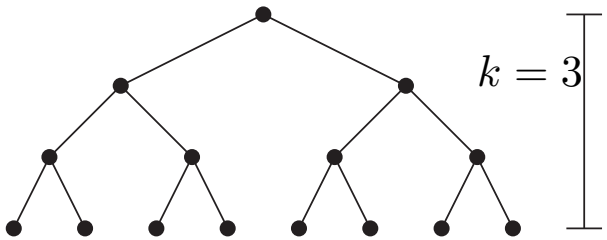
Beispiel

- $T(n,k) = T(n,k-1) + T(n,k-1)$

$\Rightarrow$ $x^k = x^{k-1} + x^{k-1} \Leftrightarrow x^{k-1}(x-2) = 0$

$\Rightarrow$ größte Nullstelle: $x = 2$

$\Rightarrow$ $T(n,k) = \mathcal{O}(2^k)$



Verzweigungsgrad $\alpha = 2$

Parametrisierte Algorithmen

Rekurrenzrelationen – Inhomogenität

$$T(n,k) = a_1 T(n, k-k_1) + a_2 T(n, k-k_2) + \ldots + a_j T(n, k-k_j) + f(n)$$

$(k_i \leq k_j, i < j)$

inhomogene Rekurrenzrelation ($f(n) \hat{=}$ Arbeit in jedem Knoten)

- Löse homogene Rekurrenz (ohne $f(n)$) $\rightarrow T'(n,k) = r(k)$

$\Rightarrow$ Lösung $T(n,k) = \mathcal{O}(r(k) \cdot f(n))$

Beispiel

- $T(n,k) = T(n,k-1) + T(n,k-1) + n$

$\Rightarrow$ homogene Lösung $T'(n,k) = 2^k$

$\Rightarrow$ $T(n,k) = \mathcal{O}(2^k \cdot n)$

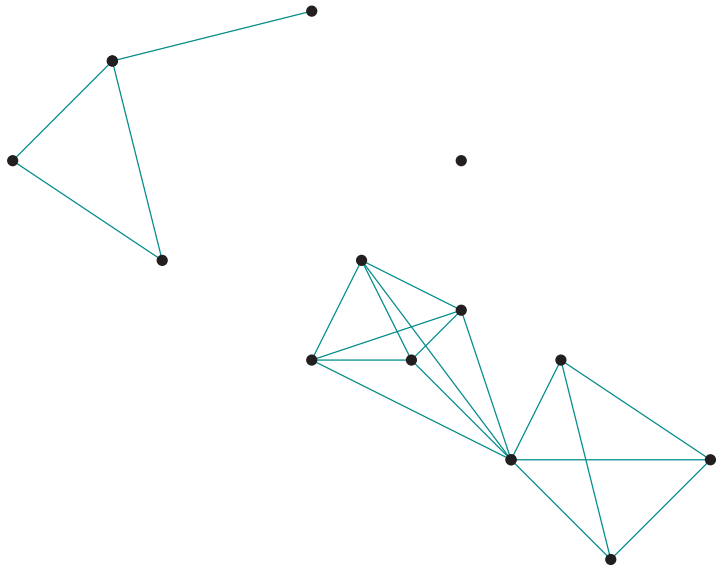
- Test: $\mathcal{O}(2^k \cdot n) = \mathcal{O}(2^{k-1} \cdot n + 2^{k-1} \cdot n + n)$ (Lösung eingesetzt)
  $\Rightarrow \mathcal{O}(2^k) = \mathcal{O}(2^{k-1} + 2^{k-1} + 1)$ (kürzen von n)
  $\Rightarrow \mathcal{O}(2^k) = \mathcal{O}(2^{k-1} + 2^{k-1})$ ($\mathcal{O}$ Kalkül ausnützen)

Parametrisierte Algorithmen

Vertex Cluster Deletion

Problemstellung

- gegeben ungerichteter Graph $G = (V, E)$, $k \in \mathbb{N}$
- existiert $S \subset V$, $|S| \leq k$, so dass $(V \setminus S, E)$ einen Clustergraph ist?
  (Clustergraph $\hat{=}$ Graph bestehend aus isolierten Cliquen)



Beobachtung

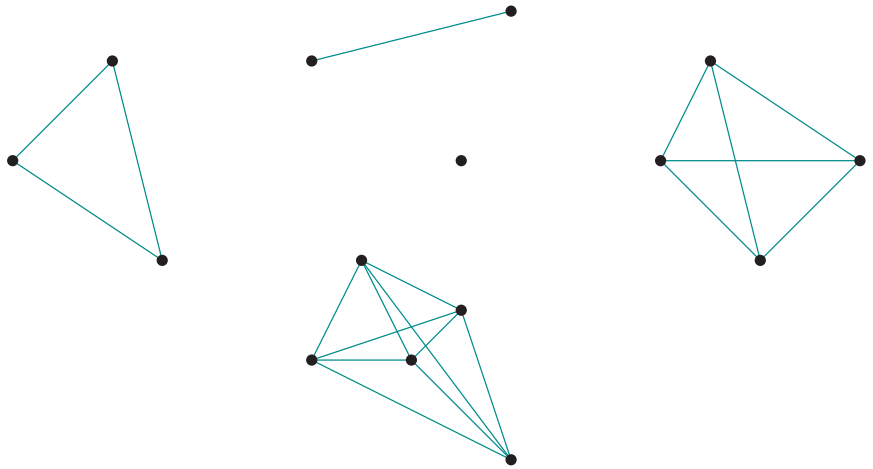
- Graph ist Clustergraph gdw. alle kürzesten Pfade 2 Knoten besitzen

