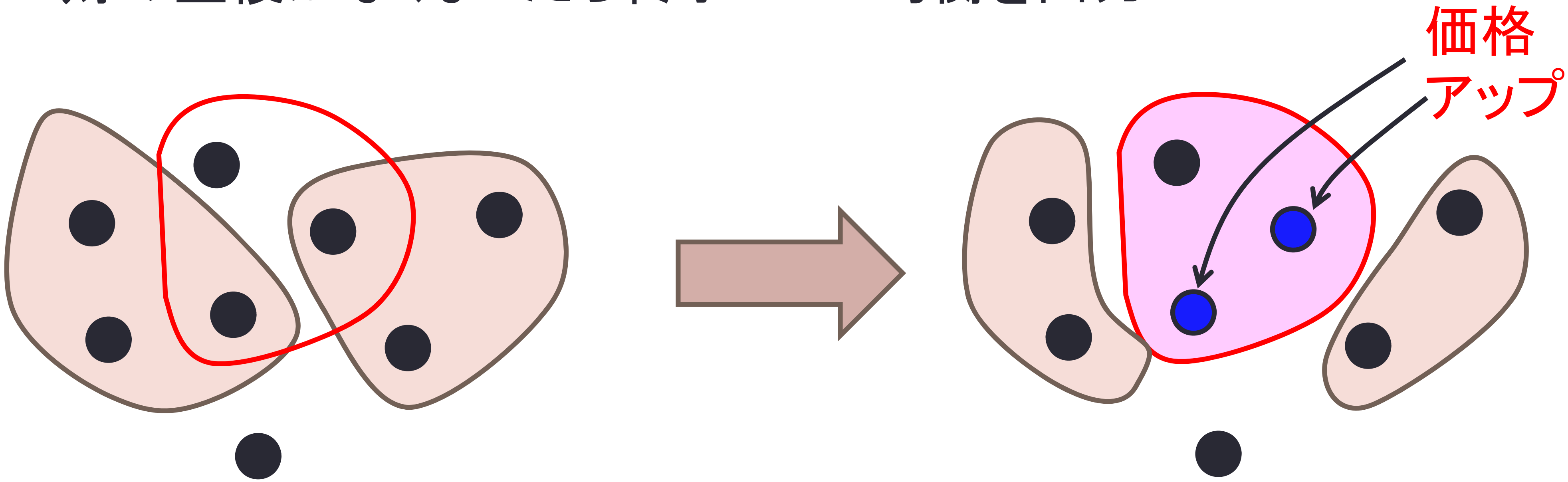
複数需要モデル:
均衡を近似的に求めるアルゴリズム

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

アルゴリズムの概要

Kelso-Crawford (1982) が提案

- 単一需要モデルにおける近似アルゴリズムの拡張
- 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 → δ 均衡を出力



δ均衡の定義

δ均衡 --- 以下の条件を満たす

財配分$S_1, \dots, S_m$と価格$p(1), \dots, p(n)$

(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）

- 普通の均衡の条件と異なるのは条件1のみ
- 0均衡 = 普通の均衡

アルゴリズムの詳細

ステップ0: $p(j) := 0 \ (j \in N)$, $S_i := \emptyset \ (i \in B)$, $B' := B$.

ステップ1: $B' = \emptyset$ ➔ 終了.

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

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

での利得最大の財集合で, S_i を含むものを D とおく

ステップ4: $p(j) := p(j) + \delta \ (j \in D \setminus S_i)$

$S_i := D$, $B' := B' \setminus \{i\}$

他の入札者 h に対し, $S_h \cap D \neq \emptyset$

➔ $S_h := S_h \setminus D$, $B' := B' \cup \{h\}$

ステップ1へ.

B' =不満をもつ(可能性のある)入札者の集合

必ず存在

単一需要モデルに特殊化

単一需要モデルのときのアルゴリズムに一致する

ステップ0: $p(j) := 0, \quad j_i := 0 \ \ (j \in N), \quad B' := B$.

ステップ1: $B' = \emptyset$ ➔ 終了.

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, j_i :=価格 p での利得最大の財

ステップ4: $p(j_i) := p(j_i) + \delta$.

$B' := B' \setminus \{i\}$

j_i を持っていた入札者 h を B' に追加.

ステップ1へ.

