Approximationsalgorithmen

Lyubomira Dimitrova

06.12.2019

Universität Heidelberg
Fakultät für Mathematik und Informatik

Proseminar Theoretische Informatik

Wolfgang Merkle

WS 19/20

Übersicht

Entscheidungsprobleme

Definition

Sei $\mathcal{P}$ ein Entscheidungsproblem.

- Die Menge aller Instanzen $\boldsymbol{I}_{\mathcal{P}}$ von $\mathcal{P}$ ist unterteilt in:
  - $\boldsymbol{Y}_{\mathcal{P}}$ (positiven Instanzen, "YES")
  - $\boldsymbol{N}_{\mathcal{P}}$ (negativen Instanzen, "NO")
- Für jede Instanz $x \in \boldsymbol{I}_{\mathcal{P}}$ wird gefragt, ob $x \in \boldsymbol{Y}_{\mathcal{P}}$.

Beispiel

Satisfiability (SAT):

Instanz Boolesche Formel $\mathcal{F}(V)$ in KNF, V ist eine Menge boolesche Variablen.

Frage Ist $\mathcal{F}(V)$ erfüllbar, d.h. existiert eine Belegung $f : V \rightarrow \{1, 0\}$, für die $\mathcal{F}(V) = 1$?

Komplexitätsklassen für EP i

P

Die Klasse aller EP, die in Zeit, proportional zu einem Polynom der Eingabelänge ($\rightarrow$ "Polynomialzeit"), lösbar sind.

- Eingabelänge $|x|$
  - jede Probleminstanz wird durch einen endlichen Alphabet repräsentiert, z.B. $\{0, 1\}$
  - $=$ die Länge der Repräsentation, z.B. der Binärdarstellung
- Zu PSPACE gehören alle EP, die auf polynomiellem *Platz* lösbar sind.

Komplexitätsklassen für EP ii

NP

Die Klasse aller EP, die in Polynomialzeit **von einem nichtdeterministischen Algorithmus** lösbar sind.

NP (alternativ)

Die Klasse aller EP, deren **konstruktive Lösung** in Polynomialzeit überprüfbar ist.

Komplexitätsklassen für EP ii

NP

Die Klasse aller EP, die in Polynomialzeit **von einem nichtdeterministischen Algorithmus** lösbar sind.

NP (alternativ)

Die Klasse aller EP, deren **konstruktive Lösung** in Polynomialzeit überprüfbar ist.

"**n**ichtdeterministisch **p**olynomielle Zeit"

Nichtdeterministische Algorithmen i

Ein nichtdetermistischer Algorithmus $\mathcal{A}$ löst das EP $\mathcal{P}$ wenn:

1. für jeden Input $x \in \boldsymbol{I}_{\mathcal{P}}$, $\mathcal{A}$ terminiert.
2. $x \in \boldsymbol{Y}_{\mathcal{P}}$ gdw. mindestens eine Sequenz von Rateversuchen existiert, für die $\mathcal{A}$ 'YES' liefert.

Komplexität von ND Algorithmen

$\mathcal{A}$ löst $\mathcal{P}$ *in Zeit* $t(n)$, wenn:

1. für jeden Input $x \in \boldsymbol{I}_{\mathcal{P}}$ mit Größe $|x| = n$, $\mathcal{A}$ terminiert,
2. $x \in \boldsymbol{Y}_{\mathcal{P}}$ gdw. mindestens eine Sequenz von Rateversuchen existiert, für die $\mathcal{A}$ *in Zeit* $\leq t(n)$ 'YES' liefert.

Nichtdeterministische Algorithmen ii

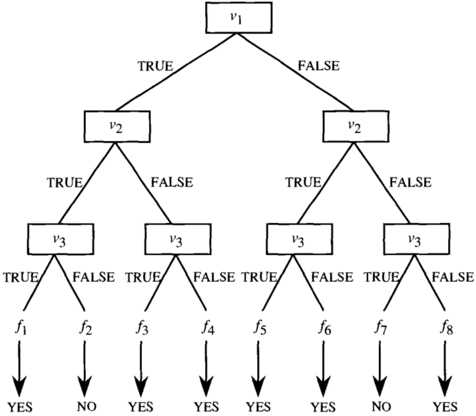


Abbildung 1: ND-Algorithmus, der entscheidet, ob $(v_1 \wedge v_2 \wedge \overline{v_3}) \vee (\overline{v_1} \wedge \overline{v_2} \wedge v_3)$ erfüllbar ist. (Ausiello et al., 2003, Figure 1.2)

