Diplomarbeit

Timm Pliefke

Ein polynominaler primaler Netzwerk Simplex Algorithmus zur Berechnung von Flüssen mit minimalen Kosten

Diplom.de

Bibliografische Information der Deutschen Nationalbibliothek:

Bibliografische Information der Deutschen Nationalbibliothek: Die Deutsche Bibliothek verzeichnet diese Publikation in der Deutschen Nationalbibliografie; detaillierte bibliografische Daten sind im Internet über http://dnb.d-nb.de/ abrufbar.

ISBN: 9783836621120

Timm Pliefke

Ein polynominaler primaler Netzwerk Simplex Algorithmus zur Berechnung von Flüssen mit minimalen Kosten

Diplom.de

Diplomarbeit

Timm Pliefke

Ein polynominaler primaler Netzwerk Simplex Algorithmus zur Berechnung von Flüssen mit minimalen Kosten

Diplom.de

Timm Pliefke
Ein polynominaler primaler Netzwerk Simplex Algorithmus zur Berechnung von Flüssen mit minimalen Kosten

ISBN: 978-3-8366-2112-0
Herstellung: Diplomica® Verlag GmbH, Hamburg, 2008
Zugl. Universität Augsburg, Augsburg, Deutschland, Diplomarbeit, 2004

Vorwort

Während der Anfertigung dieser Arbeit haben mich zahlreiche Menschen motiviert und unterstützt. Bei Ihnen möchte ich mich an dieser Stelle bedanken.

Vielen Dank an Herrn Priv.-Doz. Dr. Dirk Hachenberger, Akademischer Rat am Lehrstuhl für Optimierung und Operations Research für die wohlwollende Unterstützung und Förderung dieser Arbeit.

Bei meinen Freunden Oliver Eckelmann, Sebastian Eiteljörge und Kaya Taner bedanke ich mich für die anregenden Diskussionen und die kritische Auseinandersetzung mit meinem Diplomarbeitsthema. Überdies haben sie in mir in schwierigen Situationen immer den nötigen Beistand geleistet, um die Motivation zur Fertigstellung dieser Arbeit aufrechtzuerhalten.

Mein besonderer Dank gilt meiner Familie:

Meinen Eltern, Frau Doris und Herrn Dr. Engelbert Pliefke, sowie meiner Schwester Sabine. Erst ihre tatkräftige Unterstützung ermöglichte es mir, das Studium der Wirtschaftsmathematik an der Universität Augsburg erfolgreich zu beenden.

Ihnen widme ich diese Arbeit.

Königsbrunn, im August 2004

Timm Pliefke

Inhaltsverzeichnis

Einführung **2**

1 Grundlagen **6**

1.1 Graphen, Netzwerke und Flüsse 6
1.2 Eigenschaften maximaler Flüsse 9

2 Das Minimum Cost Flow Problem (MCFP) **17**

2.1 Die Problemstellung 17
2.2 MCFP Zulässigkeit 21
2.3 MCFP Optimalität 29

3 Der Netzwerk Simplex Algorithmus **35**

3.1 Baumlösungen und Baumstrukturen 37
3.2 Initialisierung 47
3.3 Der Netzwerk Simplex Algorithmus 54

4 Der Premultiplier Algorithmus **77**

4.1 Der allgemeine Premultiplier Algorithmus 78
4.2 Der skalierte Premultiplier Algorithmus 89

5 Details zur Implementierung **109**

5.1 Implementierung des Netzwerk Simplex Algorithmus 112
5.2 Implementierung des Premultiplier Algorithmus 134
5.3 Implementierung des skalierten Premultiplier Algorithmus . . 152

Zusammenfassung und Ausblick **159**

Literaturverzeichnis **163**

Einführung

„Es gibt zwei Wege, die Rentabilität der Arbeit eines Geschäfts, eines Unternehmens oder eines ganzen Industriezweiges zu vergrößern. Ein Weg besteht in verschiedenen Verbesserungen der Technik, z.B. neuem Zubehör für die einzelnen Maschinen, Änderungen der technischen Prozesse und der Entdeckung neuer, besserer Arten von Rohmaterial. Der andere Weg, der bis jetzt viel weniger genutzt wurde, besteht in der Verbesserung der Organisation der Planung."

