数理経済学特講

複数財オークションのアルゴリズムと離散最適化

第13回　均衡を求めるアルゴリズム

塩浦昭義
東京工業大学 経営工学系 准教授
shioura.a.aa@m.titech.ac.jp

均衡配分の計算：評価値が既知の場合

- 前回の定理より，総評価値最大の財の配分は均衡配分
- 総評価値最大配分の計算： 増加路を使ったアルゴリズムを（大幅に）一般化した方法により可能

均衡配分の計算：評価値が既知の場合

ステップ0：$X_1 = X_2 = \cdots = X_m = \emptyset$ とおく．

ステップ1：$N \setminus \cup_{i=1}^{m} X_i = \emptyset$ ならば終了．

　　現在の配分$(X_1, X_2, \ldots, X_m)$は総評価値最大の均衡配分．

ステップ2：配分に対する以下の更新方法で，総評価値が最大になるものを求める．ただし，$i_1, i_2, \ldots, i_k$ はすべて異なる．

- X_{i_1} に，ある $u_1 \in N \setminus \cup_{i=1}^{m} X_i$ を追加，ある $u_2 \in X_{i_1}$ を削除．
- X_{i_2} に u_2 を追加，ある $u_3 \in X_{i_2}$ を削除．
- ．．．
- $X_{i_{k-1}}$ に u_{k-1} を追加，ある $u_k \in X_{i_{k-1}}$ を削除．
- X_{i_k} に u_k を追加．

更新方法には，厳密には細かい条件が必要

ステップ3：上記の更新方法で配分を更新．ステップ1へ．

問題例：均衡配分の計算

- 入札者a: 重み和（①：2, ②：4, ③：3, ④：1, ⑤：4）
- 入札者b: 財集合（①：2, ②：4, ③：1, ④：4, ⑤：4）の中の

  上位2つの財に依存

  {①, ②, ③}➔評価値2+5, {③, ④, ⑤}➔評価値4+3
- 入札者c: 財の数に依存（1つ：5, 2つ：9, 3つ：11, 4つ：13, 5つ：14）

アルゴリズムによる配分の変化
(∅, ∅, {2}) ➔ (∅, ∅, {2,4}) ➔ (∅, {2}, {1,4})
➔ (∅, {2,5}, {1,4}) ➔ ({5}, {2,4}, {1,3})（総評価値最大の配分）

※ 極小均衡価格は (2,4,3,3,3)

均衡価格の計算：評価値が既知の場合

均衡配分 $X_0, X_1, \ldots, X_m$ が得られた → 均衡価格の計算へ

- $X_i \in D_i(p)\ (i = 1, \ldots, m), p(j) = 0\ (j \in X_0)$ を満たす p を求める

定理：　評価関数 v_i が粗代替性を満たすとき，
任意の価格ベクトル p に対して以下が成立：

$$v_i(X) - \sum_{j \in X} p(j) = \max\{v_i(Y) - \sum_{j \in Y} p(j) \mid Y \subseteq N\}$$

←→

$$v_i(X) - \sum_{j \in X} p(j) \geq v_i(X \setminus \{u\}) - \sum_{j \in X \setminus \{u\}} p(j)\ \ (\forall u \in X),$$
$$v_i(X) - \sum_{j \in X} p(j) \geq v_i(X \cup \{v\}) - \sum_{j \in X \cup \{v\}} p(j)\ \ (\forall v \in N \setminus X),$$
$$v_i(X) - \sum_{j \in X} p(j) \geq v_i(X \setminus \{u\} \cup \{v\}) - \sum_{j \in X \setminus \{u\} \cup \{v\}} p(j)$$
$$(\forall u \in X, \forall v \in N \setminus X)$$

←→ $p(u) \leq v_i(X) - v_i(X \setminus \{u\})\ (\forall u \in X),$
$-p(v) \leq v_i(X) - v_i(X \cup \{v\})\ (\forall v \in N \setminus X),$
$p(u) - p(v) \leq v_i(X) - v_i(X \setminus \{u\} \cup \{v\})\ (\forall u \in X, \forall v \in N \setminus X)$

