1F2-2

線形計画法を用いた提携構造形成と利得配分の同時解決

Simultaneous Solving of Coalition Structure Generation and Payoff Distribution with Linear Programming

北木 真 Makoto Kitaki　上田 俊 Suguru Ueda　岩崎 敦 Atsushi Iwasaki　横尾 真 Makoto Yokoo

九州大学大学院システム情報科学府
Graduate School of ISEE, Kyushu University

Coalitional game theory involves how a set of agents can create effective coalitions (Coalition Structure, CS) and how the gain obtained by a coalition should be divided (solution concepts). Traditionally, these two problems have been discussed separately. However, they are actually interdependent and cannot be solved independently. In this paper, we propose a method that can decide a CS and a value division among agents simultaneously. We first show that checking whether the core with an optimal CS is non-empty can be solved in polynomial time when a characteristic function is represented as a Synergy Coalition Group. Next, we develop a branch and bound algorithm for obtaining an optimal CS (as well as a value division among agents), which utilizes a primal-dual solution of an LP and an excess of each agent. Experimental results show that our algorithm is 100-times faster than an existing algorithm.

1. 序論

協力ゲーム理論は，利己的に行動するエージェント間で拘束力がある合意が可能な場合のエージェントの振る舞いに関する理論である [中山 08]．協力ゲーム理論では，(1) 最適な提携構造の形成, (2) 各提携での利得の配分の二つの問題を主な研究対象としており，マルチエージェントシステムの分野ではこれらの問題に対する計算量の分析，アルゴリズムの開発が行われている．(1) の問題は提携構造形成問題 (CSG, Coalition Structure Generation) と呼ばれ，あるエージェントの集合を，全体の利得が最大化されるように分割する問題である．(2) の各提携での利得配分の方法は解概念と呼ばれ，伝統的な協力ゲーム理論において，安定性に着目したコア，公平性に着目したシャプレイ値といった様々な解概念が提案されている [中山 08]．

従来研究では (1) の提携構造形成問題と，(2) の提携内の利得の配分は独立な問題として，逐次的に解決できるものと考えられ，個別に研究が行われてきた．しかしながら，これらの問題は相互依存しており，本来は同時に解くべき問題である．例えば，従来の利得配分における安定性は，全エージェントが一つの提携を形成する場合を対象としていたが，エージェントが複数の提携に分かれて行動する場合，与えられた提携構造を考慮して安定性を考える必要がある．

本論文では，特性関数をシナジー提携集合 (SCG, Synergy Coalition Group) [Conitzer 06] を用いて表現し，最適な提携構造におけるコアの非空判定が，SCG の要素数に関する多項式時間で行えることを示す．また，SCG を用いる場合の提携構造形成問題は組合せオークションにおける勝者決定問題と等価であり，Nisan の提案した勝者決定問題を解く線形計画法を用いたアルゴリズム [Nisan 00] をベースに，協力ゲームの概念である不満 (excess) を用いて改良したアルゴリズムを提案する．さらに，計算機実験によって，エージェント数が 100 の例題において，提案アルゴリズムが既存のアルゴリズムよりも二桁以上高速であることを示す．

連絡先: 北木 真，九州大学大学院システム情報科学府，812-0395 福岡県福岡市西区元岡 744 番地，(092)802-3576, kitaki@agent.inf.kyushu-u.ac.jp

2. 既存研究

2.1 モデル

伝統的な協力ゲームの入力は特性関数で与えられる．

定義 1 (特性関数) 特性関数 $v : 2^A \to \mathbb{R}$ は，エージェントの全体集合を A とするとき，その任意の部分集合 $S \subseteq A$ に対し，S に属するエージェントが協力した際に得る利得 $v(S)$ を与える．

$S_i \cap S_j = \emptyset$ を満たす任意の提携 S_i, S_j について，$v(S_i) + v(S_j) \leq v(S_i \cup S_j)$ が成立するとき，特性関数は優加法的であるという．この場合，全体の利得の合計を最大化するには，すべてのエージェントからなる全体提携 A を形成すればよい．しかしながら，大きな提携においては，エージェント間の調整や通信のオーバーヘッドが問題となることが通例であり，特性関数が優加法性を満たすとは限らない．このような場合にエージェント全体の利得の合計を最大化するためには，エージェントを複数の提携に適切に分割した，提携構造を考える必要がある．詳細には，提携構造 $CS = \{S_1, S_2, \ldots\}$ は以下の性質を満たす必要がある．

$$\forall i, j \ (i \neq j),\ S_i \cap S_j = \emptyset,\ \bigcup_{S_i \in CS} S_i = A.$$

提携構造のもたらす利得 $V(CS)$ は，CS に含まれるすべての提携の利得の和，すなわち $V(CS) = \sum_{S_i \in CS} v(S_i)$ である．提携構造形成問題は，$V(CS)$ が最大となる提携構造，すなわち $\forall CS,\ V(CS^*) \geq V(CS)$ を満たす提携構造 CS^* を求める問題である．

例 1 4 人のエージェント a, b, c, d による協力ゲームにおいて，以下のように特性関数が与えられていると仮定する:

$$\begin{array}{lll} v(\{a\}) = 3, & v(\{b\}) = 3, & v(\{c\}) = 2, \\ v(\{d\}) = 2, & v(\{a,b\}) = 6, & v(\{a,c\}) = 5, \\ v(\{a,d\}) = 5, & v(\{b,c\}) = 5, & v(\{b,d\}) = 5, \\ v(\{c,d\}) = 2, & v(\{a,b,c\}) = 8, & v(\{a,b,d\}) = 8, \\ v(\{a,c,d\}) = 5, & v(\{b,c,d\}) = 5, & v(\{a,b,c,d\}) = 5. \end{array}$$

このとき，$\{\{a,b,c\},\{d\}\},\{\{a,b,d\},\{c\}\}$ 等が最適な提携構造であり，その利得は 10 である．

伝統的な協力ゲーム理論では，特性関数が優加法的である場合に，全体提携の利得を各エージェントに適切に配分する基準である解概念が検討されており，コア，仁，シャプレイ値等さまざまな解概念が提案されている [中山 08]．以下，提携の安定性に着目した解概念であるコアを紹介する．

各エージェントが受け取る利得を，利得ベクトル $\pi = (\pi_1, \pi_2, \ldots)$ によって表現する．コアとは，提携の一部が不満を持ち，エージェントが提携から離脱しないような安定性という性質を持った利得ベクトルの集合のことである．通常のコアの安定性は，エージェントが全体提携を形成することを前提に定義されている．本論文では，特性関数が優加法性を満たさない場合を対象とするため，[Aumann 74] で議論されているように，提携構造における安定性を考える必要がある．以下に提携構造付きコアの定義を示す．

定義 2 (提携構造付きコア) ある提携構造 CS について，以下の条件を満たす利得ベクトル π の集合を提携構造付きコアと言う．

- $\forall S \subseteq A, \sum_{i \in S} \pi_i \geq v(S)$（提携合理性）．
- $\sum_{i \in A} \pi_i = V(CS)$（全体合理性）．

提携合理性とは，任意の提携 S において，提携に属するのエージェントの利得の和が，提携 S が実現可能な利得以上であることを意味する．提携合理性を満たすことにより，任意のエージェントの集合が，現在の提携構造から逸脱する誘因を持たないことを保証する．全体合理性は，利得ベクトルが実現可能である，すなわち，利得ベクトルの総和が，提携構造の利得に等しいことを意味する．

(提携構造付き) コアは，一般には存在することは保証されない．利得ベクトルがコアに属さない場合，各提携に属するエージェントは，割り当てられた利得ベクトルに対して不満を持つ可能性がある．

定義 3 (不満) 割り当てられた利得ベクトル π に対して，$e(S) = v(S) - \sum_{i \in S} \pi_i$ を各提携 S が持つ不満と呼ぶ．

最小コアは，コアを一般化した解概念であり，以下に定義する ε-コアを用いて定義される．

定義 4 (提携構造付き ε-コア) ある提携構造 CS について，以下の条件を満たす利得ベクトル π の集合を提携構造付き ε-コアと言う．

- $\forall S \subseteq A, e(S) \leq \varepsilon$
- $\sum_{i \in A} \pi_i = V(CS)$

