



Wintersemester 2017/2018

Lineare und kombinatorische Optimierung

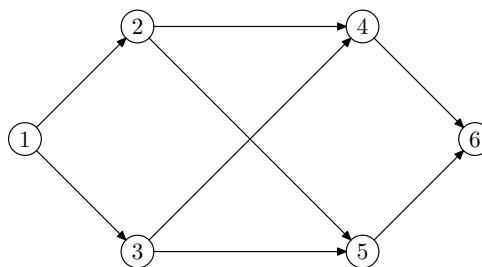
Übungsblatt 8

Gruppenübungen

Aufgabe 1. (Der Christkindlesmarkt und Min-Cost-Flow)

(0 Punkte)

Gegeben sei folgender Graph.



Bei Knoten 1 befindet sich eine Fabrik für Weihnachtslebkuchen, bei Knoten 6 befindet sich der Christkindlesmarkt, auf dem 20 Tonnen Lebkuchen verkauft werden sollen. Die Kanten zwischen den Knoten stellen Straßen dar, über die jeweils nur eine bestimmte Menge an Lebkuchen mit LKWs transportiert werden darf. An den Knoten 2, 3, 4 und 5 sind Umladestationen, an denen die Lebkuchen optimal (und kostenfrei) umsortiert werden können. An den Umladestationen werden weder Lebkuchen hergestellt noch konsumiert. In der folgenden Tabelle findest du die jeweiligen Kantenbeschränkungen und Kosten:

Bogen	(1,2)	(1,3)	(2,4)	(2,5)	(3,4)	(3,5)	(4,6)	(5,6)
Kapazität (in Tonnen)	15	13	10	10	8	9	9	17
Kosten (in Euro)	100	150	150	130	80	80	100	120

Betrachten wir nun ein leicht modifiziertes Lebkuchenproblem. Bei Knoten 5 wohnt eine fleißige Oma, die zusätzlich 5 Tonnen Lebkuchen backt, die ebenfalls verkauft werden sollen. An Knoten 2 wohnt ein hungriger Mitarbeiter der Transportfirma, der 5 Tonnen Lebkuchen konsumiert.

Formulieren Sie die beiden beschriebenen Probleme jeweils als Minimalkosten-Fluss-Problem. Das heißt, stellen Sie für die Probleme jeweils das zugehörige lineare Programm auf.

Lösung:

Sei $D = (V, A)$ der Graph aus der Abbildung. Für jeden Bogen $a \in A$ definieren wir

- c_a = Kapazität auf Bogen a ,
- w_a = Kosten von bogen a .

Weiterhin definieren wir eine Funktion $b : V \rightarrow \mathbb{R}$ mit $b(1) = 20$, $b(6) = -20$ und $b(v) = 0$ für $v = 2, 3, 4, 5$. Dann können wir das erste Problem als Minimalkosten-Fluss-Problem der Form

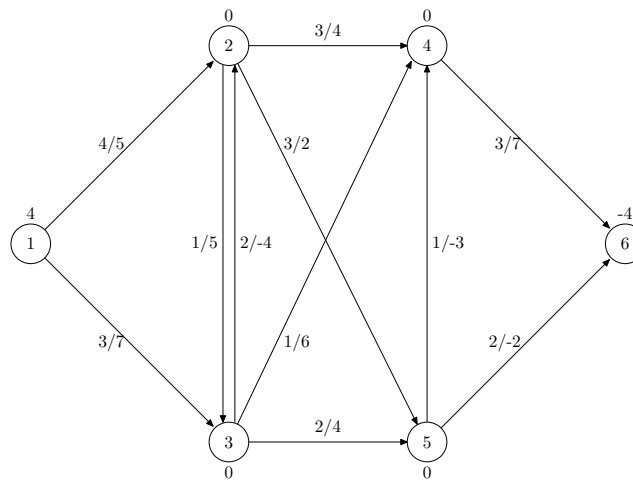
$$\begin{aligned} \min \sum_{a \in A} w_a x_a \\ x(\delta^{\text{out}}(v)) - x(\delta^{\text{in}}(v)) &= b(v) \quad \forall v \in V \\ 0 \leq x_a &\leq c_a \quad \forall a \in A \end{aligned}$$

schreiben. Im zweiten Fall haben wir einen zusätzlichen Angebots- und einen zusätzlichen Bedarfsknoten. Wir haben also anstatt $b(2) = b(5) = 0$ die Zuweisungen $b(5) = 5$ und $b(2) = -5$.