Optimierungsprobleme

Definition

Ein Optimierungsproblem $\mathcal{P}$ ist durch den 4-Tupel $(\boldsymbol{I_\mathcal{P}}, \boldsymbol{LSG_\mathcal{P}}, \boldsymbol{m_\mathcal{P}}, \boldsymbol{Ziel_\mathcal{P}})$ charakterisiert:

- $\boldsymbol{I_\mathcal{P}}$: die Menge der Probleminstanzen
- $\boldsymbol{LSG_\mathcal{P}}(x)$: die Menge zulässiger Lösungen von $x \in \boldsymbol{I_\mathcal{P}}$
- $\boldsymbol{m_\mathcal{P}}(x, y)$: für $x \in \boldsymbol{I_\mathcal{P}}$, $y \in \boldsymbol{LSG_\mathcal{P}}(x)$, der Maß der Lösung y
- $\boldsymbol{Ziel_\mathcal{P}} \in \{\textsc{Min}, \textsc{Max}\}$: ob $\mathcal{P}$ ein Minimierungs- oder Maximierungsproblem ist.

Beispiel

Minimum Vertex Cover:

$\boldsymbol{I} = \{G = (V, E) \mid \text{G ist ein Graph}\}$

$\boldsymbol{LSG}(G) = \{U \subseteq V \mid \forall(v_i, v_j) \in E: \ v_i \in U \ \vee \ v_j \in U\}$

$\boldsymbol{m}(G, U) = |U|$

$\boldsymbol{Ziel} = \textsc{Min}$

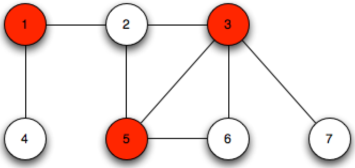


Abbildung 2: Graph und MVC, Quelle

Problemstellungen für OP

Konstruktionsproblem $\mathcal{P}_\mathcal{C}$:

- Gegeben: Instanz $x \in \boldsymbol{I}$
- Gesucht: eine optimale Lösung $y^* \in \boldsymbol{LSG}^*(x)$
- Gesucht: der Maß $\boldsymbol{m}(x, y^*)$

Evaluationsproblem $\mathcal{P}_\mathcal{E}$:

- Gegeben: Instanz $x \in \boldsymbol{I}$
- Gesucht: der Maß $\boldsymbol{m}(x, y^*)$

Entscheidungsproblem $\mathcal{P}_\mathcal{D}$:

- Gegeben: Instanz $x \in \boldsymbol{I}$ und $k \in \mathbb{N}$
- Gesucht: ob $\boldsymbol{m}(x, y^*) \leq k$, (***Ziel*** $=$ MIN)

Entscheidungsproblem für Vertex Cover

Instanz Graph $G = (V, E)$, $k \in \mathbb{N}$

Frage Existiert eine Knotenüberdeckung U mit Größe $|U| \leq k$,
so dass $\forall (v_i, v_j) \in E: \; v_i \in U \; \vee \; v_j \in U$?

NPO

Ein OP $\mathcal{P} = (\boldsymbol{I}, \boldsymbol{LSG}, \boldsymbol{m}, \boldsymbol{Ziel})$ gehört zu der Klasse NPO, wenn das Folgende gilt:

1. Die Menge der Instanzen $\boldsymbol{I}$ ist in Polynomialzeit erkennbar.
2. Es existiert ein Polynom q, so dass, gegeben Instanz $x \in \boldsymbol{I}$, $\forall y \in \boldsymbol{LSG}(x)$: $|y| \leq q(|x|)$
3. Für alle y mit $|y| \leq q(|x|)$ ist in Polynomialzeit entscheidbar, ob $y \in \boldsymbol{LSG}(x)$
4. $\boldsymbol{m}(x, y)$ ist in Polynomialzeit berechenbar

PO

NPO Probleme, für die einen deterministischen Algorithmus mit polynomieller Laufzeit bekannt ist.

NPO Beispiel

Minimum Vertex Cover gehört zu NPO.

