

2022_t2_A

⚠ Disclaimer

Alles was hier drinnen steht kann Fehler enthalten! Falls dir etwas auffällt melde dich gerne auf Discord bei mir ([@xmozz](#))

A1: P und NP, Spezialfälle

Stoff: [9. Polynominalzeitreduktionen](#) und [sem_2/AlgoDat/vo/2. Test/10. NP-Vollständigkeit Spezialfälle](#)

a)

- a) (8 Punkte) Betrachten Sie einen Graphen $G = (V, E)$. Eine Menge von Knoten $D \subseteq V$ heißt dominierende Menge, wenn jeder Knoten außerhalb von D adjazent zu mindestens einem Knoten in D ist. Das heißt, für jeden Knoten $u \in V \setminus D$ gibt es einen Knoten $v \in D$, sodass $(u, v) \in E$. Das DOMINATING SET Problem ist wie folgt definiert.

DOMINATING SET: Gegeben sei ein Graph $G = (V, E)$ mit $|V| = n$ Knoten und eine ganze Zahl $k > 0$. Existiert eine dominierende Menge von Knoten $D \subseteq V$, sodass $|D| \leq k$?

Wir wollen zeigen, dass DOMINATING SET in NP liegt, indem wir ein geeignetes Zertifikat und einen passenden Zertifizierer finden.

(i) Welche Eigenschaften muss ein geeigneter Zertifizierer erfüllen?

Ein Zertifizierer (polynomieller Größe) muss ein Zertifikat auswerten sodass:

- Er eine ja Antwort gibt, wenn die Probleminstanz eine Ja Antwort auf die Frage gibt
- Genau das Gegenteil bei einer Nein Instanz.
- Das muss er in polynomieller Zeit machen

(ii) Geben Sie ein geeignetes Zertifikat und einen passenden Zertifizierer an. Argumentieren Sie, dass Ihr Zertifizierer die Eigenschaften aus Punkt (i) erfüllt.

Ein Zertifikat C wäre eine Menge D von Knoten $D \subseteq V$.

Ein Zertifizierer nimmt als Eingabe den Graphen $G(V, E)$, die Zahl k und D und:

- Prüft die Größe
 - Prüft die Dominanz Eigenschaft
- Wenn beides Zutrifft wird die Instanz akzeptiert.

b)

b) (12 Punkte) Seien A, B und C Ja/Nein-Probleme und n deren Eingabegröße.
Nehmen Sie an, es gibt

- eine Reduktion von 3-SAT nach DOMINATING SET in Zeit n^3 ,
- eine Reduktion von DOMINATING SET nach A in Zeit n ,
- eine Reduktion von DOMINATING SET nach B in Zeit n^3 ,
- eine Reduktion von DOMINATING SET nach C in Zeit 4^n ,
- eine Reduktion von A nach 3-SAT in Zeit n ,
- eine Reduktion von C nach A in Zeit n^2 .

Aus Unteraufgabe a) wissen wir, dass DOMINATING SET in NP liegt.

(i) Geben Sie die engste obere Schranke für die Laufzeit einer Reduktion von C auf Dominating Set in O-Notation an, die sich aus diesen Annahmen ableiten lässt. Begründen Sie Ihre Antwort kurz.

$$C \rightarrow A \rightarrow 3SAT \rightarrow DominatingSet$$

$$O(((n^2)^1)^3) = O(n^{2 \cdot 1 \cdot 3}) = \boxed{O(n^6)}$$

(ii) Was folgt aus den obigen Reduktionen für die Komplexität der Probleme Dominating Set, A, B, und C? Beantworten Sie die Frage, indem Sie alle zutreffenden Felder in der folgenden Tabelle ankreuzen

	in P	in NP	NP-schwer	NP-vollständig
DOMINATING SET		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
A		<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
B			<input checked="" type="checkbox"/>	
C		<input checked="" type="checkbox"/>		

- eine Reduktion von 3-SAT nach DOMINATING SET in Zeit n^3 ,
- eine Reduktion von DOMINATING SET nach A in Zeit n ,
- eine Reduktion von DOMINATING SET nach B in Zeit n^3 ,
- eine Reduktion von DOMINATING SET nach C in Zeit 4^n ,
- eine Reduktion von A nach 3-SAT in Zeit n ,
- eine Reduktion von C nach A in Zeit n^2 .

