

Algorithmische Graphentheorie

6. Übungsblatt

Jasper Gude

Pia Röttgers

2. Juni 2026

21 / 25

Aufgabe 1 – LP-Runden

- a) Wir suchen in U nach dem Knoten v mit dem kleinsten x_v , unser ϵ , und setzen alle Knoten $u \in U$ auf $x_u - \epsilon$ und alle Knoten $w \in W$ auf $\min(x_w + \epsilon, 1)$.

Das macht unsere Beschränkung $x_u + x_w \geq 1$ nicht kaputt, da ein Nachbar w von einem Knoten $u \in U$ mindestens den Wert $\frac{1}{2} + \epsilon$ hat, also 1 ist oder in W liegt.

Somit hat sich $|U| + |W|$ um mindestens 1 verkleinert, da x_v nun 0 ist und somit nicht in $U \cup W$ liegt und zu W keine weiteren Knoten dazukommen können.

Die Veränderungen der Variablen läuft in $\mathcal{O}(V)$, da im schlimmsten Fall alle Knoten entweder in U oder in W liegen.

- b) Wir teilen die Knoten $v \in V$ in drei Mengen ein.



$$A = \left\{ v \mid x_v \leq \frac{1}{2} - \epsilon \right\}$$

$$B = \{ v \mid v \notin A \cup C \}$$

$$C = \left\{ v \mid x_v \geq \frac{1}{2} + \epsilon \right\}$$

Für alle $v \in A$ setzen wir $x_v = 0$, für alle $v \in B$ setzen wir $x_v = \frac{1}{2}$ und für alle $v \in C$ setzen wir $x_v = 1$. Als ϵ wählen wir das kleinste x_v .

Das dürfen wir, da wir damit die LP-Beschränkungen aufrechterhalten.

Mindestens 1-epsilon, nicht mind. 1/2 + epsilon.

Es wäre geschickter, entweder Knoten in U und W anzuschauen, oder sich auf das Maximum in W zu beschränken statt das Minimum in U .

Wir nehmen an, dass $|U| + |W| \geq 1$ und $|W| \geq |U|$ gilt.

^Daraus alleine können wir noch nicht folgern, dass U nicht-leer ist. Ohne weitere Begründung ist die Lösung nicht vollständig. (Sie funktioniert trotzdem, aber das folgt nicht aus den gegebenen Annahmen)

3 / 4

1

b) soll einen Algorithmus angeben, der eine optimale Lösung für eine LP-Relaxierung mit gegebenen Eigenschaften findet. Dafür muss der Algo ein LP lösen, was in der Laufzeit zu berücksichtigen ist.

Karmakars Algorithmus aus der VL löst LPs in $\mathcal{O}(L^2 n^{3.5})$

Ziel der Aufgabe war unter anderem, sich zu überlegen, wie man L (Größe des zu lösenden LPs) in Abhängigkeit von $n = |V|$ angeben kann; also wie viele Constraints gebraucht werden und wie diese fürs LP dargestellt werden.

1. Für alle Variablen x_v gilt $0 \leq x_v \leq 1$.

2. Für alle Paare v, u gilt $x_v + x_u \geq 1$, da

Fall 1: O.B.d.A gilt: Wenn $x_v \leq \frac{1}{2} - \epsilon$, dann muss $x_u \geq \frac{1}{2} + \epsilon$, wird $x_v = 0$ und $x_u = 1$.

Fall 2: Wenn x_v oder $x_u \geq \frac{1}{2} + \epsilon$, werden x_v oder $x_u = 1$.

Fall 3: Wenn $x_v, x_u \notin A \cup C$, werden $x_v = x_u = \frac{1}{2}$.

Die Laufzeit beläuft sich auf $\mathcal{O}(V)$ da sich das $\min_{v \in V} x_v$ in $\mathcal{O}(V)$ Zeit finden und sich jeder Knoten in $\mathcal{O}(V)$ in eine der drei Mengen einteilen lassen kann.