差分不等式系

最短路問題を解くことにより，均衡価格が計算可能

問題例の続き：均衡価格の計算

- 入札者a: 重み和（①:2, ②:4, ③:3, ④:1, ⑤:4）
- 入札者b: 財集合（①:2, ②:4, ③:1, ④:4, ⑤:4）の中の上位2つの財に依存
- 入札者c: 財の数に依存（1つ:5, 2つ:9, 3つ:11, 4つ:13, 5つ:14）

均衡配分は ({5}, {2,4}, {1,3})（総評価値最大の配分）

均衡価格の条件

極小均衡価格は (2,4,3,3,3)

入札者a: p(5)≦4-0,
-p(1)≦4-6, -p(2)≦4-8, -p(3)≦4-7, -p(4)≦4-5,
p(5)-p(1)≦4-2, p(5)-p(2)≦4-4, p(5)-p(3)≦4-3, p(5)-p(4)≦4-1

入札者b: p(2)≦8-4, p(4)≦8-4,
-p(1)≦8-8, -p(3)≦8-8, -p(5)≦8-8,
p(2)-p(1)≦8-6, p(2)-p(3)≦8-5, p(2)-p(5)≦8-8,
p(4)-p(1)≦8-6, p(4)-p(3)≦8-5, p(4)-p(5)≦8-8

入札者c: p(1), p(3) ≦9-5, -p(2), -p(4), -p(5) ≦9-11,
p(1)-p(2), p(1)-p(4), p(1)-p(5), p(3)-p(2), p(3)-p(4), p(3)-p(5)≦9-9

反復オークション

反復オークションのアルゴリズム

以下の2つを紹介

- その1：均衡を近似的に計算 [Kelso-Crawford 1982]
  - 単調に価格を増加，均衡配分（および均衡価格の近似値）を求める
  - 各反復で，入札者の利得最大の財集合ひとつの情報が必要
  - 価格増加のルールは簡単：　希望が重複→価格を増やす

- その2：均衡を厳密に計算 [Gul-Stacchetti 2000]
  - 単調に価格を増加，均衡価格（および均衡配分）を求める
  - 各反復で，入札者の利得最大の財集合すべての情報が必要
  - 価格増加のルールは複雑：
    - 得た情報を使い，価格を増やす財をうまく選ぶ

