3F3-5

列生成法を用いた提携形ゲームのコア非空性判定アルゴリズム

Column Generation Algorithm for Deciding Core Non-emptiness of Coalitional Games

神谷 竜平*1　花田研太　平山 勝敏
Kamitani Ryouhei　Hanada Kenta　Hirayama Katsutoshi

神戸大学大学院海事科学研究科
Graduate School of Maritime Sciences, Kobe University

We present a new algorithm for determining whether the core is non-empty for a given game in coalitional form represented by MC-nets, one of well-known concise representations of characteristic function. This problem is shown to be coNP-complete and typically solved by formulating it as a linear programming (LP) problem to apply a general LP solver. One drawback of this solution is that such a LP problem has a set of constraints whose size increases exponentially with the number of agents. To alleviate this drawback, we present an algorithm based on the column generation technique, in which starting from a LP problem with a limited number of constraints, it repeatedly solves LP problems while adding the constraint one by one that is identified by solving the pricing problem, which is an integer programming problem obtained through exploiting MC-nets. By using randomly generated MC-nets instances of 10, 20, 30, 40 and 50 agents, we compared the column generation with naive LP solving. We observed that naive LP solving failed due to running out of memory (8GB) on any instance with 20 or more agents, since it had to create more than one million constraints for such instances. On the other hand, we also observed column generation was able to solve all of the instances while generating no more than a few hundreds of constraints.

1. はじめに

協力ゲーム理論における代表的なゲームに提携形ゲームがある．提携形ゲームの問題例は，エージェントの集合 $A = \{1, \ldots, n\}$，および，エージェント間の可能な部分集合（提携）に対する利得を計算する特性関数 $v : 2^A \to \Re$ で構成される．

提携形ゲームの代表的な問題は提携構造形成問題と利得分配問題である．前者は，総利得の最大値 $p_{\max}$ を達成する提携構造 CS^* を求める問題であり，後者は，獲得した総利得 $p_{\max}$ 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問題である．本稿では後者の利得分配問題を扱う．

利得分配問題の重要な解概念の一つにコアがある．コアとは，CS^* の提携に属するどのエージェントも自分が受け取る利得に関して不満をもたない利得ベクトルの集合である．しかしながら，提携形ゲームの任意の問題例に対してそのような利得ベクトルが存在するとは限らず，問題例によってはコアが空となる場合があり得る．そこで，与えられた提携形ゲームの問題例に対して，コアが非空か否かを判定する問題が重要となる．

一方，提携形ゲームでは，従来，特性関数はいわゆるブラックボックス関数とされ，エージェント間の可能な提携を与えれば値を返すものとされていた．しかし，計算機科学の立場ではこの特性関数を具体的にどう実現するかが重要な課題であり，近年，特性関数の簡略記述法として SCG (Synergy Coalition Group) [Conitzer 06] や MC-nets (Marginal Contribution Networks) [Ieong 05] 等が提案されている．

SCG の基本的な考え方は，エージェントの組合せにより新たな利得が生じる場合（シナジー効果がある場合）のみ，提携とその利得を明示的に記述するというものである．一般に SCG による特性関数の簡略記述を得るにはシナジーの有無をチェックしなければならず，そのための計算が最悪時にエージェント数に対して指数的となる．しかし，SCG による特性関数の簡略記述が得られれば，コア非空性判定問題はその記述サイズの多項式時間で解けることが分かっている [Conitzer 06].

一方，MC-nets では，エージェントの有無に関するルールの集合で特性関数を簡略記述する．そこでは，各ルールに利得が定義されており，ある提携の利得は，その提携に対して適用可能なルール集合の利得和となる．なお，MC-nets で表現された提携形ゲームのコア非空性判定問題は，まず coNP 困難であることが示され [Ieong 05]，その後，coNP 完全であることが示された [Malizia 07].

本稿では，提携形ゲームの特性関数が MC-nets で表現されている場合のコア非空性判定問題を解くアルゴリズムを提案し，実験によりその性能を評価する．一般に，コアの非空性を判定する際には線形計画問題を解くことになるが，その線形計画問題には潜在的に膨大な数の制約が存在する．そのため，提案するアルゴリズムでは，MC-nets の特徴を利用した価格問題を解いて必要な制約を適宜生成する列生成法という手法を採用している．これにより，素朴なアルゴリズムでは解くことができなかった比較的大きな規模の問題例においても，コアの非空性が判定できることを示す．

以下，2. 章では，準備として提携形ゲームの基本概念，および，MC-nets の概要を述べる．3. 章では，列生成法の概要を紹介し，MC-nets の特徴を利用した価格問題を導入して，提案アルゴリズムの詳細を具体例を用いて説明する．4. 章では，提案アルゴリズムの評価実験とその結果を報告し，5. 章で本稿をまとめる．

2. 準備

2.1 提携形ゲームの基本概念

提携形ゲームの問題例は，エージェントの集合 $A = \{1, \ldots, n\}$，および，エージェント間の可能な部分集合（提携）

連絡先: 平山勝敏，神戸大学大学院海事科学研究科，〒 658-0022 神戸市東灘区深江南町 5-1-1，hirayama@maritime.kobe-u.ac.jp

*1 現在，関電システムソリューションズ株式会社所属

に対する利得を計算する特性関数 $v : 2^A \to \Re$ で構成される．以下，3 エージェントからなる提携形ゲームの特性関数の例を示す．

例 1（特性関数） エージェント集合 $A = \{1, 2, 3\}$ に対し，以下は特性関数の例である．

$$v(\{\}) = 0,\ v(\{1\}) = 1,\ v(\{2\}) = 2,\ v(\{3\}) = 3,$$
$$v(\{1, 2\}) = 3,\ v(\{1, 3\}) = 3,\ v(\{2, 3\}) = 7,$$
$$v(\{1, 2, 3\}) = 7$$

例えば，$v(\{2, 3\}) = 7$ は，2 と 3 が提携して協力した場合に利得 7 が得られることを示す．

提携形ゲームにおける代表的な問題の 1 つに提携構造形成問題がある．提携構造とは，エージェント集合 A に対する分割であり，提携構造形成問題では，提携構造による総利得の最大値 $p_{\max}$ を達成する A の分割 CS^* を求める．この問題は NP 困難な問題に属する．

例 2（提携構造形成問題） 例 1 の提携形ゲームに対する提携構造形成問題の解は，$CS^* = \{\{1\}, \{2, 3\}\}$ であり，そのときの総利得の最大値は $p_{\max} = v(\{1\}) + v(\{2, 3\}) = 8$ である．すなわち，この例では，エージェント 2 と 3 が提携し，エージェント 1 が単独提携となることで最も高い総利得 8 を得る．

提携形ゲームにおけるもう 1 つの問題は利得分配問題である．これは，最適な提携構造により獲得した総利得 $p_{\max}$ を各エージェントにどのように分配するかを決める問題である．以降，分配の結果，各エージェント i は x_i の利得を得るものとし，それを利得ベクトル $x = (x_1, \ldots, x_n)$ で表す．利得分配問題に対してはいくつかの解概念が提案されている．そのうち，不満という概念に基づくコア，ε-コア，仁，さらには，貢献度という概念に基づくシャプレイ値が良く知られた代表的な解概念である．本稿ではコアを扱う．

コアとは，CS^* の提携に属するどのエージェントも自分が受け取る利得に関して不満をもたない利得ベクトルの集合である．形式的には，以下の 2 つの条件を満たす利得ベクトルの集合となる．

$$\sum_{i \in S} x_i \geq v(S),\ \forall S \subseteq A, \tag{1}$$

$$\sum_{i \in A} x_i = p_{\max}. \tag{2}$$

ここで，式（1）の条件を提携合理性，式（2）の条件を全体合理性という．

例 3（コア） 例 1 の提携形ゲームにおいて，利得ベクトル $x = (1, 3, 4)$ は提携合理性と全体合理性を満たすためコアに属する．一方，$x' = (1, 1, 6)$ は，$v(\{1, 2\}) > 2$ ゆえ，エージェント 1 と 2 が不満をもつ（提携 $\{1, 2\}$ の提携合理性が満たされない）ためコアには属さない．

提携形ゲームの任意の問題例に対して上記のような利得ベクトルが存在するとは限らない．すなわち，問題例によってはコアが空となる場合があり得る．そこで，与えられた提携形ゲームの問題例に対して，コアが非空か否かを判定することが重要となる．この問題はコア非空性判定問題とよばれ，一般には以下の線形計画問題

$$\begin{aligned} p_{\min} \ = \ & \min_{x \in \Re^n} . \sum_{i \in A} x_i \\ & \text{s.t.} \sum_{i \in S} x_i - v(S) \geq 0,\ \forall S \in 2^A - CS^* - \emptyset \end{aligned} \tag{3}$$

を解き，その最適値 $p_{\min}$ が，提携構造形成問題による利得の総和の最大値 $p_{\max}$ 以下であればコアは非空，そうでなければコアは空，という手続きで解くことができる．ここで，$p_{\min}$ は，最適な提携構造 CS^* に含まれない任意の提携に対して提携合理性を満たすような利得ベクトルを構成するのに必要な最小の利得和に相当する．ただし，(3) では，可能な提携 S について 2 行目の不等式制約をすべて列挙しなければならず，エージェント数が大きい場合には事実上解くことは不可能となる．

2.2 MC-nets

提携形ゲームでは，従来，特性関数はいわゆるブラックボックス関数とされ，エージェント間の可能な提携を与えれば値を返すものとされていた．近年，特性関数の簡略記述法として SCG [Conitzer 06] や MC-nets [Ieong 05] 等が提案されている．本稿では，後者の MC-nets を扱う．

MC-nets では，提携が満たすべきルールの集合 R によって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 は存在しなければならないエージェントの集合，N_r は存在してはならないエージェントの集合であり，$P_r \cap N_r = \emptyset$ である．また，$v_r \in \Re$ は，ルール r の条件部が満たされた場合の利得である．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．S に適用可能なルールの全体集合を R_S とするとき，任意の提携 S の利得は $v(S) = \sum_{r \in R_S} v_r$ で与えられる．

例 4（MC-nets） ルール集合 $R = \{r_1, r_2, r_3, r_4\}$, ただし，

$$r_1 : \ (\{1\}, \{3\}) \to 1,$$
$$r_2 : \ (\{2\}, \emptyset) \to 2,$$
$$r_3 : \ (\{3\}, \emptyset) \to 3,$$
$$r_4 : \ (\{2, 3\}, \emptyset) \to 2,$$

は，任意の提携に対して例 1 と同じ特性関数値を与える．例えば，提携 $\{2, 3\}$ に対して，ルール r_2, r_3, r_4 が適用可能であり特性関数値は 7 となる．

なお，MC-nets は fully expressive，すなわち，任意の特性関数を表現できることが示されている [Ieong 05].

3. MC-nets におけるコア非空性判定

MC-nets で表現された提携形ゲームのコア非空性判定問題は coNP 完全であり [Malizia 07]，任意の問題例に対して効率的なアルゴリズムを設計することは事実上不可能である．本稿では，ある程度の規模の問題例を比較的効率よく解くことができるアルゴリズムを考案する．なお，[Ieong 05] では，独自に設計されたコアメンバシップアルゴリズムと楕円体法を組合わせてコア非空性判定問題を解くことが示唆されているが，評価実験等はなされておらずアルゴリズムの実際の性能は明らかにされていない．

本稿では，列生成法という手法により，すべての不等式制約をあらかじめ列挙することなく（3）の線形計画問題を解いてコアの非空性を判定する新しい解法を提案し，実験によりその性能を評価する．提案する解法では，MC-nets の特徴を利用した固有の価格問題を解くことにより必要な不等式制約を適宜生成する．もちろん，最悪時にはすべての制約を列挙する必要があるが，ランダムな問題例を用いた実験によれば，ごく少数の制約を生成するだけでコアの非空性を判定できることが分かる．

3.1 列生成法の概要

一般に列生成法とは，潜在的に膨大な数の変数あるいは膨大な数の制約をもつ線形計画問題のための解法である．以下，潜在的に膨大な数の制約をもつ線形計画問題（3）を主マスター問題とよぶ．列生成法では以下の処理を繰り返す.

ステップ 1 主マスター問題 MP の一部の制約集合（初期制約集合）からなる制限された主マスター問題 RMP を作る.

ステップ 2 RMP を解く.

