8

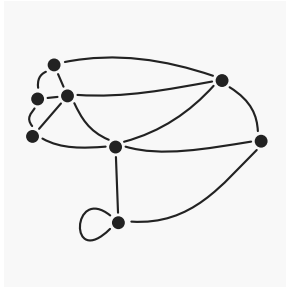
Netzwerkoptimierung

In der Netzwerkoptimierung werden graphentheoretische Probleme behandelt, welche häufig einfach zu beschreiben sind, jedoch äußerst schwierig zu lösen. Teilweise kommen hier bekannte Methoden etwa der ganzzahligen Optimierung zum Einsatz, oder aber es werden spezielle Verfahren angewandt. Zudem besteht aus didaktischer Sicht eine Herausforderung in der großen Anzahl an Definitionen und Bezeichnungen: Die Ideen und Verfahren sind zum Teil vergleichsweise simpel, allerdings aufgrund der ganzen Bezeichnungen und Schreibweisen schwer zu verstehen. Wir untersuchen in diesem Kapitel daher nur eine kleine Auswahl an graphentheoretischen Problemen, um die Anzahl der Definitionen und Notationen auf ein Minimum zu reduzieren, aber dennoch einen guten Einblick in die Grundlagen der Netzwerkoptimierung geben zu können. Wir beginnen in Abschn. 8.1 mit den wichtigsten Definitionen im Zusammenhang mit gerichteten sowie ungerichteten Graphen. Darauf aufbauend werden in den folgenden Abschnitten vier ausgewählte Probleme behandelt: Wir untersuchen spannende Bäume und lernen ein Verfahren kennen, wie diese effizient bestimmt werden können (Abschn. 8.2). Anschließend definieren wir Wege in gerichteten Graphen und stellen ein Verfahren vor, um kürzeste Wege zu bestimmen (Abschn. 8.3). Darauf aufbauend führen wir Flüsse ein, wobei das Problem darin besteht, maximale Flüsse unter Berücksichtigung von Kapazitätsschranken zu finden (Abschn. 8.4). Schließlich formulieren wir das Knotenfärbungsproblem, welches im Allgemeinen äußerst schwer zu lösen ist (Abschn. 8.5). Für spezielle Klassen von Graphen gibt es jedoch sehr effiziente Algorithmen. Genauer definieren wir chordale Graphen und zeigen, wie das Knotenfärbungsproblem dank eines perfekten Eliminationsschemas chordaler Graphen effizient gelöst werden kann.

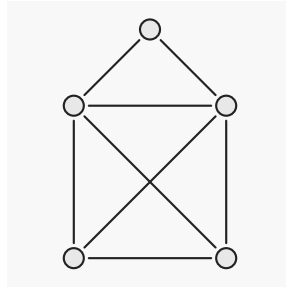
8.1 Grundlagen

Bevor wir in den folgenden Abschnitten zentrale Probleme der Netzwerkoptimierung formulieren, fassen wir zunächst eine ganze Reihe grundlegender Begriffe und Konzepte der Graphentheorie zusammen. Wir orientieren uns dabei teilweise an

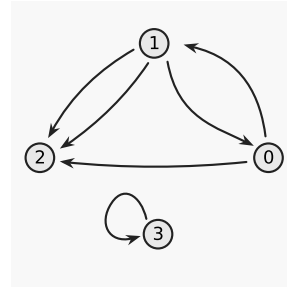
D. Scholz, *Optimierung interaktiv*, https://doi.org/10.1007/978-3-662-57953-4_8



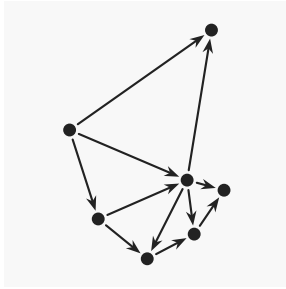
a Ungerichteter Graph



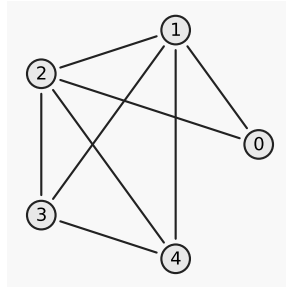
b Ungerichteter Graph



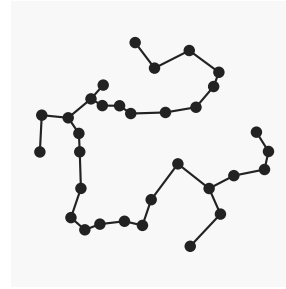
c Gerichteter Graph



d Gerichteter Graph



e Ungerichteter Graph



f Ungerichteter Graph

Abb. 8.1 Beispiele zur Visualisierung einiger Graphen mit unterschiedlichen Eigenschaften

den Notationen aus Krumke und Noltemeier (2005) sowie Domschke et al. (2015). Darin können auch die Anwendungen der folgenden Abschnitte nachgeschlagen und vertieft werden.

Definition 8.1 *Ein* ***Graph*** $G = (V, E)$ *besteht aus einer (endlichen) Menge* V *an* ***Knoten*** *sowie einer (endlichen) Menge* E *an* ***Kanten****.*

Wir unterscheiden zwischen ***gerichteten*** *sowie* ***ungerichteten*** *Graphen. In gerichteten Graphen gelte* $e = (u, v)$ *für alle* $e \in E$ *mit* $u, v \in V$ *und in ungerichteten Graphen gelte* $e = \{u, v\}$ *für alle* $e \in E$ *mit* $u, v \in V$*.*

Im Folgenden werden wir gerichtete Graphen auch als ***Digraphen*** *bezeichnen.*

Neben der formalen Definition eines Graphen lassen sich diese auch durch Skizzen beschreiben. Einige Beispiele dazu können Abb. 8.1 entnommen werden, wobei die Kanten in gerichteten Graphen in der Regel als ***Pfeile*** dargestellt werden. Es muss jedoch stets bedacht werden, dass es sehr unterschiedliche Darstellungen von ein und demselben Graphen geben kann. So sind beispielsweise die Graphen aus

Abb. 8.1b und e identisch. Die Visualisierung von Graphen ist ein große Herausforderung an sich, auf die wir hier aber nicht weiter eingehen wollen.

Aufgabe 8.1 Finde eine Nummerierung der Knoten des Graphen aus Abb. 8.1b, sodass deutlich wird, dass die beiden Graphen aus Abb. 8.1b und e identisch sind.