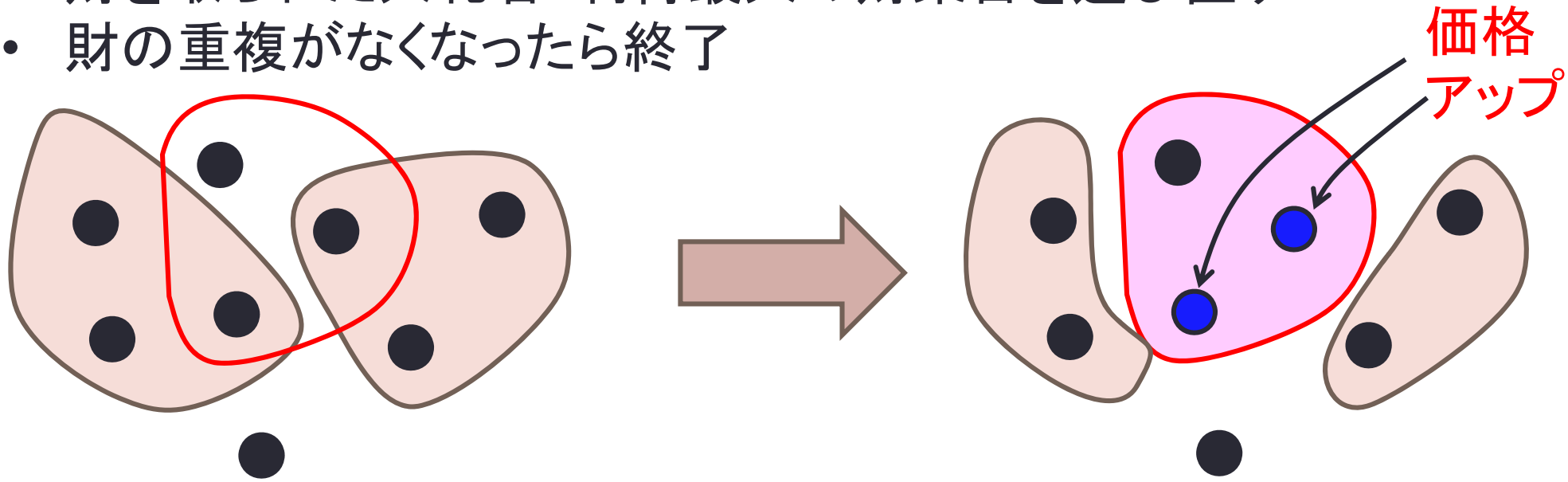
均衡を近似的に計算するアルゴリズム

アルゴリズムの概要

Kelso-Crawford (1982) が提案（Blumrosen-Nisan(2007)も参照）

アルゴリズムの流れ

- 各入札者は順番に，
  現在の価格の下で（ほぼ）利得最大の財集合を一つ選ぶ
- 選んだ財集合に，他の入札者が既に選んだ財が含まれる
  →重複した財を奪い，価格を上げる
- 財を取られた入札者：利得最大の財集合を選び直す
- 財の重複がなくなったら終了



均衡を近似的に計算：変数と出力

δ: アルゴリズムのパラメータ，> 0

変数：p(j) --- 財 j の暫定価格，δの整数倍

S_i --- 入札者 i に割り当ての財集合，常に重複無し

出力：δ均衡 --- 以下の条件を満たす

財の配分$S_1, \ldots, S_m$ と価格$p(1), \ldots, p(n)$

(条件1) 各入札者 i に対し，S_i は価格 p’ において利得最大

$$p'(j) = \begin{cases} p(j) & (j \in S_i) \\ p(j) + \delta & (j \in N \setminus S_i) \end{cases}$$

（価格 p において，ほぼ利得最大の財集合が割り当て）

(条件2) $p(j) = 0 \ (\forall j \in N \setminus \bigcup_{i \in B} S_i)$

（未割り当ての財の価格＝0）

均衡を近似的に計算：アルゴリズム

ステップ0：各財 j に対し p(j) = 0.　　各入札者 i に対し $S_i=\emptyset$.

ステップ1：各入札者 i に対し，価格 p’ での利得最大の
財集合で，S_i を含むものを D_i とおく.

$$p'(j)=\begin{cases} p(j) & (j \in S_i) \\ p(j)+\delta & (j \in N \setminus S_i) \end{cases}$$

ステップ2：各入札者 i に対し $S_i = D_i$ ➔ δ均衡（終了）

ステップ3：$S_i \neq D_i$ なる入札者 i を選ぶ.

ステップ4：各 $j \in D_i \setminus S_i$ に対し，　p(j):=p(j) + δ.
S_i を D_i に置き換える.
他の入札者 h に対し，S_h と D_i に重複があれば，
それを削除.

ステップ1へ戻る.

均衡を近似的に計算：詳しいアルゴリズム

ステップ0：各財 j に対し p(j) = 0，各入札者 i に対し S_i=∅，
B’=B とおく．

ステップ1：B’= ∅ ならば終了．

B’=不満をもっている可能性のある入札者の集合

ステップ2：B’ から 入札者 i を選ぶ．

ステップ3：入札者 i に対し，価格 q での利得最大の
財集合で，S_i を含むものを D とおく（←必ず存在）

$$q(j) = \begin{cases} p(j) & (j \in S_i) \\ p(j) + \delta & (j \in N \setminus S_i) \end{cases}$$

ステップ4：各 $j \in D \setminus S_i$ に対し，p(j):=p(j) + δ．
S_i を D に置き換える．B’ から i を削除．
他の入札者 h に対し，S_h と D に重複があれば，
それをS_h から削除し，h を B’ に追加．
ステップ1へ戻る．

反復オークションのための問題例

- 入札者a: 重み和(①:2, ②:4, ③:3, ④:1, ⑤:5)
- 入札者b: 財集合(①:2, ②:5, ③:1, ④:4, ⑤:3) の中の

  上位2つの財に依存

  {①, ②, ③}➔評価値2+5, {③, ④, ⑤}➔評価値4+3
- 入札者c: 財の数に依存(1つ:4, 2つ:7, 3つ:10, 4つ:12, 5つ:13)

均衡価格=(2,4,3,3,3)　　←極小均衡価格
均衡配分: a {5}, b {2,4}, c {1,3}
　　または a {3,5}, b {2,4}, c {1}

アルゴリズムの実行例(1)

