5U-02

MC-nets における利得分配問題の最小コアを求める複数制約生成法

小浦 隆之 *　　平山 勝敏 *　　沖本 天太 *

1 はじめに

提携形ゲームは，エージェントの集合 $A = \{1, \ldots, n\}$ およびエージェントの任意の部分集合（提携）の利得を計算する特性関数 $v : 2^A \to \Re$ で構成される．提携形ゲームにおける利得分配問題では，所与の総利得 p_{max} を各エージェントにどのように分配するかを示す利得ベクトル $x = (x_1, \ldots, x_n)$ を求める．利得分配問題の代表的な解概念としてコアとシャープレイ値がよく知られており，また，コアの拡張として ε-コア，最小コア，仁などがある．

一方，従来の提携形ゲームでは，特性関数 v は，任意の提携を与えれば値を返すブラックボックス関数とされ，それを具体的にどう表記するか明確ではなかった．近年，特性関数を簡略表記する基本的な表記法の一つとして MC-nets（Marginal Contribution Networks）[5] が提案されている．MC-nets に対しては，シャープレイ値が入力サイズの線形オーダーで計算できるのに対して，コア非空性判定問題は coNP 完全に属することが知られている [2]．

文献 [4] では，MC-nets に対して最小コアに属する利得ベクトル（準配分）を求めるアルゴリズムが提案されている．最小コアに属する利得ベクトルを求める問題はコア非空性判定問題をその一部に含む計算困難な最適化問題である．この問題は，線形計画問題として定式化することができるが，制約式の数がエージェント数に対して指数関数的に増加するという問題がある．そのため [4] では，線形計画問題の制約式を 1 つずつ生成しながら解く制約生成法（列生成法）というアルゴリズムが提案されている．本稿では，[4] の制約生成法を基礎として，各繰り返しで複数の制約式をまとめて生成することができる複数制約生成法（Multi-Constraint Generation）という新しい手法を提案し，提案手法を実装したアルゴリズムの性能を実験で評価する．

2 準備

2.1 MC-nets

MC-nets では，提携が満たすべきルールの集合 R により特性関数を表現する．各ルール $r \in R$ は，$(P_r, N_r) \to v_r$ という形式で記述され，P_r は存在すべきエージェントの集合，N_r は存在すべきでないエージェントの集合であり，$P_r \cap N_r = \emptyset$ を満たす．また，$v_r \in \Re$ は，ルール r の条件部が満たされた場合の（正または負の）利得である．ある提携 $S(\subset A)$ について $P_r \subseteq S$ かつ $N_r \cap S = \emptyset$ のとき，ルール r は提携 S に適用可能であるという．任意の提携 S に適用可能なルール全体の集合を R_S とするとき，S の特性関数値は $v(S) = \sum_{r \in R_S} v_r$ で与えられる．

2.2 最小コア

ある利得ベクトル $x = (x_1, \ldots, x_n)$ とある提携 S 対して，$e(x, S) = v(S) - \sum_{i \in S} x_i$ で計算される値を利得ベクトル x に対する提携 S の不満（excess）という．ここで，ある利得ベクトル x に対してすべての可能な部分提携を範囲としてその不満の最大値を対応させることとし，その最大値を最小化するような利得ベクトル x を求める．このような利得ベクトルの集合を最小コア（least core）という．最小コアは以下の線形計画問題（マスター問題）(1) の最適解の集合に対応させることができる．

$$\begin{array}{ll} \min. & \varepsilon \\ \text{s.t.} & e(x, S) \le \varepsilon, \quad \forall S \subset A, S \neq \emptyset, \\ & \sum_{i \in A} x_i = p_{max}, \end{array} \tag{1}$$

3 提案手法

文献 [4] の手法を基礎として，各繰り返しで複数の制約式をまとめて生成することができる複数制約生成法を提案する．基礎となる [4] の制約生成法の手続きは，以下の Step 1 から Step 6 で構成される．