Parametrisierte Algorithmen

Vertex Cluster Deletion

Problemstellung

- gegeben ungerichteter Graph $G = (V, E)$, $k \in \mathbb{N}$
- existiert $S \subset V$, $|S| \leq k$, so dass $(V \setminus S, E)$ einen Clustergraph ist?
  (Clustergraph $\hat{=}$ Graph bestehend aus isolierten Cliquen)



Beobachtung

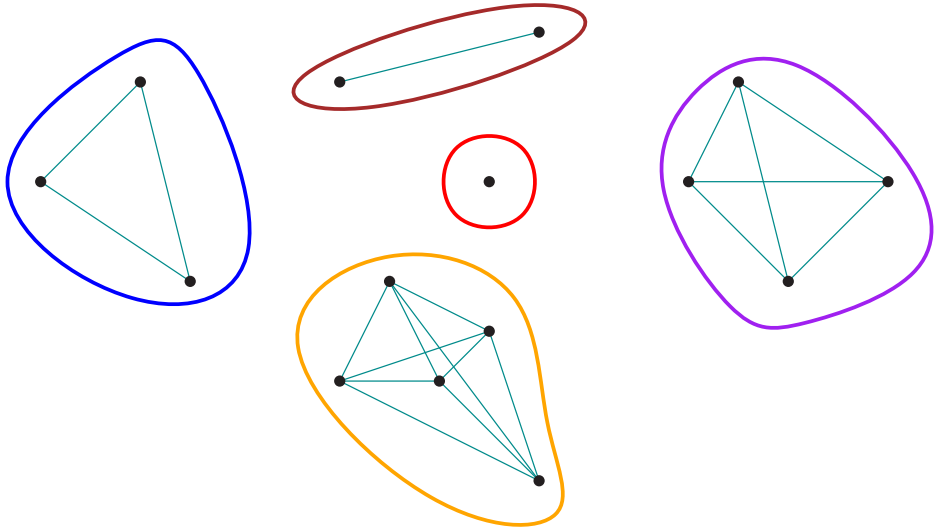
- Graph ist Clustergraph gdw. alle kürzesten Pfade 2 Knoten besitzen

Parametrisierte Algorithmen

Vertex Cluster Deletion

Problemstellung

- gegeben ungerichteter Graph $G = (V, E)$, $k \in \mathbb{N}$
- existiert $S \subset V$, $|S| \leq k$, so dass $(V \setminus S, E)$ einen Clustergraph ist?
  (Clustergraph $\hat{=}$ Graph bestehend aus isolierten Cliquen)



Beobachtung

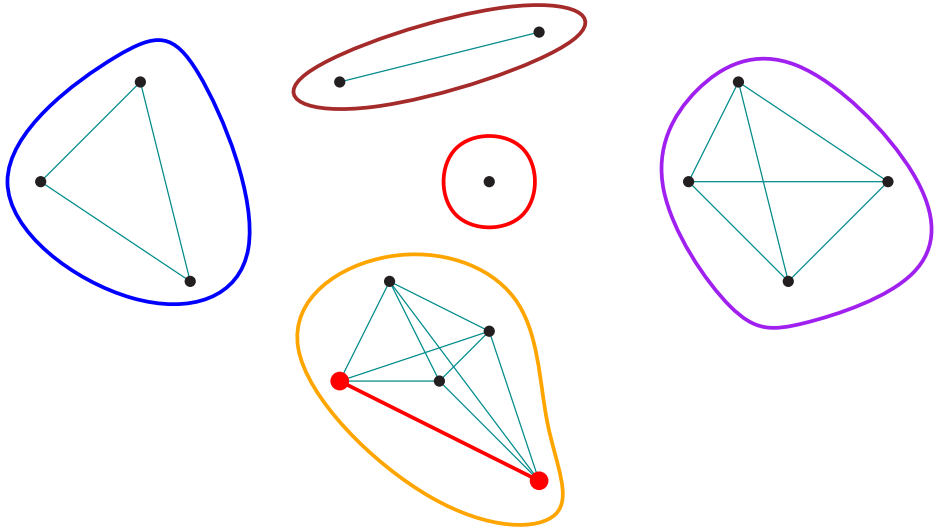
- Graph ist Clustergraph gdw. alle kürzesten Pfade 2 Knoten besitzen

Parametrisierte Algorithmen

Vertex Cluster Deletion

Problemstellung

- gegeben ungerichteter Graph $G = (V, E)$, $k \in \mathbb{N}$
- existiert $S \subset V$, $|S| \leq k$, so dass $(V \setminus S, E)$ einen Clustergraph ist?
  (Clustergraph $\hat{=}$ Graph bestehend aus isolierten Cliquen)