1. Die Menge der Instanzen (alle ungerichteten Graphen) ist in Polynomialzeit erkennbar.
2. Jede zulässige Lösung (eine Teilmenge der Knoten) ist kleiner als die Instanz selbst.
3. Es lässt sich in polynomieller Zeit überprüfen, ob eine Knotenmenge eine zulässige Lösung ist.
4. Die Maßfunktion (Kardinalität einer Menge) ist trivial zu berechnen.

Beziehung NP und NPO

Theorem

Für ein Optimierungsproblem $\mathcal{P} \in NPO$ gehört das entsprechende Entscheidungsproblem $\mathcal{P}_{\mathcal{D}}$ zu NP.

Beziehung NP und NPO

Theorem

Für ein Optimierungsproblem $\mathcal{P} \in NPO$ gehört das entsprechende Entscheidungsproblem $\mathcal{P}_{\mathcal{D}}$ zu NP.

O.E. sei $\mathcal{P}$ ein Maximierungsproblem. Für $x \in \boldsymbol{I}$ und $k \in \mathbb{N}$ können wir $\mathcal{P}_{\mathcal{D}}$ mit dem folgenden ND-Algorithmus lösen:

- In Zeit $q(|x|)$ (q ist ein Polynom) einen String y, $|y| \leq q(|x|)$, 'raten'.
- Auch in Polynomialzeit überprüfen, ob $y \in \boldsymbol{LSG}(x)$.
- Wenn ja, $\boldsymbol{m}(x, y)$ auch in polynomieller Zeit berechnen.
- Wenn $\boldsymbol{m}(x, y) \geq k$, 'YES' zurückgeben, sonst 'NO'.

Übergang

- $\triangleright$ NP-schwer sind die schwierigsten Probleme in NPO, genauso wie NP-vollständige Probleme in NP.
- $\triangleright$ NP-schwere Probleme lassen sich nicht effizient lösen $\rightarrow$ man entscheidet sich meistens für eine Lösung, die “*gut genug*” ist.

Approximationsalgorithmen

Gegeben ein OP $\mathcal{P} = (\boldsymbol{I}, \boldsymbol{LSG}, \boldsymbol{m}, \boldsymbol{Ziel})$, ein Algorithmus $\mathcal{A}$ heißt ein Approximationsalgorithmus für $\mathcal{P}$, wenn er für jede Instanz $x \in \boldsymbol{I}$ eine zulässige Lösung $\mathcal{A}(x) \in \boldsymbol{LSG}(x)$ liefert.

Warum ist diese Definition unbefriedigend?

Güte einer Lösung

Absoluter Fehler

Für $\mathcal{P} = (\boldsymbol{I}, \boldsymbol{LSG}, \boldsymbol{m}, \boldsymbol{Ziel})$, $x \in \boldsymbol{I}$, der absolute Fehler einer Lösung $y \in \boldsymbol{LSG}(x)$ im Bezug auf einer optimalen Lösung y^* ist:

$$D(x, y) = |\boldsymbol{m}(x, y^*) - \boldsymbol{m}(x, y)|$$

Relativer Fehler

Für $\mathcal{P} = (\boldsymbol{I}, \boldsymbol{LSG}, \boldsymbol{m}, \boldsymbol{Ziel})$, $x \in \boldsymbol{I}$, der relative Fehler einer Lösung $y \in \boldsymbol{LSG}(x)$ im Bezug auf einer optimalen Lösung y^* ist:

$$E(x, y) = \frac{|\boldsymbol{m}(x, y^*) - \boldsymbol{m}(x, y)|}{\max\{\boldsymbol{m}(x, y^*), \boldsymbol{m}(x, y)\}}$$

Absoluter Approximationsalgorithmus

Gegeben ein OP $\mathcal{P} = (\boldsymbol{I}, \boldsymbol{LSG}, \boldsymbol{m}, \boldsymbol{Ziel})$ und ein Approximationsalgorithmus $\mathcal{A}$ für $\mathcal{P}$. $\mathcal{A}$ ist ein absoluter Approximationsalgorithmus, wenn eine Konstante k existiert, so dass:

$$\forall x \in \boldsymbol{I} : \; D(x, \mathcal{A}(x)) \leq k$$

"In general, we cannot expect such a good performance from an approximation algorithm." (Ausiello et al., 2003, Section 3.1)