In der Regel werden wir mit n die Anzahl der Knoten und mit m die Anzahl der Kanten beschreiben, d.h. $n = |V|$ und $m = |E|$. Zudem werden wir die Knoten eines Graphen $G = (V, E)$ häufig mit $v_1, \ldots, v_n$ bezeichnen, sodass ein Graph einzig und allein durch die Menge E der Kanten definiert wird (wobei weiterhin spezifiziert werden muss, ob es sich um einen gerichteten oder einen ungerichteten Graphen handelt).

Notation 8.2 *Sei $G = (V, E)$ ein gerichteter Graph. Eine Kante $e = (u, v) \in E$ heißt **Schlinge**, falls $u = v$ gilt. Zwei Kanten $e_1 = (u_1, v_1)$ und $e_2 = (u_2, v_2)$ heißen **parallel**, falls $u_1 = u_2$ und $v_1 = v_2$ gilt.*

*Sei $G = (V, E)$ ein ungerichteter Graph. Eine Kante $e = \{u, v\} \in E$ heißt **Schlinge**, falls $u = v$ gilt. Zwei Kanten $e_1 = \{u_1, v_1\}$ und $e_2 = \{u_2, v_2\}$ heißen **parallel**, falls $e_1 = e_2$ gilt (wobei e_1 und e_2 als Mengen aufgefasst werden).*

Am geeignetsten lässt sich die Notation anhand von Beispielen verstehen:

Aufgabe 8.2 Welche der Graphen aus Abb. 8.1 besitzen Schlingen? Und welche besitzen parallele Kanten?

In vielen Anwendungsfällen treten weder Schlingen noch parallele Kanten auf, sodass wir folgende Definition einführen:

Definition 8.3 *Ein Graph $G = (V, E)$ heißt **einfach**, falls G weder Schlingen noch parallele Kanten besitzt.*

Um eine weitere Eigenschaft von Graphen zu definieren, benötigen wir zunächst den Begriff von benachbarten Knoten:

Definition 8.4 *Sei $G = (V, E)$ ein Graph. Zwei Knoten $u, v \in V$ heißen **benachbart**, falls es eine Kante $e \in E$ zwischen u und v gibt. Weiter sei*

$$c(v) = \{u \in V : u \text{ und } v \text{ sind benachbart}\} \subset V \tag{8.1}$$

die Menge aller Knoten $u \in V$, welche zu v benachbart sind.

Falls G ein gerichteter Graph ist, spielt die Richtung der Kante keine Rollte, d.h., u und v sind benachbart, falls es ein $e \in E$ mit $e = (u, v)$ oder mit $e = (v, u)$ gibt. Damit erhalten wir folgende Definition, welche anschaulich direkt verständlich sein sollte:

Definition 8.5 *Ein Graph $G = (V, E)$ heißt **zusammenhängend**, falls zwischen zwei beliebigen Knoten $u, v \in V$ eine Folge benachbarter Knoten von u nach v existiert.*

Auch hier lässt sich die Definition anhand einfacher Beispiele verdeutlichen:

Aufgabe 8.3 Finde ein Beispiel eines Graphen $G = (V, E)$ mit $c(v) \neq \emptyset$ für alle $v \in V$, der nicht zusammenhängend ist.

Neben einem Graphen bestehend aus Knoten und Kanten werden wir im Folgenden auch häufig eine Gewichtung der Kanten benötigen:

Definition 8.6 *Sei $G = (V, E)$ ein Graph. Eine zugehörige **Gewichtung** wird gegeben durch eine Funktion $w : E \to \mathbb{R}$, welche jeder Kante ein Gewicht zuordnet.*

Für einen gewichteten Graphen verwenden wir die Notation $G = (V, E, w)$.

Damit haben wir bereits die wesentlichen Grundbegriffe zur Definition von Graphen zusammengefasst. Wir werden in den folgenden Abschnitten jedoch eine ganze Reihe von weiteren Begriffen einführen, wobei alle folgenden Definitionen teilweise nur im jeweiligen Abschnitt benötigt werden. Mit anderen Worten können einige der folgenden Abschnitte je nach Interesse auch in einer beliebigen Reihenfolge studiert werden.

8.2 Spannende Bäume

In diesem sowie in den folgenden Abschnitten stellen wir unterschiedliche Probleme der Netzwerkoptimierung vor, welche jeweils einen Graphen zugrunde legen. Zunächst beschäftigen wir uns mit einer speziellen Klassen von Graphen:

Definition 8.7 *Ein ungerichteter, einfacher und zusammenhängender Graph $G = (V, E)$ heißt **Baum**, falls G genau*

$$m = n - 1$$

Kanten besitzt. Dabei ist n die Anzahl der Knoten von G.

Ohne es formal im Detail zu definieren, ist ein Baum G stets ***kreisfrei***, d.h., es gibt keine Folge benachbarter Knoten, die einen Kreis bilden.

Aufgabe 8.4 Sei $G = (V, E)$ ein ungerichteter, einfacher und zusammenhängender Graph. Wie viele Kanten besitzt G unter diesen Voraussetzungen mindestens?

Das Ziel, welches wir in diesem Abschnitt verfolgen, ist, einen (minimal) spannenden Baum eines gewichteten Graphen zu finden:

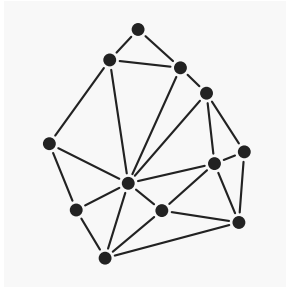
Definition 8.8 *Sei $G = (V, E, w)$ ein ungerichteter, einfacher, zusammenhängender und gewichteter Graph. Ein **spannender Baum** von G ist ein Baum $T = (V, F, w)$ mit $F \subset E$.*

Genauer besteht die Aufgabe im Folgenden darin, einen spannenden Baum T derart zu wählen, dass die Summe der Gewichte des spannenden Baumes, d.h.

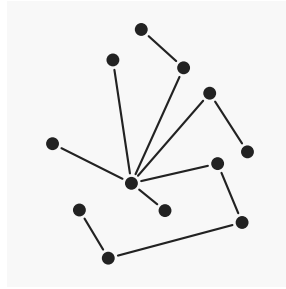
$$\sum_{e \in F} w(e), \tag{8.2}$$

minimiert wird (Abb. 8.2).