Beobachtung

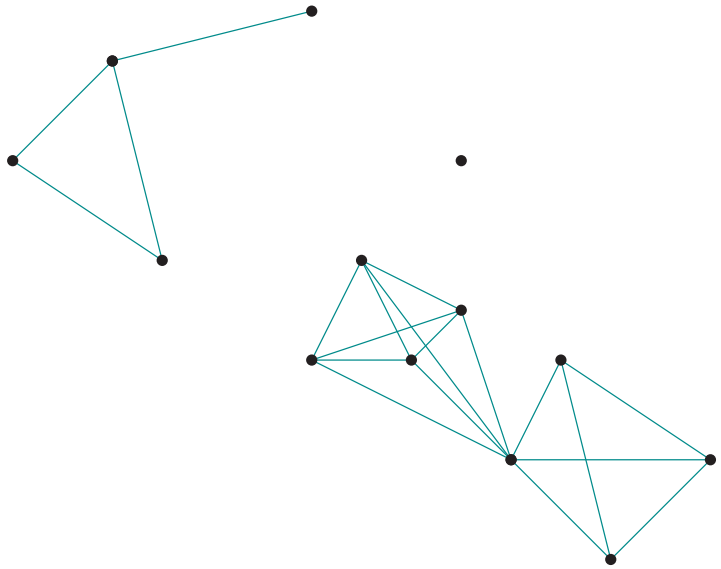
- Graph ist Clustergraph gdw. alle kürzesten Pfade 2 Knoten besitzen

Parametrisierte Algorithmen

Vertex Cluster Deletion

Problemstellung

- gegeben ungerichteter Graph $G = (V, E)$, $k \in \mathbb{N}$
- existiert $S \subset V$, $|S| \leq k$, so dass $(V \setminus S, E)$ einen Clustergraph ist?
  (Clustergraph $\hat{=}$ Graph bestehend aus isolierten Cliquen)



Beobachtung

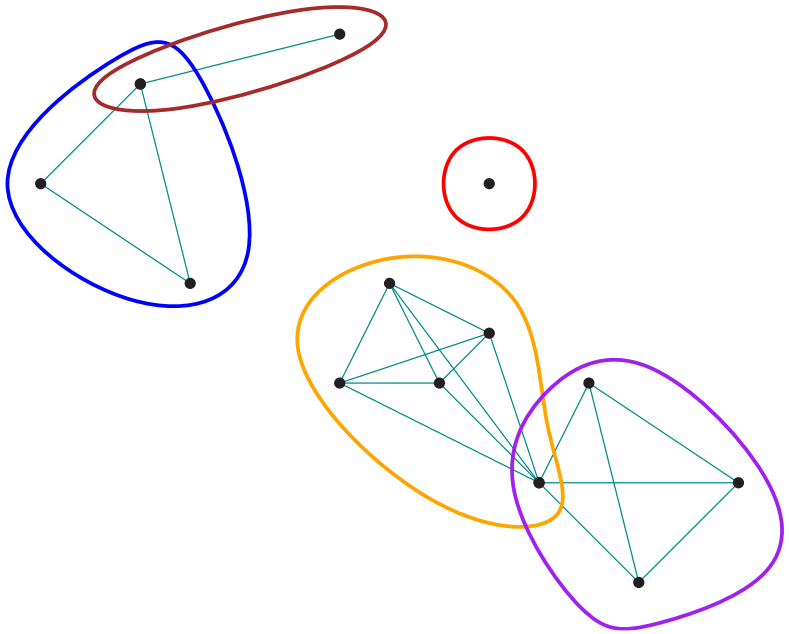
- Graph ist Clustergraph gdw. alle kürzesten Pfade 2 Knoten besitzen

Parametrisierte Algorithmen

Vertex Cluster Deletion

Problemstellung

- gegeben ungerichteter Graph $G = (V, E)$, $k \in \mathbb{N}$
- existiert $S \subset V$, $|S| \leq k$, so dass $(V \setminus S, E)$ einen Clustergraph ist?
  (Clustergraph $\hat{=}$ Graph bestehend aus isolierten Cliquen)



Beobachtung

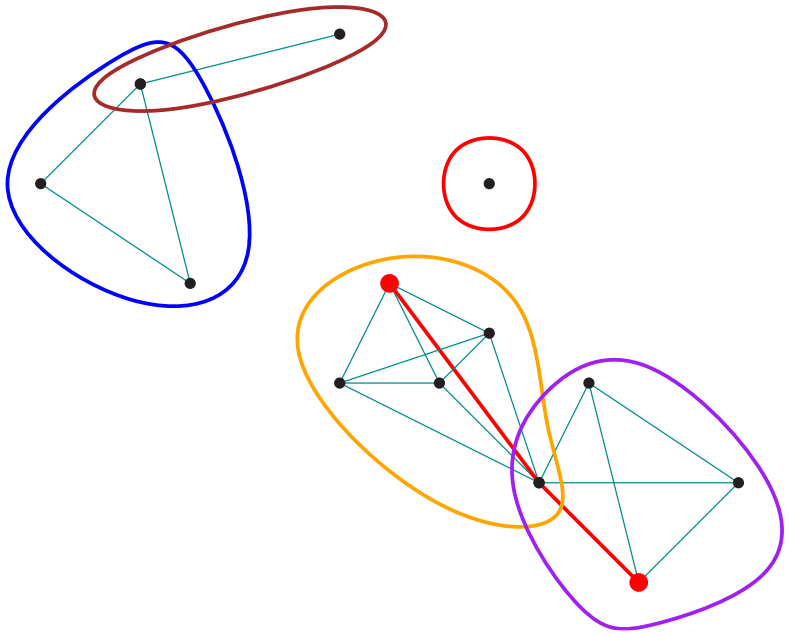
- Graph ist Clustergraph gdw. alle kürzesten Pfade 2 Knoten besitzen