ε-approximierender Algorithmus

Gegeben ein OP $\mathcal{P} = (\boldsymbol{I}, \boldsymbol{LSG}, \boldsymbol{m}, \boldsymbol{Ziel})$ und ein Approximationsalgorithmus $\mathcal{A}$ für $\mathcal{P}$. $\mathcal{A}$ ist ε-approximierend, wenn eine Konstante ε existiert, so dass:

$$\forall x \in \boldsymbol{I}: \; E(x, \mathcal{A}(x)) \leq \varepsilon$$

- Ein 1/2-approximierender Algorithmus liefert immer eine Lösung, deren relativer Fehler höchstens 1/2 ist.

Performanzrate

Gegeben ein OP $\mathcal{P}$ für jede Instanz $x \in \boldsymbol{I}$ und Lösung $y \in \boldsymbol{LSG}(x)$ ist die Performanzrate:

$$R(x, y) = \max\left(\frac{\boldsymbol{m}(x,y)}{\boldsymbol{m}(x,y^*)}, \frac{\boldsymbol{m}(x,y^*)}{\boldsymbol{m}(x,y)}\right)$$

- $R(x, y) = 1$ wenn y optimal ist, und arbiträr groß bei schlechten Approximationen.
- Relativer Fehler und Performanzrate: $E(x, y) = 1 - 1/R(x, y)$

r-approximierender Algorithmus

Gegeben ein OP $\mathcal{P} = (\boldsymbol{I}, \boldsymbol{LSG}, \boldsymbol{m}, \boldsymbol{Ziel})$ und ein Approximationsalgorithmus $\mathcal{A}$ für $\mathcal{P}$. $\mathcal{A}$ ist r-approximierend, wenn eine Konstante r existiert, so dass:

$$\forall x \in \boldsymbol{I} : \; R(x, \mathcal{A}(x)) \leq r$$

Die Klasse APX

r-approximierbare Probleme

Ein NPO Problem $\mathcal{P}$ ist r-approximierbar, wenn ein Polynomialzeit-, r-Approximationsalgorithmus für $\mathcal{P}$ existiert.

APX

Die Klasse aller NPO Probleme, für die ein Polynomialzeit-, r-Approximationsalgorithmus ($r \geq 1$) existiert.

Z.B. Minimum Vertex Cover ist 2-approximierbar

$\Rightarrow$ Minimum Vertex Cover $\in$ APX

Beispiel i

Minimum Vertex Cover ist 2-approximierbar. Der folgende Approximationsalgorithmus liefert eine Lösung, die höchstens zweimal größer als die optimale ist.

$C = \emptyset$
while $E \neq \emptyset$ **do**
    choose $(u, v) \in E$
    $C \leftarrow C \cup \{u, v\}$
    $E \leftarrow E - \{(u, w) \mid w \in V\}$
        $- \{(w, v) \mid w \in V\}$
end
return C

Beispiel i

Minimum Vertex Cover ist 2-approximierbar. Der folgende Approximationsalgorithmus liefert eine Lösung, die höchstens zweimal größer als die optimale ist.

$C = \varnothing$
while $E \neq \varnothing$ **do**
    choose $(u, v) \in E$
    $C \leftarrow C \cup \{u, v\}$
    $E \leftarrow E - \{(u, w) \mid w \in V\}$
        $- \{(w, v) \mid w \in V\}$
end
return C

Wieso 2-approximierbar?

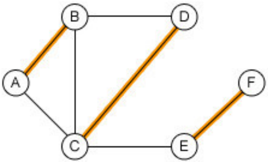
Beispiel ii

Matching

Eine Teilmenge *H* der Kanten, so dass keine zwei Kanten in *H* einen gemeinsamen Endpunkt haben.

Maximales Matching

Würde man eine weitere Kante hinzufügen, wäre *H* kein Matching mehr.



Beobachtung: Die Endpunkte in einem maximalen Matching *H* bilden eine Knotenüberdeckung.

