3C-05

MC-nets に基づく大規模提携形ゲームのための上界保証付き ε-コア *

赤木 純 [1†]　　平山 勝敏 [2‡]　　沖本 天太 [2§]

神戸大学海事科学部 [1]　神戸大学大学院海事科学研究科 [2]

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可能な部分集合（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，仁がある．

一方，従来の提携形ゲームでは，特性関数 v は，可能な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．アルゴリズム的ゲーム理論の一分野では，特性関数を簡略表記する基本的な表記法として SCG（Synergy Coalition Group）[1] や MC-nets（Marginal Contribution Networks）[4] 等が提案されており，それらを入力として受理した上で特定の解概念を直接計算するアルゴリズムやその計算コストの研究が進められている．例えば，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]．

平山らは，線形計画問題を解く制約生成法（列生成法）を用いて，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を提案した [3]．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含む計算困難な最適化問題である．[3] では，100 エージェントからなる問題例で最大約 2 時間（6997 秒）で最適解を求めることができたと報告されているが，その一方でそれ以上の規模の問題例では，何らかの近似処理が必要であることが示唆されている．

本論文では，MC-nets に対して最小コアの近似解に相当する ε-コアに属する利得ベクトルを求めるアルゴリズムを提案する．最小コアが，可能なすべての提携上での最大不満を最小化（最大不満の最小値を ε^* とする）する利得ベクトルの集合であるのに対し，ε-コアはその最大不満を ε 以下にする利得ベクトルの集合である．提案アルゴリズムの特徴は，単に ε-コアに属する利得ベクトルを求めるだけでなく，対応する ε の値とその最小値 ε^* の差の上界保証を与えることにある．以下，2 節でアルゴリズムの概要を述べ，3 節で評価実験の結果を報告する．

2 アルゴリズム

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 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，提携 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 手続きとその性質

提案するアルゴリズムの手続きは以下の通りである．

Step 1. 提携集合 $\mathcal{T}$ の初期値をすべての可能な単独提携の集合とする．終了条件である限界値 *bound* を 0 以上の任意の値に設定する．

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．

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

Step 3. $(\hat{x}_1, \ldots, \hat{x}_n)$ 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 $(\alpha_1^*, \ldots, \alpha_n^*)$ 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．

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

Step 4. z^* と $\hat{\varepsilon}$ の差が限界値 *bound* 以下であれば $(\hat{x}_1, \ldots, \hat{x}_n)$ を出力して終了する．そうでなければ $(\alpha_1^*, \ldots, \alpha_n^*)$ で 1 の値をとるエージェントのみを選択して提携を構成し，提携集合 $\mathcal{T}$ に追加する．Step 2 に戻る．

---

* ε-core with Upper Bound Guarantee for MC-nets based Large-scale Coalitional Game

† Jun Akagi, Faculty of Maritime Sciences, Kobe University

‡ Katsutoshi Hirayama, Graduate School of Maritime Sciences, Kobe University

§ Tenda Okimoto, Graduate School of Maritime Sciences, Kobe University

この手続きに関して以下の補題および定理が成立する．

補題 1. $\hat{\varepsilon}$ は最大不満の最小値 ε^* の下界値となる．

略証： 制限されたマスター問題と最小コアを求める線形計画問題の目的関数が同じで，前者の制約集合が後者の制約集合の部分集合であることから自明に導ける．□

補題 2. z^* は最大不満の最小値 ε^* の上界値となる．

略証： 価格問題は，利得ベクトル $(\hat{x}_1, \ldots, \hat{x}_n)$ において最大の不満をもつ提携を，最適解における 01 ベクトル $(\alpha_1^*, \ldots, \alpha_n^*)$ の形で出力し，そのときの最大不満の値を最適値 z^* として出力するよう設計されている．従って z^* は ε^* の上界値となる．□

定理 1. 上記のアルゴリズムは，z^*-コアに属する利得ベクトルを返す．その最大不満の値 z^* とその最小値 ε^* との差は *bound* 以下である．

略証： 補題 1 と 2 が成り立つことと Step 4 の手続きから自明に導ける．□

3 評価実験

提案したアルゴリズムにおいて，*bound* の値を変化させたときのアルゴリズムの性能を実験により評価する．*bound* の値は，任意に与えた分配すべき総利得 p_{max} の値を k 倍するものとし，$k \in \{0.001, 0.005, 0.01, 0.05, 0.1\}$ の範囲でその値を変化させた．k の値が小さいほど最小コアに「近い」利得ベクトルを求めることになる．