Dieses Zitat geht auf den russischen Mathematiker Kantorowicz zurück, der im Jahre 1939 mit seinem Buch „Mathematische Methoden in der Organisation und Planung der Produktion" die erste Arbeit auf dem Gebiet der mathematischen Optimierung veröffentlichte und somit den Grundstein für den großen Aufschwung der linearen Optimierung in den Folgejahren legte. Als Meilenstein in der Geschichte der linearen Optimierung gilt die Entwicklung des **Simplex Algorithmus** durch G.B. Dantzig [13] im Jahre 1947. Bis heute ist der Simplex Algorithmus eines der mächtigsten und das in der Praxis am weitesten verbreitete Verfahren zur Lösung linearer Programme. Komplexe Problemstellungen mit tausenden von Variablen und Restriktionen lassen sich mit modernen Implementierungen des Algorithmus [1] innerhalb kürzester Zeit lösen.

Zahlreiche Optimierungsprobleme aus dem täglichen Anwendungsbereich, wie z.B

- die Ermittlung von kürzesten Wegen innerhalb von Verkehrs- und Kommunikationsnetzen *(Shortest Path Problem, SPP)*,
- die Maximierung des Verkehrsflusses auf einem vorgegebenen Straßennetz *(Maximum Flow Problem, MFP)*,
- oder der kostenminimale Transport von Gütern zwischen Produzenten und Käufern **(Minimum Cost Flow Problem, MCFP)**,

[1] wie z.B. CPLEX

weisen eine spezielle Struktur auf, welche es erlaubt, diese, selbst bei komplexen Zusammenhängen, anhand von geeigneten Graphen und Netzwerken zu modellieren. Probleme dieser Art werden in der Kategorie **Netzwerkprobleme** zusammengefasst. Innerhalb dieser Kategorie nimmt das zuletzt aufgeführte *MCFP* eine zentrale Position ein, da es möglich ist, eine Vielzahl anderer Netzwerkprobleme auf dieses zurückzuführen.

Obwohl die oben skizzierten Netzwerkprobleme eine spezielle Klasse linearer Programme bilden, ist eine Darstellung dieser Probleme als lineare Programme denkbar ungünstig, da die resultierenden Programme sehr groß und mit einer immensen Degeneriertheit behaftet sind. Somit ist eine Lösung dieser Netzwerkprobleme mittels des traditionellen Simplex Algorithmus nur mit sehr hohem Zeit- und Rechenaufwand möglich und folglich nicht diskutabel.

Hier gelang es Dantzig [12] 1951 das Kernkonzept des von ihm entwickelten Simplex Algorithmus auf die strukturellen Besonderheiten von Netzwerken zu transformieren und somit eine graphentheoretische Spezialisierung des Verfahrens, den sogenannten **Netzwerk Simplex Algorithmus**, zu entwickeln. Obwohl sich dieser in zahlreichen Varianten in der Praxis bewährt hat, blieb es über lange Zeit ein offenes Problem, die polynomiale Beschränktheit der Komplexität eines Netzwerk Simplex Algorithmus für das MCFP nachzuweisen. Dieses Problem wurde 1995 von Orlin gelöst, der mit dem sogenannten **Premultiplier Algorithmus** eine Spezialform des Netzwerk Simplex Algorithmus entwickelte, welcher für ein MCFP mit polynomialer Komplexität eine Optimallösung generiert.

Das **Ziel der vorliegenden Arbeit** besteht darin, durch systematisches Vorgehen ein weit reichendes Verständnis für die Problematik des Netzwerk Simplex Algorithmus zur Lösung eines MCFP zu schaffen, die Einflussgrößen der Komplexität des Verfahrens darzulegen und aufbauend auf den erzielten Befunden die Polynomialität des Premultiplier Algorithmus nachzuweisen. Ferner soll auf Grundlage geeigneter Datenstrukturen eine Implementierung der Algorithmen auf Basis der in der Literatur gebräuchlichen, an PASCAL orientierten Pseudocodes dokumentiert werden.