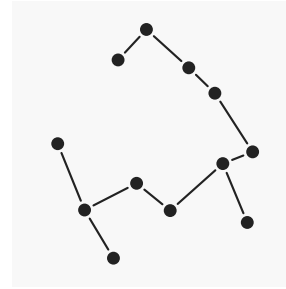
Beispiel 8.1 *Angenommen, es müssen n Orte über ein Versorgungsnetz miteinander verbunden werden. Dabei steht eine endliche Anzahl an Verbindungsstrecken zum Bau des Versorgungsnetzes zur Verfügung, und es sind die Baukosten der einzelnen Abschnitte zwischen zwei Orten des Netzes bekannt. Um die Gesamtbaukosten zu minimieren, kann dieses Problem als Graph modelliert werden, sodass anschließend ein minimal spannender Baum zu bestimmen ist.*



a Gegebener Graph



b Spannender Baum



c Minimal spannender Baum

Abb. 8.2 Zum gegebenen Graphen aus **a** veranschaulicht **b** einen (beliebigen) spannenden Baum. Hingegen zeigt **c** einen minimal spannenden Baum, wobei die Kantengewichte proportional zur Länge der Kanten in der Darstellung aus **a** gewählt wurden

Neben praktischen Anwendungen wie im Beispiel werden spannende Bäume häufig im Rahmen von Heuristiken verwendet, da sich spannende Bäume sehr einfach und effizient finden lassen. Wir fassen ein mögliches Verfahren direkt zusammen, s. Kruskal (1956), wobei dank der Voraussetzungen stets eine Optimallösung existiert:

#

Zusammenfassung 8.1 (Kruskal) Gegeben sei ein ungerichteter, einfacher, zusammenhängender und gewichteter Graph $G = (V, E, w)$.

Setze $F = \emptyset$ und führe für $k = 1, \ldots, n-1$ folgende Schritte aus: Wähle eine Kante e aus $E \setminus F$ mit kleinstem Gewicht, sodass e zusammen mit den bereits gewählten Kanten F keinen Kreis bildet. Füge e zur Kantenmenge F hinzu.

Das Verfahren terminiert mit einem minimal spannenden Baum $T = (V, F, w)$.

Obwohl sich der Algorithmus denkbar einfach beschreiben lässt, kann eine effiziente Implementierung durchaus zur kleinen Herausforderung werden. Denn nur dank geeigneter Datenstrukturen lässt sich effizient prüfen, ob die Teilgraphen zur Kantenmenge F während des Algorithmus kreisfrei sind oder nicht. Dennoch besitzt der Algorithmus von Kruskal bei geeigneter Implementierung eine Komplexität von

$$\mathcal{O}\Big(m \cdot \log(m)\Big), \tag{8.3}$$

d.h., die Komplexität wird durch das Sortieren der Kantengewichte bestimmt.

Zudem lassen sich anhand des Findens eines minimal spannenden Baumes viele weiterführende Konzepte der Graphentheorie erläutern. Wir geben daher einen Ausblick, wie das Problem auch als ganzzahliges (lineares) Programm formuliert werden kann:

Ausblick *Gegeben sei ein ungerichteter, einfacher, zusammenhängender und gewichteter Graph $G = (V, E, w)$ mit Knoten $v_1, \ldots, v_n$ und Kanten $e_1, \ldots, e_m$.*

Um das Finden eines minimal spannenden Baumes als ganzzahliges Programm zu formulieren, definieren wir $x = (x_1, \ldots, x_m) \in \mathbb{R}^m$ folgendermaßen:

$$x_j \;=\; \begin{cases} 1 & \text{falls die Kante } e_j \text{ gewählt wird} \\ 0 & \text{falls die Kante } e_j \text{ nicht gewählt wird} \end{cases} .$$

Zudem definieren wir eine Matrix $A = (a_{i,j}) \in \mathbb{R}^{n \times m}$ mit

$$a_{i,j} \;=\; \begin{cases} 1 & \text{falls die Kante } e_j \text{ den Knoten } v_i \text{ enthält} \\ 0 & \text{sonst} \end{cases}$$

sowie den Vektor $b = (1, \ldots, 1) \in \mathbb{R}^n$. Es sei bemerkt, dass jede Spalte der Matrix A einer Kante des Graphen entspricht und daher genau zwei 1-Einträge besitzt.

Damit erhalten wir das folgende Programm:

$$\min \sum_{j=1}^{m} w(e_j) \cdot x_j$$

unter den Nebenbedingungen

$$\sum_{j=1}^{m} x_j \;=\; n-1 \qquad \text{und} \qquad A \cdot x \;\geq\; b$$

sowie mit $x \in \{0,1\}^m$. Dabei ist das $\geq$ bezogen auf Vektoren komponentenweise zu verstehen.

Schließlich sei bemerkt, dass die im Ausblick definierte $(n \times m)$-Matrix A der ***Inzidenzmatrix*** des Graphen G entspricht.

8.3 Kürzeste Wege

In diesem Abschnitt untersuchen wir zwecks einer leichter verständlichen Darstellung ausschließlich gerichtete Graphen, obwohl fast alle Erkenntnisse recht einfach auch auf ungerichtete Graphen übertragen werden können. Zudem werden wir auch an anderen Stellen nicht den allgemeinsten Fall betrachten, um unübersichtliche Fallunterscheidungen zu vermeiden.

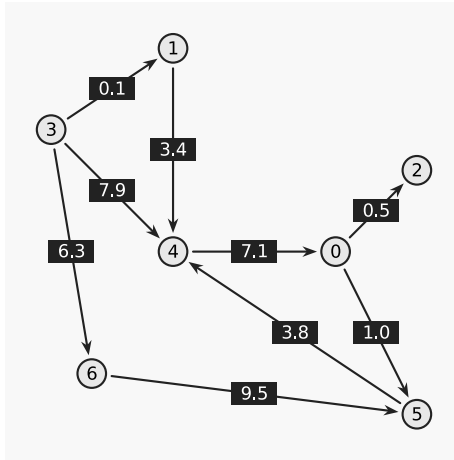


Abb. 8.3 Beispiel eines Graphen zur Berechnung aller Wege von $s = 3$ nach $t = 5$. Dargestellt sind auch die zugehörigen Kantengewichte

Definition 8.9 *Sei $G = (V, E, w)$ ein gerichteter und gewichteter Graph. Weiter seien $s \in V$ und $t \in V$. Ein* ***Weg***

$$(p_1, \ldots, p_r)$$

von s nach t in G ist eine Folge von Knoten $p_1, \ldots, p_r \in V$ mit $p_1 = s$ sowie mit $p_r = t$. Dabei gelte zudem

$$(p_k, p_{k+1}) \in E$$

für $k = 1, \ldots, r-1$.