今回の実験では，[3] の実験で使用された $n \in \{10, 20, \ldots, 100\}$ の各 n エージェントからなる MC-nets の問題例を 100 問ずつ，合計 1000 問を使用した．実験環境は，Intel Core i7-3960X (3.30GHz, 12 CPUs)，メモリ 32GB，Windows 7 Professional 64bit であり，線形計画問題および整数計画問題を解く際には IBM ILOG CPLEX 12.6.3.0 を使用した．

bound の各値に対するアルゴリズムの累積実行時間（累積追加制約数）のカクタスプロット（cactus plot）を図 1（図 2）に示す．カクタスプロットとは，各アルゴリズムで 1000 問の問題例を実際に解き，その際の実行時間（追加制約数）で 1000 問の問題例を昇順に並べ替えて x 軸に等間隔で配置し，対応する y 座標には 1 番目から x 番目までの問題例の累積実行時間（累積追加制約数）を選択して，x-y 平面上にプロットしたものである．x 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）アルゴリズムの効率が良いことを意味する．図中の各プロットには，*bound* の値とともに 1000 問すべての問題例を解くのに要した累積実行時間（累積追加制約数）が括弧内に示されている．なお，制限時間を 7200 秒（2 時間）に設定し，実行時間がそれを越えた場合には処理を打ち切った．その場合の実行時間はその制限時間とした．そのような問題例は *bound* の値が $p_{max} \times 0.001$ のときに 1 例だけ観測された．図から明らかな通り，*bound* の値（k の値）を増やすにつれて，実行時間および追加制約数ともに大きく減少することが分かる．

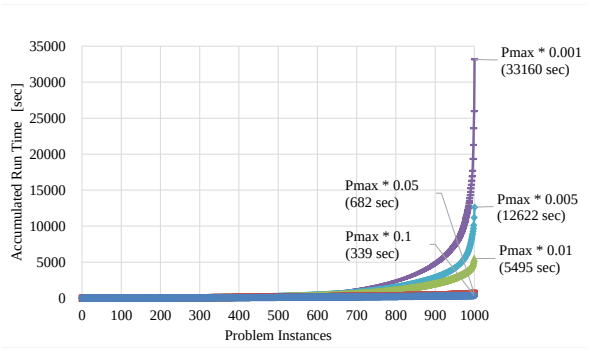

図 1　*bound* の各値に対するアルゴリズムの累積実行時間

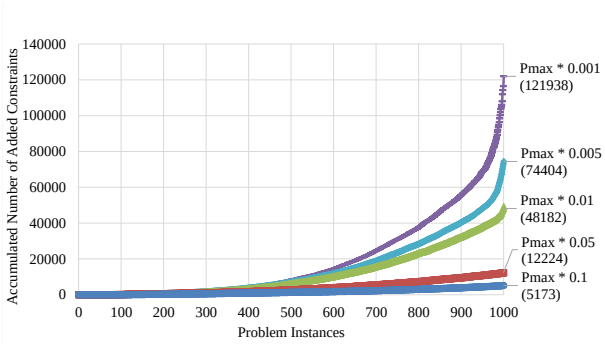

図 2　*bound* の各値に対するアルゴリズムの累積追加制約数

4 おわりに

本論文では，MC-nets に対して，最大不満の値 ε が最小値 ε^* プラス *bound* 以下である ε-コアに属する利得ベクトルを求めるアルゴリズムを提案し，その性能評価を行った．今後の課題は，k の値とコストの関係を詳細に調べること，より規模の大きな問題例で性能評価を行うこと等が挙げられる．

参考文献

[1] Conitzer, V., Sandholm, T.: Complexity of constructing solutions in the core based on synergies among coalitions. *Artificial Intelligence* 170, 607–619 (2006)

[2] Greco, G., Malizia, E., Palopoli, L., Scarcello, F.: On the complexity of core, kernel, and bargaining set. *Artificial Intelligence* 175(12–13), 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 *Proc. of PRIMA-2014*, pp. 319–332 (2014)

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