Hierfür werden wir im ersten Kapitel zunächst die grundlegenden Begrifflichkeiten und Ergebnisse der algorithmischen Graphentheorie und der Flusstheorie bereitstellen.
Eine Einführung in die Problematik des Minimum Cost Flow Problems werden wir im zweiten Kapitel dieser Arbeit aufführen und überdies bereits erste Grundgedanken zur Optimierung des Problems entwickeln.
Im dritten Kapitel werden wir den Netzwerk Simplex Algorithmus in seiner

allgemeinen Form vorstellen und die wesentlichen Einflussgrößen der Laufzeit des Verfahrens diskutieren.
Aufbauend auf diesen Erkenntnissen werden wir im vierten Kapitel dieser Arbeit mit dem Premultiplier Algorithmus eine Spezialform des Netzwerk Simplex Algorithmus kennen lernen und durch eine detaillierte Laufzeitanalyse die theoretische Effizienz des Verfahrens nachweisen.
Eine ausführlich dokumentierte Implementierung der vorgestellten Netzwerk Simplex Algorithmen auf Grundlage von Pseudocodes werden wir in Kapitel fünf dieser Arbeit darlegen.
Mit einer komprimierten Zusammenstellung der entwickelten Ergebnisse, sowie mit Vorschlägen zur Verbesserung bestehender Implementierungen werden wir diese Arbeit abschließen.

Kapitel 1

Grundlagen

In diesem einführenden Kapitel werden wir einige grundlegende Begriffe und Ergebnisse aus dem Bereich der algorithmischen Graphentheorie, sowie der Flusstheorie zusammenstellen, welche im Folgenden als theoretisches Fundament dieser Arbeit dienen werden. Im Fokus der Betrachtung wird hierbei stehen, charakteristische Eigenschaften von Flüssen auf Netzwerken zu studieren, deren Kenntnis für die sich anschließende Diskussion des Minimum Cost Flow Problems eine notwendige Voraussetzung darstellt.

Im ersten Abschnitt dieses Kapitels werden wir zunächst einige elementare Definitionen der Graphentheorie bereitstellen und schließlich den Übergang von der Graphen- zur Flusstheorie einleiten. Der darauf folgende zweite Abschnitt wird sich im Wesentlichen auf die Herleitung bestimmter Optimalitätseigenschaften von maximalen Flüssen auf Netzwerken konzentrieren.

1.1 Graphen, Netzwerke und Flüsse

Bevor wir uns in diesem Abschnitt der Charakterisierung von Netzwerken und Flüssen widmen werden, ist es zunächst notwendig, hierfür die fundamentalsten Begriffe der algorithmischen Graphentheorie bereitzustellen.

1.1.1 Definition (gerichteter Graph). Ein *gerichteter Graph* $G = (V, E)$ ist ein Paar bestehend aus einer Menge V von *Knoten* und einer Menge $E \subseteq V \times V$ von *Kanten*, wobei für jede Kante zwischen dem Anfangs- und Endknoten zu unterscheiden ist. Bezeichnet $e = (v, w)$ eine Kante der Menge E, so heißt $e^- := v$ der *Anfangs-* und $e^+ := w$ der *Endknoten* von e. Schließlich sind durch $|V| = n$ und durch $|E| = m$ die Mächtigkeiten der beiden Mengen V und E gegeben. □

Wir werden im weiteren Verlauf immer davon ausgehen, dass auf den betrachteten Graphen jeder Knoten von jedem anderen Knoten aus über eine Folge von paarweise verschiedenen Kanten, einem so genannten *einfachen Kantenzug* oder *Weg*, erreicht werden kann und die Graphen somit *zusammenhängend* sind. Dies wird keineswegs eine Einschränkung der Allgemeinheit darstellen, da wir bei nicht zusammenhängenden Graphen unsere Betrachtungen für jede Zusammenhangskomponente getrennt durchführen können.

In den folgenden Kapiteln werden wir besonders an speziellen Unterstrukturen gerichteter Graphen interessiert sein, welche durch die nächsten beiden Definitionen beschrieben werden.

