Experimentelle Ergebnisse zum Network-Simplex-Algorithmus

Max Kanold

19. September 2018

Inhaltsverzeichnis

Kapitel 1

Einführung

Bla. Zum Beispiel in Abschnitt 2.3 habe ich programmiert.

Kapitel 2

Network-Simplex-Algorithmus

Das Simplex-Verfahren, zu welchem eine Einführung in [1] gefunden werden kann, löst Lineare Programme in der Praxis sehr schnell, obwohl die Worst-Case-Laufzeit nicht polynomiell ist. Jedes Netzwerkproblem lässt sich als Lineares Programm darstellen und somit durch das Simplex-Verfahren lösen, durch die konkrete Struktur solcher Probleme genügt jedoch der vereinfachte Network-Simplex-Algorithmus. Auch für diesen gibt es exponentielle Instanzen (siehe [2]), in der Praxis wird er trotzdem vielfach verwendet.
TODO gibt es?

2.1 Min-Cost-Flow-Problem

Definition 1. Ein **Netzwerk** ist ein Tupel (G, b, c, u), wobei $G = (V, E)$ ein gerichteter Graph, $b : V \to \mathbb{R}$ eine b-Wert-Funktion, $c : E \to \mathbb{R}$ eine Kostenfunktion und $u : E \to \mathbb{R}_{\geq 0} \cup \infty$ eine Kapazitätsfunktion seien.

Anmerkung. Knoten mit positiven b-Wert werden als Quellen, solche mit negativen als Senken bezeichnet.
Ein ungerichteter Graph kann durch das Verwandeln jeder Kante $\{v, w\}$ in zwei Kanten (v, w) und (w, v) zu einem gerichteten modifiziert werden.

Definition 2. Ein **maximaler Fluss** auf einem Netzwerk $(G = (V, E), b, c, u)$ ist eine Abbildung $f : E \to \mathbb{R}_{\geq 0}$, die folgende Eigenschaften erfüllt:

(i) $\forall e \in E : f(e) \leq u(e)$

(ii) $\forall v \in V : \sum_{(w,v)\in E} f((w, v)) - \sum_{(v,w)\in E} f((v, w)) + b(v) = 0$

Der **Wert** von f ist $v(f) = \frac{1}{2} \cdot \sum_{v\in V} |b(v)|$
und die **Kosten** von f sind $c(f) = \sum_{e\in E} f(e) \cdot c(e)$.

Beim *Min-Cost-Flow-Problem* wird unter allen maximalen Flüssen einer mit minimalen Kosten gesucht. Sind die Kapazitäten unbeschränkt, so wird es als *Transportproblem* bezeichnet.

In dieser Bachelorarbeit wird angenommen, dass u und c auf $\mathbb{N}$ sowie b auf $\mathbb{Z}$ abbildet, um Gleitkommazahlungenauigkeit zu vermeiden. Durch eine entsprechende Skalierung des Problems können die Funktionen nach $\mathbb{R}_{\geq 0}$ bzw. $\mathbb{R}$ hinreichend genug angenähert werden. Durch die eingeschränkte Kostenfunktion hat kein maximaler Fluss negative Kosten; es gibt keine unbeschränkten Instanzen. Unbeschränkte Kapazitäten können somit o. B. d. A. durch $\frac{1}{2} \cdot \sum_{v \in V} |b(v)| + 1$ approximiert werden.

Zusätzlich wird davon ausgegangen, dass $\sum_{v \in V(G)} b(v) = 0$ ist, Angebot und Nachfrage also ausgeglichen sind. Des Weiteren ist in der konkreten Implementierung E keine Multimenge; es sind keine parallelen Kanten vorgesehen.

2.2 Algorithmus

[3, Dantzig, 1951] und [4, Orden, 1956] vereinfachten das Simplex-Verfahren zum Netzwerk-Simplex-Algorithmus; die folgende Beschreibung orientiert sich zuerst an [1, S. 291 ff.] zur Lösung des Transportproblems, danach wird der Algorithmus anhand von TODO auf den allgemeinen, durch Kapazitäten beschränkten Fall erweitert.