- c) Wir nehmen als Basis den Algorithmus aus Punkt b). Um nun eine eindeutige 2-Approximation für die Knotenüberdeckung zu bekommen, runden wir alle $x_v = \frac{1}{2}$ auf 1 auf.

Dadurch erhalten wir eine Lösung die maximal doppelt so viele Knoten enthält, wie eine optimale Lösung, da für ein Knotenpaar v, u für das $x_v = x_u = \frac{1}{2}$ mindestens ein Knoten in der Knotenüberdeckung enthalten sein muss. Wenn wir beide Knoten nehmen, haben wir doppelt so viele.

Wäre gut, irgendwo in b) und/oder c) nochmal zu wiederholen, dass $x_v=1 \Leftrightarrow v$ in Überdeckung.

Aber es ist hinreichend klar impliziert.

Sowas bitte für Vollständigkeit der Lösung immer mit angeben.
Aber zwei erreichbare Punkte sind zu wenig, um einen angemessenen Abzug unterzubringen.

Gäbs 3 oder mehr Punkte auf Teilaufgabe c, dann hätte ich keine volle Punktzahl gegeben ;)

2 / 3

Ich gehe gerade nochmal durch meine Korrekturen durch und nehme Punkte weg, wenn c) eine 2-Approximation für den ursprünglichen Algorithmus, nicht für das eigentliche Problem diskutiert.

Das haben fast alle übersehen. Freue mich, hier zu sehen dass die Approximation tatsächlich auf die optimale Lösung bezogen ist. Lob =)

2 / 2

Aufgabe 2 – Christofides' Algorithmus

- a) Siehe Abbildung 2.

4 / 4

- b) Siehe Abbildung 1. Diese Tour kann nicht von CHRISTOFIDES berechnet werden, denn egal mit welcher Kante in welche Richtung gestartet wird, nimmt CRISTOFIDES immer eine Kante in die Tour, die nicht in Abbildung 1 ist.

1 / 1

Ich würde mich über 2 Dinge beschweren:

1) in a) und b) sollte irgendwo das Gesamtgewicht der gefundenen Tour stehen.

2) Für die Nachvollziehbarkeit der a) wäre gut, mit anzugeben in welchem Knoten der Rundlauf startet, am besten auch jeweils nochmal dazu schreiben, welche Kante in einem Schritt der Rundtour abgelaufen oder übersprungen wird.

Die Lösung ist so hübsch dass mir die Augen feucht werden und fast lesaber genug. (mit Beschwerde 1 erfüllt, würde ichs ggf auch "gerade lesbar genug" aufrunden).

Aber ich kann es nicht verantworten, hier Punkte abzuziehen.

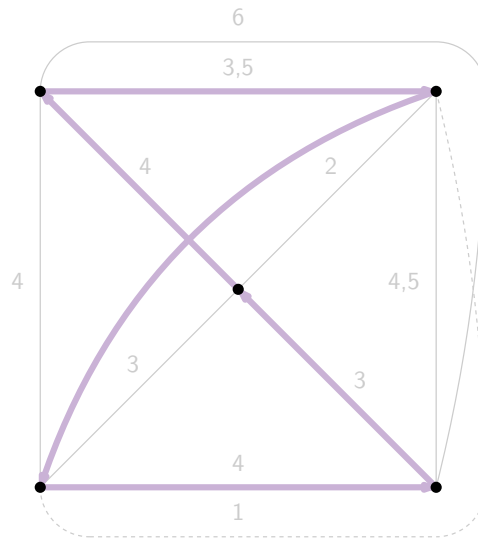


Abbildung 1: Eine kleinere TSP-Tour, die von CRISTOFIDES nicht berechnet werden kann.

Aufgabe 3 – Matchings in allgemeinen Graphen

a)

$\text{MATCHINGS}(G = (V, E))$

$M = \emptyset$

$\text{visited} = \text{Array von False der Größe } |V| \text{ // Markiert gematchte Knoten}$

for e **in** E **do**

if $\neg \text{visited}[u] \wedge \neg \text{visited}[v]$ **then**

$M = M \cup \{e\}$

$\text{visited}[u] = \text{True}$

$\text{visited}[v] = \text{True}$

return M

Meistens geschickter, solche Information als Knoten- oder Kantenattribute zu speichern statt eine separate Datenstruktur anzulegen.

Macht aber hier keinen (nennenswerten) Unterschied.

(ich könnte mich beschweren, dass u und v Knoten und keine Integer-Werte sind, und deswegen die Feld-Adressierung nicht funktioniert. Aber mir ist klar genug was gemeint ist und wie man es umsetzen würde, für Pseudocode sehe ich daher kein Problem)

Angenommen M wäre erweiterbar (d.h. es gibt eine Kante $\{u, v\}$ mit $u, v \notin V(M)$). Dann wurden sowohl u als auch v während des Algorithmus beim Durchlaufen der Kante $\{u, v\}$ als frei angesehen und wäre somit der Menge M hinzugefügt worden. Das ist ein Widerspruch.

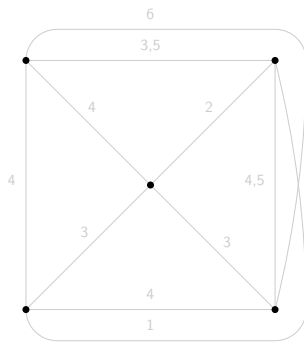
Laufzeit:

Initialisierung des Arrays: $\mathcal{O}(V)$

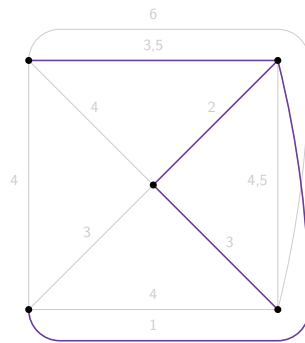
Schleife: Jede Kante wird genau einmal betrachtet ($\mathcal{O}(V)$) und die Überprüfung und Markierung passieren in $\mathcal{O}(1)$. Also insgesamt $\mathcal{O}(E)$

Das ergibt eine Gesamtlaufzeit von $\mathcal{O}(V + E)$

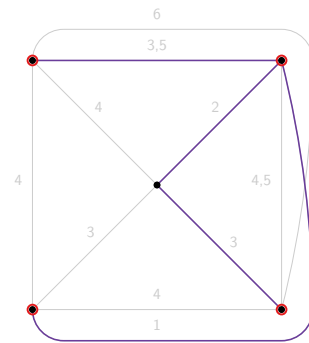
3 / 3



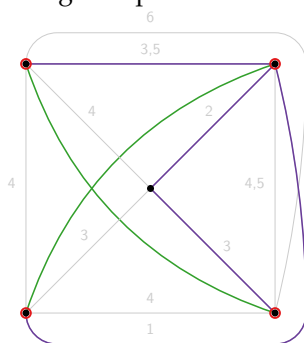
(a) Der gewichtete, vollständige Graph G .



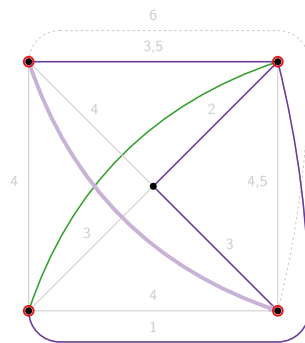
(b) Minimalen Spannbaum finden.



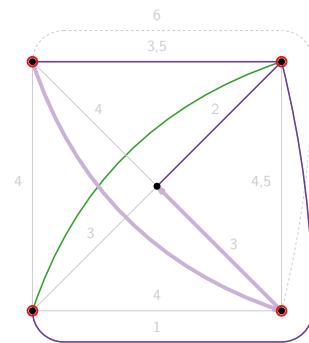
(c) Knoten U mit ungeraden Graden finden.



