Vol.2015-ICS-179 No.8
2015/3/20

ロバストな提携構造形成問題のための問題変換アルゴリズムについて

中野 克哉[1,a)]　白松 俊[1]　大囿 忠親[1]　新谷 虎松[1]

概要：提携構造形成（CSG）問題の新しい形態の一つとして，提携構造のロバスト性に着目したロバストな提携構造形成（RCSG）問題がある．ロバストな提携構造とは，提携構造内の任意のエージェントが離脱しても，社会的余剰（提携の形成によって得られる利得の総和）が最大のままである提携構造を指す．RCSG を解くためには，全ての提携構造に対して，任意のエージェントが離脱する場合を検討する必要があり，従来の CSG の求解アルゴリズムを利用できない．本研究では，RCSG をより効率的に求解可能な CSG に変換する手法を提案する．RCSG を考慮した上界および下界を用いて RCSG を CSG に変換することで，従来の CSG の求解アルゴリズムでの求解が可能となる．

キーワード：ロバストな提携構造形成問題，提携の上界および下界，提携束，問題変換アルゴリズム

1. はじめに

提携構造形成（Coalition Structure Generation，CSG）問題 [1] は，マルチエージェントシステムにおける重要な問題の一つである．CSG では，あるエージェントの集団に対して，社会的余剰が最大化するような提携を形成させることを目的としている．CSG における社会的余剰とは，各エージェントが提携を形成することによって得られる利得の総和を指している．

提携構造は最適であるのみならず，エージェントの離脱に対して頑健でなければならない．エージェントの離脱には次の二つの問題が存在する．一つ目の問題は，エージェント離脱後の解の非効率性に関する問題である．エージェント離脱前の最適解と新たな状況における最適解との差異が大きい場合，新たな最適解を適用するよりも，元の解を実行した場合が好ましい場合もある．二つ目の問題は，エージェント離脱後の求解に利用可能な時間が不十分な場合が存在するという問題である．このような問題を想定して，予めエージェントの離脱を考慮しておくことは自然な考えである．

提携構造からのエージェントの離脱を想定した新しい形態の CSG として，ロバストな提携構造形成問題（Robust Coalition Structure Generation, RCSG）問題 [2] がある．RCSG におけるロバストな提携構造とは，提携構造内の任意のエージェントが離脱しても，社会的余剰が最大のままである提携構造を指す．沖本ら [4] は，ロバストなチーム編成問題 (Robust Team Formation Problem, RTFP) のフレームワークを定義している．RTFP は，RCSG における頑健性を考慮した提携を形成することと同義である．この研究では，形成するチームの頑健性を考慮しても，RTFP の計算複雑度は頑健性を考慮しない場合と比較しても，変化しないことを示している．

RCSG を解くためには，全ての提携構造に対して，任意のエージェントが離脱する場合を検討する必要があり，従来の CSG の求解アルゴリズムを利用できない．先行研究では，RCSG を効率的に求解するための新しいデータ構造として提携の包含関係を明確化した提携束 (Coalition Lattice) を提案した．提携束を形成することにより，頑健性を持つ提携を効率的に判別可能となった．従来の手法には，提携束を形成する際，提携の下界の計算部分に改善の余地があった．

本稿では，RCSG をより効率的に求解可能な CSG に変換する手法を提案する．具体的には，提携束の形成アルゴリズムを利用し，提携の上界と下界によって余分な部分の枝刈りを行うことで，先行研究で提案した形成アルゴリズムを改良し，問題変換アルゴリズムとして新たに提案する．本稿で提案する問題変換アルゴリズムによって，RCSG を CSG に変換することが可能であり，従来の CSG の求解アルゴリズムでの求解が可能となる．

[1] 名古屋工業大学大学院 情報工学専攻
Department of Computer Science and Engineering, Graduate School of Engineering, Nagoya Institute of Technology
[a)] nkatsu@toralab.org

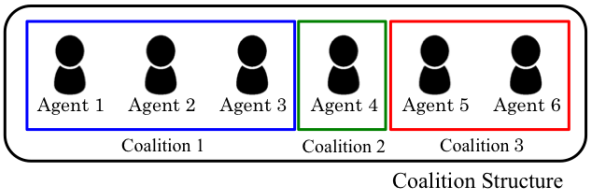

図 1 提携構造の例
Fig. 1 An example of coalition structures.

2. ロバストな提携構造形成問題

最適な提携構造を形成した後に，あるエージェントが事故や病気などの突発的な理由で提携から離脱する場合，残りのエージェントで最適な提携構造を形成し直すことが必要になる可能性がある．その場合，最適な提携構造を再計算するために膨大な時間がかかってしまう．現実的な時間で再計算が可能だとしても，初期に形成された提携構造とは全く異なる提携構造が形成される可能性がある．初めからエージェントの突発的な離脱を想定し，提携構造を形成することは自然な考えである．提携構造のロバスト性に着目した研究 [2] がある．この研究では，ロバストな提携構造形成（RCSG）問題のフレームワークを定義している．

提携構造形成（CSG）問題では，A をエージェントの集合とした場合，A の部分集合 $S \subseteq A$ を “提携（Coalition）” と呼び，式 (1) を満たすものを A の “提携構造（Coalition Structure）” と呼ぶ．

$$\forall i, j (i \neq j),\ \ S_i \cap S_j = \emptyset,\ \ \bigcup_{S_i \in CS} S_i = A \qquad (1)$$

図 1 は，エージェント 6 体で形成可能な提携構造の一つを表している．各エージェントは単一の提携に属し，同時に複数の提携に属することはない．ある提携 S に属するエージェントが相互に協力することで得られる利得は，$v : 2^A \rightarrow \mathbb{N}$ を”特性関数”としたとき，$v(S)$ により与えられる．特性関数 v は多項式時間内に計算されるものであるとする．提携構造 CS の利得 $V(CS)$ は，式 (2) のように提携構造内の全提携の利得の総和で与えられる．

$$V(CS) = \sum_{S_i \in CS} v(S_i) \qquad (2)$$

CSG では，式 (3) を満たすような CS^* を，“最適な提携構造” と呼ぶ．

$$\forall CS : V(CS) \leq V(CS^*) \qquad (3)$$

k を非負整数とし，A の部分集合を $A' \subseteq A$ とする．全ての $k \leq |A'|$ $(0 \leq k \leq |A| - 2)$ に関して，式 (4) を満たすような $A \setminus A'$ における提携構造 CS' が存在しない場合，CS を “k-robust な提携構造” と呼ぶ．CSG において，任意の提携構造 CS は 0-robust である．また，全ての CS は

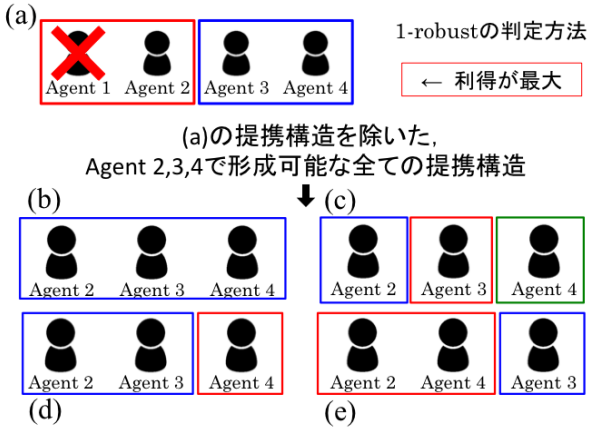

図 2 1-robust な提携構造の判定
Fig. 2 Judgment of the 1-robust coalition structure.

$(|A| - 1)$-robust であるため，k の範囲は $0 \leq k \leq |A| - 2$ としている．

$$V(CS \setminus A') < V(CS') \qquad (4)$$

A における，ある提携構造 CS が k-robust であることを判定するためには，CS からエージェントが離脱する全ての場合（${}_{|A|}C_k$ 通り）を調べる必要がある．図 2 は，ある提携構造が 1-robust な提携構造かどうかを判定する例を示している．図 2 の提携構造 (a) は，エージェント数が 4（$|A| = 4$）の場合に形成可能な提携構造の一つである．(a) が 1-robust な提携構造であるためには，(a) から任意のエージェントが離脱しても，残りのエージェントで形成される他の全ての提携構造と比較して，利得が最大である必要がある．図 2 は Agent1 が (a) から離脱する場合を示している．図 2 の提携構造 (b),(c),(d),(e) は，Agent2,3,4 で形成可能な全ての提携構造である（(a) から Agent1 が離脱した状態を除く．）(a) が 1-robust な提携構造であるためには，(b),(c),(d),(e) と比較して，(a) から Agent1 が離脱した状態の提携構造の利得が最大である必要がある．加えて，(a) から Agent2,3,4 がそれぞれ離脱した全ての状態で利得が最大である必要がある．