Aufgabe 2. (Kreislöschungsalgorithmus)

(0 Punkte)

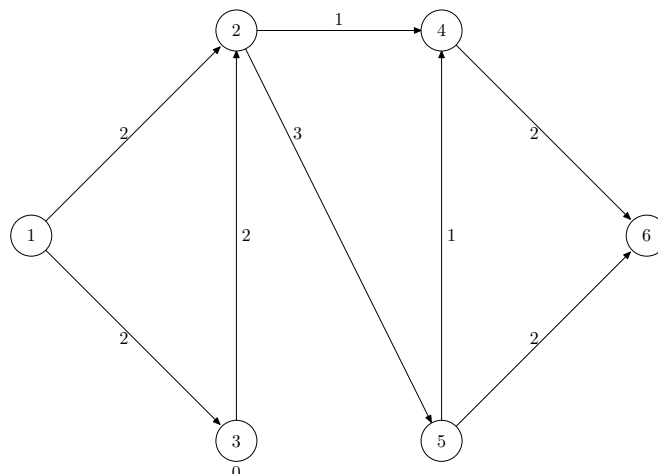
Bestimmen Sie im folgenden Graphen mit dem Kreislöschungsalgorithmus einen kostenminimalen Fluss.



Dabei stehen im Graph der erste Wert an den Bögen für die Kapazitäten des Bogens und der zweite Wert für die Transportkosten pro Flusseinheit. Die Werte an den Knoten geben den Bedarf in diesem Knoten an.

Lösung:

Sei $D = (V, A)$ der gegebene Graph. Zuerst bestimmen wir einen zulässigen Fluss x mithilfe eines Max-Flow-Algorithmus (z.B. Augmentierende Wege-Algorithmus). Ein zulässiger Fluss x ist z.B. $x_{12} = x_{24} = x_{46} = x_{13} = x_{35} = x_{56} = 2$. (Der Bedarf an den Knoten muss eingehalten werden.) Im Residualgraphen suchen wir nun nach negativen Kreisen. Wir wählen z.B. $2 \rightarrow 5 \rightarrow 4 \rightarrow 2$ als negativen Kreis und augmentieren um $\epsilon = 1$. Danach wählen wir $2 \rightarrow 5 \rightarrow 3 \rightarrow 2$ und augmentieren um $\epsilon = 2$. Es gibt keine weiteren negativen Kreise und wir sind fertig. Der Fluss hat Gesamtkosten 33 und ist in folgender Abbildung dargestellt.



Aufgabe 3. (Reduzierung von Angebot und Bedarf)

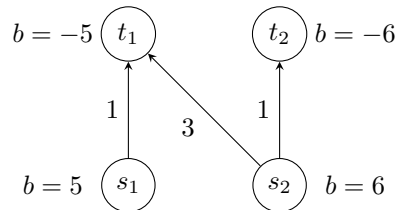
(0 Punkte)

Gegeben sei ein Minimalkosten-Fluss-Problem mit positiven Kosten an den Bögen.

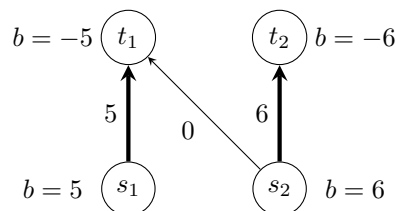
Beweisen oder widerlegen Sie folgende Eigenschaft: Werden das Angebot an einer Quelle und der Bedarf an einer Senke simultan um eine Einheit verringert, dann verringern sich auch die Gesamtkosten des optimalen Minimalkosten-Flusses.

Lösung:

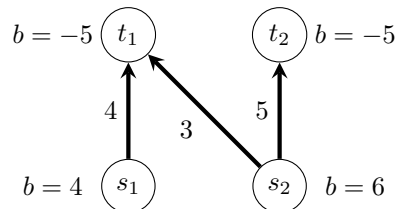
Diese Aussage ist nicht unbedingt immer richtig, wie folgendes Gegenbeispiel zeigt (an den Bögen stehen die Kosten, die Kapazität nehmen wir als unendlich an):



Die optimale Lösung hat Gesamtkosten $5 + 6 = 11$:



Verringert man nun das Angebot bzw. den Bedarf von s_1 und t_2 , dann muss man ein Gut mit Kosten von drei von s_2 nach t_1 schicken. Dann hat die optimale Lösung die Gesamtkosten $4 + 3 + 5 = 12$:

**Aufgabe 4.** (Spezialfälle des Minimalkosten-Fluss-Problems)

(0 Punkte)

- Wie kann man mit einem Algorithmus zur Lösung des Minimalkosten-Fluss-Problems auch das Kürzeste-Wege-Problem zwischen zwei Knoten lösen?
- Wie kann man mit einem Algorithmus zur Lösung des Minimalkosten-Fluss-Problems auch das Maximal-Fluss-Problem zwischen zwei Knoten lösen?