Parametrisierte Algorithmen

Vertex Cluster Deletion

Problemstellung

- gegeben ungerichteter Graph $G = (V, E)$, $k \in \mathbb{N}$
- existiert $S \subset V$, $|S| \leq k$, so dass $(V \setminus S, E)$ einen Clustergraph ist?
  (Clustergraph $\hat{=}$ Graph bestehend aus isolierten Cliquen)



Beobachtung

- Graph ist Clustergraph gdw. alle kürzesten Pfade 2 Knoten besitzen

Parametrisierte Algorithmen

Vertex Cluster Deletion

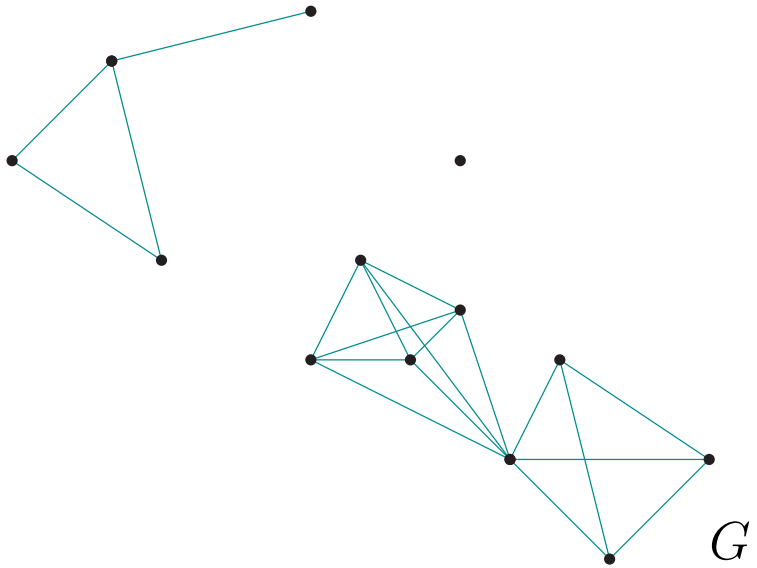
Algorithmus

- Für $k < 0$, gebe NEIN zurück
- Suche in $\mathcal{O}(n^3)$ nach kürzestem Pfad P mit 3 Knoten
  - existiert keiner, gebe JA zurück
  - sonst, verzweige $3\times$ durch Löschen je eines Knotens von P aus G, wiederhole rekursiv mit $k \to k-1$

Laufzeit

- Baumgröße $\mathcal{O}(3^k)$
  (Suchbaum mit Höhe k, Verzweigungsgrad ≤ 3)
- Arbeit pro Knoten $\mathcal{O}(poly(n))$

$\Rightarrow$ Gesamtlaufzeit: $\mathcal{O}(3^k poly(n)) \Rightarrow$ FPT
($T(n,k) = 3T(n,k-1) + \mathcal{O}(n^3)$)



Parametrisierte Algorithmen

Vertex Cluster Deletion

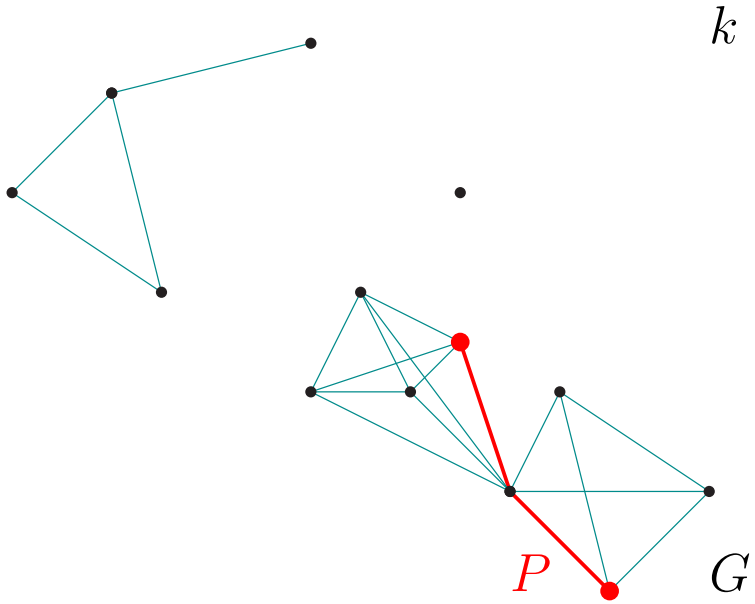
Algorithmus

- Für $k < 0$, gebe NEIN zurück
- Suche in $\mathcal{O}(n^3)$ nach kürzestem Pfad P mit 3 Knoten
  - existiert keiner, gebe JA zurück
  - sonst, verzweige $3\times$ durch Löschen je eines Knotens von P aus G, wiederhole rekursiv mit $k \to k-1$

Laufzeit

- Baumgröße $\mathcal{O}(3^k)$
  (Suchbaum mit Höhe k, Verzweigungsgrad ≤ 3)
- Arbeit pro Knoten $\mathcal{O}(poly(n))$

$\Rightarrow$ Gesamtlaufzeit: $\mathcal{O}(3^k poly(n)) \Rightarrow$ FPT
($T(n,k) = 3T(n,k-1) + \mathcal{O}(n^3)$)