Beispiel iii

- Der vorgeschlagene Algorithmus bildet ein maximales Matching.
- Sei U^* eine minimale Knotenüberdeckung in G. Dann haben wir für jedes maximales Matching H: $|U^*| \geq |H|$.
- Für die Approximation U gilt $|U| = 2|H|$.

$$\Rightarrow |U| \leq 2|U^*|$$

$C = \varnothing$
while $E \neq \varnothing$ **do**
  choose $(u, v) \in E$
  $C \leftarrow C \cup \{u, v\}$
  $E \leftarrow E - \{(u, w) \mid w \in V\}$
    $- \{(w, v) \mid w \in V\}$
end
return C

Problem des Handlungsreisenden (TSP)

Minimum Travelling Salesperson Problem:

Instanz: Menge an 'Städten' $\{c_1, ..., c_n\}$

$n \times n$ Matrix D mit Distanzen $\in \mathbb{Z}^+$.

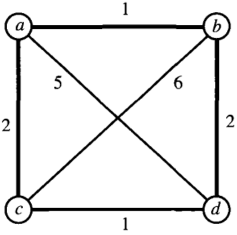
Lösung: Eine Tour aller Städte, d.h. eine Permutation

$\{c_{i_1}, ..., c_{i_n}\}$

Maß: $\sum_{k=1}^{n-1} D(i_k, i_{k+1}) \ + \ D(i_n, i_1)$

	a	b	c	d
a	0	1	2	5
b	1	0	6	2
c	2	6	0	1
d	5	2	1	0

(a)

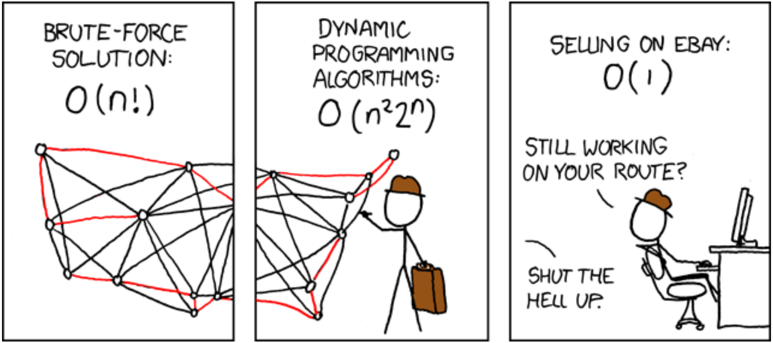


(b)

Abbildung 3: Eine TSP-Instanz, daargestellt als Matrix (a) und als Graph (b).

(Ausiello et al., 2003, Figure 1.5)

Definition iii



Travelling Salesman Problem

Approximierbarkeit von TSP

TSP ist **nicht *r*-approximierbar**, auch für beliebig große *r*.

$\Rightarrow$ TSP $\notin$ APX

Approximierbarkeit von TSP

TSP ist **nicht r-approximierbar**, auch für beliebig große r.

$\Rightarrow$ TSP $\notin$ APX

Theorem

Wenn TSP $\in$ APX, dann P = NP.

Hamiltonkreis

Ein Hamiltonkreis ist ein geschlossener Pfad in einem Graphen, der **jeden Knoten genau einmal** enthält.

Hamiltonkreisproblem

Existiert ein solcher Kreis in einem gegebenen Graphen?

- ▷ Das Hamiltonkreisproblem gehört zu **NP**. (Karp, 1972)

Wir zeigen, dass eine *r*-Approximation in polynomieller Zeit von TSP nur dann möglich ist, wenn HCP $\in$ P $\quad\Rightarrow$ P = NP

Sei $G = (V, E)$ eine Instanz des HCP, $n = |V|$.

Für jedes $r \geq 1$ bilden wir aus G eine TSP Instanz G':

- vollständiger Graph $G' = (V, E')$
- Gewichte: $\quad d(v_i, v_j) = \begin{cases} 1 & \text{if } (v_i, v_j) \in E \\ 1 + nr & \text{else} \end{cases}$

TSP $\in$ APX $\rightarrow$ P = NP

G' **hat eine optimale Tour der Größe** n **gdw. G einen Hamiltonkreis enthält.**

1. Die nächstkleinere Lösung ist $n(1+r)$. Also wird die Performanzrate $\frac{n(1+r)}{n} = 1 + r > r$
2. Man könnte HCP dann lösen, indem man TSP löst und schaut, ob es eine Tour der Größe n gibt. Bei polynomieller r-Approximierbarkeit von TSP, kann man auch HCP in Polynomialzeit lösen. $\Rightarrow$ P=NP

Metrisches TSP

- Die Distanzen sind symmetrisch: $D(u, v) = D(v, u)$
- Es gilt die Dreiecksungleichung: $D(u, v) + D(v, w) \geq D(u, w)$
- Mit Euklidischen Distanzen: Euklidisches TSP

Das metrische TSP ist genauso 'schwer' zu lösen, aber einfacher zu approximieren.