Lösung:

- Setze Kosten $w = c$, Kapazitäten $c = 1$ und Bedarf

$$b(v) = \begin{cases} 1, & \text{falls } v = s \\ -1, & \text{falls } v = t \\ 0, & v = V \setminus \{s, t\}. \end{cases}$$

- ii) Setze $b(v) = 0$ für alle $v \in V$ und $w_a = 0$ für alle $a \in A$. Alle Kapazitäten lassen wir wie gehabt. Dann führen wir einen neuen Bogen $\check{a} = (t, s)$ ein. Für \check{a} setzen wir $w_{\check{a}} = -1$ und $c_{\check{a}} = \infty$.

Hausübungen

Aufgabe 5. (Knotenkapazitäten)

(2 Punkte)

Gegeben sei ein Minimalkosten-Fluss-Problem

$$\begin{aligned} \min_x \quad & \sum_{a \in A} w_a x_a \\ \text{s.t.} \quad & x(\delta^{\text{out}}(v)) - x(\delta^{\text{in}}(v)) = b(v) \quad \text{für alle } v \in V, \\ & 0 \leq x_a \leq c_a \quad \text{für alle } a \in A \end{aligned}$$

mit $b \in \mathbb{R}^{|V|}$ so dass $b(v) = 0$ für alle $v \in V \setminus \{s, t\}$ und $b(s) > 0$ sowie $b(t) = -b(s)$.

Für alle Knoten $v \in V \setminus \{s, t\}$ seien zusätzliche Knotenkapazitäten gegeben, die angeben, wie viel Fluss maximal durch sie fließen darf.

Reduzieren Sie dieses Flussproblem auf das Minimalkosten-Fluss-Problem ohne Knotenkapazitäten. Das heißt, zeigen Sie, dass man jede dieser speziellen Instanzen in eine allgemeine Instanz umformen kann.

Lösung:

Sei v ein Knoten mit einer Knotenkapazität c . Wir löschen nun v und führen an seiner Stelle zwei neue Knoten v^{in} und v^{out} ein. Für alle Bögen aus $\delta^{\text{in}}(v)$ führen wir neue Bögen zu v^{in} mit gleicher Kapazität und Kosten ein; analog hierzu verfahren wir mit Bögen aus $\delta^{\text{out}}(v)$ und v^{out} . Nun verbinden wir v^{in} und v^{out} durch einen Bogen von v^{in} nach v^{out} mit Kosten 0 und Kapazität c . So haben wir eine äquivalente Formulierung für unser Netzwerk gefunden, ohne Knotenkapazitäten zu verwenden.

Aufgabe 6. (Modellierung)

(4 Punkte)

Eine Firma muss die folgenden Rechnungen am Ende der jeweiligen sechs Monate begleichen:

Monat	1	2	3	4	5	6
Rechnungsbetrag in Euro	200	100	50	80	160	140

Am Anfang des ersten Monats hat die Firma 150 Euro in bar und A-Bonds (verzinsliches Wertpapier) im Wert von 200 Euro, B-Bonds im Wert von 100 Euro und C-Bonds im Wert von 400 Euro.

Offensichtlich muss die Firma einige Bonds verkaufen, um die Rechnungen zu bezahlen. Der Verkauf von Bonds vor Ende der sechs Monaten verursacht allerdings Kosten (z.B. durch Wertverlust). Die Kosten in Euro für den Verkauf von Bondsanteilen im Wert von 1 Euro sind in folgender Tabelle dargestellt:

Monat	1	2	3	4	5	6
Bond A	0.7	0.4	0.4	0.3	0.2	0.1
Bond B	0.8	0.6	0.4	0.2	0	0
Bond C	1	1	1	0.5	0.5	0.5

Alle Rechnung müssen rechtzeitig beglichen werden, die Kosten für den Verkauf der Bondsanteile müssen allerdings erst in Zukunft (also nach den sechs Monaten) bezahlt werden. Formulieren Sie ein Min-Cost-Flow Problem, welches die Kosten für den Verkauf der Bondsanteile minimiert.