Dabei nehmen wir stets an, dass kein Knoten doppelt besucht wird, d.h., es gelte $p_i \neq p_j$ für alle $i, j \in \{1, \ldots, r\}$ mit $i \neq j$.

Genauer untersuchen wir in diesem Abschnitt folgendes Problem: Gegeben sei ein gerichteter und gewichteter Graph $G = (V, E, w)$. Zudem seien zwei Knoten $s \in V$ sowie $t \in V$ gegeben. Die Aufgabe besteht darin, einen Weg $(p_1, \ldots, p_r)$ von $p_1 = s$ nach $p_r = t$ in G derart zu wählen, dass die Summe der Gewichte des Weges, d.h.

$$\sum_{k=1}^{r-1} w\Big((p_k, p_{k+1})\Big),$$

minimiert wird. Dabei ist keineswegs gesagt, dass überhaupt ein Weg von s nach t existiert. Beispielsweise haben wir nicht gefordert, dass G zusammenhängend ist und selbst wenn, kann es sein, dass es keinen (gerichteten) Weg von s nach t gibt.

Aufgabe 8.5 Betrachte den Graphen aus Abb. 8.3. Finde alle Wege von Knoten $s = 3$ nach Knoten $t = 5$. Welcher dieser Wege ist der kürzeste?

Zur weiteren Vereinfachung nehmen wir im Folgenden an, dass $w(e) \geq 0$ für alle $e \in E$ gilt. Das Kürzeste-Wege-Problem lässt sich auch für Graphen mit negativen Kanten definieren, wobei dann einige Fallunterscheidungen zu berücksichtigen sind.

Um schließlich ein Verfahren zum Finden eines kürzesten Weges angeben zu können, verallgemeinern wir Definition 8.4 für gerichtete Graphen:

Definition 8.10 *Sei $G = (V, E)$ ein gerichteter Graph und sei $v \in V$. Dann definieren wir*

$$a(v) \;=\; \{u \in V \;:\; (v, u) \in E\} \;\subset\; V \tag{8.4}$$

als die Menge aller ***Nachfolger*** *von v und*

$$b(v) \;=\; \{u \in V \;:\; (u, v) \in E\} \;\subset\; V \tag{8.5}$$

als die Menge aller ***Vorgänger*** *von v.*

Die Definition lässt sich am besten anhand eines Beispiels verstehen:

Aufgabe 8.6 Betrachte den Graphen aus Abb. 8.3. Finde alle Nachfolger und alle Vorgänger von Knoten 4.

Unter Verwendung der Notation $c(v)$ für benachbarte Knoten (Definition 8.4) erhalten wir zudem folgende Aussage:

Aufgabe 8.7 Sei $G = (V, E)$ ein gerichteter Graph. Weise nach, dass

$$a(v) \cup b(v) \;=\; c(v)$$

für alle $v \in V$ gilt (s. auch Gl. 8.1). Unter welchen Bedingungen gilt

$$a(v) \cap b(v) \;\neq\; \emptyset$$

für mindestens einen Knoten $v \in V$?

Es gibt mehrere Verfahren, welche einen kürzesten Weg innerhalb eines Netzwerks finden. Wir stellen den Algorithmus von Dijkstra vor, s. Dijkstra (1959), welchen wir unter Verwendung der obigen Notationen direkt zusammenfassen können:

Zusammenfassung 8.2 (Dijkstra) Gegeben sei ein gerichteter und gewichteter Graph $G = (V, E, w)$ mit $w(e) \geq 0$ für alle $e \in E$. Gesucht wird ein kürzester Weg von $s \in V$ nach $t \in V$.

(1) Weise allen Knoten $v \in V$ die folgenden drei Eigenschaften zu:

$$\text{Vorgänger}(v) = \text{unbekannt}, \quad \text{besucht}(v) = \text{nein}, \quad \text{Distanz}(v) = \infty.$$

(2) Weise dem Startknoten $s \in V$ eine Distanz von 0 zu, d.h., setze

$$\text{Distanz}(s) \;=\; 0.$$

(3) Wähle unter allen Knoten $v \in V$ mit den Eigenschaften

$$\text{besucht}(v) = \text{nein} \qquad \text{und} \qquad \text{Distanz}(v) < \infty$$

einen Knoten $z \in V$ mit kleinster Distanz aus. Falls es keinen Knoten mit diesen Eigenschaften gibt, beende das Verfahren.

(4) Markiere z als besucht, d.h., setze $\text{besucht}(z) = \text{ja}$.

(5) Für alle $u \in a(z)$: Berechne

$$q(u) \;=\; \text{Distanz}(z) \;+\; w\Big((z,u)\Big),$$

und falls $q(u) < \text{Distanz}(u)$ gilt, setze

$$\text{Distanz}(u) = q(u) \qquad \text{und} \qquad \text{Vorgänger}(u) = z.$$

(6) Gehe zurück zu Schritt **(3)**.

Falls $\text{Vorgänger}(t) = \text{unbekannt}$, so existiert kein Weg von s nach t in G. Anderenfalls liefert die Kette der Vorgänger von t nach s (rückwärts) einen kürzesten Weg von s nach t in G.

Der Algorithmus in der oben angegebenen Form findet nicht nur den kürzesten Weg von s nach t, sondern die kürzesten Wege von s zu allen anderen Knoten. Ist dies gar nicht notwendig, so kann das Verfahren beendet werden, sobald der Knoten t besucht wurde, d.h., falls $\text{besucht}(t) = \text{ja}$.