$\rightarrow$ **einfache 2-Approximation möglich**

2-Approximation für das metrische TSP

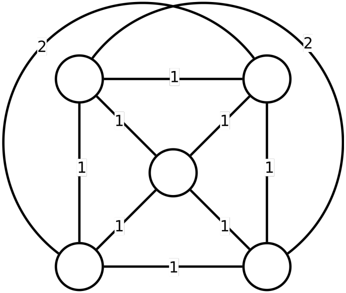
Input: Vollständiger Graph $G = (V, E)$ mit Gewichten

Output: Tour t

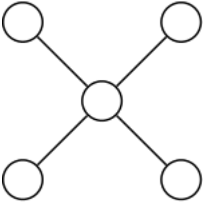
```
Konstruiere einen minimalen Spannbaum T = (V, E_T) von G
Erstelle einen Multigraphen M durch Verdoppelung der Kanten von T
Finde einen Eulerkreis w in M
Extrahiere die Tour t von w
return t
```

2-Approximation: Input und Schritt 1

Vollständiger Graph $G = (V, E)$ mit Gewichten.



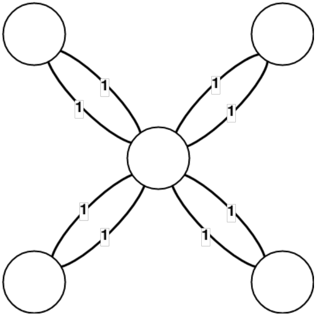
Finde einen minimalen Spannbaum $T = (V, E_T)$ in G.



- In Polynomialzeit lösbar (z.B. Prims Algorithmus)

2-Approximation: Schritt 2

Erstelle einen Multigraphen M durch Verdoppelung der Kanten von T.

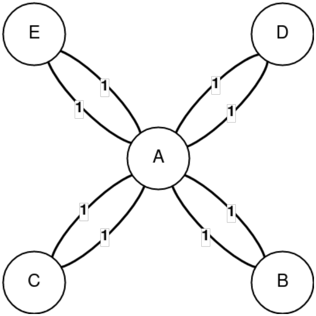


Multigraph

Zwei Knoten können auch durch mehrere Kanten verbunden sein, d.h. E ist eine Multimenge.

2-Approximation: Schritt 3

Finde einen Eulerkreis w in M.



A-B-A-D-A-C-A-E-A

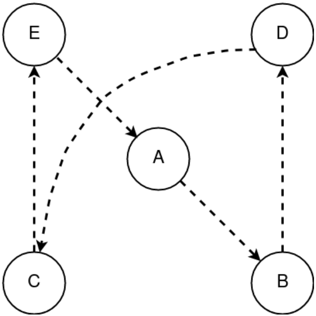
Eulerkreis

Ein geschlossener Pfad in einem Graphen, der **jede Kante genau einmal** und jeden Knoten mindestens einmal enthält.

- ▷ Ein Eulerkreis existiert gdw. alle Knoten einen **geraden Grad** haben.
- ▷ Das Eulerkreisproblem gehört zu **P**. (Karp, 1972)

2-Approximation: Schritt 4 und Output

Extrahiere die Tour t von w.



A-B-D-C-E-A

'Shortcutting':

In dem Eulerkreis wiederholt vorkommende Knoten entfernen, und durch Direktverbindung ersetzen:

A-B**-A-**D**-A-**C**-A-**E-A $\rightarrow$ A-B**-**D**-**C**-**E-A

Diese Tour ist nie länger als der Eulerkreis *w* (folgt aus der Dreiecksungleichung).

2-Approximation: Verbesserungen

$r = 2$ ergibt sich aus der Verdoppelung der Kanten in T.

- ▷ Einen günstigeren Eulerkreis finden, also den Multigraphen M geschickter bilden.
- ▷ Wichtig ist nur, dass alle Knoten einen geraden Grad haben.