Step 1. 提携集合 $\mathcal{T}$ の初期値をすべての単独提携を含む任意の提携集合とする．

Step 2. 次の線形計画問題（制限されたマスター問題）の最適解 $(\hat{x}_1, \ldots, \hat{x}_n, \hat{\varepsilon})$ を求める．

$$\begin{array}{ll} \min. & \varepsilon \\ \text{s.t.} & e(x, S) \le \varepsilon, \quad \forall S \in \mathcal{T}, S \neq \emptyset, \\ & \sum_{i \in A} x_i = p_{max}, \end{array}$$

Step 3. $\hat{x} = (\hat{x}_1, \ldots, \hat{x}_n)$ を用いて次の整数計画問題（価格問題）を構成し，その最適値 z^* と最適解の一部である $\alpha^* = (\alpha_1^*, \ldots, \alpha_n^*)$ を求める．なお，R^+ は正の利得をもつルールの集合，R^- は負の利得をもつルールの集合である．

$$\begin{array}{ll} \max. & \sum_{r \in R} v_r \beta_r - \sum_{i \in A} \hat{x}_i \alpha_i \\ \text{s.t.} & \sum_{i \in P_r} \alpha_i + \sum_{i \in N_r} (1 - \alpha_i) \ge |P_r \cup N_r| \beta_r, \ \forall r \in R^+, \\ & \sum_{i \in P_r} (1 - \alpha_i) + \sum_{i \in N_r} \alpha_i \ge 1 - \beta_r, \ \forall r \in R^-, \\ & \sum_{i \in A} \alpha_i \le |A| - 1, \\ & \sum_{i \in A} \alpha_i \ge 1, \\ & \alpha_i, \beta_r \in \{0, 1\}, \ \forall i \in A, \forall r \in R. \end{array}$$

Step 4. マスター問題 (1) の最適値の上界に相当する z^* と下界に相当する $\hat{\varepsilon}$ の値が一致すれば，$\hat{x}$ を出力して終了する．

* 神戸大学大学院海事科学研究科

Step 5. α^* において 1 の値をとるエージェントを選んで提携 S を構成し，S を提携集合 $\mathcal{T}$ に追加する．

Step 6. Step 2 に戻る．

提案手法では，上の Step 5 を次のように変更する．

Step 5′ α^* において 1 の値をとるエージェントを選んで提携 S を構成し，S を提携集合 $\mathcal{T}$ に追加する．さらに，提携 S から k 個以下の任意のエージェントを削除した提携 S' を作り，利得ベクトル $\hat{x}$ に対する提携 S' の不満 $e(\hat{x}, S')$ の値が次の選択基準 1 あるいは 2 の条件を満たすときのみ提携 S' も提携集合 $\mathcal{T}$ に追加する．

選択基準 1（max） $e(\hat{x}, S')$ の値が価格問題の最適値 z^* に一致している場合

選択基準 2（any） $e(\hat{x}, S')$ の値が制限されたマスター問題の最適値 $\hat{\varepsilon}$ を越えている場合

すなわち，選択基準 1 では，提携 S の「$-k$ 近傍」（任意の高々 k 個のエージェントを削除して得られる提携集合）の範囲内で，利得ベクトル $\hat{x}$ に対して S と同じ最大の不満をもつ任意の部分提携 S' をすべて追加する．一方，選択基準 2 では，同じく提携 S の「$-k$ 近傍」の範囲内で，利得ベクトル $\hat{x}$ と $\hat{\varepsilon}$ に対して，マスター問題 (1) の制約式に違反している任意の部分提携 S' をすべて追加する．

本稿では，Step 5′ において選択基準 1 を用いたアルゴリズムを MCG($-k$,max)，選択基準 2 を用いたものを MCG($-k$,any) とよぶ．一方，従来手法を実装したアルゴリズムを SCG（Single-Constraint Generation）とよぶ．