ステップ 3 RMP の最適解 $\tilde{x}$ から価格問題 PP を構成して解く．その結果,（もしあれば）RMP に追加すべき制約が求められる．もしそのような制約がなければ，$\tilde{x}$ を MP の最適解として終了する．一方，もし追加すべき制約があれば，それを RMP に追加してステップ 2 へ戻る.

なお，初期制約集合を構成する際には，RMP が非有界にならないように（3）の制約集合から制約を任意に選択する．典型的には，最適な提携構造 CS^* の提携を含まない任意の提携構造を選択し，その各提携について提携合理性の制約を作って初期制約集合とする.

例 5（制限された主マスター問題） 例 4（例 1）に対する最初の制限された主マスター問題は例えば以下の通りである.

$$\begin{aligned} &\min . \ x_1 + x_2 + x_3 \\ &\text{s.t. } x_1 + x_2 + x_3 - 7 \geq 0, \end{aligned} \tag{4}$$

最適解の 1 つは例えば $\tilde{x} = (1, 1, 5)$ となる.

3.2 価格問題

主マスター問題 MP とその双対問題の相補性の条件から，制限された主マスター問題 RMP の最適解が MP の最適解に一致するための必要かつ十分な条件は，RMP の最適解 $\tilde{x}$ が MP の実行可能解であることが導ける．実行可能性をチェックするにはすべての制約をチェックすることが基本だが，主マスター問題の制約の数は膨大であるため，それは事実上不可能である．しかしながら，特性関数が MC-nets で記述されている場合には，価格問題とよばれる以下の整数計画問題の最適値を求め，それが 0 以上であれば $\tilde{x}$ が MP の実行可能解であり，かつ，最適解であることがいえる.

$$\begin{aligned} &\min . \ \sum_{i \in A} \tilde{x}_i \alpha_i - \sum_{r \in R} v_r \beta_r \\ &\text{s.t. } \sum_{i \in P_r} \alpha_i + \sum_{i \in N_r} (1 - \alpha_i) \geq |P_r \cup N_r| \beta_r, \ \forall r \in R, \\ &\quad \sum_{i \in S} \alpha_i + \sum_{i \in A - S} (1 - \alpha_i) \leq |A| - 1, \ \forall S \in CS^*, \\ &\quad \alpha_i, \beta_r \in \{0, 1\}, \ \forall i \in A, \ \forall r \in R. \end{aligned} \tag{5}$$

この価格問題の最適値は，(3) のすべての制約の左辺に $\tilde{x}$ を代入した場合の最小値を与える．また，最適解 $(\tilde{\alpha}, \tilde{\beta})$ のうち $\tilde{\alpha}$ は，制約の左辺の最小値を与える提携 $\tilde{S}$ に対応する．従って，この最小値が 0 以上であれば，$\tilde{x}$ は（3）のすべての制約を満たしており，一方，0 未満であれば，$\tilde{x}$ は提携 $\tilde{S}$ の提携合理性の条件を壊しているため，制約として

$$\sum_{i \in \tilde{S}} x_i - v(\tilde{S}) \geq 0$$

を追加して RMP を更新し，ステップ 2 に戻って同じ手続きを繰り返す.

例 6（価格問題 1） （4）の最適解 $\tilde{x} = (1, 1, 5)$ に対する価格問題は以下の通りである.

$$\begin{aligned} &\min . \ \alpha_1 + \alpha_2 + 5\alpha_3 - \beta_1 - 2\beta_2 - 3\beta_3 - 2\beta_4 \\ &\text{s.t. } \alpha_1 + (1 - \alpha_3) \geq 2\beta_1, \\ &\quad \alpha_2 \geq \beta_2, \\ &\quad \alpha_3 \geq \beta_3, \\ &\quad \alpha_2 + \alpha_3 \geq 2\beta_4, \\ &\quad \alpha_1 + (1 - \alpha_2) + (1 - \alpha_3) \leq 2, \\ &\quad \alpha_2 + \alpha_3 + (1 - \alpha_1) \leq 2, \\ &\quad \alpha_i, \beta_r \in \{0, 1\}, \ \forall i \in A, \ \forall r \in R. \end{aligned} \tag{6}$$

これを解けば，最適値が -1，最適解 $\alpha = (1, 1, 0)$, $\beta = (1, 1, 0, 0)$ ゆえ，提携 $\{1, 2\}$ の提携合理性の条件，すなわち，$x_1 + x_2 - 3 \geq 0$ を（4）に追加し，以下の新しい制限された主マスター問題を得る.

$$\begin{aligned} &\min . \ x_1 + x_2 + x_3 \\ &\text{s.t. } x_1 + x_2 + x_3 - 7 \geq 0, \\ &\quad x_1 + x_2 - 3 \geq 0. \end{aligned} \tag{7}$$

最適解の 1 つは例えば $\tilde{x} = (1, 2, 4)$ となる.