定義 5 (提携構造付き最小コア) 提携構造付き最小コアとは，$\varepsilon > 0$ である非空な提携構造付き ε-コアの中で，ε の値が最も小さい提携構造付き ε-コアである．

つまり，最小コアは，全体合理性を満たし，各提携の不満の最大値が最小である利得ベクトルの集合である．

2.2 特性関数の簡略記述法

本節では特性関数の簡略表現法であるシナジー提携集合 (SCG, Synergy Coalition Group) [Conitzer 06] を紹介する．SCG の基本的なアイデアは，エージェントの組合せによって新たな利得が生まれる提携 (シナジー提携) のみに関して利得を記述するというものである．

定義 6 (SCG) SCG は提携 S とその利得 $v(S)$ のペア $(S, v(S))$ の集合である．SCG に直接記述されていない提携 S に関して，その提携の利得 $v(S)$ は $v(S) = \max\{\sum_{S_i \in p_S} v(S_i)\}$ で計算される．ここで p_S は，p_S の任意の要素 S_i は互いに共通部分を持たず，$\bigcup_{S_i \in p_S} S_i = S$ であり，$(S_i, v(S_i)) \in SCG$ であるような S の分割である．また，S が必ず分割可能であるようにするため，$|S| = 1$ なる提携はすべて SCG に含まれるとする．

SCG は優加法的な特性関数しか表現できないため，そのままでは提携構造形成問題に用いることはできないが，Ohta らにより，$|p'_S| \geq 2$ である任意の分割 $p'_S \subseteq p_S$ について，$(\cup_{S_i \in p'_S} S_i, v(\cup_{S_i \in p'_S} S_i))$ は SCG の要素でないという制約を加えることで，優加法的でない特性関数を表記できることが示されている [Ohta 09]．以後，本論文では，特性関数はこの拡張された SCG を用いて表現されていると仮定する．

2.3 組合せオークションにおける勝者決定問題

SCG を用いる場合の提携構造形成問題は重み付き集合充填問題，および従来，AI 分野で盛んに研究が進められてきた，組合せオークションにおける勝者決定問題と等価である．本節では，組合せオークションにおける勝者決定問題について概説する [Nisan 00]．

勝者決定問題では，財の集合 $M = \{1, \cdots, m\}$ が存在し，入札者 i は M の部分集合 $S_i \subseteq M$ に対して，入札額 p_i を提示する．目的は，入札額の合計を最大化するような勝利入札の集合を求めることである．

勝者決定問題は整数計画問題として定式化可能であるが，この問題は NP 困難であり，この問題を解くために必要な時間は入札者数の増加に伴って指数的に増加する．Nisan は整数計画問題を線形緩和した問題と，その双対問題の解を用いて勝者決定問題を解く 2 つのアルゴリズムを提案している [Nisan 00]．

定義 7 (勝者決定問題の線形緩和表現) 勝者決定問題を線形緩和した線形計画問題は以下のように表現される．

$$\begin{array}{ll} \max & \sum x_i p_i \\ \textit{subject to} & \sum_{i|j \in S_i} x_i \leq 1, \ \forall\, j \\ & 0 \leq x_i \leq 1, \ \forall\, i \end{array}$$

ここで x_i は i の入札が勝利入札であることを示す決定変数であり，整数計画法であれば 0/1 の整数値を取るが，緩和問題では 0 から 1 の間の任意の値を取りうる．

定義 8 (勝者決定問題の線形緩和表現の双対問題) 定義 7 の線形計画問題の双対問題は以下のように表現できる．

$$\begin{array}{ll} \min & \sum y_j \\ \textit{subject to} & \sum_{j \in S_i} y_j \geq p_i, \ \forall\, i \\ & y_j \geq 0, \ \forall\, j \end{array}$$