Parametrisierte Algorithmen

Vertex Cluster Deletion

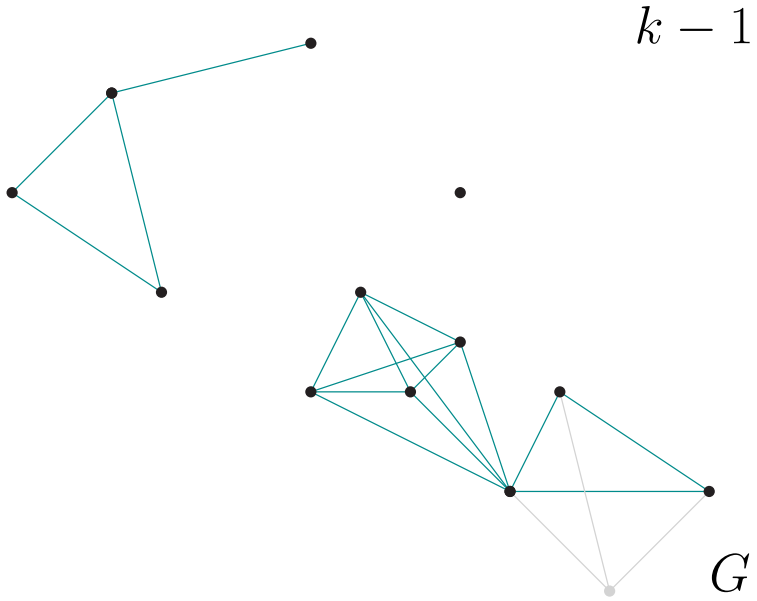
Algorithmus

- Für $k < 0$, gebe NEIN zurück
- Suche in $\mathcal{O}(n^3)$ nach kürzestem Pfad P mit 3 Knoten
  - existiert keiner, gebe JA zurück
  - sonst, verzweige $3\times$ durch Löschen je eines Knotens von P aus G, wiederhole rekursiv mit $k \to k-1$

Laufzeit

- Baumgröße $\mathcal{O}(3^k)$
  (Suchbaum mit Höhe k, Verzweigungsgrad ≤ 3)
- Arbeit pro Knoten $\mathcal{O}(poly(n))$

$\Rightarrow$ Gesamtlaufzeit: $\mathcal{O}(3^k poly(n)) \Rightarrow$ FPT
($T(n,k) = 3T(n,k-1) + \mathcal{O}(n^3)$)



Parametrisierte Algorithmen

Vertex Cluster Deletion

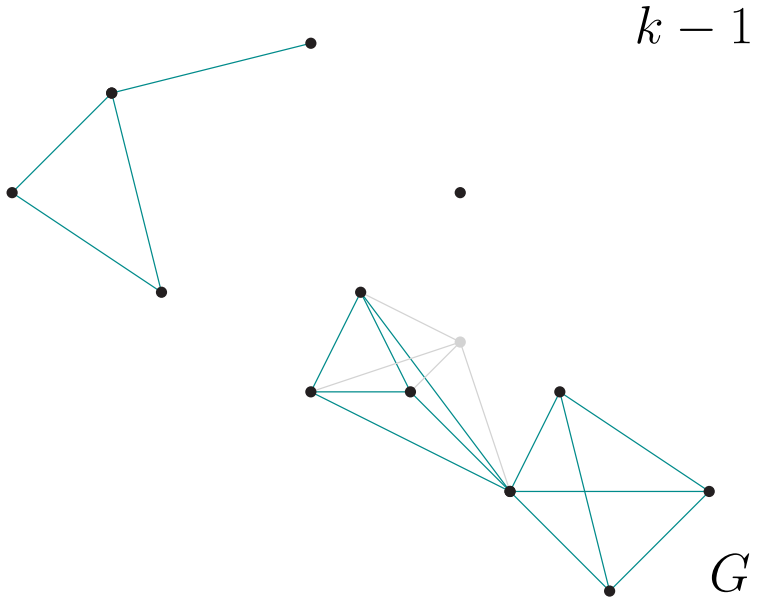
Algorithmus

- Für $k < 0$, gebe NEIN zurück
- Suche in $\mathcal{O}(n^3)$ nach kürzestem Pfad P mit 3 Knoten
  - existiert keiner, gebe JA zurück
  - sonst, verzweige $3\times$ durch Löschen je eines Knotens von P aus G, wiederhole rekursiv mit $k \to k-1$

Laufzeit

- Baumgröße $\mathcal{O}(3^k)$
  (Suchbaum mit Höhe k, Verzweigungsgrad ≤ 3)
- Arbeit pro Knoten $\mathcal{O}(poly(n))$

$\Rightarrow$ Gesamtlaufzeit: $\mathcal{O}(3^k poly(n)) \Rightarrow$ FPT
($T(n,k) = 3T(n,k-1) + \mathcal{O}(n^3)$)