例 7（価格問題 2） （7）の最適解 $\tilde{x} = (1, 2, 4)$ に対する価格問題は以下の通りである.

$$\begin{aligned} &\min . \ \alpha_1 + 2\alpha_2 + 4\alpha_3 - \beta_1 - 2\beta_2 - 3\beta_3 - 2\beta_4 \\ &\text{s.t. } \alpha_1 + (1 - \alpha_3) \geq 2\beta_1, \\ &\quad \alpha_2 \geq \beta_2, \\ &\quad \alpha_3 \geq \beta_3, \\ &\quad \alpha_2 + \alpha_3 \geq 2\beta_4, \\ &\quad \alpha_1 + (1 - \alpha_2) + (1 - \alpha_3) \leq 2, \\ &\quad \alpha_2 + \alpha_3 + (1 - \alpha_1) \leq 2, \\ &\quad \alpha_i, \beta_r \in \{0, 1\}, \ \forall i \in A, \ \forall r \in R. \end{aligned} \tag{8}$$

これを解けば，最適値が 0 ゆえ，$\tilde{x} = (1, 2, 4)$ は主マスター問題の制約をすべて満たしており，その最適解となる．主マスター問題の最適値は $p_{\min} = 7$ で，例 2 より提携構造形成問題の最適値 $p_{\max} = 8$ 以下なので，例 4 の MC-nets の問題例ではコアは非空である.

以上，例 4 の MC-nets の問題例に対するコアの非空性を判定するにあたり，提携合理性の制約をすべて生成することなく，(7) に示した 2 つの制約を生成するだけでコアは非空と判定できた次第である.

表 1: 各アルゴリズムの平均実行時間 Time [sec] と平均追加制約数 Cons

		$\|A\|=10$		$\|A\|=20$		$\|A\|=30$		$\|A\|=40$		$\|A\|=50$	
		Naive	Col.Gen.	Naive	Col.Gen.	Naive	Col.Gen.	Naive	Col.Gen.	Naive	Col.Gen.
$\|R\|=10$	Time	**0.016**	0.14	n/a	**0.39**	n/a	**0.66**	n/a	**1.00**	n/a	**1.47**
	Cons	1022	**21.4**	$\approx 10^6$	**50.0**	$\approx 10^9$	**77.1**	$\approx 10^{12}$	**111.3**	$\approx 10^{15}$	**151.3**
$\|R\|=50$	Time	**0.019**	1.74	n/a	**4.68**	n/a	**9.00**	n/a	**13.5**	n/a	**16.4**
	Cons	1022	**26.0**	$\approx 10^6$	**61.5**	$\approx 10^9$	**102.4**	$\approx 10^{12}$	**136.1**	$\approx 10^{15}$	**171.1**
$\|R\|=100$	Time	**0.025**	2.63	n/a	**9.36**	n/a	**22.0**	n/a	**36.9**	n/a	**64.7**
	Cons	1022	**28.2**	$\approx 10^6$	**63.4**	$\approx 10^9$	**111.6**	$\approx 10^{12}$	**160.3**	$\approx 10^{15}$	**210.8**

4. 実験

提案した列生成法の効果を確認するための実験を行う．本実験では，ランダムに生成した MC-nets の問題例を用いて，列生成法によるアルゴリズム（Column Generation）と（3）の線形計画問題の制約を事前に全列挙して解く素朴なアルゴリズム（Naive）を比較する．

4.1 設定

エージェント数 $|A| \in \{10, 20, 30, 40, 50\}$，MC-nets のルール数 $|R| \in \{10, 50, 100\}$ とし，各ルール r について，存在しなければならないエージェントの集合 P_r，存在してはならないエージェントの集合 N_r を互いに交わりを持たないようにランダムに生成し，条件が満たされたときの利得 v_r は 1 以上 10 以下の整数をランダムに設定した．以上，エージェント数 $|A|$ とルール数 $|R|$ の可能な組合せ 15 通りのそれぞれについて，100 個の問題例をランダムに生成し，線形計画問題の最適解を発見するまでの平均実行時間，および，平均追加制約数を測定した．