k-robust な提携構造 CS_R および k'-robust な提携構造 CS'_R に関して，$k \geq k'$ かつ $V(CS_R) > V(CS'_R)$，または $k > k'$ かつ $V(CS_R) \geq V(CS'_R)$ が成立するとき，CS_R は CS'_R を支配するという．また，ある提携構造 CS_R を支配するような他の提携構造 CS'_R が存在しない場合，CS_R は “パレート最適なロバストな提携構造” であるという．RCSG のゴールは，提携構造の利得および k の値がトレードオフとなるような，全ての提携構造を見つけることである．全てのパレート最適なロバストな提携構造を見つけるためには，効果的なアルゴリズムが必要である．

表 1 提携，提携構造，および特性関数
Table 1 Coalitions, coalition structures, and characteristic function.

提携 S	(v,δ)	提携構造 CS	(V,Δ)
$\{a_1\}$	$(1,0)$	$\{\{a_1\}\{a_2\}\{a_3\}\{a_4\}\}$	$(13,0)$
$\{a_2\}$	$(5,0)$	$\{\{a_1,a_2\}\{a_3\}\{a_4\}\}$	$(15,2)$
$\{a_3\}$	$(3,0)$	$\{\{a_1,a_2\}\{a_3,a_4\}\}$	$(17,4)$
$\{a_4\}$	$(4,0)$	$\{\{a_1\}\{a_2\}\{a_3,a_4\}\}$	$(15,2)$
$\{a_1,a_2\}$	$(8,2)$	$\{\{a_1,a_3\}\{a_2\}\{a_4\}\}$	$(14,1)$
$\{a_1,a_3\}$	$(5,1)$	$\{\{a_1,a_3\}\{a_2,a_4\}\}$	$(9,-4)$
$\{a_1,a_4\}$	$(3,-2)$	$\{\{a_1\}\{a_3\}\{a_2,a_4\}\}$	$(8,-5)$
$\{a_2,a_3\}$	$(5,-3)$	$\{\{a_1,a_4\}\{a_2\}\{a_3\}\}$	$(12,-1)$
$\{a_2,a_4\}$	$(4,-5)$	$\{\{a_1,a_4\}\{a_2,a_3\}\}$	$(8,-5)$
$\{a_3,a_4\}$	$(9,2)$	$\{\{a_1\}\{a_4\}\{a_2,a_3\}\}$	$(10,-3)$
$\{a_1,a_2,a_3\}$	$(10,1)$	$\{\{a_1,a_2,a_3\}\{a_4\}\}$	$(14,1)$
$\{a_1,a_2,a_4\}$	$(7,-3)$	$\{\{a_1,a_2,a_4\}\{a_3\}\}$	$(10,-3)$
$\{a_1,a_3,a_4\}$	$(10,2)$	$\{\{a_1,a_3,a_4\}\{a_2\}\}$	$(15,2)$
$\{a_2,a_3,a_4\}$	$(8,-4)$	$\{\{a_2,a_3,a_4\}\{a_1\}\}$	$(9,-4)$
$\{a_1,a_2,a_3,a_4\}$	$(19,6)$	$\{\{a_1,a_2,a_3,a_4\}\}$	$(19,6)$

3. 頑健性を考慮した提携の上界と下界

本節では，頑健性を考慮した上界および下界の計算手法を示す．これにより，RCSG における効率的な求解手法が実現可能になる．

3.1 上界および下界

本研究では，エージェント集合 A における，提携 S および提携構造 CS の利得の計算において，次式のように線形和からの差分 $\delta(S)$ および $\Delta(CS)$ を用いる．

$$\delta(S) = v(S) - \sum_{a_i \in S} v(\{a_i\}) \tag{5}$$

$$\Delta(CS) = \sum_{S_i \in CS} \delta(S_i) \tag{6}$$

ここで，$|S| = 1$ なる任意の提携 S，すなわち単体のエージェントから構成される提携は，$\delta(S) = 0$ である．このとき，$\delta(S) > 0$ となる S を “正の提携” と呼び，$\delta(S) < 0$ となる S を “負の提携” と呼ぶ．本稿では，ある CS が次式を満たすとき，“自由な提携構造” と呼ぶ．また，$|S| = 1$ のとき，S を “自由な提携” と呼び，エージェント $a \in S$ を “自由なエージェント” と呼ぶ．すなわち，自由な提携構造は，自由な提携のみから構成される．

$$\forall S \in CS,\ |S| = 1 \tag{7}$$

また，A における最適な提携構造 CS^* において，

$$LB_A \leq \Delta(CS^*) \leq UB_A \tag{8}$$

が成り立つとき，LB_A および UB_A を．それぞれ A の提携構造の上界および下界という．このとき，式 (9) を満たす提携 S を “CAS” と呼ぶ．

$$\delta(S) + UB_{A\setminus S} \leq LB_A \tag{9}$$

すなわち CAS を含む提携構造は，最適解とはなり得ないことを意味している．このような CAS を取り除くことで，提携構造の求解を高速化することが可能になる．従来手法において，分枝限定法を用いることで，このような CAS を効果的に除外しつつ，求解することが可能である．しかしながら，従来手法では，上界および下界の計算において，頑健性を考慮しておらず，検討の余地がある．本研究では，頑健性を考慮した上界および下界の計算手法を提案する．これにより，RCSG における効率的な求解手法が実現可能になる．すなわち，次式を満たす LB_A^k および UB_A^k を求めるための手法を示す．

$$LB_A^k \leq \Delta(CS^* \setminus A') \leq UB_A^k,\ |A'| \leq k \tag{10}$$

3.2 頑健性を考慮した下界 LB_A^k

エージェント集合 A に関する，k-robust な CSG の下界 LB_A^k は式 (12) のように求めることができる．以降，LB_A^k について説明する．

提携構造の頑健性を考慮しない場合，提携構造には，必ず一つ以上の提携が含まれることから，LB_A の例として次 (11) が考えられる．

$$LB_A = max(\{\delta(S)|\ S \in \mathcal{P}(A)\}) \geq 0 \tag{11}$$

ここで，$\mathcal{P}(A)$ には，自由な提携が含まれていることから，$LB_A \geq 0$ である．

LB_A は，RCSG における下界であることが保証されない．RCSG の解に関して，次式が成り立つ．

$$\Delta(CS^* \setminus A') \leq \Delta(CS^*)$$

ここで，式 (8) および式 (10) より，LB_A および LB_A^k の大小関係を特定することはできない．すなわち，次式が成立しない場合がある．

$$LB_A \leq \Delta(CS^* \setminus A')$$

ここで A' は A から離脱するエージェントの集合である．

提携構造の頑健性を考慮した場合，k-robust な RCSG において，式 (12) が成り立つ．

$$LB_A^k = min(\{LB_{A'}|\ A' \in \mathcal{R}(A,k)\}) \geq 0 \tag{12}$$

ここで，$\mathcal{R}(A,k)$ は，A から任意のエージェントが高々 k 体 $(k > 0)$ だけ離脱する場合の組合せの集合であり，次式で示される．

$$\mathcal{R}(A,k) = \{A \setminus A' | A' = 2^A, |A'| = k\}$$

例えば，A から 1 体のエージェントが離脱する場合について説明する．LB_A は，A の部分集合 A' における最適な提携構造の利得の下界 LB'_A の最小値である．これは，A から高々 k 体までの任意のエージェントが離脱することを考慮した場合においても，同様に成り立つ．

表 1 にエージェント数が 4 体 ($A = \{a_1, a_2, a_3, a_4\}$) の場合の特性関数の例，および形成可能な全ての提携と提携構造をまとめた．例として，式 (12) を利用して，表 1 における LB^1_A を計算する．ここで，$\mathcal{R}(A, 1) = \{\{a_1, a_2, a_3\},\ \{a_1, a_2, a_4\},\ \{a_1, a_3, a_4\},\ \{a_2, a_3, a_4\}\} = \{A'_1, A'_2, A'_3, A'_4\}$ である．まず，A'_1 の下界 $LB_{A'_1}$ を求める．式 (11) から，$\{a_1, a_2, a_3\}$ で形成可能な全ての提携 $\{\{a_1\}, \{a_2\}, \{a_3\}, \{a_1, a_2\}, \{a_1, a_3\}, \{a_2, a_3\}, \{a_1, a_2, a_3\}\}$ の内, $\delta(S)$ の最大値が $LB_{A'_1}$ となる．したがって, $LB_{A'_1} = 2$ となる．同様に計算を行うことで，$LB_{A'_2} = 2$，$LB_{A'_3} = 2$，$LB_{A'_4} = 2$ となる．式 (12) より, $LB_{A'_1}, LB_{A'_2}, LB_{A'_3}, LB_{A'_4}$ の内，最小値をとることから，$LB^1_A = 2$ と計算することができる．$LB^1_A = 2$ は，1-robust な提携構造の社会的余剰 $\Delta \geq 2$ となることを示している．

3.3 頑健性を考慮した上界 UB^k_A

エージェント集合 A の CSG の上界 UB_A は，そのまま k-robust な RCSG の上界 UB^k_A とすることが可能である．なぜならば，上界に関して，

$$\Delta(CS^* \setminus A') \leq \Delta(CS^*) \leq UB_A$$

が成り立つからである．よって，

$$UB^k_A = UB_A$$

とする[*1]．

ここでは，提携構造 CS の上界 UB_A を求めるために，サイズ k の提携 S の $\delta(S)$ の最大値 δ^k_{max} を求める．ここで，

$$CS = \{S_1, S_2, \ldots, S_n\} \cup A'$$
$$A' \subset A,\ A' \cap S_i = \emptyset, |S_i| \geq 2\ (1 \leq i \leq n)$$

のとき，すなわち，CS において $S_1 \sim S_n$ が自由な提携ではなく，それ以外のエージェント集合 A' に関する提携構造が未確定な場合，

$$UB_A = \sum_{i=1 \ldots n} \delta(S_i) + \delta^{|A'|}_{max}$$

と置くことができる．ここでは，

$$\begin{aligned}
\delta^1_{max} &= 0 \\
\delta^2_{max} &= \eta_A(2) \\
\delta^k_{max} &= max\left(\eta_A(k), \frac{k}{k-1} delta^{k-1}_{max}\right), k > 2 \\
\eta_A(k) &= max\left(\{\delta(S) |\ S \in \mathcal{P}(A), |S| = k\}\right)
\end{aligned}$$

とした．

例として，表 1 における上界 UB_A を計算するために必要な $\delta^1_{max}, \delta^2_{max}, \delta^3_{max}, \delta^4_{max}$ を計算する．δ^1_{max} は定義より，$\delta^1_{max} = 0$ である．$\delta^2_{max} = \eta_A(2)$ より，エージェント数が 2 体である提携 $\{\{a_1, a_2\}, \{a_1, a_3\}, \{a_1, a_4\}, \{a_2, a_3\}, \{a_2, a_4\}, \{a_3, a_4\}\}$ 内で，$\delta(S)$ の最大値を計算すれば良い．この場合，最大値は $\delta(\{a_1, a_2\}) = 2$ または $\delta(\{a_3, a_4\}) = 2$ である．したがって，$\delta^2_{max} = \eta_A(2) = 2$ である．

次に，δ^3_{max} を計算する．$\delta^3_{max} = max\left(\eta_A(3), \frac{3}{2}\delta^2_{max}\right)$ である．$\eta_A(3)$ は，エージェント数が 3 体である提携 $\{\{a_1, a_2, a_3\}, \{a_1, a_2, a_4\}, \{a_1, a_3, a_4\}, \{a_2, a_3, a_4\}$ 内で，$\delta(S)$ の最大値を計算すれば良い．この場合，最大値は $\delta(\{a_1, a_3, a_4\}) = 2$ であることから，$\eta_A(3) = 2$ である．ここで，$\frac{3}{2}\delta^2_{max} = 3$ となるので，$\delta^3_{max} = 3$ である．

最後に，δ^4_{max} を計算する．$\delta^4_{max} = max\left(\eta_A(4), \frac{4}{3}\delta^3_{max}\right)$ である．$\eta_A(4) = \delta(\{a_1, a_2, a_3, a_4\})$ であるから，$\eta_A(4) = 6$ である．また，$\frac{4}{3}\delta^3_{max} = 4$ となるので，$\delta^4_{max} = 6$ である．$\delta^1_{max}, \delta^2_{max}, \delta^3_{max}, \delta^4_{max}$ を計算することで，上界 UB_A の見積もりが可能となる．

