10.2 Reduktionen auf lineare Optimierung

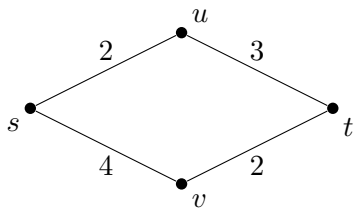
Auf den ersten Blick erweckt das Problem der linearen Optimierung vielleicht den Eindruck recht spezifisch für diese Art von Produktionsplanung und verwandte Situationen zu sein. In der Tat ist das Problem aber sehr allgemein in dem Sinn dass für viele Probleme recht direkte Reduktionen auf lineare Optimierung existieren. Um das zu illustrieren werden wir hier einige solche Reduktionen angeben.

Kürzester Pfad. Gegeben ein zusammenhängender Graph $G = (V, E)$ und eine Kostenfunktion $c : E \to \mathbb{R}_{\geq 0}$ sowie einen Startknoten $s \in V$ und einen Zielknoten $t \in V$ wollen wir die Länge des kürzesten (d.h. billigsten) Pfades von s nach t bestimmen. In Abschnitt 7.3 haben wir den Algorithmus von Dijkstra kennengelernt, der einen solchen Pfad berechnet. Um dieses Problem als lineares Programm darzustellen führen wir für jedes $v \in V$ eine reellwertige Variable d_v ein, die die Länge des kürzesten Pfades von s nach v darstellen soll. Dann gilt

$$\begin{aligned} &d_s = 0 \quad \text{und} \\ &d_v \leq d_u + c(\{u, v\}) \quad \text{für alle } \{u, v\} \in E. \end{aligned}$$

Wenn wir unter diesen Bedingungen d_t maximieren, stellen wir sicher, dass implizit ein Pfad von s nach t ausgewählt wird, da für jedes d_v für $v = t, \ldots, s$ eine der Ungleichungen mit Gleichheit erfüllt wird. Jeder Knoten u der das erreicht ist der v vorausgehende auf einem kürzesten Pfad.

Beispiel 10.2. Der Graph



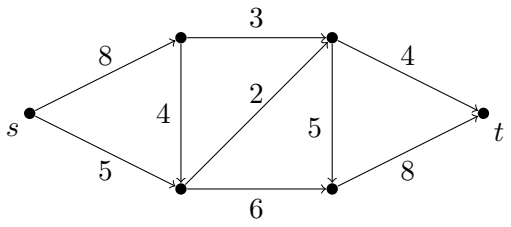
führt zu dem linearen Programm

$$\begin{aligned} &d_t \text{ max!} \\ &d_s = 0, d_u \leq d_s + 2, d_v \leq d_s + 4, d_t \leq d_u + 3, d_t \leq d_v + 2. \end{aligned}$$

von dem $(d_s, d_u, d_v, d_t) = (0, 2, 0, 5)$ eine optimale Lösung ist.

Maximaler Fluss. Wir betrachten jetzt das Problem des maximalen Flusses (in einem Transportnetzwerk). Das modellieren wir wie folgt: Wir gehen von einem gerichteten, zusammenhängenden Graphen $G = (V, E)$ aus sowie von einem ausgezeichneten Quellknoten $s \in V$ und einem ausgezeichneten Zielknoten $t \in V$. Weiters hat jede Kante eine gewisse (Transport-)kapazität die durch eine Funktion $c : E \to \mathbb{R}_{\geq 0}$ angegeben wird. Es gibt an den Knoten außer den im Graph angegebenen Kanten keine Möglichkeit der Entnahme oder Einspeisung. Weiters ist nirgendwo eine Lagerung möglich. Wir wollen so viel wie möglich von s nach t transportieren. Als Anwendungsbeispiel können wir uns etwa den Transport von Rohöl durch ein Netzwerk von Pipelines vorstellen.

Beispiel 10.3. Ein Beispiel für einen gerichteten Graphen mit Kapazitätsbeschriftungen auf den Kanten ist:



Definition 10.2. Sei $G = (V, E)$ ein gerichteter Graph, $s, t \in V$ und $c : E \to \mathbb{R}_{\geq 0}$ eine Kapazitätsfunktion. Ein *Fluss* für G und e ist dann eine Funktion $f : E \to \mathbb{R}_{\geq 0}$ so dass

1. $f(e) \leq c(e)$ für alle $e \in E$ und
2. $\sum_{(u,v)\in E} f(u, v) = \sum_{(v,w)\in E} f(v, w)$ für alle $v \in V \setminus \{s, t\}$.