- 入札者a: 重み和(①:2, ②:4, ③:3, ④:1, ⑤:5)
- 入札者b: 財集合(①:2, ②:5, ③:1, ④:4, ⑤:3) の中の上位2つの財に依存
- 入札者c: 財の数に依存(1つ:4, 2つ:7, 3つ:10, 4つ:12, 5つ:13)

- ステップ0: 価格 p=(0,0,0,0,0), 配分 ∅, ∅, ∅, B'={a,b,c}
- ステップ2: i=aを選択
- ステップ3: 入札者aの価格 (1,1,1,1,1)での利得最大の集合D=N
- ステップ4:
  - D - S_a = Nに含まれる財の価格を上げる→p=(1,1,1,1,1)
  - S_a = D = N, S_b = ∅, S_c = ∅,　　B' = {b,c}

アルゴリズムの実行例(2)

- 入札者a: 重み和(①:2, ②:4, ③:3, ④:1, ⑤:5)
- 入札者b: 財集合(①:2, ②:5, ③:1, ④:4, ⑤:3) の中の上位2つの財に依存
- 入札者c: 財の数に依存(1つ:4, 2つ:7, 3つ:10, 4つ:12, 5つ:13)

- p=(1,1,1,1,1)
- S_a = N, S_b = ∅, S_c = ∅,
- B' = {b,c}
- ステップ2: i=bを選択
- ステップ3: 入札者bの価格 (2,2,2,2,2)での利得最大の集合
  D={2,4}
- ステップ4:
  - D - S_b = {2,4} に含まれる財の価格を上げる→p=(1,2,1,2,1)
  - S_b = D = {2,4}, S_a = N-D={1,3,5}, S_c = ∅,
  - B' = {a,c}

アルゴリズムの実行例(3)

- 入札者a: 重み和(①:2, ②:4, ③:3, ④:1, ⑤:5)
- 入札者b: 財集合(①:2, ②:5, ③:1, ④:4, ⑤:3) の中の上位2つの財に依存
- 入札者c: 財の数に依存(1つ:4, 2つ:7, 3つ:10, 4つ:12, 5つ:13)

- p=(1,2,1,2,1)
- S_b = {2,4}, S_a = {1,3,5}, S_c = ∅
- B' = {a,c}

- ステップ2: i=cを選択
- ステップ3: 入札者cの価格 (2,3,2,3,2)での利得最大の集合
  D={1,3,5}
- ステップ4:
  - D - S_c ={1,3,5}に含まれる財の価格を上げる→(2,2,2,2,2)
  - S_c = D = {1,3,5}, S_a = {1,3,5}-D= ∅, S_b = {2,4}-D={2,4},
  - B' = {a}

アルゴリズムの実行例(4)

- 入札者a: 重み和(①:2, ②:4, ③:3, ④:1, ⑤:5)
- 入札者b: 財集合(①:2, ②:5, ③:1, ④:4, ⑤:3) の中の上位2つの財に依存
- 入札者c: 財の数に依存(1つ:4, 2つ:7, 3つ:10, 4つ:12, 5つ:13)

- p=(2,2,2,2,2)
- S_c = {1,3,5}, S_a = ∅, S_b = {2,4},　B' = {a}

- ステップ2: i=aを選択
- ステップ3: 入札者cの価格 (3,3,3,3,3)での利得最大の集合
  D={2,5}
- ステップ4:
  - D - S_a ={2,5} に含まれる財の価格を上げる→(2,3,2,2,3)
  - S_a = D = {2,5}, S_b = {2,4}-D={4}, S_c = {1,3,5}-D={1,3},
  - B' = {b,c}

アルゴリズムの実行例(5)

- 入札者a: 重み和(①:2, ②:4, ③:3, ④:1, ⑤:5)
- 入札者b: 財集合(①:2, ②:5, ③:1, ④:4, ⑤:3) の中の上位2つの財に依存
- 入札者c: 財の数に依存(1つ:4, 2つ:7, 3つ:10, 4つ:12, 5つ:13)

- p=(2,3,2,2,3)
- S_a = {2,5}, S_b = {4}, S_c = {1,3}, B' = {b,c}

- ステップ2: i=bを選択
- ステップ3: 入札者cの価格 (3,4,3,2,4)での利得最大の集合 D={2,4}
- ステップ4:
  - D - S_b ={2} に含まれる財の価格を上げる→(2,4,2,2,3)
  - S_b = D = {2,4}, S_c = {1,3}-D={1,3}, S_a = {2,5}-D={5},
  - B' = {a,c}