A2: Branch-and-Bound

Stoff: sem_2/AlgoDat/vo/2. Test/11. Branch and Bound

a)

- a) (4 Punkte) Welche Aussagen für Branch-and-Bound Verfahren sind korrekt? Kreuzen Sie Zutreffendes an.

(+1 Punkt für jede richtige, -1 Punkt für jede falsche und 0 Punkte für keine Antwort, keine negativen Gesamtpunkte auf diese Unteraufgabe)

- (Q1) Eine korrekte optimale Lösung kann nur gefunden werden, wenn alle Teilprobleme betrachtet werden.

Wahr Falsch

- (Q2) Branch-and-Bound für ein Maximierungsproblem funktioniert im Allgemeinen umso besser, je größer die lokalen oberen Schranken sind.

Wahr Falsch

- (Q3) Wenn ein Branch-and-Bound Algorithmus die nächste Teilinstanz schlecht auswählt, kann es sein, dass keine optimale Lösung mehr gefunden wird.

Wahr Falsch

- (Q4) Bei einem Minimierungsproblem kann die Branch-and-Bound Suche für eine Teilinstanz abgebrochen werden, wenn die globale untere Schranke größer als lokale obere Schranke ist.

Wahr Falsch

Q1: Das ist der Kern hinter Brachn-and-Bound

Q2: Falsch, da *kleinere* lokale obere Schranken es ermöglichen, mehr Teillösungen frühzeitig abzuschneiden, weil sie schlechter sind als die aktuell beste gefundene globale Lösung.

Q3: Eine "schlechte" Auswahl der nächsten Teilinstanz bewirkt nur, dass die optimale Lösung *langsamer* gefunden wird

Q4: Bei einem Minimierungsproblem kann die Branch-and-Bound-Suche für eine Teilinstanz abgebrochen werden, wenn die lokale untere Schranke größer als die *globale* obere Schranke ist

b)

b) (8 Punkte) Wir betrachten einen Branch-and-Bound Algorithmus für ein Maximierungsproblem. Betrachten Sie die folgenden Teilstücke, die in diesem Branch-and-Bound Algorithmus behandelt werden könnten. Gehen Sie jeweils von einer globalen unteren Schranke von 70 aus, bevor die Schranken der Teilstücke berechnet werden.

- (I₁) Teilstück mit $L' = 60, U' = 80$
- (I₂) Teilstück mit $L' = 78, U' = 78$
- (I₃) Teilstück mit $L' = 55, U' = 55$
- (I₄) Teilstück mit $L' = 40, U' = 65$
- (I₅) Teilstück mit $L' = 70, U' = 75$

Geben Sie an, welche der Teilstücke im Branch-and-Bound Algorithmus weiter aufgespalten werden müssen.

Teilstücke:

Bei welcher der Teilstücke würde sich die globale untere Schranke ändern?

Teilstücke:

c)

- c) (8 Punkte) Betrachten Sie die folgende Instanz des Rucksackproblems. Die Gewichte und Werte der Gegenstände sind in der Tabelle angegeben. Die Kapazität des Rucksacks beträgt $G = 10$.

Gegenstand	A	B	C	D
Gewicht	3	7	2	6
Wert	15	21	14	12
Verhältnis	5	3	7	2

Berechnen Sie die Wert-Gewichts-Verhältnisse aller Gegenstände und tragen Sie diese in obiger Tabelle ein. Geben Sie die Reihenfolge an, in der die Gegenstände betrachtet werden, wenn Sie den **verbesserten** Branch-and-Bound Algorithmus der Vorlesung anwenden.