$\rightarrow$ **3/2-Approximation mit dem Algorithmus von Christofides** (Christofides, 1976)

Algorithmus von Christofides

Input: Vollständiger Graph $G = (V, E)$ mit Gewichten

Output: Tour t

Konstruiere einen minimalen Spannbaum $T = (V, E_T)$ von G

$C \leftarrow$ {die Knoten in T mit ungeradem Grad}

Finde ein minimales perfektes Matching H in dem Teilgraph (C, E_C)

Erstelle einen Multigraphen $M = (V, E_T \cup H)$

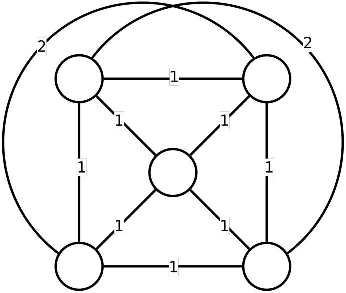
Finde einen Eulerkreis w in M

Extrahiere die Tour t von w

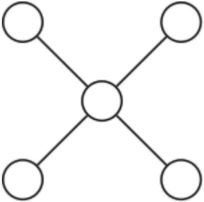
return t

Christofides: Input und Schritt 1

Vollständiger Graph $G = (V, E)$ mit Gewichten.



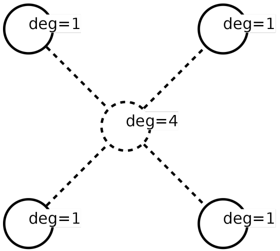
Finde einen minimalen Spannbaum $T = (V, E_T)$ in G.



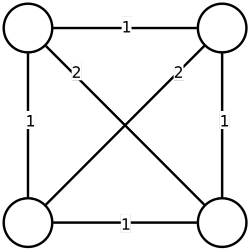
- In Polynomialzeit lösbar (z.B. Prims Algorithmus)

Christofides: Schritt 2

$C \leftarrow \{$*die Knoten in T mit ungeradem Grad*$\}$

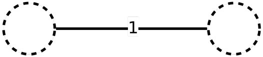


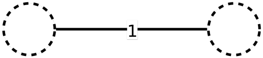
Reduziere G auf die Knoten aus C $\Rightarrow$ *Teilgraph* (C, E_C)



Christofides: Schritt 3

Finde ein minimales perfektes Matching H.





- ▷ In Polynomialzeit lösbar (z.B. Blossom-Algorithmus)

Matching (Wdh.)

$H \subseteq E$, so dass keine zwei Kanten in H einen gemeinsamen Endpunkt haben.

Maximales Matching (Wdh.)

H ist nicht erweiterbar.

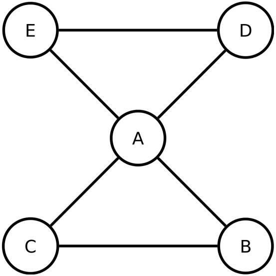
Perfektes Matching

Falls $2|H| = |V|$ (Jeder Knoten ist in einer Kante des Matchings enthalten.)

Christofides: Schritt 4

Erstelle einen Multigraphen

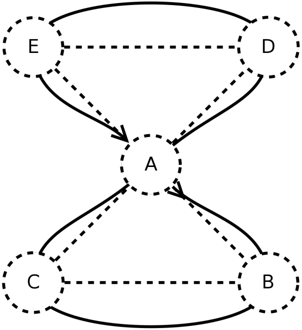
$M = (V, E_T \cup H)$.



Die Vereinigung $E_T \cup H$ sorgt dafür, dass Knoten mit vormals ungeradem Grad nun einen geraden Grad aufweisen.

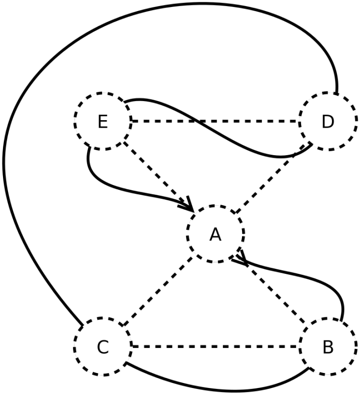
Christofides: Schritte 5 und 6

Finde einen Eulerkreis w in M.