アルゴリズムの実行例(6)

- 入札者a: 重み和(①:2, ②:4, ③:3, ④:1, ⑤:5)
- 入札者b: 財集合(①:2, ②:5, ③:1, ④:4, ⑤:3) の中の上位2つの財に依存
- 入札者c: 財の数に依存(1つ:4, 2つ:7, 3つ:10, 4つ:12, 5つ:13)

- p=(2,4,2,2,3)
- S_a = {5}, S_b = {2,4}, S_c = {1,3}, B' = {a,c}

- ステップ2: i=cを選択
- ステップ3: 入札者cの価格 (2,5,2,3,4)での利得最大の集合 D={1,3}
- ステップ4:
  - D - S_c = ∅ に含まれる財の価格を上げる→ p 変更せず
  - S_a = {5}, S_b = {2,4}, S_c = {1,3}, 変更せず
  - B' = {a}

アルゴリズムの実行例(7)

- 入札者a: 重み和(①:2, ②:4, ③:3, ④:1, ⑤:5)
- 入札者b: 財集合(①:2, ②:5, ③:1, ④:4, ⑤:3) の中の上位2つの財に依存
- 入札者c: 財の数に依存(1つ:4, 2つ:7, 3つ:10, 4つ:12, 5つ:13)

- p=(2,4,2,2,3)
- S_a = {5}, S_b = {2,4}, S_c = {1,3}, B' = {a}

- ステップ2: i=aを選択
- ステップ3: 入札者cの価格 (3,5,3,3,3)での利得最大の集合
  D={5}
- ステップ4:
  - D - S_a = ∅ に含まれる財の価格を上げる→ p 変更せず
  - S_a = {5}, S_b = {2,4}, S_c = {1,3}, 変更せず
  - B' = ∅ ➔ 次の反復でアルゴリズム終了

得られた財配分の近似最適性

定理 アルゴリズムで得られた財配分の総評価値
≧ 財配分の総評価値の最大値 – nδ

系 評価値が整数値のとき，δ=1/(n+1) とおくと，
アルゴリズムで得られた財配分 ＝ 総評価値最大の財配分

[証明] $(S_1, \ldots, S_m)$: アルゴリズムの財配分，
$(X_1, \ldots, X_m)$: 総評価値最大の財配分
(条件1) 各入札者 i に対し，S_i は価格 p’ において利得最大

$$p'(j) = \begin{cases} p(j) & (j \in S_i) \\ p(j) + \delta & (j \in N \setminus S_i) \end{cases}$$

(条件2) $p(j) = 0 \ \ (\forall j \in N \setminus \cup_{i \in B} S_i)$

得られた財配分の近似最適性

$p(S_i) \equiv \sum_{j \in S_i} p(j)$

[証明のつづき]

条件1より，$v_i(S_i) - p(S_i) = v_i(S_i) - p'(S_i) \geq v_i(X_i) - p'(X_i)$

$$= v_i(X_i) - p(X_i) - |X_i \cap S_i|\delta$$

$\therefore \sum_{i \in B} v_i(S_i) - \sum_{i \in B} p(S_i)$

$$\geq \sum_{i \in B} v_i(X_i) - \sum_{i \in B} p(X_i) - \sum_{i \in B} |X_i \cap S_i|\delta$$

条件2より，$\sum_{i \in B} p(S_i) = p(N)$

価格の非負性より，$\sum_{i \in B} p(X_i) \leq p(N)$

$X_i \cap S_i$ は互いに素なので，$\sum_{i \in B} |X_i \cap S_i|\delta \leq n\delta$

$\therefore \sum_{i \in B} v_i(S_i) - p(N) \geq \sum_{i \in B} v_i(X_i) - p(N) - n\delta$

$\therefore \sum_{i \in B} v_i(S_i) \geq \sum_{i \in B} v_i(X_i) - n\delta$ ■

アルゴリズムの正当性

命題 各反復のステップ1において，各入札者 h に対し，

$$\exists X \in D_h(p') \text{ s.t. } X \supseteq S_h$$

[証明] 命題が k-1 回目のステップ1で成立と仮定，
k 回目の反復でも成り立つことを証明.

