MULTICOMMODITY FLOW

ÜBER MEHRGÜTERFLÜSSE UND MEHRFACHSCHNITTE

STEFFEN KIONKE

18. September 2007

Ausarbeitung zum Vortrag bei der Sommerakademie 2007 in Görlitz.

Ziel dieses Artikels ist es, zwei Probleme (das Mehrgüterfluss- und das Mehrfachschnittproblem) vorzustellen und einen Approximationsalgorithmus für eben diese Probleme anzugeben und zu diskutieren. Dazu wird zunächst das sogenannte *Primal-Dual-Schema* kurz zu vorgestellt, welches in der Konstruktion unseres Algorithmuses Verwendung findet. Es wird vorausgesetzt, dass der Leser mit den Grundkonzepten der linearen Programmierung vertraut ist.

1. Das Primal-Dual-Schema

Beginnen wir also mit dem theoretischen Teil dieses Artikels. Wir betrachten im Folgenden ein lineares Programm

$$\begin{aligned} \text{minimiere} \quad & c^T x \\ \text{unter den Bedingungen} \quad & Ax \geq b \\ & \forall\, j \in \{1, \ldots, n\} \quad x_j \geq 0 \end{aligned}$$

welches wir auch das *primale* Programm nennen. Es sei auch nochmals erwähnt, dass jeder Vektor x, welcher die Nebenbedingungen erfüllt (aber nicht notwendigerweise minimal ist), *zulässige* Lösung des Programms genannt wird. Wir bilden außerdem das zugehörige duale Programm:

$$\begin{aligned} \text{maximiere} \quad & b^T y \\ \text{unter den Bedingungen} \quad & A^T y \leq c \\ & \forall\, i \in \{1, \ldots, m\} \quad y_i \geq 0 \end{aligned}$$

Betrachten wir nun den (hoffentlich) bekannten Satz über den komplementären Schlupf

Satz 1 (vom komplementären Schlupf)**.** *Seien $x = (x_1, \ldots, x_n)$ und $y = (y_1, \ldots, y_m)$ zulässige Lösungen für ein primales und sein duales lineares Programm. Beide Lösungen sind optimal genau dann, wenn gilt:*

(1) *Für alle $j \in \{1, \ldots, n\}$ gilt $x_j = 0$ oder $\sum_{i=1}^{m} a_{ij} y_i = c_j$*
(2) *Für alle $i \in \{1, \ldots, m\}$ gilt $y_i = 0$ oder $\sum_{j=1}^{n} a_{ij} x_j = b_i$*

Wobei wir die Bedingungen primal *bzw.* dual *nennen wollen.*

Die Idee hinter dem Primal-Dual-Schema, welches nun erläutert werden soll, ist folgende: Eine gute, aber nicht optimale Lösung des linearen Programms erfüllt diese Schlupfbedingungen nicht exakt, aber *fast*. Es soll nun eben formalisiert werden, was *fast* in diesem Zusammenhang heißen soll.

Geben wir uns also zwei reelle Zahlen $\alpha, \beta \geq 1$ vor. Wir wollen diese verwenden, um die Schlupfbedingungen abzuschwächen. Denken wir uns die Gleichheit in den Schlupfbedingungen als zwei Ungleichungen, so wird klar, dass nur diejenigen dieser Ungleichungen relaxiert werden müssen, welche nicht aus der Zulässigkeit einer Lösung folgen. Wir nennen also die folgenden Bedingungen *α-β relaxierte* Schlupfbedingungen.

(1) Für alle $j \in \{1, \ldots, n\}$ gilt $x_j = 0$ oder $\frac{c_j}{\alpha} \leq \sum_{i=1}^{m} a_{ij} y_i \leq c_j$

(2) Für alle $i \in \{1, \ldots, m\}$ gilt $y_i = 0$ oder $b_i \leq \sum_{j=1}^{n} a_{ij} x_j \leq \beta b_i$

Unter dem Begriff *Primal-Dual-Schema* werden nun Vorgehensweisen zusammengefasst, die für primales und duales Programm ein Paar zulässiger Lösungen finden, die immer α-β *relaxierten* Schlupfbedingungen genügen. Der folgende Satz zeigt uns direkt welche Güte solche Lösungen haben.

Satz 2. *Seien x und y zulässige Lösungen des primalen bzw. dualen Programms welche α-β relaxierten Schlupfbedigungen genügen, so gilt*

$$\sum_{j=1}^{n} c_j x_j \;\leq\; \alpha\beta \sum_{i=1}^{m} b_i y_i$$

In anderen Worten: Konstruiert ein Algorithmus für ein primales und duales lineares Progamm α-β relaxierte Lösungen, so wissen wir, dass dieser Algorithmus die Approximationsgüte $\alpha\beta$ besitzt.

Beweis. Der Beweis besteht aus drei Schritten. Wir verwenden zuerst die primale relaxierte Schlupfbedingung, dann tauschen wir die Summen und verwenden den dualen relaxierten Schlupf.

$$\begin{aligned}\sum_{j=1}^{n} c_j x_j \;&\leq\; \sum_{j=1}^{n} \alpha \sum_{i=1}^{m} a_{ij} y_i x_j \;=\; \alpha \sum_{i=1}^{m} y_i \sum_{j=1}^{n} a_{ij} x_j \\ &\leq\; \alpha\beta \sum_{i=1}^{m} y_i b_i\end{aligned}$$

□

2. Mehrgüterfluss- und Mehrfachschnittproblem

Der theoretische Teil ist nun abgeschlossen und wir wollen uns zwei konkreten Problemen widmen, die nun vorgestellt werden. Beginnen wir mit dem *ganzzahligen Mehrgüterfluss-Problem*, das wir im Folgenden kurz *IMF* (integer multicommodity flow) nennen wollen.

Problem 3 (IMF)**.** *Gegeben sei ein ungerichteter Graph $G = (V, E)$ mit positiven ganzzahligen Kantengewichten (genannt* Kapazitäten*) c_e für jede Kante $e \in E$. Weiter sei eine Menge von Paaren von Knoten $L = \{ (s_1, t_1), \ldots, (s_k, t_k) \}$ gegeben. Für alle $i = 1, \ldots, k$ soll ein* Gut *von s_i nach t_i transportiert werden.*
Ziel ist es möglichst viele Güter gleichzeitig zu versenden, ohne die Kapazitäten der Kanten zu überschreiten. Die Quantitäten der versendeten Güter müssen dabei ganzzahlig sein.
Kurz: Der gesamte Güterfluss soll ganzzahlig maximiert werden.

Dieses Problem hat natürlich vielfache praktische Bedeutung, der Leser möge an Routing- oder Vekehrsflussmodelle denken. Gehen wir aber sofort weiter zu einem zweiten Problem, dem minimalen Mehrfachschnittproblem (kurz *MM*).

Wir betrachten wieder einen ungerichteten Graphen $G = (V, E)$ mit positiven ganzzahligen Kantengewichten $c_e \in \mathbb{N}$. Weiter sei eine Menge von Knotenpaaren $L = \{ (s_1, t_1), \ldots, (s_k, t_k) \}$ gegeben. Wir starten mit einer Definition.

Definition 4 (multicut)**.** *Ein* Mehrfachschnitt *(multicut) ist eine Teilmenge $D \subseteq E$ der Kanten so, dass das Entfernen der Kanten in D jeweils s_i von t_i trennt.*
Weiter nennen wir $c(D) \overset{def}{=} \sum_{e \in D} c_e$ die Kosten *des Schnittes.*

Das zweite Problem mit dem wir uns also beschäftigen wollen ist folgendes:

Problem 5 (MM)**.** *Zu gegebenem Graphen G finde einen Mehrfachschnitt D mit minimalen Kosten $c(D)$.*

Der Grund warum wir diese Probleme gemeinsam betrachten wird hoffentlich bald klar werden. Es sei aber bereits erwähnt, dass es sich gewissermaßen um LP duale Probleme handelt. Dies ist auch der Grund warum wir später das Primal-Dual-Schema anwenden wollen.

Außerdem sei noch auf dem Sonderfall $k = 1$ verwiesen. Dieser lässt sich in polynomieller Zeit bearbeiten. Varianten des *Algorithmus von Ford und Fulkerson* tun dies beispielsweise in $O(|V||E|^2)$. Des Weiteren sei an das *MinCut-MaxFlow*-Theorem erinnert, das für den Fall $k = 1$ Gleichheit zwischen maximalem Fluss und den Kosten des minimalen Schnittes herstellt. Im allgemeinen Fall ist ein solcher Zusammenhang nicht gegeben. Außerdem ist eine effiziente Lösung nicht zu erwarten, denn es handelt sich bei beiden Problemen für $k \geq 3$ um NP-schwere Probleme. Wir werden die Probleme des Mehrfachschnittes (MM) und des Mehrgüterflusses (IMF) im Folgenden auf *Bäume* einschränken. Dies stellt eine deutliche Vereinfachung dar, allerdings werden die Probleme dadurch sicherlich nicht trivial. MM ist selbst für Bäume der Höhe 1 und Kanten mit Gewicht 1 noch *NP*-schwer. Auch das Flussproblem IMF ist für Bäume der Höhe 3 bereits *NP*-schwer. Folglich werden wir auch für diese Probleme nur schwer exakte Lösungen finden. Eine Eigenschaft von Bäumen wird es uns aber ermöglichen dieses Problem effizient zu approximieren. Diese Eigenschaft ist die folgende: Zwischen zwei Knoten gibt es immer *genau einen* einfachen Weg. Das heißt eine Menge von Kanten ist ein Mehrfachschnitt, wenn für alle i auf *dem* Weg s_i nach t_i mindestens eine Kante ausgewählt ist.
Wir wollen noch kurz verwendete Bezeichnungen einführen. Sei G ein gewurzelter Baum und u, v zwei beliebige Knoten. Die *Tiefe* von u ist die Länge des Pfades von u zur Wurzel. Außerdem nennen wir den höchsten Knoten auf dem Weg von u nach v den *tiefsten gemeinsamen Vorfahren* $lca(u, v)$.

3. Formulierung als lineares Programm

Beginnen wir nun mit der Konstruktion eines Approximationsalgorithmus für unsere beiden Probleme. Gegeben sei eine Instanz von MM mit k Paaren (s_i, t_i). Wir bezeichnen mit P_i die Menge aller Kanten auf dem Weg von s_i nach t_i. Wir wollen das Mehrfachschnittproblem als lineares Programm formulieren. Dazu führen wir die Variablen $d_e \in \{0, 1\}$ für alle $e \in E$ ein. Wir erhalten einen Schnitt D nach Lösung des linearen Programms über die Beziehung $e \in D \iff d_e = 1$. Das ganzzahlige lineare Programm hat dann die Form:

$$\begin{aligned} \text{minimiere} \quad & \sum_{e \in E} c_e d_e \\ \text{u.d.B.} \quad & \forall\, i \in \{1, \ldots, k\} \sum_{e \in P_i} d_e \geq 1 \\ & \forall\, e \in E \; d_e \in \{0, 1\} \end{aligned}$$

Wir minimieren die Schnittkosten und die Nebenbedingungen garantieren uns, dass es sich um einen Schnitt handelt. Wir relaxieren das ganzzahlige Programm, indem wir die letzte Zeile durch

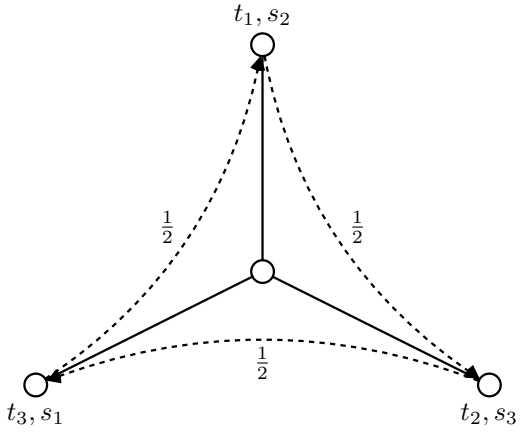
$$\forall\, e \in E \; d_e \geq 0$$

ersetzen.
Als nächstes betrachten wir das zugehörige duale Programm, welches gegeben ist durch:

$$\begin{aligned} \text{maximiere} \quad & \sum_{i=1}^{k} f_i \\ \text{u.d.B.} \quad & \forall\, e \in E \sum_{\substack{i=1 \\ e \in P_i}}^{k} f_i \leq c_e \\ & \forall\, i \in \{1, \ldots, k\} \; f_i \geq 0 \end{aligned}$$

Es ist naheliegend die neu eingeführten Variablen f_i als Fluss zu interpretieren. Die Bedingungen garantieren die Einhaltung der Kapazitäten auf jeder Kante. Wir erhalten eine LP-Form unsers Mehrgüterflussproblems mit dem Unterschied, dass ganzzahlige Flusswerte nicht verlangt sind. Wir werden dieses Problem mit *FMF* (fractional multicommodity flow) bezeichnen.
Betrachten wir zunächst ein Beispiel, an dem wir erkennen können, dass eine Aussage wie die des

MinCut-MaxFlow Theorems nicht zu erhalten sind. Man sieht auch, dass die optimale rationale Lösung nicht ganzzahlig sein muss.



Man erkennt leicht, dass der minimale Mehrfachschnitt (MM) genau 2 Kanten benötigt, der maximale ganzzahlige Fluss (IMF) allerdings genau 1 beträgt, weil dann schon 2 Kanten gesättigt sind. Der maximale rationale Fluss (FMF) beträgt allerdings $\frac{3}{2}$, wenn man für jedes Paar den Fluss $\frac{1}{2}$ wählt.

4. Ein 2-Approximationsalgorithmus

Wir wollen nun einen 2-Approximationsalgorithmus vorstellen. Unser Ziel ist es das Primal-Dual-Schema anzuwenden. Wir wählen hierzu $\alpha = 1$ und $\beta = 2$. Dadurch erhalten wir die folgenden relaxierten Schlupfbedingungen:

(1) Für alle $e \in E$ gilt $d_e = 0$ oder $\sum_{\substack{i=1 \\ e \in P_i}}^{k} f_i \ = \ c_e$.

(2) Für alle $i \in \{1, \ldots, k\}$ gilt $f_i = 0$ oder $1 \ \leq \ \sum_{e \in P_i} d_e \ \leq \ 2$.

Versuchen wir die relaxierten Bedingugen in unserem Kontext zu interpretieren, so wird ersichtlich, dass wir einen Schnitt suchen, der nur gesättigte (Fluss = Kapazität) Kanten enthält und für jedes Paar (s_i, t_i) mit positivem Fluss höchstens zwei Kanten auf dem Weg s_i nach t_i im Schnitt sein dürfen.
Der Algorithmus, den wir jetzt angeben werden, wird genau diese Bedingungen erfüllen und natürlich zulässig ganzzahlige Lösungen konstruieren. Dazu sind drei Schritte notwendig. Nach einer kurzen Initialisation folgt die so genannte *flow routing* Phase die dazu da ist die primale Bedingung zu erfüllen. Danach kommt ein *reverse delete* Schritt der die duale Bedingung erfüllen wird.

Algorithmus 6. *Initialisation:*
wähle eine beliebige Wurzel r;
for $i = 1$ *to* k *do* $f_i := 0$*;*
$D := \emptyset$*;*

Flow routing:
Ordne die Knoten in abnehmender Tiefe: $v_1, \ldots, v_n$*;*
for $i = 1$ *to* n *do*
  for $j = 1$ *to* k *do*
    if $lca(s_j, t_j) = v_i$ *then*
      -sende soviel Fluss f_j *wie möglich*
      von s_j *nach* t_j*;*
      -füge jede neu gesättigte Kante zu D hinzu;

Reverse delete:

Seien $e_1, \ldots, e_p$ die Kanten in D in der Reihenfolge in der sie zugefügt wurden;

for $i = p$ *downto 1 do*
    wenn $D \setminus \{e_i\}$ *noch multicut ist* dann
        lösche e_i aus D ;

5. Analyse

Wir wollen nun den vorgestellten Algorithmus analysieren und beweisen, dass dieser die Approximationsgüte 2 besitzt. Wir werden schlussendlich auf Satz 2 zurückgreifen. Um dies zu tun, ist zu zeigen, dass alle vom Algorithmus ausgegebenen Lösungen die relaxierten Schlupfbedingungen erfüllen. Außerdem müssen wir einsehen, dass alle Lösungen zulässig und ganzzahlig sind, was jedoch einfach ist. Zeigen wir zunächst, dass der duale relaxierte Schlupf erfüllt ist. Dazu zitieren wir folgendes Lemma aus [V2001].

Lemma 7. *Sei (s_i, t_i) ein* source-sink *Paar mit positivem Fluss ($f_i \neq 0$) und sei $lca(s_i, t_i) = v$. Dann enthält D höchstens je eine Kante in beiden Pfaden s_i nach v und v nach t_i.*

Der Beweis erfolgt doch Widerspruch und ist nicht kompliziert, allerdings muss er sorgfälltig Schritt für Schritt durchgeführt werden. Aus Platzgründen verzichten wir darauf und verweisen nochmals auf [V2001] Kapitel 18. Aus diesem Lemma folgt natürlich direkt, dass für jedes Paar (s_i, t_i) gilt $1 \leq |D \cap P_i| \leq 2$, sofern $f_i > 0$. Dies ist gerade die duale relaxierte Schlupfbedingung.

Satz 8. *Der angegebene Algorithmus erreicht für beide Probleme (MM und IMF auf Bäumen) die Approximationsgüte* 2*.*

Beweis. Wir wollen Satz 2 anwenden. Dazu müssen wir zunächst erkennen, dass die erzeugten Lösungen zulässig sind. Für den Fluss ist das offensichtlich, da die *flow routing* Phase nur positiven Fluss zulässt und Kapazitäten respektiert. Die Menge D ist vor Beginn des *reverse delete* Schrittes ein Schnitt, da auf jedem Weg s_i nach t_i mindestens eine Kante gesättigt wurde und somit in D ist. Der *reverse delete* Schritt entfernt nur Kanten, wenn D dann weiterhin ein Schnitt bleibt. Folglich ist D auch am Ende der Ausführung ein Schnitt.
Wenden wir uns jetzt den relaxierten Schlupfbedingungen zu. Der primale Schlupf ist offensichtlich erfüllt, da wir zu D nur gesättigte Kanten hinzufügen. Der relaxierte duale Schlupf ist wegen Lemma 7 erfüllt.
Wir haben folglich einen 2-Approximationsalgorithmus, wobei wir noch erwähnen wollen, dass die konstruierten Lösungen offensichtlich ganzzahlig sind. □

Betrachten wir nochmals kurz den Algorithmus, so fällt natürlich auf, dass der Fluss nur in der *flow routing* Phase verändert wird. Wenn wir also nur eine Näherungslösung für den maximalen Fluss wollen, so genügt es den Algorithmus nach der *flow routing* Phase zu beenden und D zu ignorieren. Für das Mehrfachschnittproblem ist der *reverse delete* Schritt allerding essentiell. Man erkennt leicht, dass ohne löschen von Kanten der Schnitt beliebing schlecht werden kann. Auch ein *forward delete* funktioniert nicht, in dem Sinn, dass die Approximation an den minimalen Schnitt beliebig schlecht werden kann. Man kann sich diesen Sachverhalt vereinfacht vorstellen, indem man sich klarmacht, dass tendenziell teurere Kanten später gesättigt werden und deshalb bei *reverse delete* zuerst betrachtet werden. Eine andere Weise zu löschen ist folgende: Sortiere alle Kanten in D nach absteigenden Kosten und sortiere der Reihe nach aus, wenn die Restmenge immer noch ein Mehrfachschnitt ist. Verwendet man diese Löschmethode, so erhält man mit einem ähnlichen Beweis auch Lemma 7. Man bekommt so also auch einen 2-Approximationsalgorithmus.

Die Anwendung des Primal-Dual-Schemas scheint in unserem Fall sehr elegant und vielversprechend zu sein. Tatsächlich aber hat die Anwendung von Satz 2 im Beweis von Satz 8 eine wichtige Konsequenz, denn Satz 2 stellt nicht nur eine Beziehung der Lösung zum Optimum her, sondern eine Beziehung der Lösungen untereinander.

Folgerung 9. *Für jeden Baum mit ganzzahligen Kantengewichten und gegebener Menge von Knotenpaaren L gilt immer*

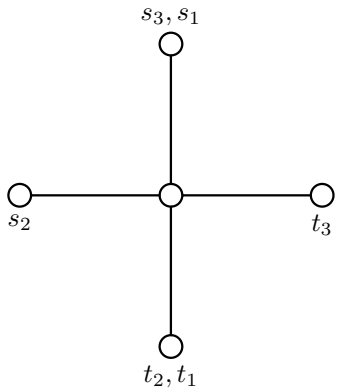
$$OPT_{IMF} \leq \min_{multicut\ D} c(D) \leq 2\,OPT_{IMF}$$

Wobei OPT_{IMF} den maximalen ganzzahligen Fluss bezeichnet.

Daraus ergibt sich sofort, dass eine Anwendung des Primal-Dual-Schemas nur Sinn ergibt, sofern das zugrundeliegende Problem eine entsprechende Struktur aufweist. Kurz gesagt: die beiden ganzzahligen Lösungen dürfen nur um den Faktor $\alpha\beta$ auseinander liegen. Ein Indikator dafür ist zum Beispiel die Ganzzahligkeitslücke. Wächst diese in Abhängigkeit von der Eingabegröße, so ist eine Anwendung des Primal-Dual-Schemas mit konstanten α und β nicht möglich. Das ist auch der Grund, warum unsere Technik nicht für allgemeine Graphen angewandt werden kann. Es ist einfach eine Folge planarer Graphen anzugeben, sodass die Ganzzahligkeitslücke für IMF mit der Eingabegröße wächst.

Lemma 10. *Die Mehrgüterfluss-Ganzzahligkeitslücke in Abhängigkeit von der Anzahl der Paare k auf allgemeinen Graphen ist mindestens $\frac{k}{2}$.*

Ein entsprechendes Beispiel findet sich in Vaziranis Buch [V2001] in Kapitel 18. Beide Probleme werden in allgemeinen Graphen wesentlich komplizierter. Für MM in allgemeinen Graphen gibt es Algorithmen der relativen Güte $O(\log k)$ (siehe [V2001]). Wir wollen uns zum Schluss noch die Frage stellen, ob die Güteabschätzung für unseren Algorithmus berhaupt scharf ist. In der Tat ist sie das, betrachten wir folgenden Zeugen:



Ist die Wurzel der Knoten in der Mitte, so kann es passieren, dass der Algorithmus zunächst von s_1 nach t_1 gierig *routet*. Dann sind die anderen Verbindungen dicht und der gefundene Fluss beträgt 1. Der maximale Fluss beträgt allerdings 2, indem man den Fluss zwischen (s_2, t_2) und (s_3, t_3) gleich 1 wählt. Man kann nun beliebig große Beispiele konstruieren, indem man mehrere solcher *Bausteine* in einen Baum packt.

Literatur

[V2001] Vazirani, V., *Approximation Algorithms*. Springer-Verlag, Berlin, 2001.
ISBN: 3-540-65367-8