Aufgabe 8.8 Führe den Algorithmus von Dijkstra anhand des Graphen aus Abb. 8.4 unter Verwendung von $s = 0$ und $t = 4$ durch. Fertige zu jeder Iteration eine Skizze

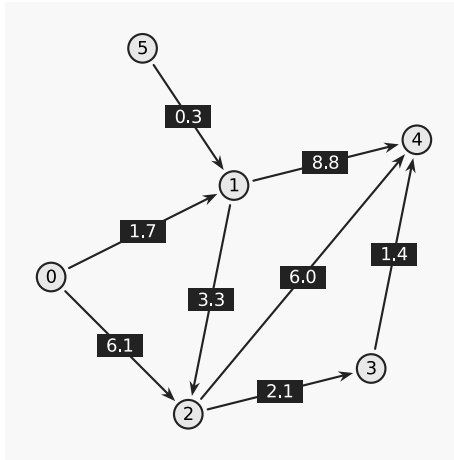


Abb. 8.4 Beispiel eines Graphen zur Berechnung eines kürzesten Weges von $s = 0$ nach $t = 4$

des Graphen in Schritt **(3)** an, welche jeweils die drei Eigenschaften aller Knoten beinhaltet.

Offensichtlich wird im ersten Durchlauf von Schritt **(3)** für z jeweils der Startknoten s gewählt. Zudem ist auch die Endlichkeit des Verfahrens stets gegeben:

Aufgabe 8.9 Begründe, warum der Algorithmus von Dijkstra stets endlich ist.

?

Anwendungen des Kürzesten-Wege-Problems aus diesem Abschnitt sind ziemlich naheliegend: Als anschauliches Beispiel lässt sich die Routenplanung im Straßennetz erwähnen. Dabei wird das Straßennetz als gerichteter Graph modelliert, und die Gewichte definieren beispielsweise die erwartete Reisezeit entlang einer Kante bzw. entlang des entsprechenden Straßenabschnitts.

8.4 Maximale Flüsse

In diesem Abschnitt bauen wir teilweise auf kürzesten Wegen auf und beschäftigen uns mit (maximalen) Flüssen. Dazu führen wir zunächst folgende Definition ein (vgl. auch Definition 8.10 bezüglich Knoten statt Kanten):

Definition 8.11 *Sei $G = (V, E)$ ein gerichteter Graph und sei $v \in V$. Dann definieren wir*

$$A(v) \;=\; \{(v,u) \;:\; u \in V \text{ und } (v,u) \in E\} \;\subset\; E \tag{8.6}$$

*als die Menge aller **Ausgangskanten** von v und*

$$B(v) \;=\; \{(u,v) \;:\; u \in V \text{ und } (u,v) \in E\} \;\subset\; E \tag{8.7}$$

*als die Menge aller **Eingangskanten** von v.*

Aufbauend auf diesen Notationen können wir zunächst den Begriff eines Flusses definieren:

Definition 8.12 *Sei $G = (V, E, w)$ ein gerichteter und gewichteter Graph. Weiter seien $s \in V$ und $t \in V$. Ein **Fluss** von s nach t in G wird gegeben durch eine Abbildung $f : E \to \mathbb{R}$, welche die **Flusserhaltungsbedingungen***

$$\sum_{e \in A(v)} f(e) \;=\; \sum_{e \in B(v)} f(e) \qquad \text{für alle } v \in V \setminus \{s,t\}$$

*erfüllt und für welche $f(e) = 0$ für alle $e \in B(s)$ und alle $e \in A(t)$ gilt. Ein Fluss heißt **zulässig**, falls zusätzlich*

$$f(e) \;\leq\; w(e) \qquad \text{für alle } e \in E$$

gilt.

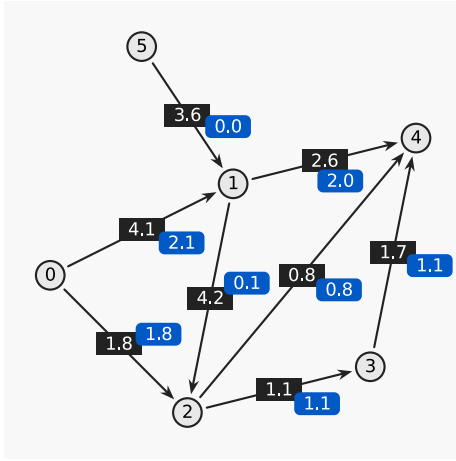
Nach dieser Definition ist ein Fluss f eine Gewichtung der Kanten, sodass mit Ausnahme der ***Quelle*** s sowie der ***Senke*** t in allen Knoten der eingehende Fluss gleich dem ausgehenden Fluss ist. Abb. 8.5a zeigt ein Netzwerk samt zulässigem Fluss. Weiterhin ist die Definition auch nur dann sinnvoll, falls $w(e) \geq 0$ und $f(e) \geq 0$ für alle $e \in E$ gilt.

Eine weitere direkte Folgerung aus der Definition wird als Aufgabe gestellt:

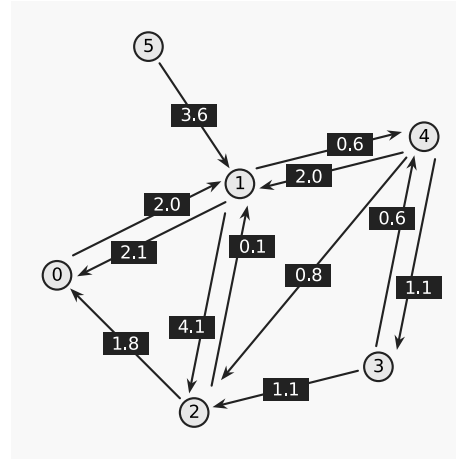
Aufgabe 8.10 Sei $G = (V, E, w)$ ein gerichteter und gewichteter Graph und weiter sei $f : E \to \mathbb{R}$ ein zulässiger Fluss von s nach t in G. Weise nach, dass

$$\sum_{e \in A(s)} f(e) \;=\; \sum_{e \in B(t)} f(e)$$

gilt. Gilt die Aussage auch für unzulässige Flüsse?



a Beispiel eines Graphen samt Fluss



b Zugehöriger Residualgraph

Abb. 8.5 Beispiel eines Graphen zur Veranschaulichung der Definition eines Flusses. **a** zeigt einen Graphen mit sechs Knoten samt Kantengewichtung sowie einen Fluss zur Quelle $s = 0$ und Senke $t = 4$. Der Fluss ist dabei durch die Zahlen in den blauen Rechtecken gekennzeichnet. Der zugehörige Residualgraph kann **b** entnommen werden

Mit den Notationen und Definitionen zuvor können wir das Problem, welches wir in diesem Abschnitt genauer analysieren werden, folgendermaßen beschreiben: Gegeben seien ein gerichteter und gewichteter Graph $G = (V, E, w)$ sowie eine Quelle $s \in V$ und eine Senke $t \in V$. Die Aufgabe besteht darin, einen zulässigen Fluss $f : E \to \mathbb{R}$ derart zu wählen, dass der Fluss maximiert wird. Dies bedeutet, dass der Wert

$$M = \sum_{e \in A(s)} f(e) = \sum_{e \in B(t)} f(e)$$

zu maximieren ist; eine Optimallösung wird als ***maximaler Fluss*** bezeichnet.