仮定：第 k-1 回目のステップ1：$\exists X \in D_h(p')$ s.t. $X \supseteq S_h^{(k-1)}$

$S_h^{(k-1)}$：第 k-1 回目の反復におけるS_h

k 回目のステップ1の価格 p’ を q’ とおく

→$\exists Y \in D_h(q')$ s.t. $Y \supseteq S_h^{(k)}$を示せば良い

2つのケース

① h=i（入札者 h=i が直前のステップ4 で他人の財を奪っている場合）

$S_h^{(k)} = X \in D_h(p')$ なので，q’ = p’ を示せばよい (Y=$S_h^{(k)}$ とすればよい)

p’ = p において，$S_h^{(k-1)}$ 以外の財のみ価格をδだけアップ

q = p において，$S_h^{(k)}$ - $S_h^{(k-1)}$ の財のみ価格をδだけアップ

q’ = q において，$S_h^{(k)}$ 以外の財のみ価格をδだけアップ

= p において，$S_h^{(k-1)}$ 以外の財のみ価格をδだけアップ = p’

アルゴリズムの正当性

② h≠i（h は財を奪われる側の入札者の場合）

第 k-1 回目のステップ4で $S_h^{(k-1)}$ から財が奪われる（かもしれない）

→その残りの財集合が $S_h^{(k)}$

入札者 i が他者から奪った財の価格のみアップ

つまり，$q(j) = p(j) + \delta \left(j \in D_i \setminus S_i^{(k-1)} \right)$,

$q(j) = p(j)$ (それ以外の財 j)　← $S_h^{(k)}$ の財はこちら

p’ = p において，$S_h^{(k-1)}$ 以外の財のみ価格をδだけアップ

q’ = q において，$S_h^{(k)}$ ($\subseteq S_h^{(k-1)}$) 以外の財のみ価格をδだけアップ

よって，p’ ≦ q’ かつ p’(j) = q’(j) ($j \in S_h^{(k)}$) 成立

粗代替性より，$\exists Y \in D_i(q')$

s.t. $Y \supseteq D_h$ の中で価格不変の財すべて $\supseteq S_h^{(k)}$

均衡を厳密に計算するアルゴリズム

均衡価格

- 定義：価格ベクトルp^*と財の配分$(X_0, X_1, \dots, X_m)$の組はワルラス均衡

  ←→ $X_i \in D_i(p^*)\ (i = 1, \dots, m), \quad p(j) = 0\ (\forall j \in X_0)$

- 定義：価格ベクトルp^*は均衡価格

  ←→ ある財の配分$(X_0, X_1, \dots, X_m)$ が存在して

  $X_i \in D_i(p^*)\ (i = 1, \dots, m), \qquad p(j) = 0\ (\forall j \in X_0)$

均衡価格の必要十分条件

- 任意の財集合 X⊆N に対し,
  η(i, X) ≡ 入札者 i が利得最大の財集合を得るとき,
  X から選ぶ必要のある財の最小数
  $= \min\{|Y \cap X| \mid Y \in D_i(p)\}$

$\sum_{i \in B} \eta(i, X) > |X|$ ➔ X の財は，配分するには不足

- 任意の価格>0の財集合 X に対し,
  μ(i, X) ≡ 入札者 i が利得最大の財集合を得るとき,
  X から選ぶことが可能な財の最大数
  $= \max\{|Y \cap X| \mid Y \in D_i(p)\}$

$\sum_{i \in B} \mu(i, X) < |X|$ ➔ X の財は，配分するには多すぎ

定理 価格ベクトル p に対し,
$X_i \in D_i(p)\ (i = 1, \ldots, m)$ を満たす財配分$(X_0, X_1, \ldots, X_m)$が存在
➔➔ ∀X⊆N: $\sum_{i \in B} \eta(i, X) \leq |X|$
∀X⊆N: $\sum_{i \in B} \mu(i, X) \geq |X|$

具体例

- 入札者a: 重み和(①:2, ②:4, ③:3, ④:1, ⑤:5)
- 入札者b: 財集合(①:2, ②:5, ③:1, ④:4, ⑤:3) の中の
  上位2つの財に依存
- 入札者c: 財の数に依存(1つ:4, 2つ:7, 3つ:10, 4つ:12, 5つ:13)