Reihenfolge: C, A, B, D

Betrachten Sie nun das initiale Teilproblem des **verbesserten** Branch-and-Bound Algorithmus aus der Vorlesung bei einer Rucksackkapazität von 10 und geben Sie die lokale obere Schranke U' , die lokale untere Schranke L' , sowie eine passende Auswahl von Gegenständen mit Gesamtwert L' an.

$$L' = 15 + 21 = 36$$

$$U' = 15 + 21 = 36$$

$$S = A, B$$

$$L' = 15 + 15 = 29$$

$$U' = 15 + 15 + \frac{5}{7} \cdot 21 = \frac{105}{7} + 29$$

$$C=1$$

$$L' = 15 + 15 = 29$$

$$U' = 15 + 15 + \frac{5}{7} \cdot 21 = \frac{105}{7} + 29$$

$$C=0$$

$$L' = 15 + 21 = 36$$

$$U' = 15 + 21 = 36$$

$$A=1$$

$$L' = 15 + 15 = 29$$

$$U' = 15 + 15 + \frac{5}{7} \cdot 21 = \frac{105}{7} + 29$$

$$A=0$$

$$L' = 15 + 21 = 25$$

$$U' = 15 + 21 + \frac{1}{6} \cdot 12 = 27$$

Perfekte Lp

:

:

d)

- d) (2 Punkte) Wie viele Teilstufen betrachtet der Branch-and-Bound Algorithmus für das Rucksackproblem im Allgemeinen im Worst-Case? Kreuzen Sie **alle** zutreffenden Schranken an, wobei n der Anzahl der Gegenstände entspricht.
(alles korrekt: 2 Punkte, sonst: 0 Punkte)

$O(n^3)$ $O(n^2)$ $O(2^n)$ $\Theta(3^n)$

A3: Dynamisches Programmieren

Stoff: [sem_2/AlgoDat/vo/2. Test/12. Dynamische Programmierung](#)

Folgendes Problem ist als das GOLDGRÄBER Problem bekannt: Gegeben sei eine Matrix M der Dimension $m \times n$ die eine Goldmine repräsentiert. Jede Zelle der Matrix beinhaltet einen ganzzahligen Wert ≥ 0 der die Menge an Gold in dieser Zelle repräsentiert. Anfänglich wählt der Goldgräber eine beliebige Position in der ersten Spalte der Matrix. Der Goldgräber kann ab nun je einen von 3 Schritten durchführen:

- er bewegt sich diagonal nach rechts oben,
- er bewegt sich nach rechts,
- er bewegt sich diagonal nach rechts unten.

Schritte nach außerhalb der Matrix sind nicht erlaubt. Der Goldgräber stoppt sobald er in der letzten Spalte angekommen ist. Was ist die maximale Menge an Gold die der Goldgräber auf seinem Pfad einsammeln kann?

Für diese Aufgabe betrachten wir folgende Instanz:

$$M = \begin{pmatrix} 1, 3, 1, 5 \\ 2, 2, 4, 1 \\ 5, 0, 2, 2 \\ 0, 6, 1, 0 \end{pmatrix}$$

a)

- a) (5 Punkte) Nehmen Sie an, der Goldgräber wählt aus den erreichbaren Zellen immer eine mit dem meisten Gold.

(i) Um welche Art von Strategie handelt es sich hier?