Parametrisierte Algorithmen

Vertex Cluster Deletion

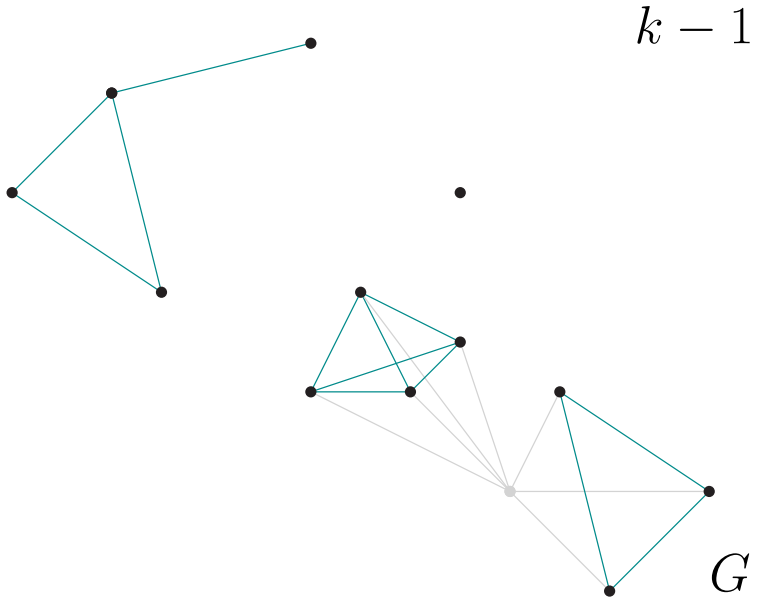
Algorithmus

- Für $k < 0$, gebe NEIN zurück
- Suche in $\mathcal{O}(n^3)$ nach kürzestem Pfad P mit 3 Knoten
  - existiert keiner, gebe JA zurück
  - sonst, verzweige $3\times$ durch Löschen je eines Knotens von P aus G, wiederhole rekursiv mit $k \to k-1$

Laufzeit

- Baumgröße $\mathcal{O}(3^k)$
  (Suchbaum mit Höhe k, Verzweigungsgrad ≤ 3)
- Arbeit pro Knoten $\mathcal{O}(poly(n))$

$\Rightarrow$ Gesamtlaufzeit: $\mathcal{O}(3^k poly(n)) \Rightarrow$ FPT
($T(n,k) = 3T(n,k-1) + \mathcal{O}(n^3)$)



Parametrisierte Algorithmen

Vertex Cluster Deletion

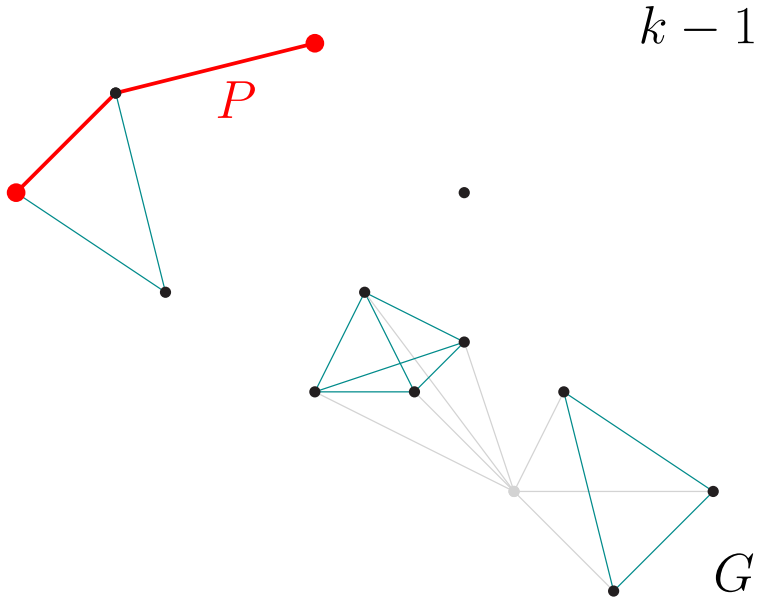
Algorithmus

- Für $k < 0$, gebe NEIN zurück
- Suche in $\mathcal{O}(n^3)$ nach kürzestem Pfad P mit 3 Knoten
  - existiert keiner, gebe JA zurück
  - sonst, verzweige $3\times$ durch Löschen je eines Knotens von P aus G, wiederhole rekursiv mit $k \to k-1$

Laufzeit

- Baumgröße $\mathcal{O}(3^k)$
  (Suchbaum mit Höhe k, Verzweigungsgrad ≤ 3)
- Arbeit pro Knoten $\mathcal{O}(poly(n))$

$\Rightarrow$ Gesamtlaufzeit: $\mathcal{O}(3^k poly(n)) \Rightarrow$ FPT
($T(n,k) = 3T(n,k-1) + \mathcal{O}(n^3)$)



Parametrisierte Algorithmen

Vertex Cluster Deletion

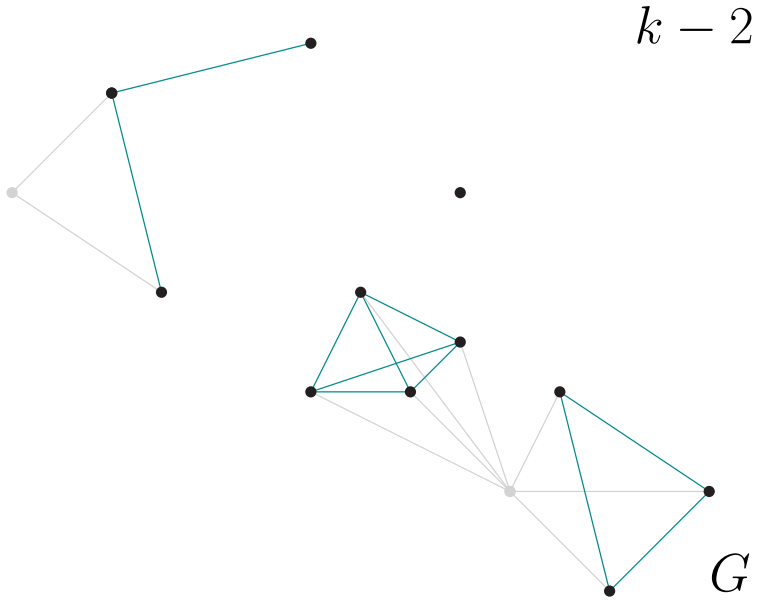
Algorithmus

- Für $k < 0$, gebe NEIN zurück
- Suche in $\mathcal{O}(n^3)$ nach kürzestem Pfad P mit 3 Knoten
  - existiert keiner, gebe JA zurück
  - sonst, verzweige $3\times$ durch Löschen je eines Knotens von P aus G, wiederhole rekursiv mit $k \to k-1$