*A-B-C-**A-**D-E-A*

Extrahiere die Tour t von w.



*A-B-C**-**D-E-A*

Christofides: Gütegarantie i

Theorem

Gegeben eine Instanz $G = (V, E)$ des metrischen TSP, der Algorithmus von Christofides liefert in Polynomialzeit eine Lösung t mit Performanzrate $\boldsymbol{r \leq 3/2}$.

- Betrachten wir den Multigraphen $M = (V, E_T \cup H)$.
- Seien $c(T)$ und $c(H)$ die Summen der Kantengewichte im minimalen Spannbaum T und im Matching H.
- Die Länge der Eulerkreis w in M: $\quad c(w) = c(T) + c(H)$
- **Der Maß der Tour $\boldsymbol{t}$:** $\quad \boldsymbol{m}(G, I) \leq c(w) = c(T) + c(H)$

Christofides: Gütegarantie ii

Beobachtung #1: $\boldsymbol{m}(G, t^*) \geq 2c(H)$

- Sei $\boldsymbol{t^* = (v_1, ..., v_n)}$ eine optimale Tour in G.
- Wir betrachten nur die Knoten, die im Matching H waren.

  $\rightarrow$ neue Sequenz $\boldsymbol{(v_{i_1}, ..., v_{i_{2|H|}})}$ mit $\leq n$ Knoten
- Sei H_1 ein Matching $\boldsymbol{\{(v_{i_1}, v_{i_2}), (v_{i_3}, v_{i_4}), ..., (v_{i_{2|H|-1}}, v_{i_{2|H|}})\}}$.
- Sei H_2 ein Matching $\boldsymbol{\{(v_{i_2}, v_{i_3}), (v_{i_4}, v_{i_5}), ..., (v_{i_{2|H|}}, v_{i_1})\}}$.
- Dreiecksungleichung: $\boldsymbol{m}(G, t^*) \geq c(H_1) + c(H_2)$
- H ist ein minimales perfektes Matching

  $\Rightarrow\ c(|H_1|) \geq c(|H|)$ und $c(|H_2|) \geq c(|H|)$

$\Rightarrow\ \boldsymbol{m}(G, t^*) \geq 2c(H)$

Christofides: Gütegarantie iii

Beobachtung #2: $c(T) \leq \boldsymbol{m}(G, t^*)$

- Sei $\boldsymbol{t^* = (v_1, ..., v_n)}$ eine optimale Tour in G.
- Würde man eine Kante entfernen, ist der verbleibende 'Pfad' auch ein Spannbaum, mit Summe der Kantengewichte $c(t^*_{(-1)}) \leq \boldsymbol{m}(G, t^*)$
- Aber T ist ein *minimaler* Spannbaum, also $c(t^*_{(-1)}) \geq c(T)$

$$\Rightarrow \quad c(T) \leq \boldsymbol{m}(G, t^*)$$

(Gilt auch für die 2-Approximation.)

Christofides: Gütegarantie iv

1. $\boldsymbol{m}(G, t) \leq c(T) + c(H)$
2. $\boldsymbol{m}(G, t^*) \geq 2c(H)$
3. $c(T) \leq \boldsymbol{m}(G, t^*)$

Wir ersetzen 2. und 3. in 1.:

$$\boldsymbol{m}(G, t) \leq \boldsymbol{m}(G, t^*) + \frac{\boldsymbol{m}(G, t^*)}{2}$$

$$\frac{\boldsymbol{m}(G, t)}{\boldsymbol{m}(G, t^*)} = r \leq \frac{3}{2}$$

Danke für die Aufmerksamkeit!

Danke für die Aufmerksamkeit!

Fragen?

Literatur

Giorgio Ausiello, Pierluigi Crescenzi, Giorgio Gambosi, Viggo Kann, Alberto Marchetti-Spaccamela, and Marco Protasi. *Complexity and approximation: Combinatorial optimization problems and their approximability properties.* Springer Science & Business Media, 2003.

Nicos Christofides. Worst-case analysis of a new heuristic for the travelling salesman problem. Technical report, Carnegie-Mellon Univ Pittsburgh Pa Management Sciences Research Group, 1976.

Richard M Karp. Reducibility among combinatorial problems. In *Complexity of computer computations*, pages 85–103. Springer, 1972.