Greedy Strategie

(ii) Geben Sie die Zeilen in der Reihenfolge wie der Goldgräber sie besucht, sowie die eingesammelte Summe an Gold an. Der Index der Zeilen startet bei 1.

$$5 \rightarrow 6 \rightarrow 2 \rightarrow 2 \implies 15$$

(iii) Geben Sie eine 3×3 Matrix an bei der diese Strategie immer die optimale Lösung liefert.

$$M = \begin{pmatrix} 5 & 5 & 5 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$$

b)

- b) (3 Punkte) Um einen optimalen Pfad des Goldgräbers Brute-Force zu finden, müssen alle Pfade durch die Matrix betrachtet werden. Geben Sie eine möglichst enge obere Schranke für die Gesamtanzahl der Pfade in Abhängigkeit von m Zeilen und n Spalten an. Sie können den Umstand ignorieren, dass Pfade außerhalb der Matrix ungültig sind.

c)

- c) (12 Punkte) Vervollständigen Sie folgendes dynamisches Programm welches eine optimale Lösung mithilfe einer Tabelle „gold“ berechnet. Wir nutzen folgende Variablen und Konstanten:

- Ein zweidimensionales Array M , das die Goldmine als Eingabe enthält.
- Die Anzahl der Zeilen m und die Anzahl der Spalten n .
- Die Variable i , welche den Index der aktuellen Zeile enthält.
- Die Variable j , welche den Index der aktuellen Spalte enthält.
- Ein zweidimensionales Array $gold$ das im Eintrag (i, j) den maximal akkumulierten Wert an Gold speichert, der in einer Zeile i nach j -Schritten erreicht werden kann.
- Die Variablen $rightUp$, $right$ und $rightDown$ speichern für den aktuellen Wert von i und j das bis dahin akkumulierte Gold mit dem die Zelle (i, j) mit einem „diagonal nach rechts oben“, „nach rechts“ oder „diagonal nach rechts unten“ Zug erreicht werden kann.

Function goldgräber(M)

Setze alle Einträge von $gold$ auf 0

for $j = 1 \dots n$

for $i = 1 \dots m$

if $j == \text{ } \textcircled{1}$ **then**

$gold[i][j] \leftarrow M[i, j]$

else

$right \leftarrow gold[i+1, j]$

if $i == \text{ } \textcircled{0}$ **then**

$rightUp \leftarrow -\infty$ //invalid move

else

$rightUp \leftarrow gold[i-1, j]$

if $i == \text{ } \textcircled{n-1}$ **then**

$rightDown \leftarrow -\infty$ //invalid move

else

$rightDown \leftarrow gold[i+1, j]$

$gold[i][j] \leftarrow \max(rightDown, rightUp, right) + M[i, j]$

A4: Approximationsalgorithmen

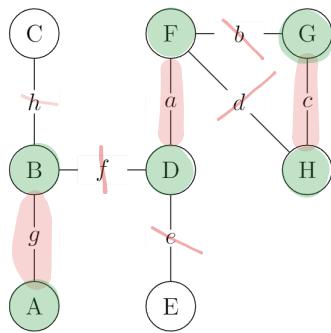
Stoff: sem_2/AlgoDat/vo/2. Test/13. Approximation

a)

- a) Beim MINIMUM VERTEX COVER Problem geht es darum, für einen gegebenen Graphen ein Vertex Cover mit minimaler Anzahl an Knoten zu finden. Betrachten Sie dazu den aus der Vorlesung bekannten 2-Approximationsalgorithmus.

(i)

- (i) (6 Punkte) Führen Sie den Algorithmus auf folgendem Graphen aus und geben Sie das resultierende Vertex Cover an. Ist dieses minimal? Wenn Sie die Auswahl zwischen mehreren Knoten haben, nehmen Sie die alphabetisch kleinste.



```
Approx-Vertex-Cover( $G$ ):
 $C \leftarrow \emptyset$ 
while  $E \neq \emptyset$ 
    Wähle eine beliebige Kante  $(u, v) \in E$ 
     $C \leftarrow C \cup \{u, v\}$ 
    Entferne aus  $E$  alle Kanten, die inzident
        zu  $u$  oder  $v$  sind
return  $C$ 
```

$$C = \{A, B, D, F, G, H\}$$

Ist es Minimal?

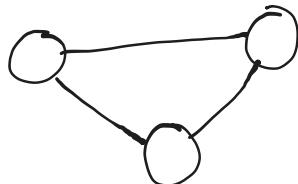
Nein minimal wäre $C = \{B, D, F, H\}$

(ii)

- (ii) (6 Punkte) Geben Sie zwei zusammenhängende Graphen G_1 und G_2 mit jeweils mindestens drei Knoten an, sodass **unabhängig von der Kantenwahl** im Algorithmus:

- für G_1 immer ein minimales Vertex Cover gefunden wird, also $c(G_1) = c_{\text{opt}}(G_1)$ gilt;
- für G_2 immer eine schlechteste mögliche Lösung mit $c(G_2) = 2 \cdot c_{\text{opt}}(G_2)$ gefunden wird.

$G_1 :$



$G_2 :$



b)

- b) (4 Punkte) Kreuzen Sie an, ob die folgenden Aussagen wahr oder falsch sind.

(+1 Punkt für jede richtige, -1 Punkt für jede falsche und 0 Punkte für keine Antwort, keine negativen Gesamtpunkte auf diese Unteraufgabe)

- (Q1) Falls $P = NP$ gilt, dann existiert für das MINIMUM VERTEX COVER Problem ein polynomieller 1.25-Approximationsalgorithmus.

Wahr Falsch

- (Q2) Da im symmetrischen TSP die Dreiecksungleichung gilt, gibt es dafür Approximationsalgorithmen mit besserer Gütegarantie als im metrischen TSP.

Wahr Falsch

- (Q3) Da das Finden eines Hamiltonkreises NP-vollständig ist, gibt es unter der Annahme von $P \neq NP$ keinen polynomiellen 2-Approximationsalgorithmus für das symmetrische TSP.

Wahr Falsch

- (Q4) Ein 2-Approximationsalgorithmus für ein Minimierungsproblem findet eine Lösung die höchstens doppelt so groß wie die optimale Lösung ist.

Wahr Falsch

Q1: Dann hätte man einen 1-Approximationsalgorithmus und da der besser ist als 1.25, schließt dieser den 1.25er Algorithmus mit ein --> Daher Wahr.

Q2: Andersherum

c)

- c) (4 Punkte) Für ein Maximierungsproblem \mathcal{P} existieren zwei Approximationsalgorithmen A und B mit den Gütegarantien $\varepsilon_A = \frac{1}{2}$ und $\varepsilon_B = \frac{1}{3}$. Für eine Instanz I von \mathcal{P} berechnen die Algorithmen die Werte $c_A(I) = 2022$ und $c_B(I) = 909$. In welchem Bereich kann der tatsächliche optimale Lösungswert $c_{\text{opt}}(I)$ liegen?

Das Ergebnis liegt zwischen $2022 \cdot 2 = 4044$ und $909 \cdot 3 = 2727$

Also zwischen 2022 und 2727

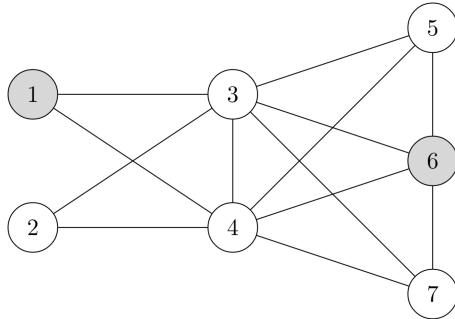
A5: Heuristische Verfahren

Stoff: 14. Heuristiken und Lokale Suche

Beim MAXIMUM INDEPENDENT SET Problem geht es darum, für einen gegebenen Graphen $G = (V, E)$ ein Independent Set $S \subseteq V$ mit der maximalen Anzahl an Knoten zu finden. Wir definieren die folgenden zwei Nachbarschaftsstrukturen N_1 und N_2 für dieses Problem:

- $S' \in N_1(S)$ genau dann wenn S' aus S durch Hinzufügen eines einzigen Knotens in $V \setminus S$ erzeugt werden kann und noch immer ein Independent Set von G ist;
- $S' \in N_2(S)$ genau dann wenn $S' \in N_1(S)$ oder wenn S' durch Entfernen eines Knotens aus S und Hinzufügen von zwei Knoten aus $V \setminus S$ gebildet werden kann und noch immer ein Independent Set von G ist.

Gegeben ist folgender Graph H in dem $T = \{1, 6\}$ ein Independent Set ist.



a)

a) (6 Punkte) Bestimmen Sie $N_1(T)$ und $N_2(T)$ für den obigen Graphen H .

$$\begin{aligned}N_1(T) &= \{1, 2, 6\} \\N_2(T) &= \{\{1, 2, 6\}, \{1, 5, 7\}\}\end{aligned}$$

b)

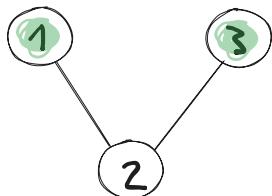
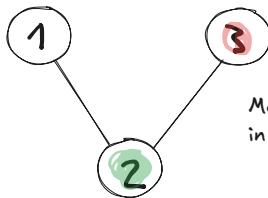
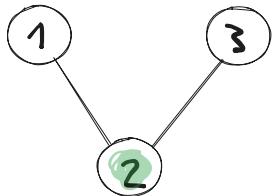
b) (4 Punkte) Wenden Sie lokale Suche mit Nachbarschaftsstruktur N_2 auf H mit Ausgangslösung T an, bis ein lokales Optimum erreicht wird. Schrittfolge und Durchmusterungsreihenfolge der Nachbarlösungen können frei gewählt werden. Geben Sie die Zwischenschritte an, also jedes Independet Set welches im Laufe der lokalen Suche ausgewählt wird.

Wir nehmen statt $T \{1, 2, 6\} \rightarrow$ es ist besser $\rightarrow \{1, 2, 6\}$ wird unser neues T

Wir nehmen statt $\{1, 2, 6\}$ jetzt $\{1, 5, 7\} \rightarrow$ gleich gut \rightarrow bleibt gleich, weil wir schon am lokalen Optimum sind.

c)

- c) (4 Punkte) Bestimmen Sie einen ungerichteten Graphen mit genau drei Knoten und eine Ausgangslösung (ein Independent Set), sodass eine lokale Suche mit N_1 kein globales Optimum erreicht, eine lokale Suche mit N_2 aber schon. Geben Sie eindeutig an, welche Knoten Ihres Graphen zur Ausgangslösung gehören.



d)

- d) (4 Punkte) Kreuzen Sie an, ob die folgenden Aussagen wahr oder falsch sind.

(+1 Punkt für jede richtige, -1 Punkt für jede falsche und 0 Punkte für keine Antwort, keine negativen Gesamtpunkte auf diese Unteraufgabe)

- (Q1) Lokale Suche mit N_1 ist ein $\frac{1}{2}$ -Approximationsalgorithmus für das MAXIMUM INDEPENDENT SET Problem.

Wahr Falsch

- (Q2) Sei G ein beliebiger Graph und S ein Independent Set von G . Dann gilt $|N_1(S)| \leq |N_2(S)|$.

Wahr Falsch

- (Q3) Angenommen man findet bei einer lokalen Suche mit N_1 ein maximales Independent Set S für einen Graphen G . Dann ist S auch ein minimales Vertex Cover für G .

Wahr Falsch

- (Q4) Angenommen man könnte bei lokaler Suche mit N_2 immer garantieren ein globales Optimum (ein maximales Independent Set) finden, egal welcher Graph und welche Ausgangslösung verwendet werden. Dann gilt $P = NP$.

Wahr Falsch

3. Nein es wäre die Komplementärmenge