Laufzeit

- Baumgröße $\mathcal{O}(3^k)$
  (Suchbaum mit Höhe k, Verzweigungsgrad ≤ 3)
- Arbeit pro Knoten $\mathcal{O}(poly(n))$

$\Rightarrow$ Gesamtlaufzeit: $\mathcal{O}(3^k poly(n)) \Rightarrow$ FPT
($T(n,k) = 3T(n,k-1) + \mathcal{O}(n^3)$)



Parametrisierte Algorithmen

Vertex Cluster Deletion

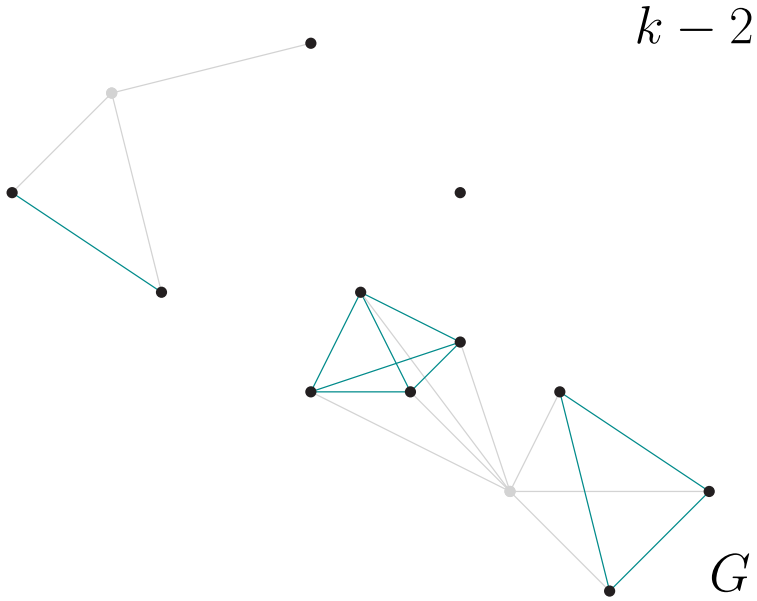
Algorithmus

- Für $k < 0$, gebe NEIN zurück
- Suche in $\mathcal{O}(n^3)$ nach kürzestem Pfad P mit 3 Knoten
  - existiert keiner, gebe JA zurück
  - sonst, verzweige $3\times$ durch Löschen je eines Knotens von P aus G, wiederhole rekursiv mit $k \to k-1$

Laufzeit

- Baumgröße $\mathcal{O}(3^k)$
  (Suchbaum mit Höhe k, Verzweigungsgrad ≤ 3)
- Arbeit pro Knoten $\mathcal{O}(poly(n))$

$\Rightarrow$ Gesamtlaufzeit: $\mathcal{O}(3^k poly(n)) \Rightarrow$ FPT
($T(n,k) = 3T(n,k-1) + \mathcal{O}(n^3)$)



Parametrisierte Algorithmen

Vertex Cluster Deletion

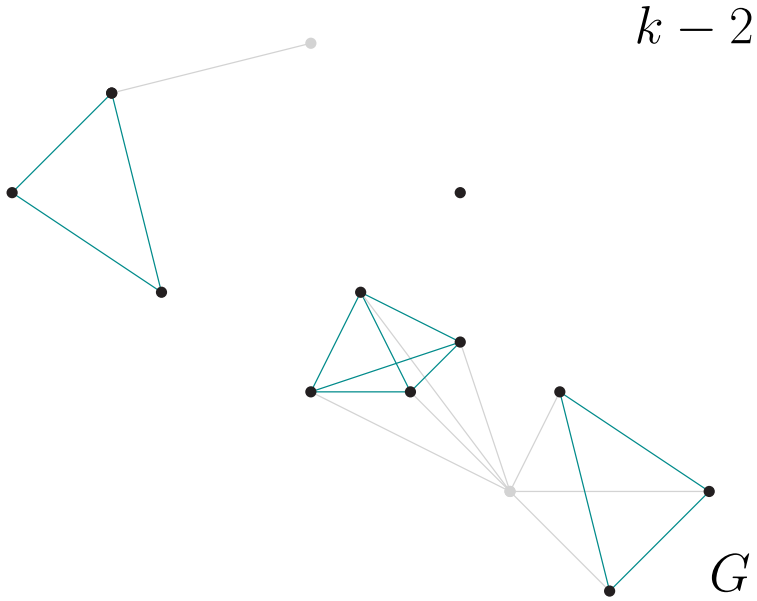
Algorithmus

- Für $k < 0$, gebe NEIN zurück
- Suche in $\mathcal{O}(n^3)$ nach kürzestem Pfad P mit 3 Knoten
  - existiert keiner, gebe JA zurück
  - sonst, verzweige $3\times$ durch Löschen je eines Knotens von P aus G, wiederhole rekursiv mit $k \to k-1$