双対問題の解 y は，各財の潜在価格を表している．

Nisan は，定義 7 と定義 8 の線形計画問題を用いた 2 つのアルゴリズムを提案している [Nisan 00]．1 つは，Greedy アルゴリズムである．このアルゴリズムは，計算の効率が良いため多項式時間で実行できるが，得られた解が最適解である保証はない．もう 1 つは，Branch and Bound アルゴリズムである．このアルゴリズムは最適解を得ることを保証するが，実行時間が指数的に長くなる場合がある．

3. 提携構造形成問題と利得配分の同時解決

3.1 勝者決定問題と提携構造形成問題

勝者決定問題の財全体の集合 M をエージェントの全体集合 A，入札 (S_i, p_i) を $p_i = v(S_i)$ となるような SCG の要素 $(S_i, v(S_i))$ に対応させて考えると，勝者決定問題は提携構造形成問題と等価である．このとき，勝者決定問題を解くことで得られる最適な入札の組合せは，提携構造形成問題の最適な提携構造に相当する．また，定義 8 の線形計画問題で得られる解 y は，提携構造付きコアの提携合理性を満たす (一方，全体合理性を満たすとは限らない) 利得ベクトル π を求めていることに相当する．このため，2.3 節で紹介した勝者決定問題を解くアルゴリズムは，提携構造形成問題と利得配分を同時に解決するアルゴリズムとしてそのまま使うことができる．このとき，提携構造形成問題と利得配分を同時に解決する過程における，提携構造付きコアに関する計算量について定理 1 が成り立つ．

定理 1 *最適な提携構造における提携構造付きコアの非空判定は，SCG の要素数に関する多項式時間で行うことができる．*

証明の詳細は省略するが，双対問題の解が全体合理性を満たすとき，かつその時に限り，提携構造付きコアが存在することから導かれる．従来研究 [Conitzer 06] では，通常のコアの非空判定が，全体提携の値が与えられている場合に SCG の要素数に関する多項式時間で実行可能なことを示しているのに対して，本定理では，より一般化的な，提携構造付きコアの非空判定が，最適な提携構造の値が与えられていない場合においても多項式時間で実現可能であるという，より強い結果を与えている．

3.2 提案アルゴリズム

提携構造付きコアが存在する場合には，定義 7 と定義 8 の線形計画問題を解くことにより整数解が得られるが，提携構造付きコアが存在しない場合は整数解が得られない．本節では提携構造付きコアが存在しない場合に，最適な提携構造と最小コアを同時に求める二つのアルゴリズムを提案する．一つは Nisan の提案した Greedy アルゴリズムに基づくものであり，もう一つは Nisan の提案した Branch and Bound アルゴリズムを，不満を用いて改良したものである．

定義 9 (Greedy アルゴリズム)

1. 定義 7 と定義 8 の線形計画問題を解き，その (整数でない) 解 $x_1, \ldots, x_n$ と双対問題の解 $y_1, \ldots, y_m$ を得る．
2. SCG の要素 $(S_i, v(S_i))$ を $v(S_i)/\sum_{j \in S_i} y_j$ の値で降順に並べ替える．$x_i \neq 0$ である SCG の要素 $(v(S_i)/\sum_{j \in S_i} y_j = 1)$ は，x_i の降順で並べ替える．
3. 変数 $WinningBids$ と $AllocatedItems$ を定義し，空集合に初期化する．
4. $i = 1, \cdots, n$ について，$S_i \cap AllocatedItems = \emptyset$ であれば以下の操作を行う．
   (a) $WinningBids$ に $(S_i, v(S_i))$ を加える．
   (b) 提携 S_i に含まれるエージェントをすべて $AllocatedItems$ に加える．
5. $WinningBids$ に含まれる提携によって形成される提携構造が解である．

定義 10 (提案アルゴリズム) 現在求められている最も利得の大きい解の値を表す変数 $LowValue$ を定義する．アルゴリズムがこの値より良い解を発見できない場合，失敗を返す．再帰の一番初めに $LowValue$ を 0 に初期化し，以後は再帰呼び出しで $LowValue$ を更新する．以下にアルゴリズムを示す．