Aufgabe 8.11 Handelt es sich bei dem Fluss aus Abb. 8.5a um einen maximalen Fluss? Begründe deine Antwort.

Um ein Verfahren zur Berechnung eines maximalen Flusses präsentieren zu können, benötigen wir noch eine weitere Notation und eine weitere Definition:

Notation 8.13 *Sei $G = (V, E)$ ein gerichteter Graph. Dann bezeichnen wir für alle $e = (u, v) \in E$ die zugehörige **Rückkante** mit $e_b = (v, u)$. Die Menge aller Rückkanten von E bezeichnen wir mit E_b.*

Dank dieser Notation können wir nun folgende zentrale Definition einführen:

Definition 8.14 *Sei $G = (V, E, w)$ ein gerichteter und gewichteter Graph mit $w(e) \geq 0$ für alle $e \in E$. Weiter seien $s, t \in V$ und $f : E \to \mathbb{R}$ sei ein zulässiger Fluss von s nach t in G.*

*Der **Residualgraph** von G und f ist ein gerichteter und gewichteter Graph $G_f(V, F, z)$ mit folgenden Eigenschaften:*

(1) *Die Kantenmenge F besteht aus allen Kanten $e \in E$, für welche $w(e) > 0$ und $f(e) < w(e)$ gilt. Für das zugehörige Gewicht gelte $z(e) = w(e) - f(e)$.*

(2) *Zusätzlich beinhaltet F für alle $e = (u, v) \in E$ mit $f(e) > 0$ die Rückkante $e_b = (v, u)$. Für das zugehörige Gewicht gelte dabei $z(e_b) = f(e)$.*

Die Menge der Knoten von G und G_f ist jeweils identisch.

Die Definition des Residualgraphen ist zwar nicht schwer, lässt sich aber kaum verständlich formulieren. Ein anschauliches Beispiel kann daher Abb. 8.5b entnommen werden. Schließlich soll die folgende Zusammenfassung die Definition nochmals greifbarer machen:

Zusammenfassung 8.3 (Residualgraph) Sei $G = (V, E, w)$ ein gerichteter und gewichteter Graph mit $w(e) \geq 0$ für alle $e \in E$. Weiter seien $s, t \in V$ und $f : E \to \mathbb{R}$ sei ein zulässiger Fluss von s nach t in G.

(1) Definiere den gerichteten und gewichteten Graphen $G_f = (V, F, z)$, wobei $F = E \cup E_b$ gelte. F beinhaltet damit alle Kanten aus $e \in E$ sowie alle Rückkanten $e_b \in E_b$.

Für die zugehörigen Gewichte gelte $z(e) = w(e) - f(e)$ für alle $e \in E$ und $z(e_b) = f(e)$ für alle $e_b \in E_b$.

(2) Entferne alle Kanten $e \in F$ aus dem Graphen G_f, für die $z(e) \leq 0$ gilt.

Nach dieser Vorgehensweise ist G_f der Residualgraph von G und f.

Dank der Definition des Residualgraphen lässt sich schließlich ein Verfahren zur Berechnung eines maximalen Flusses folgendermaßen formulieren, s. Ford und Fulkerson (1956):

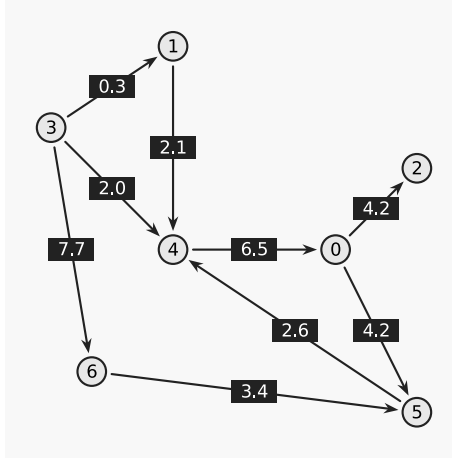


Abb. 8.6 Beispiel eines Graphen zur Berechnung eines maximalen Flusses von $s = 3$ nach $t = 5$

Zusammenfassung 8.4 (Ford und Fulkerson) Gegeben sei ein gerichteter und gewichteter Graph $G = (V, E, w)$ mit $w(e) \geq 0$ für alle $e \in E$. Gesucht wird ein maximaler Fluss von $s \in V$ nach $t \in V$.

(1) Sei $f : E \to \mathbb{R}$ der (zulässige) Fluss von s nach t in $G = (V, E, w)$ mit $f(e) = 0$ für alle $e \in E$.

(2) Bestimme den Residualgraphen $G_f = (V, F, z)$ von G und f.

(3) Finden einen kürzesten Weg von s nach t in $G_f = (V, F, z)$ und definiere E_p als die Menge aller Kanten des kürzesten Weges. Falls kein Weg von s nach t existiert, beende das Verfahren.

(4) Bestimme das minimale Kantengewicht q aller Kanten des zuvor bestimmten Weges E_p, d.h.

$$q \;=\; \min\{z(e) \;:\; e \in E_p\}.$$

(5) Für alle $e \in E_p \cap E$ addiere q zu $f(e)$.

(6) Für alle $e \in E_p \cap E_b$ subtrahiere q von $f(e)$. Dabei ist E_b die Menge aller Rückkanten von E (Notation 8.13).

(7) Gehe zurück zu Schritt **(2)**.

Das Verfahren terminiert mit einem maximalen Fluss f von s nach t in G.

In der Regel wird aufgrund numerischer Rundungsfehler bei der Implementierung des Verfahrens an einigen Stellen nicht auf kleiner oder gleich 0 verglichen, sondern gegen einen sehr kleinen Wert, etwa $\varepsilon = 10^{-6}$. Anderenfalls kann es (theoretisch) auch vorkommen, dass das Verfahren nicht endlich ist, s. Ford und Fulkerson (1962).