4. 提携束（Coalition Lattice）

RCSG のためのデータ構造として “提携束（Coalition Lattice）” が，提案されている [3]．図 2 のように 1-robust な提携構造の判定を行う場合，各提携の包含関係を明確にする必要がある．提携束とは，特性関数内に記述されている各提携をノードとし，包含関係のある提携をアークによって結合したデータ構造である．提携束を利用することで，“提携の頑健性” を効率的に計算することができる．

4.1 提携の頑健性

先行研究では，提携の頑健性を，ある提携 S から任意のエージェントの離脱が発生しても，残りのエージェントで構成される S の部分集合である提携 $S' \subset S$ が負の提携ではない性質と定義した [3]．RCSG では，エージェントの集合 A において，ある提携構造 CS から任意の k 体のエージェントが離脱した場合，残りのエージェントで形成される全ての提携構造 CS' のどれと比較しても，既存の提携構造の利得が最大のままであるような CS を k-robust な提携構造としている．CS が k-robust であるためには，CS から任意の k 体のエージェントが離脱する場合を考慮する必要がある．すなわち，CS に含まれる各 S から高々 k 体の

[*1] 当然，$UB^k_A < UB_A$ なるような UB^k_A を効率的に求めることが可能ならば，そのような UB^k_A を利用した方がよい．

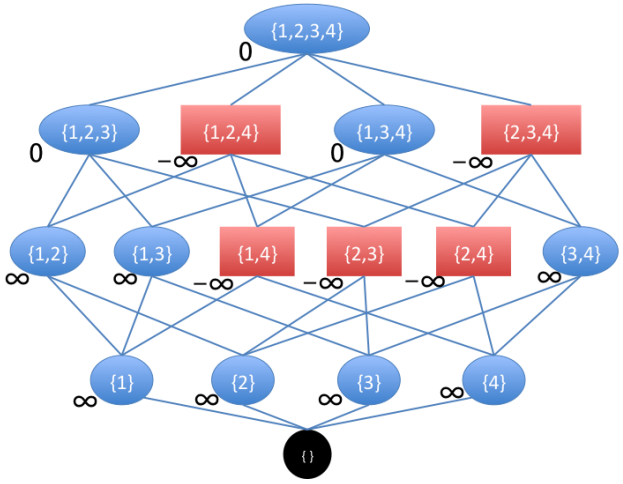

図 3 表 1 における提携束
Fig. 3 The coalition lattice in case of Table 1.

エージェントが離脱する場合を想定する．S から高々 k 体のエージェントの集合 A' が離脱した場合，$S \setminus A'$ の利得を考慮することが重要である．

本稿における頑健な提携とは，S から任意のエージェントの離脱が発生しても，残りのエージェントで構成される S' が CAS ではないことと定義する．先行研究では，自明な下界である $LB_A = 0$ を基準として，提携の頑健性を定義したともいえる．本研究では，下界として $LB_A > 0$ となるような，LB_A を用いることで，ロバストな提携構造形成問題を効率的に解くことを目指す．

ある正の提携 S_p を考える．S_p を含む提携構造 CS が k-robust であるためには，CS から任意の k 体のエージェントが離脱しても，既存の提携構造の利得が最大のままである必要がある．そのため，CS の構成要素の一つである S_p から高々 k 体のエージェントが離脱する場合があり，残りのエージェントで構成される $S' \subset S_p$ は CAS であってはならない．S' が CAS の場合，CS が最適な提携構造とはなり得ないからである．S_p の部分集合に CAS である提携 $S_n \subset S_p$ が存在し，かつ全ての S_n の中で，$|S_n|$ が最大となるものを S_n^* とした場合，S_p は式 (13) を満たすような”k-L-robust”であるとする．ある提携 S が k-L-robust であるとは，S から任意の k 体のエージェントが離脱しても，残りのエージェントで構成される提携 $S' \subset S$ は CAS ではないことを指す．

$$k = |S_p| - |S_n^*| - 1 \tag{13}$$

4.2 データ構造

図 3 は，表 1 の特性関数から生成される提携束の図である．提携束は，上限を全体提携，下限を空集合としている．提携束は，$|S|$ で階層分けされた提携 S により，各提携の包含関係を表現する．図 1 の丸いノードは正の提携を表し，四角いノードは負の提携を表す．各ノードの左下の数値は，提携の頑健性を表す数値化である．ただし，特性関数内には空集合は存在しないため，下限ノードには頑健性の値を記述していない．図 1 のような提携束を構築することで，各提携の包含関係から，各提携の頑健性を容易に計算することが可能になる．