1. 定義 7 の線形計画問題を解き，得られた $\sum x_i v(S_i)$ の値を $UpperBound$ とする．
   - $UpperBound < LowValue$ ならば，失敗を返す．
2. Greedy アルゴリズムを解き，得られた提携構造の利得を $LowerBound$ とする．
   (a) $LowerBound > LowValue$ ならば，$LowValue$ に $LowerBound$ の値を入れ，提携構造および利得ベクトルを記憶する．
   (b) $UpperBound = LowerBound$ ならば，成功を返す．
3. 提携構造付き最小コアを求め，SCG の各要素 $(S_i, v(S_i))$ について $e(S_i)$ の値を計算する．
4. SCG の要素を $e(S_i)$ の値の降順で並べ替える．$e(S_i)$ が同じ場合は，定義 7 の線形計画問題で得られた解 $x(S_i)$ の降順で並べ替える．$x(S_i)$ も同じ場合は，$v(S_i)$ の降順で並べ替える．
5. $(S_i, v(S_i))$ を，並べ替えた SCG の最初の要素とする．
6. $(S_i, v(S_i))$ が最適な提携構造に含まれる場合を考える．
   (a) SCG から $(S_i, v(S_i))$ と S_i に属するエージェントを含む他の SCG の要素を取り除く．
   (b) $LowValue$ から $v(S_i)$ を引いた値を新たな $LowValue$ とし，Branch and Bound アルゴリズムを再帰的に呼び出す．
   (c) 再帰で呼び出した関数から成功が返ってきた場合は，返り値に $v(S_i)$ を足した値を $LowValue$ に代入し，提携構造および利得ベクトルを記憶する．
7. $(S_i, v(S_i))$ が最適な提携構造に含まれない場合を考える．
   (a) SCG から $(S_i, v(S_i))$ を取り除く．
   (b) $LowValue$ はそのままの値を用いて，Branch and Bound アルゴリズムを再帰的に呼び出す．
   (c) 再帰で呼び出した関数から成功が返ってきた場合は $LowValue$ を返り値に更新し，提携構造および利得ベクトルを記憶する．
8. 2,6,7 のいずれかの操作において $LowValue$ が更新された場合は成功を返す．そうでない場合は失敗を返す．

本アルゴリズムでの，Nisan の提案した Branch and Bound アルゴリズムからの改良点は，Greedy アルゴリズムで並べ替えた SCG の要素を，さらに不満を用いて並べ替える点である．すなわち，提案アルゴリズムでは，Greedy アルゴリズムで得られた提携構造における提携構造付き最小コアでの不満が大きい提携から探索を始める．この理由は，現在の提携構造に対してより大きな不満を持つ提携を用いて提携構造を形成した方が，全体で得られる利得をより向上させる可能性があるためである．

4. 実験

本章では，提案アルゴリズムを用いて計算機実験を行い，得られた結果を用いてその性能を評価する．本論文で提示されるすべての実験は，Core 2 Quad Q9650 3GHz プロセッサと 16GB のメモリを搭載した Windows 7 Enterprise x64 Edition マシンで行われた．また，提案した手法で線形計画問題を解くため，市販の最適化エンジンである CPLEX version 12.1 を使用した．

4.1 実験設定

本論文では，以下に提示する 2 つの実験を行った．実験 1 では，SCG の要素数とエージェント数が等しい条件の下，エージェント数を 10 から 100 の間で変化させ，最適な提携構造において提携構造付きコアがある問題の割合を測定した．実験 2 では，実験 1 と同様の設定で，Nisan の提案した Branch and Bound アルゴリズムと提案アルゴリズムに関して，それぞれ探索を終了するまでの時間を測定し比較した．評価のための提携構造形成問題は，以下に示す減衰分布 (decay distribution) によって問題を生成した [Ohta 09]．減衰分布では，まず，ランダムなエージェント 1 人からなる提携を形成し，その提携に確率 α でランダムなエージェントを追加する．エージェントを追加しなくなるか，すべてのエージェントを提携に追加した場合は動作を終了し，1 つの提携を決定する．この動作を繰り返し，生成された提携を SCG の要素として追加していくことで，任意個の提携を生成する．ただし，提携の重複は許さないとする．本論文では，$\alpha = 0.55$ とした．また，任意の提携のもたらす利得は，0 から提携に含まれるエージェント数× 10 の間の一様分布によって決定される (ただし 0 は除く)．この条件の下で，実験 1 と実験 2 ともに問題を 100 問ずつ生成した．また，実験 2 において，生成された問題の最適な提携構造において提携構造付きコアが存在する問題は除去した上で問題を 100 問生成した．