価格=(0,0,0,0,0), X={1,2,3} のとき
aの需要集合 = {1,2,3,4,5} のみ → X の財を3つ欲しい
b の需要集合 = {2,4}を含む全ての集合 → X の財を1つ欲しい
c の需要集合 = {1,2,3,4,5} のみ → X の財を3つ欲しい
よって, 不足分は 3+1+3 – 3 = 4

アルゴリズムの概要

Gul-Stacchetti (2000) が提案

アルゴリズムの考え方

価格ベクトルp^*は均衡価格

　⇔ ある財の配分$(X_0, X_1, \ldots, X_m)$ が存在して

①$\forall X \subseteq N$:　$\sum_{i \in B} \eta(i, X) \leq |X|$,

②$\forall X \subseteq N$:　$\sum_{i \in B} \mu(i, X) \geq |X|$,

③ $p(j) = 0$ $(\forall j \in X_0)$

②, ③の条件を満たしつつ, ①を満たすように価格を増加する

アルゴリズムの流れ

初期価格 p=(0,⋯,0)
各反復で以下を実行:

- 全ての X に対し $\sum_{i\in B}\eta(i,X) \le |X|$ （つまり，①成立）
  ➔ 入札者全員が利得最大の財集合を得ることが可能（要証明）
- ある X に対し，$\sum_{i\in B}\eta(i,X) > |X|$
  ➔ X の財は，配分するには不足 ➔ X の財の価格アップ
- X の選び方
  - 不足分 $\sum_{i\in B}\eta(i,X) - |X|$ が最大の X を選ぶ
  - [極小な均衡価格が欲しい場合] 極小な X を選ぶ

アルゴリズムの実行例(1)

- 入札者a: 重み和(①:2, ②:4, ③:3, ④:1, ⑤:5)
- 入札者b: 財集合(①:2, ②:5, ③:1, ④:4, ⑤:3) の中の上位2つの財に依存
- 入札者c: 財の数に依存(1つ:4, 2つ:7, 3つ:10, 4つ:12, 5つ:13)

- 初期価格 (0,0,0,0,0)
  - a の需要集合 = {1,2,3,4,5} のみ
  - b の需要集合 = {2,4} を含む任意の財集合
  - c の需要集合 = {1,2,3,4,5} のみ
- X = {1,2,3,4,5} とすると，不足分が最大 5+2+5-5=7
- 価格 (1,1,1,1,1)
  - a の需要集合 = {1,2,3,5}, {1,2,3,4,5}
  - b の需要集合 = {2,4} のみ
  - c の需要集合 = {1,2,3,4,5} または要素数4の任意の財集合
- X = {1,2,3,4,5} とすると，不足分が最大 4+2+4-5=6

アルゴリズムの実行例(2)

- 入札者a: 重み和(①:2, ②:4, ③:3, ④:1, ⑤:5)
- 入札者b: 財集合(①:2, ②:5, ③:1, ④:4, ⑤:3) の中の上位2つの財に依存
- 入札者c: 財の数に依存(1つ:4, 2つ:7, 3つ:10, 4つ:12, 5つ:13)

- 価格 (2,2,2,2,2)
  - a の需要集合 = {2,3,5}, {1,2,3,5}
  - b の需要集合 = {2,4} のみ
  - c の需要集合 =要素数3または4の任意の財集合
- X = {2,3,4,5} とすると，不足分が最大 3+2+2-4=3
- 価格 (2,3,3,3,3)
  - a の需要集合 = {2,5}, {1,2,5}, {2,3,5}, {1,2,3,5}
  - b の需要集合 = {2,4} のみ
  - c の需要集合 = {1}を含む，要素数1,2または3の任意の財集合
- X = {2} とすると，不足分が最大 1+1+0-1=2

アルゴリズムの実行例(3)

- 入札者a: 重み和(①:2, ②:4, ③:3, ④:1, ⑤:5)
- 入札者b: 財集合(①:2, ②:5, ③:1, ④:4, ⑤:3) の中の上位2つの財に依存
- 入札者c: 財の数に依存(1つ:4, 2つ:7, 3つ:10, 4つ:12, 5つ:13)