Ähnlich zum Algorithmus von Dijkstra ist auch das Verfahren von Ford und Fulkerson am verständlichsten, wenn dieses anhand eines kleinen Beispiels eigenständig angewandt wird:

Aufgabe 8.12 Führe den Algorithmus von Ford und Fulkerson anhand des Graphen aus Abb. 8.6 unter Verwendung von $s = 3$ und $t = 5$ durch. Fertige zu jeder Iteration eine Skizze des Residualgraphen samt Kantengewichtung an.

8.5 Knotenfärbungen

In den vorhergehenden Abschnitten haben wir Probleme der Netzwerkoptimierung kennengelernt, welche sich alle vergleichsweise effizient lösen lassen. Dies wird (zumindest für den allgemeinen Fall) in diesem Abschnitt nicht mehr der Fall sein. Zunächst beginnen wir jedoch wieder mit einer grundlegenden Definition:

Definition 8.15 *Sei $G = (V, E)$ ein ungerichteter und einfacher Graph. Eine **Knotenfärbung** in G mit $r \in \mathbb{N}$ Farben wird gegeben durch eine Funktion*

$$c : V \to \{1, \ldots, r\}, \tag{8.8}$$

welche jedem Knoten $v \in V$ die Farbe $c(v)$ zuordnet. Dabei müssen benachbarte Knoten stets eine unterschiedliche Farbe haben, d.h., es gelte $c(v) \neq c(u)$ für alle $e = (u, v) \in E$.

Offensichtlich lässt sich für einen Graphen $G = (V, E)$ mit n Knoten sehr einfach eine Knotenfärbung finden, falls $r \geq n$ gewählt wird. Das ***Knotenfärbungsproblem***, welches wir in diesem Abschnitt anhand von Spezialfällen genauer analysieren werden, besteht daher darin, eine Knotenfärbung mit einer minimalen Anzahl von Farben zu finden:

Notation 8.16 *Sei $G = (V, E)$ ein ungerichteter und einfacher Graph. Die **chromatische Zahl** $\chi(G)$ ist die minimale Anzahl r an Farben, die für eine Knotenfärbung in G benötigt wird.*

Genauer interessieren wir uns im Folgenden dafür, wie eine Knotenfärbung gefunden werden kann, welche mit $r = \chi(G)$ eine minimale Anzahl an Farben verwendet. Alle folgenden Ergebnisse dazu können in Krumke und Noltemeier (2005) nachgelesen werden.

Zunächst sei bemerkt, dass bereits die Bestimmung der chromatische Zahl $\chi(G)$ im Allgemeinen äußerst schwer ist und schon in vergleichsweise kleinen Graphen nur näherungsweise bestimmt werden kann. Daher kommen zur Lösung von (allgemeinen) Färbungsproblemen häufig Heuristiken zum Einsatz. Wir konzentrieren uns hingegen auf spezielle Klassen von Graphen, in denen Färbungsprobleme exakt und effizient gelöst werden können:

Beispiel 8.2 *Ein Logistikunternehmen hat n Lieferaufträge zu bearbeiten, die alle in einem Zentrallager beginnen und enden. Die Zeiten der Lieferaufträge seien bekannt und gegeben durch Zeitintervalle $[a_i, b_i]$ für $i = 1, \ldots, n$. Dies soll bedeuten, dass zum Zeitpunkt a_i ein Fahrzeug beladen wird, bevor die Ware zum Kunden ausgeliefert wird und das Fahrzeug anschließend zum Zeitpunkt b_i wieder im Lager für einen neuen Auftrag zur Verfügung steht. Dabei kann ein Fahrzeug aufgrund der Größe der Waren niemals mit dem Umfang zweier Aufträge beladen werden.*

Offensichtlich können unter diesen Annahmen zwei Lieferaufträge i und j nur dann von einem Fahrzeug ausgeführt werden, wenn sich die zugehörigen Zeitintervalle $[a_i, b_i]$ und $[a_j, b_j]$ nicht überschneiden. Zu klären ist nun die Frage nach der Anzahl r von mindestens benötigten Fahrzeugen, um alle Lieferaufträge termingerecht zu erledigen.

Dieses Problem kann als Färbungsproblem modelliert werden: Wir konstruieren einen Graphen $G = (V, E)$, dessen Knoten den jeweiligen Lieferaufträgen entsprechen. Weiterhin verbinden wir zwei Knoten durch eine Kante genau dann, wenn sich die zugehörigen Zeitintervalle der Lieferaufträge überschneiden.

Angenommen, wir haben eine Knotenfärbung in G mit r Farben, so entspricht jede Farbe einem benötigten Fahrzeug. Dabei soll r natürlich möglichst klein gehalten werden.

Weitere Anwendungen von Färbungsproblemen wie beispielsweise Ressourcenplanung, Registerzuweisung in Prozessoren oder Frequenzverteilung von Mobilfunkanbietern sind auch in Marx (2004) zu finden.

Zunächst beginnen wir mit einer einfachen Abschätzung der chromatischen Zahl und benötigen dafür folgende Definition:

Definition 8.17 *Sei $G = (V, E)$ ein ungerichteter und einfacher Graph. Der* ***Knotengrad*** *$\delta(v)$ eines Knoten $v \in V$ ist gleich der Anzahl der benachbarten*

Knoten von v. Zudem ist

$$\Delta(G) \;=\; \max\{\delta(v) \;:\; v \in V\}$$

*der **Maximalgrad** von G.*

Der Maximalgrad liefert uns folgende Aussage:

Satz 8.1 *Sei $G = (V, E)$ ein ungerichteter und einfacher Graph. Dann gilt*

$$\chi(G) \;\leq\; \Delta(G) + 1,$$

d.h., die chromatische Zahl wird nach oben beschränkt durch $\Delta(G) + 1$.

Weiterhin können wir ein Verfahren angeben, welches stets eine Knotenfärbung in $G = (V, E)$ mit maximal $\Delta(G) + 1$ Farben bestimmt:

Zusammenfassung 8.5 (Einfache Knotenfärbung) Gegeben sei ein ungerichteter und einfacher Graph $G = (V, E)$ mit n Knoten.

(1) Wähle eine beliebige Nummerierung der Knoten von v_1 bis v_n.

(2) Setze $c(v) = 0$ für alle $v \in V$, d.h., alle Konten seien zunächst noch nicht eingefärbt.