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）

極小均衡価格=(2,4,3,3,3)

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)
  - $S_a = D = N$, $S_b = \emptyset$, $S_c = \emptyset$, 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)
- $S_a = N$, $S_b = \emptyset$, $S_c = \emptyset$,
- 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)
  - $S_b = D = \{2,4\}$, $S_a = N\text{-}D = \{1,3,5\}$, $S_c = \emptyset$,
  - 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 = Ø
- 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= Ø, 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

アルゴリズムの詳細（再掲）

ステップ0： $p(j) := 0\ \ (j \in N)$， $S_i := \emptyset\ \ (i \in B)$， $B' := B$.

ステップ1： $B' = \emptyset$ ➔ 終了.

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

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

での利得最大の財集合で，S_i を含むものを D とおく

ステップ4： $p(j) := p(j) + \delta\ (j \in D \setminus S_i)$

$S_i := D$, $B' := B' \setminus \{i\}$

他の入札者 i

ステップ1へ.

B' =不満をもつ（可能性のある）入札者の集合

必ず存在

定義： 評価関数 $v_i: 2^N \to \mathbb{Z}_+$は 粗代替的

⟺ $\forall p \leq \forall q$, $\forall X \in D_i(p)$,

$\exists Y \in D_i(q): \{h \in X \mid q(h) = p(h)\} \subseteq Y$

ステップ3の妥当性

粗代替性により，ステップ3が実行可能

命題　各反復のステップ3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] 入札者 h に対し，現在の反復以前で，

入札者 h が最後に財集合 S=S_h を得た反復に注目．

その反復のステップ4終了後の価格 $\tilde{p}$ に関して，$S \in D_h(\tilde{p})$.

それ以降，S の財は他の入札者に取られた可能性あり．

現在手元に残っている S_h は取られなかった財の集合．

これらについては，価格は不変．

$$\therefore S_h \subseteq \{j \in N \mid p'(j) = \tilde{p}(j)\}$$

よって，評価関数の粗代替性より，S_h を含む $X \in D_h(p')$ が存在． ■

近似精度の解析

アルゴリズムの正当性

定理1 アルゴリズムは有限回の反復後に終了し，δ均衡を求める．

単一需要モデルの証明に類似

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

[有限性の証明]

- 各財の価格：初期値＝0
- アルゴリズムの各反復：
  ある財 j の価格 p(j) が δ 増加
  → ∀入札者 i, ∀財集合S∋j, $v_i(S)-p(S)$ が δ 減少
- ∀入札者 i, ∀財集合S∋j, $v_i(S)-p(S) \leqq 0$
  ➔どの入札者も財 j を奪わなくなる
  ➔ 財 j の価格は今後変化しない

∴財 j の価格 p(j) の増加回数$\leqq \frac{1}{\delta} \max\{v_i(S) \mid S \ni j\}$

アルゴリズムの正当性

定理1 アルゴリズムは有限回の反復後に終了し，
δ均衡を求める.

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

[条件1の証明]
アルゴリズム中で，入札者 i が(最後に)B'から削除された反復に注目.
その時点では，受け取った財集合 S_i について条件１成立(ステップ3より)
他の入札者から財を奪った場合，直後に価格がδ増加
∴アルゴリズム終了時まで，S_i の財は他の入札者に
奪われないので，価格 p(j) 不変 ■

アルゴリズムの正当性

定理1 アルゴリズムは有限回の反復後に終了し，
δ均衡を求める.

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

[条件2の証明]
アルゴリズムの途中で，財の価格が増加
⇔ その財の割当が別の入札者に変更
∴アルゴリズム終了時に割り当てなしの財 j に対し，
価格 p(j) はずっと不変で 0 ■

アルゴリズムの性能評価

得られた財配分の総評価値 ≒ 最大値

定理2 δ均衡における財配分の総評価値
≧ 財配分の総評価値の最大値 ー nδ

証明は「均衡配分➔総評価値最大」と同様

評価値 $v_i(S)$ がすべて整数
→ δを調整して，均衡配分の厳密解を得ることが可能

$\delta < 1/n$ → 誤差 $n\delta < 1$
∴ 総評価値の整数性より
アルゴリズムの財配分の総評価値
＝ 財配分の総評価値の最大値

得られた財配分の近似最適性

定理2　δ 均衡における財配分の総評価値
≧ 財配分の総評価値の最大値　− nδ

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

[証明] (S_1, …, S_m): δ 均衡における財配分

(X_1, …, X_m): 総評価値最大の財配分,

一般性を失うことなく $\bigcup_{i=1}^{m} X_i = N$ を仮定

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

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

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

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

条件2より，$\sum_{i \in B} p(S_i) = p(N) = \sum_{i \in B} p(X_i)$

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

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

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