1.1.2 Definition (Kreis). Sei $G = (V, E)$ ein gerichteter Graph. Ist durch eine Folge paarweise verschiedener Kanten $(e_1, ..., e_n)$ eine Folge von Knoten $(v_0, ..., v_n)$ gegeben, so dass $e_i = (v_{i-1}, v_i)$ oder $e_i = (v_i, v_{i-1})$ erfüllt ist, so heißt der Kantenzug $(e_1, ..., e_n)$ bzw. die entsprechende Folge von Knoten $(v_0, ..., v_n)$ ein *Kreis*, falls $v_0 = v_n$ Gültigkeit besitzt. Dabei bezeichnet man $e_i = (v_{i-1}, v_i)$ als *Vorwärtskante* und $e_i = (v_i, v_{i-1})$ als *Rückwärtskante* des Kreises. Besteht der Kreis nur aus Vorwärtskanten, so handelt es sich um einen *gerichteten Kreis*. □

1.1.3 Definition (aufspannender Baum). Sei $G = (V, E)$ ein gerichteter Graph. Ein *aufspannender Baum* T des Graphen $G = (V, E)$ kennzeichnet einen speziellen Teilgraphen, welcher die Knotenmenge V vollständig überdeckt, während die Kantenmenge F des Baumes eine Teilmenge der ursprünglichen Kantenmenge E bildet. Zusätzlich hat ein aufspannender Baum T die folgenden beiden Anforderungen zu erfüllen:

- T ist zusammenhängend,
- T ist kreisfrei.

Des Weiteren bezeichnen wir einen Knoten v des Baumes T als *Blatt*, falls v inzident mit genau einer Baumkante der Menge F ist. □

Entsprechend dieser Definition ist offensichtlich, dass für zwei beliebige Knoten der Menge V auf dem aufspannenden Baum T stets ein eindeutiger Weg existiert, welcher diese beiden Knoten verbindet. Somit resultiert das Hinzufügen einer beliebige Kante der Menge $E \backslash F$ zum Baum T in der Bildung eines eindeutigen Kreises. Wird wiederum eine beliebige Kante dieses Kreises entfernt, so liegt erneut ein aufspannender Baum des Graphen $G = (V, E)$

vor. Dieser ist genau dann vom vorigen aufspannenden Baum verschieden, wenn sich die hinzugefügte und die entfernte Kante unterscheiden.

Wir werden uns mit diesen Überlegungen in den folgenden Kapiteln sehr detailliert auseinandersetzen, da sie eine zentrale Rolle bei der Diskussion des Netzwerk Simplex Algorithmus einnehmen.

Auf Grundlage gerichteter Graphen werden wir an dieser Stelle das Konstrukt erweitern, indem wir auf der Kantenmenge E des Graphen $G = (V, E)$ spezielle Abbildungen definieren werden. Somit gelangen wir von Graphen zu sogenannten Netzwerken.

1.1.4 Definition ((Fluss-)Netzwerk). Sei $G = (V, E)$ ein gerichteter Graph. Ist auf der Kantenmenge E des Graphen G eine Abbildung $w : E \to \mathbb{R}$ definiert, so bildet das Paar (G, w) ein *Netzwerk*. Die Funktion w wird in diesem Kontext häufig auch als *Länge, Kapazität, Kosten, Dauer, Gewicht* etc. interpretiert.
Zeichnet die Knotenmenge V zusätzlich zwei besondere Knoten s und t aus, wobei der Knoten s als *Quelle*, der Knoten t als *Senke* bezeichnet wird, so ist durch $N = (G, w, s, t)$ ein *Flussnetzwerk* gegeben.
Ist die Funktion w durch eine Abbildung $c : E \to \mathbb{R}^+$ spezifiziert, welche *obere Kapazitätsfunktion* heißt, so kennzeichnet $N = (G, c, s, t)$ ein *Flussnetzwerk mit oberer Kapazitätsfunktion c*. □

Aufgrund der starken Verwandtschaft der Begriffe werden wir im Folgenden die exakte Abgrenzung der Definitionen Netzwerk, Flussnetzwerk und Flussnetzwerk mit oberer Kapazitätsfunktion etwas vernachlässigen und häufig synonym verwenden.

Auf einem Flussnetzwerk $N = (G, c, s, t)$ lässt sich nun wie folgt ein Fluss definieren.