- 価格 (2,4,3,3,3)
  - a の需要集合 = {5}を含み{4}を含まない，任意の財集合
  - b の需要集合 = {2,4} のみ
  - c の需要集合 = {1}を含み{2}を含まない，
    要素数1, 2または3の任意の財集合
- 全ての X に対し，不足数≦0
- X_a={5}, X_b={2,4}, X_c={1,3} は各入札者の利得最大の財集合
  → 均衡配分

定理 (p*(1),…,p*(n)): 極小均衡価格 とすると，
アルゴリズムの反復回数＝$\max_j$ p*(j)

関数最小化と均衡の関係

総評価値最大化問題の線形計画緩和の双対問題

最小化　$\sum_{i \in B} q(i) + p(N)$

条件　$q(i) + p(Y) \geq v_i(Y) \quad (\forall i \in B, \forall Y \subseteq N)$

$p(j) \geq 0 \quad (\forall j \in N)$

定理　最適解における p ←→ 均衡価格
さらに，評価値が整数値 →整数最適解が存在

関数最小化問題に書き換えできる

リアプノフ関数

最小化　$\sum_{i \in B} \max\{v_i(Y_i) - p(Y_i) | Y_i \subseteq N\} + p(N)$

条件　$p(j) \geq 0 \quad (\forall j \in N)$

定理[Ausubel 2006]　リアプノフ関数の最小解 ←→ 均衡価格
さらに，評価値が整数値 →整数最小解が存在

均衡計算＝リアプノフ関数最小化

- Gul-Stacchetti (2006)の均衡計算のアルゴリズム
  ＝ リアプノフ関数 $L(p) = \sum_{i \in B} \max\{v_i(Y_i) - p(Y_i) | Y_i \subseteq N\} + p(N)$
  の最小化に対する最急降下アルゴリズム

- 財集合 X の価格 p における不足分 $\sum_{i \in B} \eta(i, X) - |X|$
  ＝財集合 X の価格を1ずつ上げたときの，関数値の減少量

解析において鍵となる性質

定理 リアプノフ関数はL♮凸性（離散中点凸性）をもつ

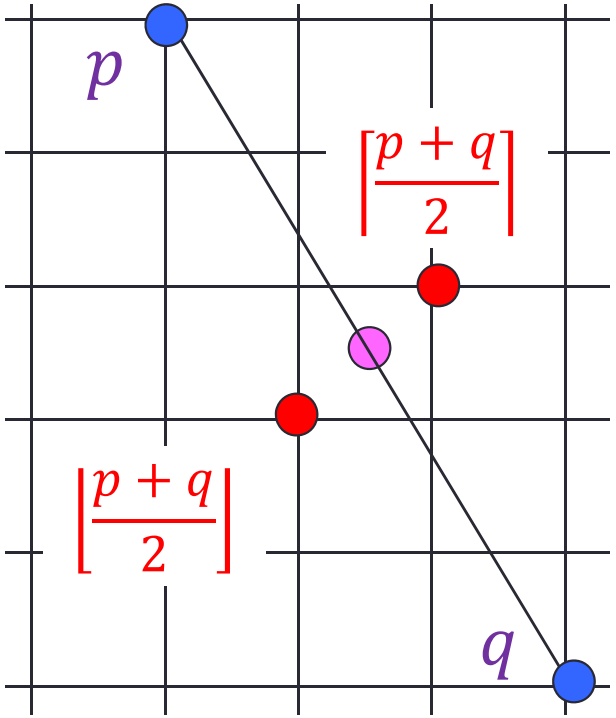
$$\forall p, q \in \mathbb{Z}^n:\ g(p) + g(q) \geq g\left(\left\lceil \frac{p+q}{2} \right\rceil\right) + g\left(\left\lfloor \frac{p+q}{2} \right\rfloor\right)$$

一般のL♮凸関数 $g: \mathbb{Z}_+^n \to \mathbb{R}$ に対し，
最急降下アルゴリズムが適用可能

ステップ0：初期ベクトル p=(0,…,0)

ステップ1：$g(p + e_X) - g(p)$ 最小
の X⊆N を求める.

ステップ2：$g(p + e_X) = g(p)$ ならば終了，
そうでなければ $p := p + e_X$ とおきステップ1へ.



定理 (p*(1),…,p*(n)): 極小最小解 とすると，
アルゴリズムの反復回数＝$\max_j$ p*(j)