4.2 考察

実験 1 では，エージェント数が 10 の問題では割合が約 93%であることに対し，エージェント数が 60 の問題では約 70%まで減少し，それ以降はエージェントが増えても割合は約 70%のままで大きな変化は見られないことを確認した．定理 1 より，利得配分を決める提携構造付きコアの非空判定は，SCG の要素数に関する多項式時間で行うことができるので，ほとんどの問題において提携構造形成と利得配分は，SCG の要素数に関する多項式時間で解決可能となっている．

次に，図 1 に実験 2 の結果を示す．横軸はエージェント数，縦軸は平均実行時間である．凡例の Nisan は Nisan の提案した Branch and Bound アルゴリズム，Proposed は提案アルゴリズムを表す．エージェント数が 10 や 20 の問題においては，その探索空間が少なく，実行時間にあまり差が見られない．一方，エージェント数が多くなって探索空間が増えると実行時間に明らかに差が出ている．例えばエージェント数 50 の問題では，既存のアルゴリズムの実行時間は 784ms であるのに対し，提案アルゴリズムの実行時間は 46ms である．さらに，エージェント数 100 の問題では二桁以上の高速化が得られている．

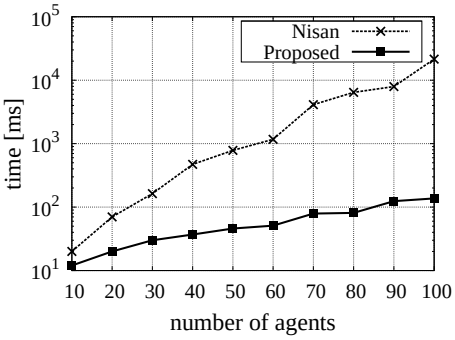

図 1: 各アルゴリズムの実行時間の比較

5. 結論

本論文では，線形計画法を用いて提携構造形成問題と利得配分を同時に解決するアルゴリズムを提案した．SCG を用いた提携構造形成問題は組合せオークションにおける勝者決定問題と等価であり，勝者決定問題における知見を利用可能である．本論文では，最適な提携構造における提携構造付きコアの非空判定が多項式時間で行えることを示した．さらに，最適な提携構造において提携構造付きコアが存在しない場合に対して，既存のアルゴリズムを，協力ゲームにおける概念である不満を用いて改良したアルゴリズムを提案した．また，計算機実験により，エージェント数が 100 の例題において，提案アルゴリズムが既存のアルゴリズムよりも二桁以上高速であることを示した．今後の課題として，Branch and Bound アルゴリズムにおいて，より効率的な探索空間の枝刈りを実現するために，よりよい上界値を得る手法を開発すること等が挙げられる．

参考文献

[Aumann 74] Aumann, R. J. and Dreze, J. H.: Cooperative games with coalition structures, *International Journal of Game Theory*, Vol. 3, pp. 217–237 (1974)

[Conitzer 06] Conitzer, V. and Sandholm, T.: Complexity of Constructing Solutions in the Core Based on Synergies Among Coalitions., *Artificial Intelligence*, Vol. 170, No. 6, pp. 607–619 (2006)

[Nisan 00] Nisan, N.: Bidding and allocation in combinatorial auctions, in *ACM Conference on Electronic Commerce*, pp. 1–12 (2000)

[Ohta 09] Ohta, N., Conitzer, V., Ichimura, R., Sakurai, Y., Iwasaki, A., and Yokoo, M.: Coalition Structure Generation Utilizing Compact Characteristic Function Representations, in *Principles and Practice of Constraint Programming - CP 2009*, pp. 623–638 (2009)

[中山 08] 中山 幹夫, 船木 由喜彦, 武藤 滋夫：協力ゲーム理論, 勁草書房 (2008)