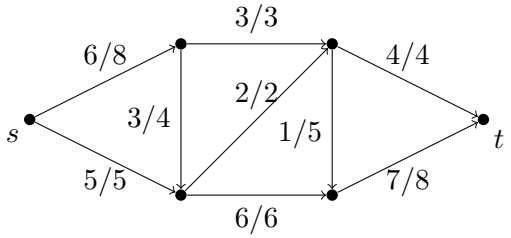
Das Problem der Bestimmung eines maximalen Flusses kann dann wie folgt formuliert werden.

Maximaler Fluss

Eingabe: gerichteter zusammenhängender Graph (V, E), $s, t \in V$, Kapazitätsfunktion $c : E \to \mathbb{R}_{\geq 0}$

Ausgabe: ein Fluss $f : E \to \mathbb{R}_{\geq 0}$ so dass $\sum_{(v,t)\in E} f(v, t)$ maximal ist

Beispiel 10.4. Ein maximaler Fluss des Graphen aus Beispiel 10.3 ist:



Zur Lösung dieses Problems mittels linearer Optimierung stellen wir einen Fluss f dar indem wir für jedes $e \in E$ eine reellwertige Variable f_e einführen, die $f(e)$ darstellen soll. Das lineare Programm ist dann:

$$
\begin{aligned}
0 \leq f_e \quad & \text{für alle } e \in E \\
f_e \leq c(e) \quad & \text{für alle } e \in E \\
\sum_{(u,v)\in E} f_{u,v} = \sum_{(v,w)\in E} f_{v,w} \quad & \text{für alle } v \in V \setminus \{s, t\} \\
\sum_{(v,t)\in E} f_{v,t} \ & \text{max!}
\end{aligned}
$$

Es ist leicht zu sehen dass eine optimale Lösung dieses linearen Programms ein maximaler Fluss ist.

Schaltkreisevaluierung. Ein letztes Problem das wir nun auf lineare Optimierung reduzieren wollen ist die Schaltkreisevaluierung. Das Problem der Schaltkreisevaluierung ist **P**-vollständig, d.h., informell gesagt, dass jedes (mit einem beliebigen Algorithmus) in polynomialer Zeit lösbare Problem durch ein Verfahren für Schaltkreisevaluierung in polynomialer Zeit gelöst werden kann[1]. Diese Reduktion zeigt also, dass auch lineare Optimierung **P**-vollständig ist,

Definition 10.3. Ein Boolescher Schaltkreis ist ein endlicher gerichteter azyklischer Graph $G = (V, E)$ mit einer Knotenbeschriftung $l : V \to \{\mathbf{w}, \mathbf{f}, \wedge, \vee, \neg\}$ so dass $\mathrm{d}^-(v) \leq 2$ für alle $v \in V$ und:

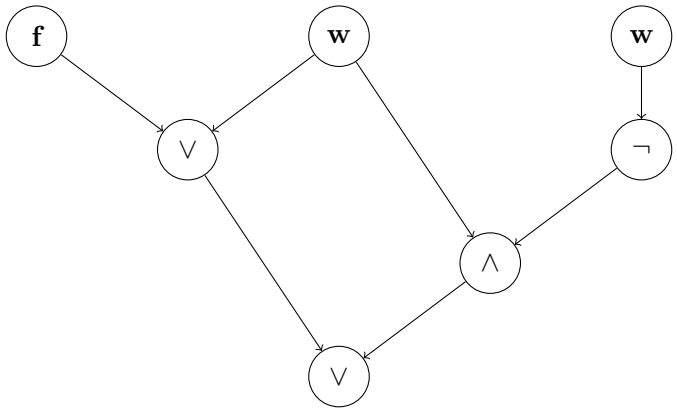
1. Falls $l(v) \in \{\mathbf{w}, \mathbf{f}\}$, dann ist $\mathrm{d}^-(v) = 0$.
2. Falls $l(v) = \neg$, dann ist $\mathrm{d}^-(v) = 1$.
3. Falls $l(v) \in \{\wedge, \vee\}$, dann ist $\mathrm{d}^-(v) = 2$.

Hier steht **w** für "wahr" und **f** für "falsch". Die logischen Operatoren $\wedge$, $\vee$ und $\neg$ werden ihre übliche Bedeutung erhalten. Ein Knoten v eines Schaltkreises mit $\mathrm{d}^+(v) = 0$ heißt *Ausgabeknoten*. Für jeden Knoten v wird jetzt induktiv sein Wert $[\![v]\!]$ der Semantik der Booleschen Logik folgend definiert: Falls $l(v) = \mathbf{w}$, dann $[\![v]\!] = 1$. Falls $l(v) = \mathbf{f}$, dann $[\![v]\!] = 0$. Falls $l(v) \in \{\neg, \wedge, \vee\}$, dann ergibt sich $[\![v]\!]$ aus den folgenden Tabellen:

a	$\neg a$
0	1
1	0

a	b	$a \wedge b$	$a \vee b$
0	0	0	0
0	1	0	1
1	0	0	1
1	1	1	1

Beispiel 10.5. Der Schaltkreis



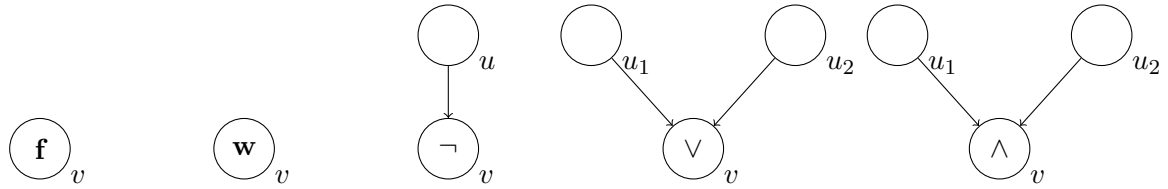
hat einen einzigen Ausgabeknoten v dessen Wert $[\![v]\!] = 1$ ist.

Das Berechnungsproblem der Schaltkreisevaluierung ist dann wie folgt:

Schaltkreisevaluierung
Eingabe: Ein Schaltkreis mit genau einem Ausgabeknoten
Ausgabe: Der Wert des Ausgabeknotens

[1] Wir geben hier keine formale Definition von **P**-Vollständigkeit und verweisen stattdessen auf Quellen über Komplexitätstheorie.

Dieses Problem wird durch das folgende lineare Programm gelöst. Wir führen für jedes $v \in V$ eine Variable x_v ein, die den Wert von v repräsentieren soll. Für jedes x_v werden die Ungleichungen $0 \leq x_v$ und $x_v \leq 1$ aufgenommen. Für jeden Knoten werden weitere Ungleichungen wie folgt aufgenommen:



$x_v = 0$	$x_v = 1$	$x_v = 1 - x_u$	$x_v \geq x_{u_1}$, $x_v \geq x_{u_2}$ $x_v \leq x_{u_1} + x_{u_2}$	$x_v \leq x_{u_1}$, $x_v \leq x_{u_2}$ $x_v \geq x_{u_1} + x_{u_2} - 1$

Dann lässt sich mit Induktion über den Schaltkreis nachweisen dass die einzige zulässige Lösung dieses linearen Programms jene ist mit $x_v = [\![v]\!]$ für alle $v \in V$. Insbesondere gilt dadurch $x_v \in \{0, 1\}$. Deshalb ist in diesem Fall auch irrelevant ob und was minimiert oder maximiert wird.

10.3 Der Simplex-Algorithmus

Wir behandeln jetzt einen Algorithmus zur Lösung des Problems der linearen Optimierung. Dazu ist es sinnvoll, den Formalismus ein wenig einzuschränken.

Definition 10.4. Ein *lineares Programm in Standardform* besteht aus einer Matrix $A \in \mathbb{R}^{m \times n}$, einem Vektor $b \in \mathbb{R}^m$ und einem Vektor $c \in \mathbb{R}^n$.

Die Matrix A gemeinsam mit dem Vektor b spezifiziert E durch die Ungleichungen $Ax \leq b$. Der Vektor c spezifiziert f als $x \mapsto c^{\mathrm{T}}x$. Zusätzlich schränken wir uns bei einem linearen Programm in Standardform darauf ein f zu maximieren und nur solche Lösungen zu betrachten wo $x_i \geq 0$ für alle $i \in \{0, \ldots, n\}$. Wir definieren also

Definition 10.5. Sei A, b, c ein lineares Programm in Standardform. Ein $x \in \mathbb{R}^n$ heißt *zulässige Lösung* falls $Ax \leq b$ und $x_i \geq 0$ für alle $i \in \{0, \ldots, n\}$. Eine zulässige Lösung $x \in \mathbb{R}^n$ heißt *optimale Lösung* falls $c^{\mathrm{T}}x$ maximal unter allen zulässigen Lösungen ist.