(3) Für $i = 1, \ldots, n$: Setze $c(v_i)$ auf die kleinstmögliche Zahl bzw. Farbe (unter Berücksichtigung aller bereits eingefärbten Knoten).

Das Verfahren liefert eine Knotenfärbung mit $r \leq \Delta(G) + 1$ Farben.

Es sei nochmals darauf hingewiesen, dass dieses Verfahren im Allgemeinen keine Färbung mit der kleinstmöglichen Anzahl an Farben findet und somit das Knotenfärbungsproblem im Allgemeinen auch nicht löst. Vielmehr hängt die bestimmte Knotenfärbung stark von der gewählten Nummerierung in Schritt **(1)** ab:

Beispiel 8.3 *In Abb. 8.7 wurde die einfache Knotenfärbung aus Zusammenfassung 8.5 auf zwei unterschiedliche Nummerierungen der Knoten angewandt.*

Wir erkennen, dass die erste Nummerierung eine Färbung mit vier Farben liefert, obwohl $\Delta(G) + 1 = 5$ gilt. Die zweite Nummerierung ergibt eine Färbung mit nur

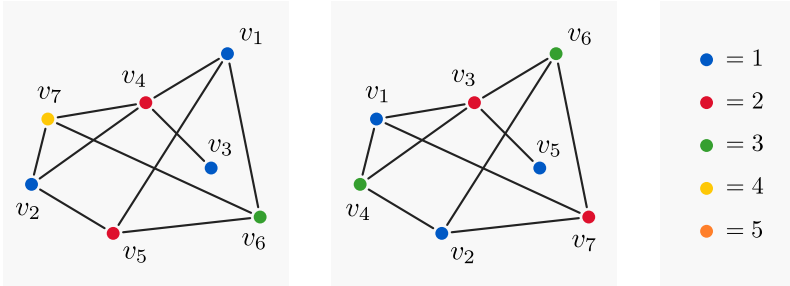


Abb. 8.7 Beispiel zur Knotenfärbung unter Verwendung der einfachen Knotenfärbung aus Zusammenfassung 8.5. Dargestellt ist die Knotenfärbung für zwei unterschiedliche Nummerierungen der Knoten

drei Farben. Es sei bemerkt, dass $\chi(G) = 3$ gilt; somit ist die zweite Färbung eine Lösung des Knotenfärbungsproblems.

Zusammengefasst liefert die einfache Knotenfärbung eine simple Heuristik zur Bestimmung einer Knotenfärbung (welche jedoch keineswegs mit einer minimalen Anzahl an Farben auskommt). Damit wenden wir uns bereits vom allgemeinen Fall ab und untersuchen spezielle Klassen von Graphen:

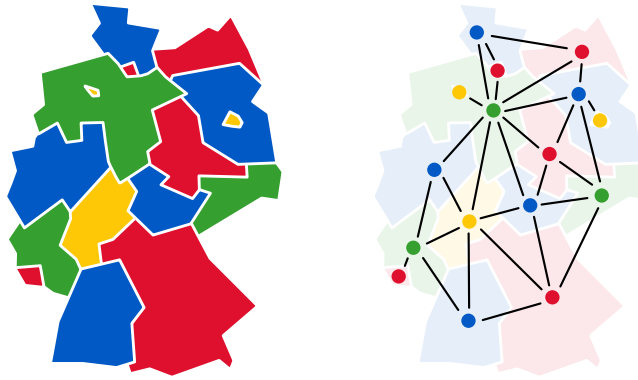
Ausblick *Ein ungerichteter und einfacher Graph $G = (V, E)$ heißt **planar**, falls der Graph derart in der Ebene gezeichnet werden kann, dass sich keine Kanten überschneiden.*

*Beispielsweise ist der Graph aus Abb. 8.7 planar (sofern zwei Kanten anders eingezeichnet werden). Planare Graphen sind hinsichtlich des Knotenfärbungsproblems von großer Bedeutung, da sie den **Vier-Farben-Satz** erfüllen: Für planare Graphen $G = (V, E)$ gilt $\chi(G) \leq 4$. Dies bedeutet, dass planare Graphen mit (maximal) vier Farben gefärbt werden können.*

Dieser Satz wurde bereits vor mehreren Hundert Jahren vermutet, jedoch erst vor wenigen Jahrzehnten bewiesen. Das Interessante dabei ist, dass der Satz auch besagt, dass sich jede Landkarte mit vier Farben färben lässt: Genauer kann jede Landkarte derart eingefärbt werden, dass keine Länder mit gemeinsamer Grenze die gleiche Farbe haben (unter der Annahme, dass alle Länder zusammenhängend sind).

Um aus einer Landkarte einen planaren Graphen zu erhalten, gehen wir folgendermaßen vor: Die Knoten entsprechen den einzelnen Ländern. Nun fügen wir jeweils Kanten hinzu, falls die Länder eine gemeinsame Grenze haben. Ein Beispiel samt Färbung kann Abb. 8.8 entnommen werden.

Obwohl wir wissen, dass sich planare Graphen mit maximal vier Farben färben lassen, kann es allerdings sehr schwierig sein, eine derartige Färbung zu finden.



a Eingefärbte Landkarte **b** Zugehöriger Graph

Abb. 8.8 Beispiel einer eingefärbten Landkarte als Anwendung des Knotenfärbungsproblems für planare Graphen. Keine zwei benachbarten Länder haben dieselbe Farbe

Im Fokus der folgenden Betrachtungen liegt nun eine Klasse von Graphen, welche beispielsweise auch in Beispiel 8.2 zum Einsatz kommt. Dazu benötigen wir zunächst folgenden Begriff:

Definition 8.18 *Ein ungerichteter und einfacher Graph* $G = (V, E)$ *heißt* ***vollständig****, falls alle Knoten miteinander verbunden sind.*

Ein vollständiger Graph mit n Knoten besitzt demnach

$$m = \frac{n \cdot (n-1)}{2}$$

Kanten (Abb. 8.9). Offensichtlich lässt sich das Knotenfärbungsproblem für vollständige Graphen einfach lösen:

Aufgabe 8.13 Wie viele Farben werden zur Färbung eines vollständigen Graphen mit n Knoten (mindestens) benötigt?

Schließlich betrachten wir folgende Klasse von Graphen, für welche wir einen effizienten Färbungsalgorithmus formulieren werden, s. auch Krumke und Noltemeier (2005):