1.1.5 Definition (Fluss). Es sei $N = (G, c, s, t)$ ein Flussnetzwerk. Eine Abbildung $f : E \to \mathbb{R}$ wird als *Fluss auf N* bezeichnet, falls sie die folgenden beiden Bedingungen erfüllt:

1. *Kapazitätsbeschränkung*:

$$0 \leq f(e) \leq c(e), \quad \text{für alle } e \in E.$$

2. *Flusserhaltung*:

$$\sum_{e^+=v} f(e) = \sum_{e^-=v} f(e), \quad \text{für alle } v \in V \backslash \{s, t\}.$$

Dabei signalisieren uns die Summationsindizes, dass die Summe über alle Kanten $e \in E$ gebildet wird, deren Anfangsknoten e^- bzw. Endknoten e^+ identisch mit dem Knoten v ist.
Schließlich ist der *Wert des Flusses f* durch

$$w(f) = \sum_{e^-=s} f(e) - \sum_{e^+=s} f(e)$$

bestimmt. Ein Fluss f^* heißt *maximal* auf N, falls die Ungleichung $w(f^*) \geq w(f)$ für alle weiteren Flüsse f auf N erfüllt ist. Die Suche nach einem maximalen Fluss auf einem Flussnetzwerk N bezeichnet man als *Maximum Flow Problem (MFP).* □

Durch die Kapazitätsbeschränkungen in Definition 1.1.5 ist für jede Kante $e \in E$ des Netzwerkes N sowohl eine minimale als auch eine maximale Flussmenge vorgegeben, die ein Fluss vom Anfangs- zum Endknoten der betrachteten Kante e transportieren darf. Überdies wird durch die Flusserhaltungsbedingungen für jeden von der Quelle s und der Senke t verschiedenen Knoten sichergestellt, dass die gleiche Flussmenge in den Knoten hinein- wie hinausfließt und somit kein Fluss innerhalb des Netzwerkes „verloren" geht. Ferner beschreibt der Wert eines Flusses f genau diejenige Flussmenge, die netto aus der Quelle s hinausfließt. Schließlich wird durch die Flusserhaltungsbedingungen, welche für alle von s und t verschiedenen Knoten des Netzwerkes zu erfüllen sind, gewährleistet, dass $w(f)$ gleichzeitig derjenigen Flussmenge entspricht, die netto in die Senke t hineinfließt.

1.2 Eigenschaften maximaler Flüsse

Nachdem wir im letzten Abschnitt die für uns relevanten Grundbegriffe der Graphen- und Flusstheorie vorgestellt haben, werden wir uns nun mit der Charakterisierung maximaler Flüsse auf Netzwerken beschäftigen. Bevor wir im weiteren Verlauf konkrete Bedingungen herleiten werden, die einen maximalen Fluss auf einem Netzwerk beschreiben, ist es notwendig, die Existenz solcher Flüsse sicherzustellen. Hierfür sind die folgenden beiden Voraussetzungen zu erfüllen:

1. **Zulässigkeit:** Das Flussnetzwerk muss so beschaffen sein, dass es möglich ist, einen Fluss auf den Kanten des Netzwerkes zu definieren, der sowohl die Kapazitätsrestriktionen, als auch die Flusserhaltungsbedingungen erfüllt.

2. **Beschränktheit:** Es muss eine obere Schranke existieren, die den Wert des Flusses nach oben hin begrenzt. Überdies sollte gewährleistet sein, dass der Wert des Flusses die durch die obere Schranke gegebene Zahl auch annehmen kann.

Die Bedingung der Zulässigkeit stellt für die in diesem Abschnitt betrachteten Flussnetzwerke keine Einschränkung dar, weil durch den Nullfluss $f(e) = 0$ für alle Kanten $e \in E$ stets ein Fluss gegeben ist, der sowohl den Kapazitätsrestriktionen, als auch den Flusserhaltungsbedingungen nachkommt. Demnach sind *MFP* auf diesen Netzwerken immer zulässig.
Für Netzwerke, die zusätzlich zur oberen Kapazitätsschranke c eine beliebige untere Kapazitätsschranke b aufweisen, ist die Zulässigkeit dagegen nicht notwendigerweise gewährleistet. In diesem Fall kann die Zulässigkeit durch geeignete Transformationen der Problemstellung überprüft werden.[1]
Da die Existenz eines Flusses auf den hier betrachteten Netzwerken durch den Nullfluss erwiesen ist, werden wir uns nun mit der Herleitung einer oberen Schranke für den Flusswert beschäftigen. Es ist leicht nachvollziehbar, dass der Wert $\omega(f)$ eines Flusses f stets durch die Zahl $\sum_{e^-=s} c(e)$ nach oben hin begrenzt ist, da jeder Fluss die Kapazitätsrestriktionen einzuhalten hat. Wegen der zusätzlich zu erfüllenden Flusserhaltungsbedingungen ist diese Lösung jedoch im Allgemeinen nicht zulässig. Für die Konstruktion einer oberen Schranke, die der Wert des Flusses auch annehmen kann, benötigen wir zunächst das Konstrukt des Schnittes.