Lineare Optimierung (Standardform)
Eingabe: ein lineares Programm A, b, c in Standardform
Ausgabe: "unbeschränkt" falls A, b, c unbeschränkt ist, "unlösbar" falls A, b, c unlösbar ist, eine optimale Lösung $x \in \mathbb{R}^n$ von A, b, c sonst

Definition 10.6. Zwei max-lineare Programme E, f und E', f' heißen *äquivalent* falls:

1. Für jede zulässige Lösung x von E existiert eine zulässige Lösung x' von E' mit $f(x) = f'(x')$.
2. Für jede zulässige Lösung x' von E' existiert eine zulässige Lösung x von E mit $f(x) = f'(x')$.

Ein min-lineares Programm E, f und ein max-lineares Programm E', f' heißen *äquivalent* falls:

1. Für jede zulässige Lösung x von E existiert eine zulässige Lösung x' von E' mit $f(x) = -f'(x')$.
2. Für jede zulässige Lösung x' von E' existiert eine zulässige Lösung x von E mit $f(x) = -f'(x')$.

Lemma 10.1. *Jedes lineare Programm kann in ein äquivalentes lineares Programm in Standardform transformiert werden.*

Beweis. Ein lineares Programm kann sich von einem linearen Programm in Standardform in den folgenden Aspekten unterscheiden: 1. Falls die Zielfunktion f einen konstanten Teil $d \in \mathbb{R}$ hat, dann fügen wir eine neue Variable v sowie die Gleichung $v = d$ hinzu und ersetzen d in f durch v. Das dadurch entstehende lineare Programm ist äquivalent zum ursprünglichen. 2. Falls die Zielfunktion f minimiert werden soll, wird stattdessen $-f$ maximiert. Diese beiden linearen Programme sind dann äquivalent. 3. Eine Variable v ohne die Bedingung $v \geq 0$ wird entfernt und durch zwei neue Variablen v^+ und v^- simuliert, indem jedes Vorkommen von v durch $v^+ - v^-$ ersetzt wird. Weiters werden die Bedingungen $v^+ \geq 0$ und $v^- \geq 0$ aufgenommen. Diese beiden linearen Programme sind äquivalent da jedes $a \in \mathbb{R}$ geschrieben werden kann als $a = a^+ - a^-$ mit $a^+, a^- \geq 0$. 4. Eine Gleichung der Form $f(\bar{x}) = g(\bar{x})$ für zwei affine Funktionen f und g wird ersetzt durch die beiden Ungleichungen $f(\bar{x}) \leq g(\bar{x})$ und $g(\bar{x}) \leq f(\bar{x})$. Diese beiden linearen Programme sind äquivalent. Schließlich werden die affinen Ungleichungen in der Form $Ax \leq b$ geschrieben und die Zielfunktion f als $x \mapsto c^{\mathrm{T}}x$. □

Wir machen nun einige grundlegende Beobachtungen über die Situation in deren Rahmen der Simplex-Algorithmus motiviert und geometrisch erklärt werden kann.

Zunächst beobachten wir dass jede der Ungleichungen eines linearen Programms in Standardform einen Halbraum von $\mathbb{R}^n$ definiert. Die Menge der zulässigen Lösungen ist also ein Schnitt von Halbräumen und damit ein *konvexes Polytop*.

Die zu maximierende Funktion $x \mapsto c^{\mathrm{T}}x$ induziert für festes $z \in \mathbb{R}$ die Hyperebene $z = c^{\mathrm{T}}x$. Der Maximierung von z entspricht eine Verschiebung der Hyperebene durch das Polytop der zulässigen Lösungen (vgl. dazu auch Beispiel 10.1 im $\mathbb{R}^2$). Falls das lineare Programm lösbar ist, endet diese Verschiebung mit einem z das eine Hyperebene definiert, die ein k-dimensionales Unterpolytop enthält, also einen Knoten, eine Kante, eine Seitenfläche, etc. des Polytops der zulässigen Lösungen[2]. Ein Unterpolytop enthält aber mindestens einen Knoten. Wir erkennen also: *Falls das lineare Programm lösbar ist, dann ist ein Knoten des Polytops eine optimale Lösung.*

Es reicht also, die Hyperebene von Knoten zu Knoten zu schieben. Die *Grundidee* für den Simplex-Algorithmus ist nun: wir starten an einem beliebigen Knoten des Polytops. In jedem Schritt bewegen wir uns zu einem benachbarten Knoten an dem der Zielwert mindestens so groß wie beim vorherigen ist.

[2]Falls das lineare Programm unlösbar ist, hat diese Verschiebung keinen Anfang, falls es unbeschränkt ist keine Ende.