Algorithm 1 RCSG to CSG

Require: $k \geq 0$
1: $CL_1 \leftarrow \{\langle i, \{a_i\}, +\infty \rangle | a_i \in A\}$
2: $CL_{2\ldots|A|} \leftarrow \{\}$
3: **for** $i = |A| + 1$ to $2^{|A|} - 1$ **do**
4: $S \leftarrow GetS(i)$
5: $p \leftarrow false$
6: **if** S is not CAS **then**
7: $l \leftarrow Calculate(S)$
8: **if** $l \geq k$ **then**
9: $CL_{|S|} \leftarrow CL_{|S|} \cup \{\langle i, S, l \rangle\}$
10: $p \leftarrow true$
11: **end if**
12: **end if**
13: **if** $p = false$ **then**
14: $CL_{|S|} \leftarrow CL_{|S|} \cup \{\langle i, S, -\infty \rangle\}$
15: **end if**
16: **end for**
17: $CSG \leftarrow \{\langle S, \delta(S) \rangle \mid \langle i, S, l \rangle \in CL_{1\ldots|A|}\}$

5. 問題変換アルゴリズム

提携束上での各提携の頑健性，つまり k-L-robust な提携の k の値を決定する方法を示す．最適な提携構造 CS 内にエージェント a で形成される自由な提携が存在する場合を考える．自由な提携 $\{a\}$ から自由なエージェント a が離脱する場合，CS 内のその他の提携には全く影響がないため，$\{a\}$ を除いた既存の提携構造の社会的余剰は最大のままである．自由なエージェントの離脱によって，自由なエージェントを除く既存の提携構造の社会的余剰は最大の状態を保つことから，自由な提携の k の値は，$k = \infty$ とする．また，CAS は CS 内に存在するべきではないことから，CAS の k の値は，$k = -\infty$ とする．CL 上において，ある正の提携 S_p の部分集合である子ノードに，CAS の提携 S_n が存在する場合，そのノードを S_n^* として，式 (13) を用いて，k の値を決定する．S_p の部分集合である子ノードに CAS が存在しない場合を考える．子ノードの中で，最も小さい k の値を k_{min}，その k_{min}-L-robust な提携を S_{min} とする．S_{min} と比較して，S_p はエージェントが $M = |S_p| - |S_{min}|$ 体だけ多いことから，S_{min} よりも M 体だけ多くエージェントが離脱しても CAS にはならない．したがって，S_p の子ノードに CAS が存在しない場合，k の値は，$k = k_{min} + M$ によって決定できる．

5.1 アルゴリズム

問題変換アルゴリズムを Algorithm1 に示す．Algo-

rithm1 において，CL_i，A，S，l はそれぞれ，エージェント数が i である提携で形成された提携束，エージェント集合，提携，提携の頑健性の数値を表している．A，特性関数 v，求める k-robust な提携構造の k の値を入力することで，Algorithm1 を適用すると，頑健性を考慮した提携の集合が出力される．すなわち，RCSG を入力することによって，Algorithm1 を適用することで，CSG に変換することが可能となる．提携束は，$\langle i, S, l\rangle$ の集合で表現している．A において，$\mathcal{P}(A)$ は提携内のエージェント数で昇順ソートされ，それぞれ全提携にインデックス i が対応付けされているとする．

Algorithm1 の 1,2 行目では，CL の初期化を行っている．1 行目では，自由な提携の提携束を形成している．CL には，自由な提携の頑健性を $+\infty$ として追加する．2 行目では，2 から $|A|$ 体までの提携の提携束の初期化をしている．

Algorithm1 の 3-16 行目では，エージェントが 2 体以上の提携に関して，エージェント数が少ない提携からボトムアップに提携束を形成している．エージェント数が $|A|$ の場合，形成可能な提携の総数は $2^{|A|}-1$(空集合を除く) であることから，3 行目では $i = |A|+1$ から $2^{|A|}-1$ までとしている．4 行目の関数 $GetS(i)$ は，i に対応した S を返す関数である．

全ての提携に関して，CAS であるか否かを判別し，それぞれを CL に追加する．6 行目の S is not CAS とは，関数 $GetS(i)$ によって得られた S が CAS ではない場合に $true$ を返し，それ以外であるならば $false$ を返す関数である．S が CAS ではない場合，7 行目の関数 $Calculate(S)$ によって，S の子ノードから，S の頑健性 l を計算する．l を計算するための関数 $Calculate(S)$ については，Algorithm2 に記述する．8 行目で，その提携が k-robust な提携構造に利用可能かどうか，すなわち k 以上の頑健性を持つ提携であるのかを判定している．S が k 以上の頑健性を持つ場合，$CS_{|S|}$ に追加する．13 行目で，S が CAS である，または S が k 以上の頑健性を持っていない場合，14 行目において，S の頑健性を $-\infty$ として，$CL_{|S|}$ に追加する．Algorithm1 の 1-16 行目で，提携内のエージェント数に対応した提携束を形成している．Algorithm1 の 17 行目で，全ての提携束に含まれる提携の集合を返している．

5.2 頑健性の計算

Algorithm2 は，Algorithm1 の 7 行目で使用する関数 $Calculate(S)$ を示している．$Calculate(S)$ は，提携 S を入力することで，S の子ノードを考慮し，S の頑健性 l を計算して返す関数である．3 行目において，l を計算するとき，S の各子ノードが CAS であるのかを調べる．子ノードに一つでも CAS が存在する場合，4 行目で，式 (13) を用いて，l を計算する．子ノードに CAS が存在しない場合，6 行目で，子ノードの頑健性から l を計算する．最終的に，

Algorithm 2 $l = Calculate(S)$ in Algorithm1

Require: $|S| > 1$
Ensure: $l_{min} = Calculate(S)$
1: $l_{min} \leftarrow +\infty$
2: **for all** $\langle S_{child}, l_{child}\rangle \in Child(S)$ **do**
3: **if** S_{child} is CAS **then**
4: $l' \leftarrow |S| - |S_{child}| - 1$
5: **else**
6: $l' \leftarrow l_{child} + |S| - |S_{child}|$
7: **end if**
8: **if** $l_{min} > l'$ **then**
9: $l_{min} \leftarrow l'$
10: **end if**
11: **end for**

計算した頑健性の値が最も小さいものを l として返す．

6. おわりに

提携構造形成（Coalition Structure Generation）問題におけるエージェントの離脱に対する頑健さを考慮したロバストな提携構造形成 (Robust Coalition Structure Generation) 問題は，マルチエージェントシステムにおける重要な問題の一つである．ロバストな提携構造形成問題を素直に解こうとすると，膨大な計算量となり現実的ではない．提携の頑健性を考慮すると，ロバストな提携構造形成問題を，頑健性を考慮する必要の無い通常の提携構造形成問題に変換可能である．提携構造形成問題への変換において，提携の頑健さを見積もる手法が必要である．

本稿では，提携の頑健さを見積もるために，エージェントの離脱に対する頑健さを考慮した最適解の上界および下界を見積もるための手法を示した．頑健さを考慮した上界に関しては，提携構造形成問題における上界を利用可能であることを示した．頑健さを考慮した下界に関しては，新たに計算方法を示した．

本手法により，ロバストな提携構造形成問題をよりコンパクトな提携構造形成問題に変換可能になり，効率的な求解が可能になると予想される．今後は，開発した手法を実装し，本手法の性能を定量的に評価する予定である．

参考文献

[1] T. Sandholm, K. Larson, M. Andersson, O. Shehory, and F. Tohmé, “Coalition structure generation with worst case guarantees” Artificial Intelligence, 111(1-2): 209-238, 1999.
[2] 沖本天太，シュウィンドニコラ，井上克巳，“ロバストな提携構造形成問題に関する一検討，”JSAI2014, 3A3-3, 2014.
[3] K. Nakano, S. Shiramatsu, T. Ozono, and T. Shintani, “Coalition Lattice: A Data Structure considering Robustness for Robust Coalition Structure Generation Problem,” ICIAE2015 (to appear), 2015.
[4] 沖本天太，シュウィンドニコラ，クレモンマキシム，井上克巳，“ロバストなチーム編成問題，” The Proceedings of JAWS, pp.341-344, 2014.