1.2.1 Definition (Schnitt). Sei $N = (G, c, s, t)$ ein Flussnetzwerk. Unter einem *Schnitt* von N versteht man eine Zerlegung $V = S \dot{\cup} T$ der Knotenmenge V in zwei disjunkte Teilmengen S und T, wobei $s \in S$ und $t \in T$.
Dabei ist die *Kapazität* des Schnittes durch die Zahl

$$c(S,T) = \sum_{\substack{e^- \in S \\ e^+ \in T}} c(e)$$

gegeben. Ein Schnitt (S,T) heißt *minimal auf* N, wenn für alle weiteren Schnitte (S',T') von N die Bedingung $c(S,T) \leq c(S',T')$ erfüllt ist. □

Die Kapazität eines Schnittes beschreibt demnach anschaulich die maximale Flussmenge, die bei gegebener Knotenpartition von der Knotenmenge S zur Knotenmenge T transportiert werden kann, ohne die Kapazitätsbeschränkungen des zugrunde liegenden Netzwerkes zu verletzen. Aus dieser Überlegung heraus können wir nun eine obere Schranke für den Flusswert herleiten.

[1] vergleiche Ahuja et al. [1] S.193

1.2.2 Lemma. *Sei $N = (G, c, s, t)$ ein Flussnetzwerk und f bezeichne einen Fluss auf N. Ist durch (S, T) ein beliebiger Schnitt von N gegeben, so ist die Ungleichung*

$$w(f) \leq c(S, T)$$

stets erfüllt.

Beweis. Nach Definition 1.1.5 ist der Wert des Flusses f durch die Gleichung

$$w(f) = \sum_{e^- = s} f(e) - \sum_{e^+ = s} f(e)$$

gegeben. Da die Partitionsmenge S des Schnittes die Quelle s, nicht aber die Senke t enthält, folgt mit den Flusserhaltungsbedingungen:

$$\begin{aligned} w(f) &= \sum_{v \in S} \left(\sum_{e^- = v} f(e) - \sum_{e^+ = v} f(e) \right) = \\ &= \sum_{\substack{e^- \in S \\ e^+ \in S}} f(e) - \sum_{\substack{e^+ \in S \\ e^- \in S}} f(e) + \sum_{\substack{e^- \in S \\ e^+ \in T}} f(e) - \sum_{\substack{e^+ \in S \\ e^- \in T}} f(e) = \\ &= \sum_{\substack{e^- \in S \\ e^+ \in T}} f(e) - \sum_{\substack{e^+ \in S \\ e^- \in T}} f(e). \end{aligned}$$

Mit Hilfe der Kapazitätsbeschränkungen lässt sich nun aus obiger Gleichungskette die folgende Abschätzung herleiten:

$$w(f) = \sum_{\substack{e^- \in S \\ e^+ \in T}} f(e) - \sum_{\substack{e^+ \in S \\ e^- \in T}} f(e) \leq \sum_{\substack{e^- \in S \\ e^+ \in T}} c(e) - \sum_{\substack{e^+ \in S \\ e^- \in T}} 0 = c(S, T),$$

so dass die Behauptung folgt. □

Da nun erwiesen ist, dass der Wert eines Flusses f auf einem Netzwerk N stets durch die Kapazität eines Schnittes (S, T) nach oben hin beschränkt ist, stellt sich nun die Frage, wie genau obige Abschätzung für den Wert eines maximalen Flusses ist. In der Absicht, dieser Frage nachzugehen, bedarf es zunächst einer Charakterisierung maximaler Flüsse, wofür die folgende Definition die Grundlage bildet.

1.2.3 Definition (augmentierender Weg). Sei f ein Fluss auf einem Flussnetzwerk $N = (G, c, s, t)$. Ein Weg P von einem Knoten $v \in V$ zu einem anderen Knoten $w \in V$ wird als *augmentierender* oder *zunehmender Weg* bezüglich f bezeichnet, falls die beiden folgenden Bedingungen erfüllt sind: