**Seminar über Algorithmen** 12.11.2013

# Kombinatorische Algorithmen zur Berechnung von Marktequilibria

*Sebastian Stugk* *Wolfgang Mulzer, Yannik Stein*

## 1 Marktmodelle und Gleichgewichtsdefinition

Die folgenden Seiten befassen sich mit dem Fisher-Marktmodell einem Spezialfall der Arrow-Debreu-Tauschökonomien (Definition 1) und stellen einen kombinatorischen Algorithmus zur Berechnung von Gleichgewichten in diesen Märkten.

**Definition 1.** *Ein **Fisher-Markt** besteht aus einer Menge $A$ von unterscheidbaren und teilbaren Gütern und einer Menge von Käufern $B$. Seien $|A| = n$ und $|B| = m$. Jeder Käufer ist mit einer Menge von Geld $e_i$ ausgestattet. Jedes Produkt ist einer Stückzahl von $b_i$ Einheiten verfügbar. Jeder Käufer hat eine konkave Nutzenfunktion $u_i : \mathbb{R}^n_+ \to \mathbb{R}$ und ein Guthaben der Höhe $e_i > 0$. Ein **Marktgleichgewicht** in diesem Modell ist ein Preisvektor $p = (p_1, \ldots, p_n)^\top \in \mathbb{R}^n_+$ (Market-Clearing-Preise), sodass für jeden Käufer $i$ eine Menge von Produkten $x_i^* = (x_{i1}^*, \ldots, x_{in}^*)^\top \in \mathbb{R}^n_+$ existiert und die folgenden beiden Bedingungen erfüllt sind:*

1. *Der Vektor $x_i^*$ maximiert $u_i(x)$ mit Rücksicht auf die Budgetbeschränkung $p \cdot x \leq e_i$.*
2. *Für jedes Produkt $j$ gilt $\sum_i x_{ij}^* = b_j$.*

Den folgenden Betrachtungen liegt die Einschränkung zugrunde, dass die Nutzenfunktionen $u_i(x)$ linear sind, d.h. $u_i(x) = \sum_{i=1}^n u_{ij} x_{ij}$ wobei $u_{ij} \in \mathbb{R}_+$ der Nutzen ist, den Käufer $i$ aus dem Kauf einer Einheit von Produkt $j$ zieht.

## 2 Das Eisenberg-Gale-Programm

Ein mögliches Mittel um die gegebenen Gleichgewichtsbedingungen zu formalisieren sind konvexe Programme, d.h Optimierungsprobleme mit konvexer Zielfunktion und linearen Nebenbedingungen. Das im Folgenden vorgestellte Eisenberg-Gale-Programm dient als Grundlage für den anschließend vorgestellten kombinatorischen Algorithmus zur Bestimmung des Marktgleichgewichts.

$$\begin{aligned} \max \quad & \sum_{i=1}^{m} e_i \log u_i & & \\ \text{mit} \quad & u_i = \sum_{j=1}^{n} u_{ij} x_{ij} & & \forall i \in B \\ \text{unter Nb.} \quad & \sum_{i=1}^{m} x_{ij} \leq 1 & & \forall j \in A \\ \text{und Vzb.} \quad & x_{ij} \geq 0 & & \forall i \in B, \forall j \in B \end{aligned} \tag{EG}$$

Wir setzen voraus, dass zu jedem Produkt $j$ ein potenzieller Käufer existiert, d.h. $u_{ij} > 0$ für ein $i \in B$. Als zugehöriges duales Lagrange-Problem formulieren wir:

$$\begin{aligned} \max_{p} \quad & L(x,p) = -\sum_{i=1}^{m} e_i \log u_i + \sum_{j=1}^{n} p_j f_j(x) \\ \text{mit} \quad & u_i = \sum_{j=1}^{n} u_{ij} x_{ij} && \forall i \in B \\ & f_j(x) = (\sum_{i=1}^{m} x_{ij}) - 1 && \forall j \in A \\ \text{unter Vzb.} \quad & p_i \geq 0 \end{aligned} \tag{$\text{EG}_D$}$$

Wie der Beweis von Satz 2 zeigt erfüllen die optimale Lösung $x^*$ von EG und die optimale Lösung $p^*$ von $\text{EG}_D$ die in Definition 1 formulierten Gleichgewichtsbedingungen. Wir greifen dabei auf die im Folgenden aufgeführten Bedingungen zurück, die aus den Karush-Kuhn-Tucker-Optimalitätsbedingungen (KKT-Bedingungen) für primal optimale Lösung $x^*$ und dual optimale Lösung $p^*$ folgen:

**(i)** $\forall j \in A : p_j \geq 0$

**(ii)** $\forall j \in A : p_j > 0 \Rightarrow \sum_{i=1}^{m} x_{ij} = 1$

**(iii)** $\forall i \in B \; \forall j \in A : \frac{u_{ij}}{p_j} \leq \frac{\sum_{j \in A} u_{ij} x_{ij}}{e_i}$

**(iv)** $\forall i \in B \; \forall j \in A : x_{ij} > 0 \Rightarrow \frac{u_{ij}}{p_j} = \frac{\sum_{j \in A} u_{ij} x_{ij}}{e_i}$

**Satz 2.** *Primal und dual optimale Lösung des Eisenberg-Gale-Programms $x^*$ und $p^*$ erfüllen die Marktgleichgewichtsbedingungen.*

*Beweis.* Da zu jedem Produkt $j$ ein Käufer $i$ mit $u_{ij} > 0$ existiert können wir aus Bedingung (ii) und (iii) folgern, dass alle Preise $p_j$ positiv sind und alle Güter verkauft werden können:

$$\begin{aligned} & p_j \geq \frac{e_i u_{ij}}{\sum_j u_{ij} x_{ij}} > 0 \\ & p_j > 0 \qquad \Rightarrow \qquad \sum_{i=1}^{m} x_{ij} = 1 \end{aligned}$$

Aus Bedingung (iii) und (iv) folgt direkt, dass jeder Käufer die Menge an Gütern erhält, die seinen Nutzen pro Einheit investiertem Guthaben und damit auch seine Nutzenfunktion $u_i(x)$ bezüglich $A$ maximiert. Aus Bedingung (iv) können wir außerdem ableiten, dass die Guthaben aller Käufer vollständig aufgebraucht werden:

$$\begin{aligned} & \forall i \in B \; \forall j \in A : \frac{e_i u_{ij} x_{ij}}{\sum_{j \in A} u_{ij} x_{ij}} = p_j x_{ij} \\ \Rightarrow \qquad & \forall i \in B : \frac{e_i \sum_j u_{ij} x_{ij}}{\sum_{j \in A} u_{ij} x_{ij}} = \sum_j p_j x_{ij} \\ \Rightarrow \qquad & \forall i \in B : e_i = \sum_j p_j x_{ij} \end{aligned}$$

□

# 3 Kombinatorische Bestimmung des Optimums von EG

Die Grundidee des Algorithmus besteht besteht in der Berechnung des Optimimums von EG und $\text{EG}_D$ durch Berechnung von maximalen Netzwerkflüssen. Dabei wird durch Wartung einer einer Invariante (siehe Abschnitt 3.2) sichergestellt, dass die Gleichgewichtspreise zu keinerzeit überschritten werden. Im Gegenzug werden die dritte und vierte der vorgestellten Optimalitätsbedingung zunächst wie folgt zu Laufzeitbedingungen (iii') und (iv') abgeschwächt:

**(iii')** $\forall i \in B, \forall j \in A : \frac{u_{ij}}{p_j} \leq \frac{\sum_{j \in A} u_{ij} x_{ij}}{m_i}$

**(iv')** $\forall i \in B, \forall j \in A : x_{ij} > 0 \Rightarrow \frac{u_{ij}}{p_j} = \frac{\sum_{j \in A} u_{ij} x_{ij}}{m_i}$

Im Laufe des Algorithmus wird der Wert der folgenden Potenzialfunktion kontinuierlich reduziert:

$$\Phi = \varphi_1^2 + \varphi_2^2 + \cdots + \varphi_m^2 \qquad \text{mit} \qquad \varphi_i = e_i - m_i$$

$m_i$ bezeichnet hier die Geldmenge, die Käufer $i$ ausgegeben hat. Wenn die Potentialfunktion den Wert 0 erreicht, sind alle 4 Originalbedingungen erfüllt.

## 3.1 Verifizieren von Gleichgewichtspreisen

Sei der Preisvektor $p = (p_1, \ldots, p_n)^\top \subseteq \mathbb{R}_+^n$ fest gegeben. Der *Maximalnutzen pro Einheit Geld* eines Käufers $i$ ist $\alpha_i = \max_j\{\frac{u_{ij}}{p_j}\}$. Produkte $p_j$, für die $\frac{u_{ij}}{p_j} = \alpha_i$ gilt, machen Käufer $i$ am glücklichsten. Sei $C = \{(a,b)|\frac{u_{ij}}{p_j} = \alpha_i\}$ die Menge der Paare von Käufern und Produkten mit Maximalnutzen pro Einheit Geld. Wir bezeichnen $G = (V, E)$ mit $V = A \cup B$ und $E = C$ als *Equality Subgraph* und darauf basierend das folgende Flussnetzwerk $N(p)$:

- $V = \{s\} \cup A \cup B \cup \{t\}$ für die Produktmenge $A$, die Menge der Käufer $B$, Quelle $s$ und Senke $t$
- $E = \{(s,a)|a \in A\} \cup \{(b,t)|b \in B\} \cup C$
- Kapazitäten $c(s,a) = p_i$ für alle $a \in A$, $c(b,t) = e_i$ für alle $b \in B$ und $c(a,b) = \infty$ für $(a,b) \in C$.

Abbildung 3.1 zeigt eine Veranschaulichung des Flussnetzwerks $N(p)$. Die Berechnung der maximalen Menge von Produkten, die bei gegebenen Preisen verkauft werden kann, entspricht der Berechnung eines maximalen Netzwerkflusses. Für einen gegebenen Fluss $f$ in $N(p)$ entspricht dann das Verhältnis $\frac{f(a_j,b_i)}{p_j}$ dem Anteil der Menge von Produkt $j$, den Käufer $b_i$ gekauft hat.

**Lemma 3.** *Ein Preisvektor p ist ein Marktgleichgewicht, gdw. im Flussnetzwerk $N(p)$ die beiden Schnitte $(\{s\}, A \cup B \cup \{t\})$ und $(\{s\} \cup A \cup B, \{t\})$ minimal sind. Jede zu einem maximalen Fluss korrespondierende Produktverteilung ist dann eine Gleichgewichtsverteilung.*

Zur effizienten Bestimmung des Marktgleichgewichts werden die Optimalitätsbedingungen (Referenz) zunächst abgeschwächt. Im Laufe des Algorithmus wird die Verletzung der zu jedem Käufer korrespondierenden Bedingungen immer weiter reduziert bis die optimale Lösung erreicht ist.

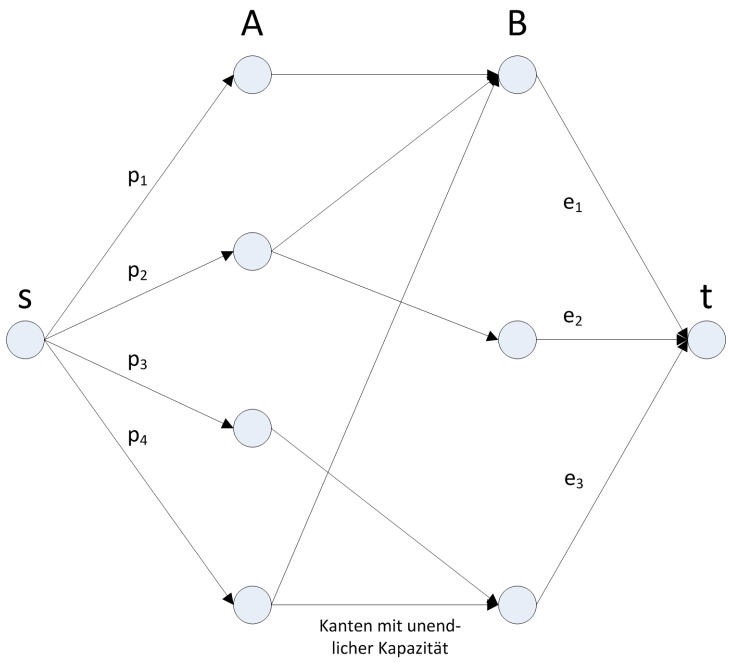


Abbildung 1: Flussnetzwerk $N(p)$

### 3.2 Invariante

Um sicherzustellen, dass mit im Verlauf des Algorithmus zu jedem Zeitpunkt alle Waren verkauft werden können, stellt der Lösungsalgorithmus die Erfüllung der folgenden Invariante sicher:

*Jeder Preisvektor $p$ gewährleistet, dass der Schnitt $(\{s\}, A \cup B \cup \{t\})$ minimal in $N(p)$ ist.*

Um diese Invariante zu warten greift der Algorithmus auf das Konzept der Nachbarschaft zurück, dass für eine Menge von Knoten alle direkt über gerichtete Kanten erreichbaren Nachbarn angibt.

**Definition 4.** *Sei der Preisvektor $p$ fest gegeben. Wir definieren die **Nachbarschaft von** $S$ **in** $N(p)$ für $S \subseteq A$ als*

$$\Gamma(S) = \{j \in B \mid \exists i \in S \text{ mit } (i,j) \in N(p)\}$$

*Bezeichne außerdem $p(S)$ den Gesamtwert der Produkte in $S$, d.h. die Summe ihrer Preise und bezeichne analog $m(T)$ das Gesamtvermögen einer Teilmenge $T \subseteq B$ von Käufern. $S$ wird als **feste Menge** bezeichnet, gdw. $p(S) = m(\Gamma(S))$.*

$\Gamma(S)$ ist also die Menge von Käufern die an Produkten in $S$ interessiert sind. Basierend auf dieser Definition wird uns das folgende Lemma als Grundlage zur Beschreibung des Algorithmus dienen.

**Lemma 5.** *Das Netzwerk $N(p)$ erfüllt die Invariante für geg. $p \Leftrightarrow \forall S \subseteq A : p(S) \leq m(\Gamma(S))$*

### 3.3 Balancierte Flüsse

Die im vorigen Abschnitt vorgestellte Invariante gewährleistet, dass die Gleichgewichtspreise zu keinem Zeitpunkt überschritten werden. Käufer können jedoch einen Überschuss an Geld haben. Der Algorithmus startet mit niedrigen Preisen (sicher kleiner als Gleichgewichtspreise) und erhöht diese schrittweise und verringert dabei den Überschuss der Käufer. Das Konzept der *balancierten Flüsse* wird sicherstellen, dass der Überschuss der Käufer sich schnell genug reduziert um eine polynomielle Laufzeit zu erreichen.

**Definition 6.** *Sei das Flussnetzwerk $N(p)$ (im Folgenden nur als $N$ bezeichnet) gegeben, sodass $(s, A \cup B \cup \{t\})$ ein min-cut in $N(p)$ ist. Sei weiterhin $f$ ein Fluss in diesem Netzwerk und $R(f)$ der Restgraph bezüglich dieses Flusses. Der **Überschuss** von Käufer $i$, bezeichnet als $\gamma_i(N, f)$, ist die Restkapazität der Kante $(i, t)$ mit Rücksicht auf $f$. Wir definieren den **Überschussvektor** ist definiert als $\gamma(N, f) := (\gamma_1(N, f), \ldots, \gamma_n(N, f))$. Ein **balancierter Fluss** in $N$ ist ein Fluss, der die $L_2$-Norm $\|\gamma(N, f)\|$ des Überschussvektors minimiert.*

Wie der Beweis von Satz 7 unten zeigt, berechnet der folgende Algorithmus einen balancierten Fluss für ein gegebenes Flussnetzwerk, dass die Invariante erfüllt:

**Algorithmus zur Berechnung eines balancierten Flusses:**

Gegeben: Flussnetzwerk $N$, sodass Invariante erfüllt ist

1. Reduziere Kapazitäten aller Kanten von Menge $B$ nach $t$ (bis zur Untergrenze von 0) bis die Kapazität des Schnittes $(\{s\} \cup A \cup B, \{t\})$ gleich der Kapazität des Schnittes $(\{s\}, \cup A \cup B \cup \{t\})$. Sei $N'$ das resultierende Netzwerk und $f'$ ein maximaler Fluss in $N'$.
2. Finde einen maximalen s-t-min-cut $(S, T)$ mit $s \in S$ und $t \in T$ in $N'$.

Fall 1: $T = \{t\}$
Finde maximalen Fluss in $N'$ und gib ihn aus.

Fall 2: Sonst.
Seien $N_1$ und $N_2$ die Teilnetzwerke von $N$ mit Knoten nur aus $S \cup \{t\}$ bzw. $T \cup \{s\}$. Seien weiterhin $A_1$ und $B_1$ die Teilmengen von $A$ bzw. $B$, die aus $N_1$ resultieren, sowie $A_2$ und $B_2$ die Teilmengen von $A$ und $B$ die aus $N_2$ resultieren. Finde rekursiv balancierte Flüsse $f_1$ und $f_2$ in $N_1$ und $N_2$. Gib den balancierten Fluss $f = f_1 \cup f_2$ von $N$ zurück.

**Satz 7.** *Der vorgestellte Algorithmus berechnet einen balancierten Fluss in $N$.*

Der Beweis von 7 ist im folgenden nur skizziert und im Original in [1] (S. 114) zu finden.

*Beweisskizze.* Sei der berechnete Fluss $f$ gegeben. Es ist zunächst zu zeigen, dass $f$ ein maximaler Fluss in $N$ ist.
Falls der Algorithmus in Fall 1 terminiert, ist $(\{s\} \cup A \cup B, \{t\})$ ein minimaler Schnitt in $N'$ und hat die gleiche Kapazität wie $(\{s\} \cup A \cup B, \{t\})$. Also ist ein maximaler Fluss $f$ von $N'$ auch ein maximaler Fluss von $N$.
Für Fall 2 des Algorithmus lässt sich zeigen, dass $f = f_1 \cup f_2$ alle Kanten von von $s$ nach $A$ in $N$ sättigt und demnach ein maximaler Fluss in $N$ ist.
Außerdem ist zu zeigen, dass $f$ balanciert ist. Dazu kann durch Induktion über die Tiefe der Rekursion gezeigt werden, dass der Algorithmus bei Terminierung in Fall 1 die $L_2$-Norm minimiert und ein bei Terminierung in Fall 2 berechneter Fluss $f$ Eigenschaft XY erfüllt. □

### 3.4 Hauptalgorithmus

Zunächst wird der Preisvektor $p$ wie folgt initialisiert um die Invariante zu gewährleisten:

- Setzte $p_i = \frac{1}{n}$ für $1 \leq i \leq m$.
- Berechne $\alpha_i$ für $1 \leq i \leq n$ und das daraus resultierende Netzwerk $N(p)$.
- Reduziere den Preis von Produkt $j$ auf $p_j = \max_i\{\frac{u_{ij}}{\alpha_i}\}$, falls es keine inzidente Kante im Graph besitzt.

Der verbleibende Teil des Algorithmus ist in *Phasen* unterteilt, an deren Ende jeweils eine neue Teilmenge von $B$ fest wird. Jede dieser Phasen ist in Iterationen unterteilt. Der Ablauf einer Phase ist im Folgenden zusammengefasst:

1. Berechnung eines balancierten Flusses $f$ im aktuellen Netzwerk $N(p)$.
2. Falls der Flussalgorithmus in Fall 1 endet, gib den Preisvektor $p$ und Produktverteilung $x$ zurück. Andernfalls fahre mit Schritt 3 fort.
3. Setze:
   $\delta$ = maximaler Überschuss der Käufer mit Rücksicht auf $f$
   $I$ = Teilmenge der Käufer $B$ mit Überschuss $\delta$
   $J \subseteq A$ Produkte mit Kanten zu $I$ in $N(p)$

   Beginne mit $\Delta_0 = 1$ und erhöhe in jeder Iteration $\Delta_i$ um einen kleinen Wert bis für $p_i = \Delta_i \cdot p$ und $N(p_i)$ eines der beiden folgenden Ereignisse eintritt:

   (a) Eine Teilmenge $S \subseteq J$ wird fest:
   In diesem Fall fahren wir mit der nächsten Phase fort.

   (b) Eine neue Kante $(i, j)$ mit $i \in I$ und $j \in A \setminus J$ tritt in den *Equality subgraph* ein:
   In diesem Fall fügen wir eine neue Kante $(i, j)$ zu $N(p)$ und berechne einen balancierten Fluss $f$.
   Falls Algorithmus XY in Fall 1 endet, geben wir Preisvektor $p$ und Produktverteilung $x$ zurück. Anderfalls bestimmen wir die Menge $B'$ aller Käufer, von denen im zu $f$ korrespondierenden Restgraphen $R$ ein gerichteter Pfad zu Käufern in $I$ existiert. Wir setzen $I := B'$ und $J = \{j \mid j \in A;\ j \text{ hat Kante zu } i \in I \text{ in } N(p)\}$ und fahren mit Iteration $i + 1$ fort.

# 4 Schlussbetrachtung

Der vorgestellte Algorithmus berechnet Marktgleichgewichte in Fisher-Märkten

$$\mathcal{O}(n^4(\log n + n \log U + \log M))$$

Berechnungen von maximalen Flüssen in $N(p)$. Dabei ist $U = \max_{i \in B, j \in A}\{u_{ij}\}$ und $M$ die Summe der Gelder aller Käufer.

Die Einschränkungen auf lineare Nutzenfunktion beschränken den praktischen Nutzen jedoch. Über das Fisher-Modell verallgemeinert das Modell der *Arrow-Debreu-Tauschökonomien* die Käufermenge zu allgemeinen Marktakteuren. Das Ziel bei der Gleichgewichtsbestimmung ist es dann Preise zu finden, sodass jeder Händler seinen Warenbestand vollständing verkauft und eine bezüglich seiner Nutzenfunktion optimale Warenmenge erhält.

Die spezielleren *Ressourcenverteilungsmärkte* betrachten eine Menge von Ressourcen $R$ mit verfügbaren Kapazitäten $c : R \to \mathbb{R}^+$ und eine Menge von Marktteilnehmern $A$, von denen jeder über eine gewisse Geldmenge $m_i$ verfügt. Jeder der Marktteilnehmer kann bestimmte Produkte $V \subseteq \mathcal{P}(R)$ produzieren und versucht möglichst viele dieser Produkte herzustellen.

# Literatur


[1] Noam Nisan, Tim Roughgarden, Èva Tardos, Vijay V. Vazirani: *Algorithmic Game Theory.* Chapter 5. Cambridge University Press, New York, 2007.