Lösung:

Um das Problem zu modellieren führen wir vier Quellknoten (bar, A-Bond, B-Bond, C-Bond) mit Angebot von 150, 200, 100 bzw. 400 Euro ein. Für jeden Monat führen wir einen Senkeknoten mit Bedarf gleich der jeweilige Rechnungsbetrag ein. Nun verbinden wir den bar-Knoten mit allen Monaten, diese Kanten haben keine Kosten. Dann verbinden wir die Bonds jeweils mit allen Monaten, diese Kanten haben die Kosten laut der Tabelle. Da das Angebot größer als die Nachfrage ist, d.h. da wir am Ende nicht alle Bonds verkaufen möchten, führen wir noch einen Dummy-Knoten ein. Dieser hat als Bedarf die

Differenz von bisherigem Angebot und Nachfrage, also $850 - 730 = 120$ (d.h. am Ende haben wir noch Geld bzw. Bonds im Wert von 120 Euro). Wir verbinden alle Quellknoten mit Kosten Null mit diesem Dummyknoten. Nicht verkaufte Bonds werden dadurch diesem Dummy-Knoten zugeführt.

Die optimale Lösung des Warenumschlagsproblems auf diesem Graphen ergibt einen Verkaufsplan, der die Kosten für die Bonds-Verkäufe minimiert.

Bemerkung: Im Prinzip können natürlich, um eine Rechnung im Monat i zu begleichen, auch Bondsanteile in den vorherigen Monaten verkauft werden. Dafür kann man einfach zusätzliche Kanten einführen. Da die Kosten aber in diesem Beispiel Monat für Monat anfallen, macht es keinen Sinn, Bonds früher zu verkaufen.

Aufgabe 7. (b -transshipment)

(4 Punkte)

Beweisen Sie folgenden Satz.

Satz 0.1. Es existiert ein b -transshipment x mit $0 \leq x \leq c$ genau dann, wenn

1. $\sum_{v \in V} b(v) = 0$,
2. $0 \leq c$ und
3. $\sum_{a \in \delta^{\text{out}}(U)} c_a \geq \sum_{v \in U} b(v)$ für alle $U \subseteq V$.

Darüber hinaus gilt, dass x ganzzahlig gewählt werden kann, wenn c und b ganzzahlig sind.

Lösung:

Es ist klar, dass die beiden ersten Bedingungen erfüllt sein müssen, damit es ein b -transshipment gibt. Nehmen wir also an, dass diese beiden Bedingungen gelten. Jetzt konstruieren wir einen Hilfsgraphen $G' = (V', A')$ mit Kapazitäten c wie folgt: Setze $V' = V \cup \{s, t\}$, wobei s und t neue Knoten sind. Setze nun

$$A' = A \cup \{(s, v) : b(v) > 0\} \cup \{(v, t) : b(v) < 0\}$$

und

$$c_a = \begin{cases} c_a, & \text{falls } a \in A, \\ b(v), & \text{falls } a = (s, v), \\ -b(v), & \text{falls } a = (v, t). \end{cases}$$

Sei jetzt

$$B := \sum_{v \in V : b(v) > 0} b(v).$$

Suche nun einen maximalen s - t -Fluss x' in G' mit Kapazitäten c . Wenn der Wert von x' gerade B ist, kann man nachrechnen, dass x' eingeschränkt auf die Kanten aus A ein b -transshipment ist. Wenn b und c ganzzahlig sind, kann x' ganzzahlig gewählt werden, somit auch die Einschränkung auf A . Wenn x' nicht existiert, muss es nach dem Max-Flow-Min-Cut-Satz eine Knotenmenge U' geben, die einen s - t -Schnitt einer Kapazität B' mit $B' < B$ induziert, wobei $s \in U'$ und $t \in V' \setminus U'$ gilt. Betrachte nun $U = U' \setminus \{s\}$. Nun gilt

$$\begin{aligned} 0 &> B' - B \\ &= \sum_{a \in \delta^{\text{out}}(U')} c_a - B \\ &= \sum_{a \in \delta^{\text{out}}(U)} c_a + \sum_{v \notin U : (s, v) \in A'} b(v) - \sum_{v \in U : (v, t) \in A'} b(v) - B \\ &= \sum_{a \in \delta^{\text{out}}(U)} c_a - \left(B - \sum_{v \notin U : (s, v) \in A'} b(v) \right) - \sum_{v \in U : (v, t) \in A'} b(v) \\ &= \sum_{a \in \delta^{\text{out}}(U)} c_a - \sum_{v \in U : (s, v) \in A'} b(v) - \sum_{v \in U : (v, t) \in A'} b(v) \\ &= \sum_{a \in \delta^{\text{out}}(U)} c_a - \sum_{v \in U} b(v). \end{aligned}$$

Somit existiert ein $U \subseteq V$, so dass Bedingung 3 verletzt ist, was zu zeigen war.

Aufgabe 8. (Transportproblem)

(2+2 Punkte)

Eine Firma hat den Auftrag, ihre Produkte p_1, p_2 und p_3 wöchentlich von den Produktionsstätten a_1, a_2 und a_3 zu den Orten b_1 und b_2 zu transportieren. Dabei werden an jedem Ort b_k jede Woche mindestens d_{ik} Einheiten des Produkts p_i benötigt ($i \in \{1, 2, 3\}, k \in \{1, 2\}$). An jeder Produktionsstätte a_j können pro Woche jedoch höchstens c_{ij} Einheiten des Produkts p_i hergestellt werden. Insgesamt können jede Woche an a_1 maximal 2500 Produkte, an a_2 höchstens 2000 und an a_3 maximal 500 Produkte hergestellt werden. Die Transportkosten einer Einheit des Produkts p_i von a_j nach b_k betragen t_{ijk} . Gesucht ist ein kostenminimaler Herstellungs- und Transportplan, der die Nachfrage bei b_1 und b_2 befriedigt.

Produkt	Produktionsstätte	Produktion	Transportkosten t_{ijk}		Bedarf d_{ik}	
			b_1	b_2	b_1	b_2
p_1	a_1	500	3	4	700	500
	a_2	1000	7	5		
	a_3	0	0	0		
p_2	a_1	1200	6	9	500	900
	a_2	0	0	0		
	a_3	400	5	5		
p_3	a_1	600	0	6	0	1600
	a_2	1000	0	6		
	a_3	200	0	3		

1. Formulieren Sie dieses Problem als Minimalkosten- (s, t) -Fluss-Problem.
2. Lösen Sie das Minimalkosten- (s, t) -Fluss-Problem mit CATBox unter Verwendung des Kreis-Löschungs-Algorithmus.¹ Geben Sie nach jeder Iteration den augmentierten negativen Kreis und am Ende die minimalen Transportkosten mit dem zugehörigen kostenminimalen (s, t) -Fluss an.

Lösung:

1. Seien $I := \{1, 2, 3\}, J := \{1, 2, 3\}, K := \{1, 2\}$. Man könnte das Problem wie folgt modellieren: Wir führen wie in Abbildung 1 einen Startknoten s mit Bedarf 4200 ein und von dort aus Kanten zu einzelnen Produktionsstätten a_j ($j \in J$), deren Kapazität der maximalen Produktion an a_j entspricht. Von jedem a_j führen wir jeweils eine Kante mit Kapazität c_{ij} zu den Knoten $p_{i,j}$, der die Produktion von Produkt p_i ($i \in I$) an der Produktionsstätte a_j repräsentiert, ein, vorausgesetzt p_i kann an a_j produziert werden. Am anderen Ende des Graphen haben wir die Knoten b_k ($k \in K$), die für die Nachfrageorte b_k stehen, und von dort aus jeweils eine Kante mit Kapazität M (mit einem $M \in \mathbb{Z}$ groß genug - in diesem Fall $M \geq 3000$) zum Endknoten t , welcher einen Bedarf von -4200 bekommt. Wir benötigen noch Knoten $b_{i,k}$, die die Nachfrage von p_i an b_k repräsentieren, und Kanten von $b_{i,k}$ nach b_k ($\forall i \in I, k \in K$) mit Kantenkapazität d_{ik} . Zuletzt führen wir Kanten $(p_{i,j}, b_{i,k})$ mit Kapazität M und Kosten t_{ijk} ein ($\forall i \in I, j \in J, k \in K$). Die Kosten aller anderen Kanten betragen 0.

Der folgende Graph zeigt eine mögliche Modellierung, wobei die erste Zahl an den Kanten die Kantenkapazität und die zweite die Kosten angibt:

¹Sie können bei der Erstellung des Graphen mit `gred` unter **Graph►Vertex/Edge Weights** Gewichte zu den Knoten bzw. ein zusätzliches Gewicht zu den Kanten hinzufügen. Geben Sie anschließend bei jeder Kante als erstes Gewicht die Kapazität und als zweites die Kosten an.

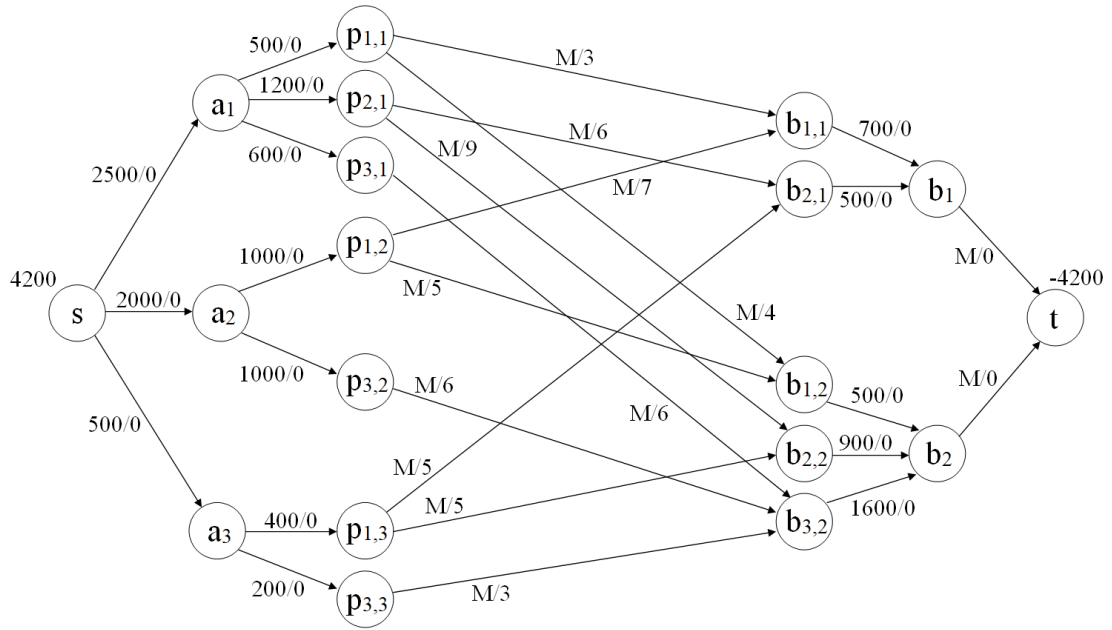
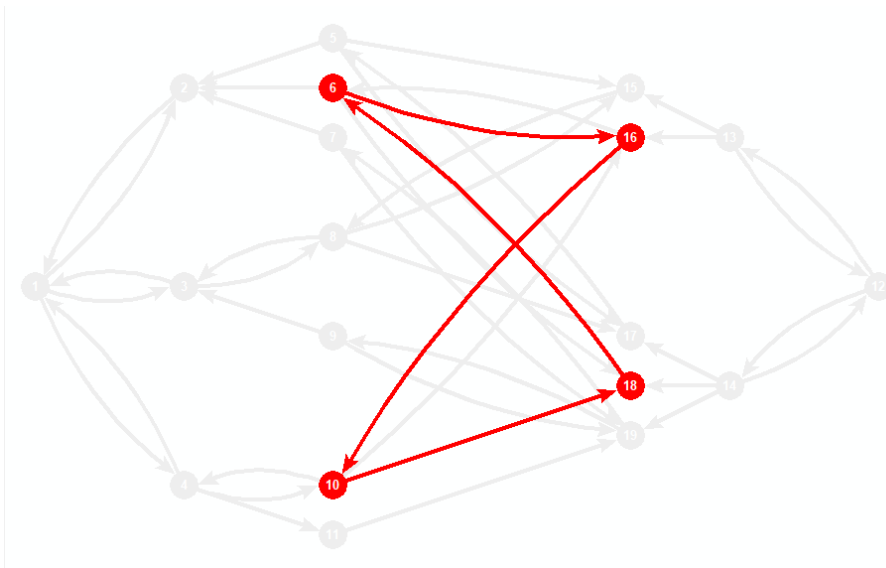


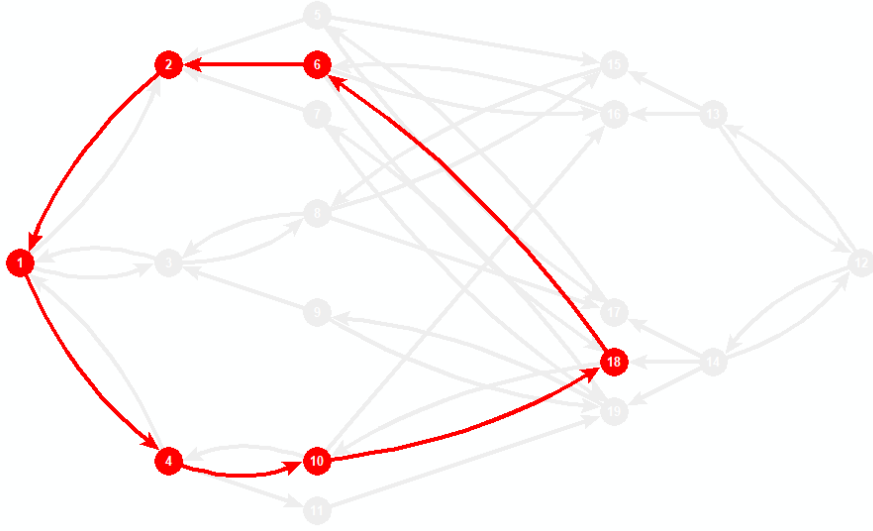
Abbildung 1: Mögliche Modellierung als Minimalkosten- (s, t) -Fluss-Problem

2. Der Kreis-Löschungs-Algorithmus hat die folgenden negativen Kreise augmentiert (C_i in der i -ten Iteration):

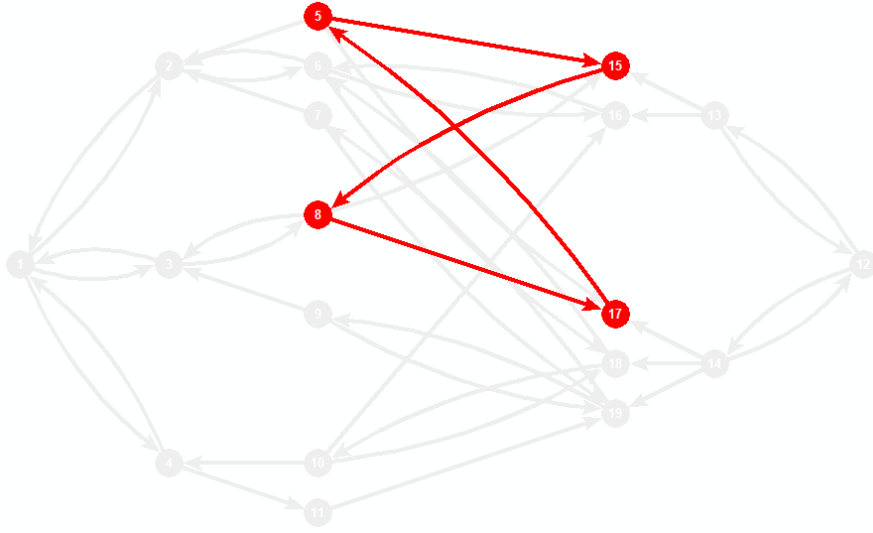
(a) $C_1 = ((p_{2,1}, b_{2,1}), (b_{2,1}, p_{2,3}), (p_{2,3}, b_{2,2}), (b_{2,2}, p_{2,1}))$



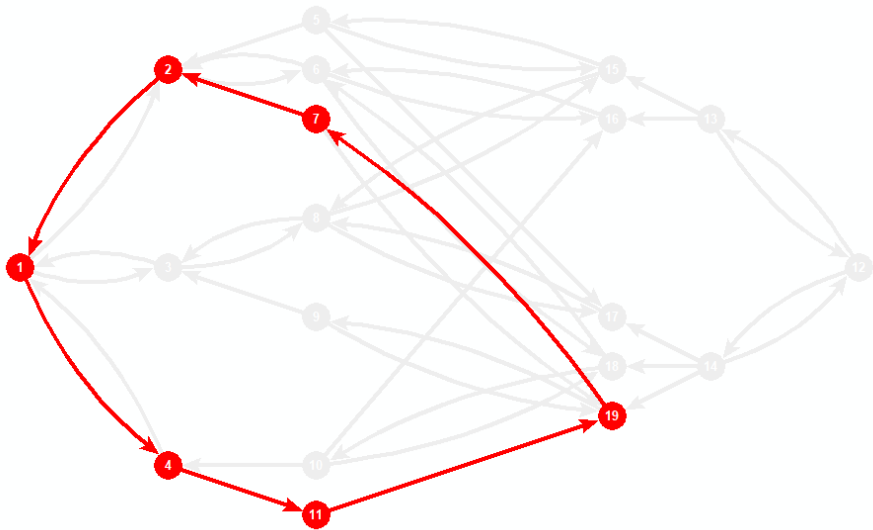
(b) $C_2 = ((s, a_3), (a_3, p_{2,3}), (p_{2,3}, b_{2,2}), (b_{2,2}, p_{2,1}), (p_{2,1}, a_1), (a_1, s))$



(c) $C_3 = ((p_{1,1}, b_{1,1}), (b_{1,1}, p_{1,2}), (p_{1,2}, b_{1,2}), (b_{1,2}, p_{1,1}))$

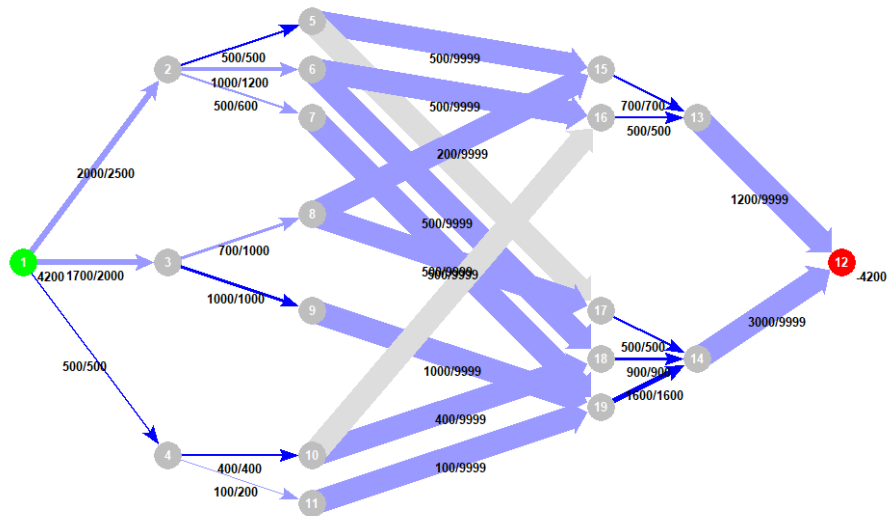


(d) $C_4 = ((s, a_3), (a_3, p_{3,3}), (p_{3,3}, b_{3,2}), (b_{3,2}, p_{3,1}), (p_{3,1}, a_1), (a_1, s))$



In der folgenden Abbildung ist der kostenminimale (s, t) -Fluss mit Gesamtkosten

$500 \cdot 3 + 0 \cdot 4 + 500 \cdot 6 + 500 \cdot 9 + 500 \cdot 6 + 200 \cdot 7 + 500 \cdot 5 + 1000 \cdot 6 + 0 \cdot 5 + 400 \cdot 5 + 100 \cdot 3 = 24200$ dargestellt:



Aufgabe 9. (Programmieraufgabe: Negative Kreise)

(4 Punkte)

Laden Sie die Datei `ProgAufgabe08.py` aus Studon herunter und fügen Sie sie in ihr PyCharm ein.

In dieser Aufgabe implementieren Sie die Methode `find_negative_cycle(G, attribute_name)`, die (falls vorhanden) einen negativen Kreis im Graph G findet und zurückgibt. Die Methode soll folgende Signatur erfüllen:

- G ist ein gerichteter `networkx`-Graph, der nicht verändert werden darf.
- `attribute_name` ist der Name des Kantenattributs, das als Kantengewicht verwendet werden soll.
- Existiert ein negativer Kreis, so ist der Rückgabewert eine Liste von Knoten $[v_0, \dots, v_k]$, die einen negativen Kreis bilden, das heißt $v_0 = v_k$, für alle $i = 1 \dots k$ existiert der Bogen (v_{i-1}, v_i) und die Summe der Kosten dieser Bögen ist echt negativ.
- Existiert kein negativer Kreis, gibt die Methode `None` zurück.

Eine Vorgehensweise wäre die Folgende:

1. Wenden sie den Bellman-Ford-Algorithmus (siehe <https://de.wikipedia.org/wiki/Bellman-Ford-Algorithmus>) auf einen beliebigen Knoten einer Zusammenhangskomponente an, um die Existenz eines negativen Kreis innerhalb dieser Zusammenhangskomponente festzustellen.
2. Existiert ein negativer Kreis, so verwenden sie die vom Bellman-Ford-Algorithmus berechneten Vorgänger um einen negativen Kreis zu berechnen. Ansonsten wenden sie Schritt 1 auf die nächste Zusammenhangskomponente an.

Nutzen sie den beigefügten Testcode, um ihre Methode ausführlich zu testen.

Lösung:

Die Musterlösung der Programmieraufgabe finden Sie im StudOn in der Datei `ProgAufgabe08_Loesung.py`.