

Übung 8 – Algorithmen II

Hans-Peter Lehmann, Daniel Seemaier – {*hans-peter.lehmann, daniel.seemaier*}@kit.edu
http://algo2.iti.kit.edu/AlgorithmenII_WS21.php

Institut für Theoretische Informatik - Algorithmik II

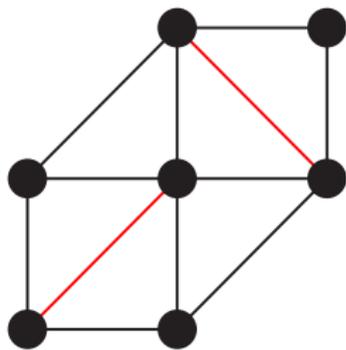
```
    result = current_weight;
    return true;
}

for( EdgeID eid = graph.edgeBegin( current ); eid != graph.edgeEnd( current ); ++eid ){
    const Edge & edge = graph.getEdge( eid );
    COUNTING( statistic_data.inc( DijkstraStatisticData::TOUCHED_EDGES ); )
    if( edge.forward ){
        COUNTING( statistic_data.inc( DijkstraStatisticData::RELAXED_EDGES ); )
        weight new_weight = edge.weight + current_weight;
        GUARANTEE( new_weight >= current_weight, std::runtime_error, "Weight overflow detected." );
        if( !priority_queue.isReached( edge.target ) ){
            COUNTING( statistic_data.inc( DijkstraStatisticData::SUCCESSFULLY_RELAXED_EDGES ); )
            COUNTING( statistic_data.inc( DijkstraStatisticData::REACHED_NODES ); )
            priority_queue.push( edge.target, new_weight );
        } else {
            if( priority_queue.getCurrentKey( edge.target ) > new_weight ){
                COUNTING( statistic_data.inc( DijkstraStatisticData::SUCCESSFULLY_RELAXED_NODES ); )
                priority_queue.decreaseKey( edge.target, new_weight );
            }
        }
    }
}
```

- Wiederholung: Matchings
- Approximationsalgorithmen
- Parameterisierte Algorithmen

Matching

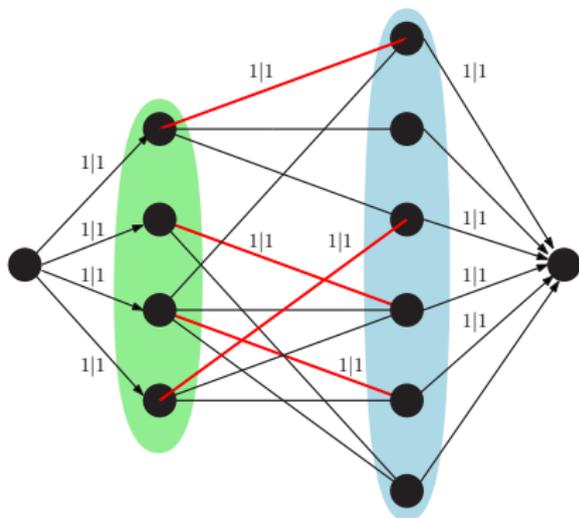
- Teilmenge von Kanten eines Graphen
- Kein Knoten inzident zu mehr als einer (gematchten) Kante
- **Kardinalität**: Zahl der Kanten im Matching
- **Maximal**: keine Kante hinzunehmbar
- **Maximum**: kein Matching hat größere Kardinalität
- **Perfekt**: jeder Knoten ist Endpunkt einer gematchten Kante
- Maximum Matching im allgemeinen über alternierende Pfade berechnet



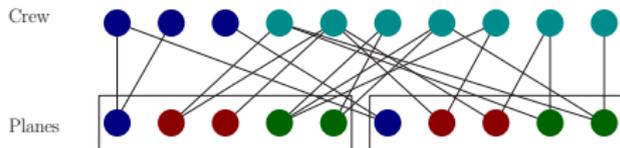
**maximal aber nicht
maximum**

Bipartite - Matching

- Bipartiter Graph: zwei Gruppen von Knoten, Kanten nicht innerhalb einer Gruppe
- Modelliere als Flussproblem
- Fluss zwischen den Teilmengen entspricht gematchten Kanten



- Anwendungen in sämtlichen Zuweisungsproblemen
- ... Jobs auf Maschinen
- ... Crewscheduling

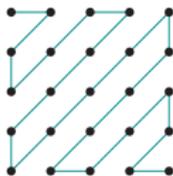


Warum Lösungen abschätzen?

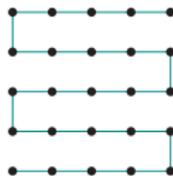
- es gibt “schwierige” Probleme z.B. TSP mit exponentieller Laufzeit
 - exakte Berechnung nicht möglich zu unseren Lebzeiten
- vernünftige Näherungen **effizient** berechnen
 - exakte/optimale Ergebnisse nicht immer wichtig
 - “gute” Lösungen genügen oft
 - aber Abstand zur korrekten Lösung wissenswert



Weglänge “lang”



Weglänge $8 + 16\sqrt{2}$



Weglänge 24

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

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

f.a. Probleminstanzen I , OPT optimale Lösung, w Bewertungsfunktion
für Minimierungsprobleme $\Rightarrow \rho > 1$, für Maximierungsprobleme $\Rightarrow \rho < 1$

Beispiel:

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

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

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

Approximationsalgorithmen

Klassen

Approximationsprobleme klassifizierbar durch

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

Klassen an Approximationsproblemen

APX *approximable*

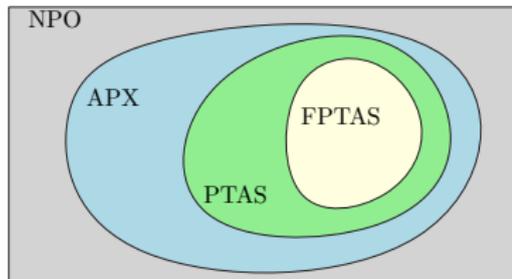
$\rho = \text{const.}$, T polynomiell in n

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:

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

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

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

Hamiltonkreis

- Besucht alle Knoten genau 1x
- Ohne Kantengewichte
- Trivial auf vollständigem Graph, sonst NP-hart

Minimum Traveling Salesman Problem

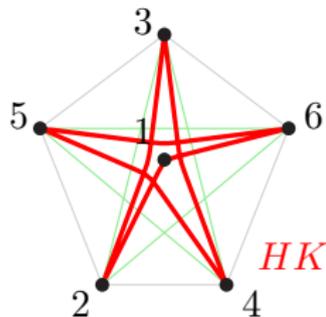
- Kürzester Hamiltonkreis
- Selbst auf vollständigem Graph NP-hart

Problemstellung

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

Wiederholung 2-Approximation (Algorithmus)

- Graph $G = (V, E)$ vollständig, ungerichtet
- bestimme MST T $w(T) \leq w(OPT)$
- verdopple Kanten von $T \rightarrow T'$ $w(T') \leq 2w(OPT)$
- bestimme Eulerkreis EK auf T' $w(EK) = w(T')$
- wandle EK zu Hamiltonkreis HK
 $w(HK) \leq w(EK) \leq 2w(OPT)$

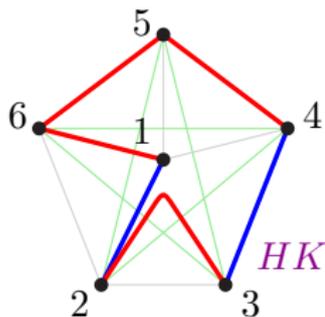


Approximationsalgorithmen

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 $T \rightarrow T'$
 $w(T') = w(T) + w(M) \leq w(OPT) + w(M)$
- bestimme Eulerkreis EK auf T'
alle Knoten haben geraden Grad
- wandle EK zu Hamiltonkreis HK
 $w(HK) \leq w(EK) = w(T') \leq w(OPT) + w(M)$



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, $HK' = M_1 \cup M_2$

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

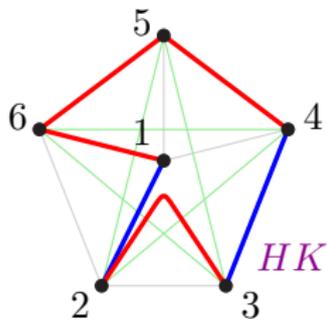
$$\Rightarrow 2 \cdot w(M) \leq w(M_1) + w(M_2) = w(HK') \leq w(OPT)$$

Überspringen gerader Knoten und Dreiecks-Ungleichung

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

- Existiert immer perfektes Matching M ?

Zeige, dass $|U|$ immer gerade ist!



Wissenswert nicht erschöpfend!

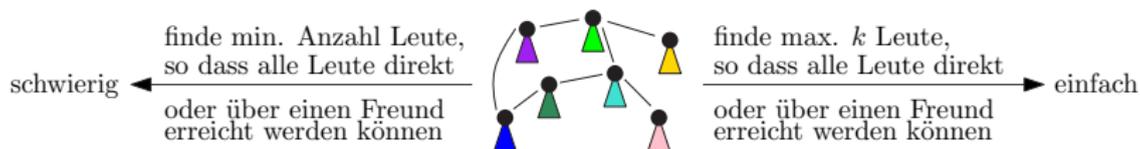
- Approximationsfaktor ρ
- APX, PTAS, FPTAS, pseudopolynomiell
- es gibt nicht gut approximierbare Probleme z.B. minimum TSP

Typische Fragestellungen

- Zu welcher Klasse gehört Algorithmus X mit $T(n, \varepsilon)$, $\rho(n, \varepsilon)$?
- Zeigen/Widerlegen Sie Approximationsfaktor ρ für Algorithmus X .
- (Bestimmen Sie einen Algorithmus mit Approximationsfaktor ρ)

Warum Probleme parametrisieren?

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



- Ein Problem heißt **fixed parameter tractable**, wenn es eine Laufzeit

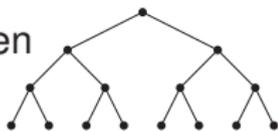
$$T(n, k) = \mathcal{O}(f(k) \cdot p(n))$$

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

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

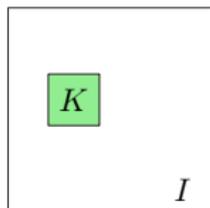
Tiefenbeschränkte Suche

- erschöpfendes Aufzählen und Testen aller Möglichkeiten
→ mit geeignetem Suchbaum beschränkter Tiefe
k gibt Hinweis, wie weit man in die Tiefe gehen muss



Kernbildung

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



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

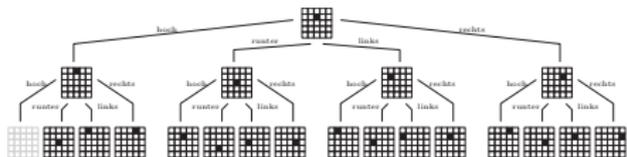
Algorithmus A

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

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

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

$$T(n, k) = 4T(n, k-1) + \text{poly}(n)$$



Ende!



Feierabend!