Definition 3. Der einem gerichteten Graphen $G = (V, E)$ **zugrundeliegende ungerichtete Graph** $G' = (V, E')$ ist definiert durch:

$$\{v, w\} \in E' \iff (v, w) \in E \vee (w, v) \in E$$

Anmerkung. Nach dieser Definition gibt es keine Bijektion zwischen gerichteten und den zugrundeliegenden ungerichteten Graphen; dafür vermeidet man parallele Kanten.

Definition 4. Ein **Baum** T ist ein ungerichteter, zusammenhängender und kreisfreier Graph. Ein **Wald** ist ein Graph, bei dem jede Zusammenhangskomponente ein Baum ist.
Ein Teilgraph $T = (V', E')$ eines ungerichteten Graphen $G = (V, E)$ heißt **aufspannender Baum**, wenn T ein Baum und $V' = V$ ist.

Anmerkung. Sprechen wir bei einem gerichteten Graphen G über einen Wald bzw. (aufspannenden) Baum, so bezieht sich das stets auf einen Teilgraphen T von G, dessen zugrundeliegender ungerichteter Graph ein Wald bzw. (aufspannender) Baum des G zugrundeliegenden ungerichteten Graphen ist.

Definition 5. Sei $N = (G, b, c, u)$ ein Netzwerk. Ein aufspannender Baum T von G und ein maximaler Fluss f auf N bilden eine **zulässige Baumlösung** (T, f), wenn $\forall e \in E(G) \backslash E(T) : f(e) = 0$.

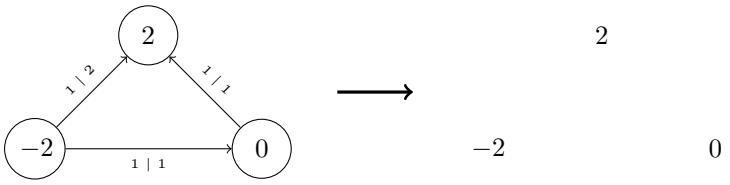
Lemma 6. *Jede zulässige Baumlösung (T, f) ist eindeutig durch den aufspannenden Baum T definiert.*

Beweis. Sei $(T = (V, E), f)$ eine zulässige Baumlösung eines Netzwerkes. Für jedes Blatt von $l \in V$ sei $e_l \in E$ die eindeutige Kante in T zwischen den Knoten k und l. Für alle Blätter l ist der Wert von $f(e_l)$ nach Definition 2 Punkt (ii) eindeutig.

Wir entfernen nun alle mit Fluss belegten Kanten e_l sowie die nun isolierten Knoten l und aktualisieren den b-Wert von k. Der dabei entstehende Graph

$T' = (V' \subseteq V, E' \subseteq E)$ bleibt weiterhin ein aufspannender Baum und $f_{|E'}$ ein maximaler Fluss. Durch Iteration ist f wohldefiniert und eindeutig. □

Wie TODO veranschaulicht, gibt es maximale Flüsse, zu denen wir keine zulässige Baumlösung finden können. Auch wenn wir uns auf die maximalen Flüsse beschränken, zu denen es zulässige Baumlösungen gibt, gilt die Gegenrichtung nach TODO nicht.



Wie wir in Korollar 10 zeigen werden, existiert eine zulässige Baumlösung (T, f), sodass $c(f)$ minimal ist. Die dem Algorithmus zugrundeliegende Idee ist es, über zulässige Baumlösungen mit sinkenden Kosten zu iterieren, bis diese minimal sind. Der Übergang basiert dabei auf dem Augmentieren negativer Kreise, eine Methode, für die das Konzept des Residualgraphen hilfreich ist.

Definition 7. Sei $N = (G = (V, E), b, c, u)$ ein Netzwerk mit einem maximalen Fluss f. Die **Residualkante** $\bar{e}$ einer Kante $e = (v, w) \in E$ verläuft von w nach v. Sei $\bar{E} = \{\bar{e} | e \in E\}$ die Menge aller Residualkanten. Die *Residualkapazität* $u_f : \bar{E} \to \mathbb{Z}$ ist bestimmt durch $u_f(\bar{e}) = u(e) - f(e)$, die *Residualkosten* $c_f : \bar{E} \to \mathbb{Z}$ durch $c_f(\bar{e}) = -c(e)$.