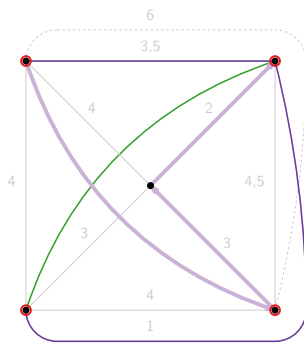
(d) Minimales, perfektes Matching auf $G[U]$ finden.



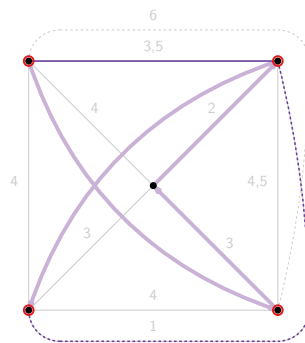
(e) Eulertour konstruieren.



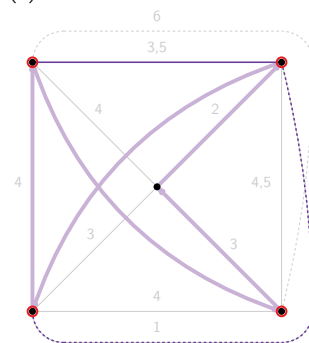
(f) Eulertour konstruieren.



(g) Eulertour konstruieren.



(h) Eulertour konstruieren.



(i) Schon besuchte Knoten überspringen. TSP vollständig.

Abbildung 2: Cristofides' Algorithmus

"maximal" ist (im Deutschen) nicht eindeutig für Matchings (mit dem Kontext ist hier aber eindeutig zu erkennen, was gemeint ist, also hier kein Problem).
 Generell besser entweder konsequent die englischen Begriffe Maximum und Maximal verwenden, oder im deutschen vom "größten" oder vom "nicht erweiterbaren" Matching sprechen.

b)

/ 2

- c) Sei M das vom Algorithmus berechnete maximale Matching und M^* das optimale maximale Matching.

Für jede Kante $e^* = \{u, v\} \in M^*$ gilt: mindestens einer der beiden Knoten u oder v muss über eine Kante aus M abgedeckt werden, sonst wäre der M nicht nicht-erweiterbar.

Eine Kante aus M hat genau zwei Endknoten und kann daher höchstens zwei verschiedene Kanten aus M^* „blockieren“. Da alle Kanten in M^* disjunkt sind kann man daraus folgern:

$$|M^*| \leq 2 \cdot |M| \implies |M| \geq \frac{1}{2} |M^*|$$

Somit ist der Algorithmus aus der Teilaufgabe a) eine $1/2$ Approximation für ein optimales Matching.

3 / 3

- d) Zielfunktion:

$$\arg \min \sum_{e \in E} x_e \geq 1$$

Entscheidungsvariablen: für jede Kante $e \in E$:

$$x_e \in \{0, 1\} \quad \forall e \in E$$

Die Variable nimmt den Wert 1 an, wenn die Kante im Matching M ist und 0 falls sie das nicht ist.

Nebenbedingungen:

1. Matching: Jeder Knoten darf von maximal einer Matching-Kante berührt werden

$$\forall v \in V: \sum_{e \in \delta(v)} x_e \leq 1$$

$\delta(v)$ ist die Menge aller Kanten welche an v anliegen

2. Nicht-Erweiterbarkeit: Für jede Kante $\{u, v\}$ muss die Summe der Matching-Kanten an u und v mindestens 1 sein

$$\forall \{u, v\} \in E: \sum_{e \in \delta(u)} x_e + \sum_{e' \in \delta(v)} x_{e'} \geq 1$$

$\delta(v)$ ist die Menge aller Kanten welche an v anliegen. Äquivalent für $\delta(u)$.

3 / 3

e)

/ 2