Laufzeit

- Baumgröße $\mathcal{O}(3^k)$
  (Suchbaum mit Höhe k, Verzweigungsgrad ≤ 3)
- Arbeit pro Knoten $\mathcal{O}(poly(n))$

$\Rightarrow$ Gesamtlaufzeit: $\mathcal{O}(3^k poly(n)) \Rightarrow$ FPT
($T(n,k) = 3T(n,k-1) + \mathcal{O}(n^3)$)



Parametrisierte Algorithmen

Vertex Cluster Deletion

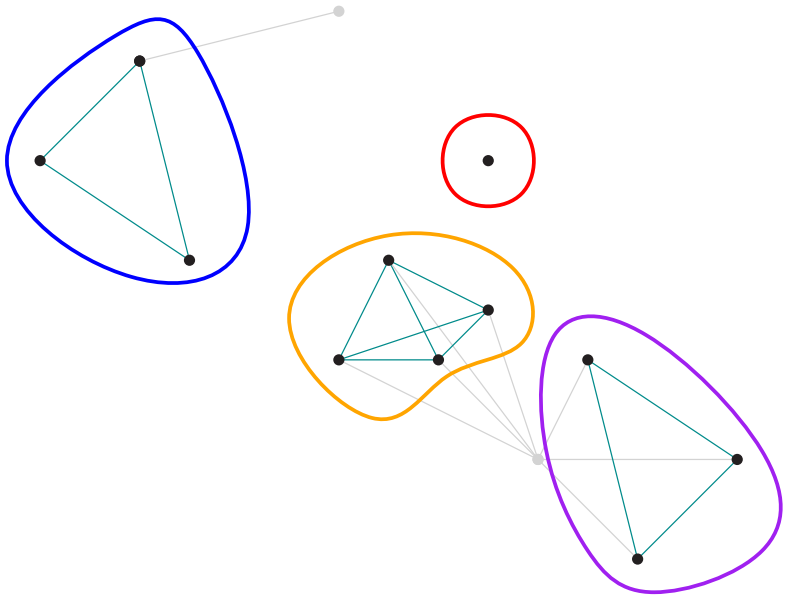
Algorithmus

- Für $k < 0$, gebe NEIN zurück
- Suche in $\mathcal{O}(n^3)$ nach kürzestem Pfad P mit 3 Knoten
  - existiert keiner, gebe JA zurück
  - sonst, verzweige $3\times$ durch Löschen je eines Knotens von P aus G, wiederhole rekursiv mit $k \to k-1$

Laufzeit

- Baumgröße $\mathcal{O}(3^k)$
  (Suchbaum mit Höhe k, Verzweigungsgrad ≤ 3)
- Arbeit pro Knoten $\mathcal{O}(poly(n))$

$\Rightarrow$ Gesamtlaufzeit: $\mathcal{O}(3^k poly(n)) \Rightarrow$ FPT
($T(n,k) = 3T(n,k-1) + \mathcal{O}(n^3)$)



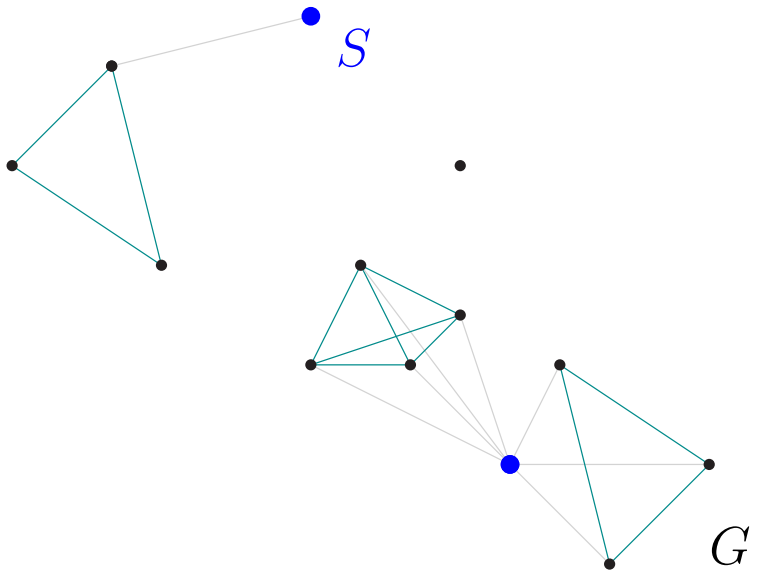
Parametrisierte Algorithmen

Vertex Cluster Deletion

Algorithmus

- Für $k < 0$, gebe NEIN zurück
- Suche in $\mathcal{O}(n^3)$ nach kürzestem Pfad P mit 3 Knoten
  - existiert keiner, gebe JA zurück
  - sonst, verzweige $3\times$ durch Löschen je eines Knotens von P aus G, wiederhole rekursiv mit $k \to k-1$

Laufzeit

- Baumgröße $\mathcal{O}(3^k)$
  (Suchbaum mit Höhe k, Verzweigungsgrad ≤ 3)
- Arbeit pro Knoten $\mathcal{O}(poly(n))$

$\Rightarrow$ Gesamtlaufzeit: $\mathcal{O}(3^k poly(n)) \Rightarrow$ FPT
($T(n,k) = 3T(n,k-1) + \mathcal{O}(n^3)$)