Der **Residualgraph** R ist ein Tupel $R_{N,f} = (\bar{G} = (V, E \amalg \bar{E}), \bar{f}, b, \bar{c}, \bar{u})$ mit dem gerichteten Multigraphen $\bar{G}$ aus der bisherigen Knotenmenge und der disjunkten Vereinigung von Kanten und Residualkanten, dem maximalen Fluss $\bar{f}$ mit $\forall e \in E : \bar{f}(e) = f(e)$ und $\forall \bar{e} \in \bar{E} : \bar{f}(\bar{e}) = 0$, der b-Wert-Funktion b wie gehabt, der Kostenfunktion $\bar{c} = c \amalg c_f : E \amalg \bar{E} \to \mathbb{Z}$ und der Kapazitätsfunktion $\bar{u} = u \amalg u_f : E \amalg \bar{E} \to \mathbb{Z}$.

Augmentieren wir in einem Residualgraphen $R_{N,f}$ einen gerichteten Kreis $C \subseteq E \amalg \bar{E}$, erhöhen also für alle Kanten $e \in C$ den Fluss um einen festen Betrag $\tau \in \mathbb{N}$, ohne die Kapazitätsschranken zu verletzen, erhalten wir einen neuen maximalen Fluss f' in N. Sind die Kosten des Kreises negativ, also $c(C) = \sum_{e \in C} \bar{c}(e) < 0$, verbilligen sich die Kosten des neuen maximalen Flusses f' um $\tau \cdot c(C)$.

Lemma 8. *Augmentiert man um Kreis maximal, ist C_f kein Kreis mehr.*

Beweis. □

Mithilfe des Konzeptes und Lemma können wir nun thm und damit kor zeigen.

Theorem 9. *Sei f ein maximaler Fluss. Es existiert ein maximaler Fluss $\hat{f}$, sodass $c(\hat{f}) \leq c(f)$ ist und eine zulässige Baumlösung $(\hat{T}, \hat{f})$ existiert.*

Beweis. Sei f ein maximaler Fluss für ein Netzwerk $(G = (V, E), b, c, u)$ und $G_f = (V, E' = \{e \in E | f(e) \neq 0\})$ der Graph aller durchflossenen Kanten. Wir werden f zu einem maximalen Fluss $\hat{f}$ umwandeln, sodass $G_{\hat{f}}$ ein Wald ist. Für

$\hat{f}$ finden wir dann leicht eine zulässige Baumlösung. Wenn wir für alle maximalen Zwischenflüsse $f = f_0, f_1, f_2, \ldots, f_n = \hat{f}$ sicherstellen, dass $c(f_{i+1}) \leq c(f_i)$ ist, gilt auch $c(\hat{f}) \leq c(f)$.

Betrachte einen maximalen Fluss f_i. Wenn G_{f_i} ein Wald ist, sind wir fertig. Ansonsten gibt es Kreis -> Lemma iterieren. □

Korollar 10. *Es existiert ein maximaler Fluss f mit minimalen Kosten und einer zulässigen Baumlösung (T, f).* □

2.3 Implementation

Hier beginnt mein schönes Werk ...

2.3.1 Spezielle Konstrukte

... und hier endet es.

Die Klasse Circle

Kreise halt.[1]

Der Rest halt

Kleinkram.

Kapitel 3

Experimentelle Ergebnisse

Alle scheiße.

Kapitel 4

Ausblick

La la la.

Literaturverzeichnis

[1] V. Chvátal, *Linear Programming*, pp. 291 ff. Series of books in the mathematical sciences, W. H. Freeman, 16 ed., 2002.

[2] N. Zadeh, "A bad network problem for the simplex method and other minimum cost flow algorithms," *Mathematical Programming*, vol. 5, no. 1, pp. 255–266, 1973.

[3] G. B. Dantzig, "Application of the simplex method to a transportation problem," in *Activity Analysis of Production and Allocation* (T. C. Koopmans, ed.), ch. XXIII, pp. 359–373, New York: Wiley, 1951.

[4] A. Orden, "The transhipment problem," *Management Science*, vol. 2, no. 3, pp. 276–285, 1956.