4 評価実験

ベンチマーク問題例を用いた評価実験により，SCG，MCG($-k$,max)，MCG($-k$,any) の性能を比較する．なお，今回の実験では $k \in \{1, 2\}$ とした．実験環境は，Intel Core i7-4770K (3.50GHz, 4 cores)，メモリ 32GB，Windows 10 Home 64bit であり，制限されたマスター問題（線形計画問題）および価格問題（01 整数計画問題）を解く際には IBM ILOG CPLEX 12.7.0.0 を使用した．

まず，文献 [4] の実験で使用されたエージェント数 100 の MC-nets の問題例 100 問を使用する．この 100 問の問題例に対する各手法による累積実行時間（ミリ秒）のカクタスプロット（cactus plot）を図 1 に示す．なお，カクタスプロットによる表記では，グラフの横軸方向に沿ったプロットの「立ち上がり」が遅いほど（プロットがグラフの右下に位置するほど），対応するアルゴリズムの求解速度が全体として速いことを意味する．

次に，重み付きグラフによる提携形ゲームを MC-nets で記述することにより問題例を作成した．重み付きグラフによる提携形ゲームでは，グラフの頂点がエージェント，枝とその重みが 2 エージェント間の「相互作用」を表し，任意の提携 S の特性関数値 $v(S)$ の値は提携 S のメンバー間に張られている枝の重み和となる．文献 [1] では，このゲームの最小コアを求める問題が NP 完全であることが示されている．実験では，頂点数 100，すべての枝のうち $\{0, 3, 5, 8\}$ 割の枝の重みが負で残りの枝の重みが正である完全グラフをそれぞれ 100 個生成して MC-nets に変換し，それらを各アルゴリズムで解いた．紙面の都合上，結果の詳細は省略するが，図 1 の結果と同様に複数制約生成法を実装した MCG(−1,any) の性能が安定して良く，従来アルゴリズムの性能を大きく上回った．

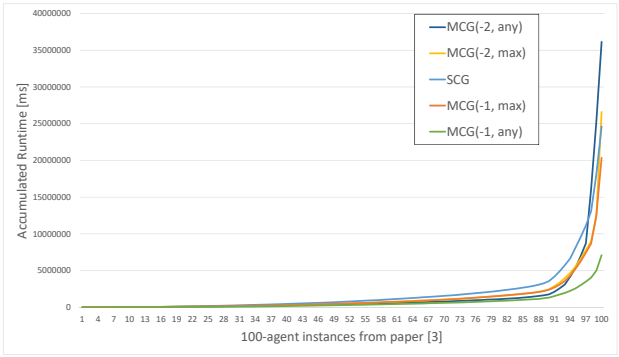

図 1 累積実行時間 [ms] のカクタスプロット

5 おわりに

本論文では，MC-nets で記述された利得分配問題の最小コアを求める複数制約生成法を提案し，実験によりその性能を評価した．その結果，提案手法を実装した MCG(−1,any) の性能が他を大きく上回ることが分かった．今後の課題として，重み付きグラフを用いた実験においてグラフの種類や構造を変えてより詳細な分析を行うこと，上界と下界の差の限界値を使った近似アルゴリズム [3] に今回の提案手法を組み込むこと，等が挙げられる．

参考文献

[1] Deng, X., Papadimitriou, C. H.: On the complexity of cooperative solution concepts. *Mathematics of Operations Research* 19(2), pp. 257–266 (1994)

[2] Greco, G., Malizia, E., Palopoli, L., Scarcello, F.: On the complexity of core, kernel, and bargaining set. *Artificial Intelligence* 175(12–13), pp. 1877–1910 (2011)

[3] Hirayama, K., Hanada, K., Ueda, S., Yokoo, M., Iwasaki, A.: Computing a payoff division in the least core for MC-nets coalitional games. *PRIMA-2014*, pp. 319–332 (2014)

[4] 平山 勝敏，赤木 純，沖本 天太: MC-nets における利得分配：上界保証付き ε-コアを求めるアルゴリズム. *JAWS-2017*, pp. 146–151 (2017)

[5] Ieong, S., Shoham, Y.: Marginal contribution nets: a compact representation scheme for coalitional games. *EC-2005*, pp. 193–202 (2005)