なお，本来，事前に提携構造形成問題を解いて最適な提携構造 CS^* を求める必要があるが，本実験では仮想的に全体提携が最適な提携構造であると仮定して，価格問題を構成している．また，制限された主マスター問題の初期制約集合は，単独提携の提携合理性の条件（すなわち $|A|$ 個の不等式）とした．

各アルゴリズムは Java で実装され，線形計画問題と整数計画問題を解く際には CPLEX 12.3 のライブラリを使用した．実験環境は，CPU: Core i7 2600（3.4GHz, 4 cores, 8 threads），Memory: 8GB，OS: Ubuntu 11.10 (64bit)，Java の実行環境: Java 1.6.0_27 である．

4.2 結果

実験結果を表 1 に示す．エージェント数 $|A|$ が 10 のとき，素朴なアルゴリズム（Naive）は，MC-nets の記述から約 1000 個の制約を事前に生成して（3）の線形計画問題を解くが，この程度の規模の問題例であれば CPLEX の線形計画ソルバーは 0.1 秒未満で解くことがわかる．一方，列生成法によるアルゴリズム（Col.Gen.）では，追加する制約の数は平均 20 から 30 個と僅かだが，平均すると 10 から 20 回程度制限された主マスター問題（線形計画問題）と価格問題（整数計画問題）を繰り返し解くため，全体としては素朴なアルゴリズムよりも多くの実行時間を要する．ルール数 $|R|$ が 100 の場合には，列生成法によるアルゴリズムの平均実行時間は素朴なアルゴリズムの約 100 倍である．

一方，エージェント数 $|A|$ が 20 以上になると，素朴なアルゴリズムは 100 万を超える制約を事前に生成する必要があり，今回の実験環境である合計 8GB のメモリ空間内にデータを収めることができなかった．その場合，表 1 の平均実行時間の欄には n/a と表示している．一方で列生成法によるアルゴリズムが生成する制約の数は 50 から 210 個程度であり，大幅に少ない数の制約を生成するだけでコアの非空性を判定できることがわかる．

ルール数を固定してエージェント数を増加させた場合，素朴なアルゴリズムでは，制約数は明らかに指数関数的に増加するが，列生成法によるアルゴリズムでは，平均追加制約数の増加率は相対的に非常に低い値となっている．また，エージェント数を固定してルール数を増加させた場合も同様である．

5. おわりに

本稿では，MC-nets で記述された提携形ゲームのコア非空性判定問題を解く新たなアルゴリズムとして，列生成法によるアルゴリズムを提案した．ランダムな問題例を用いた評価実験によれば，エージェント数が 20 以上になると，素朴なアルゴリズムではすべての制約を列挙して線形計画問題を事前に作るためメモリを極端に消費する．一方，列生成法によるアルゴリズムでは，繰り返し線形計画問題と整数計画問題を解きながら必要な制約を生成するため，ある程度の規模の問題例であれば基本的にメモリの問題を気にする必要はない．しかしながら，列生成法によるアルゴリズムでも最悪時にはすべての制約を生成することになるため，当然限界は存在する．現実的にどの程度の規模の問題例が列生成法によるアルゴリズムで解けるのかは今後の実験等で明らかにしたい．

参考文献

[Conitzer 06] Conitzer, V. and Sandholm, T.: Complexity of constructing solutions in the core based on synergies among coalitions, *Artificial Intelligence*, Vol. 170, pp. 607–619 (2006)

[Ieong 05] Ieong, S. and Shoham, Y.: Marginal Contribution Nets: A Compact Representation Scheme for Coalitional Games, in *Proceedings of the 6th ACM conference on Electronic Commerce (EC-2005)*, pp. 193–202 (2005)

[Malizia 07] Malizia, E., Palopoli, L., and Scarcello, F.: Infeasibility Certificates and the Complexity of the Core in Coalitional Games, in *Proceedings of the 20th International Joint Conference on Artificial Intelligence (IJCAI-2007)*, pp. 1402–1407 (2007)