特別研究報告書

混合相補性問題に対する分枝限定法と双行列ゲームへの適用

指導教員

福嶋 雅夫　教授
山下 信雄　助教授

京都大学工学部情報学科
数理工学コース
平成14年4月入学
平成18年3月卒業

高木　潤

平成18年1月31日提出

混合相補性問題に対する分枝限定法と双行列ゲームへの適用

高木　潤

摘要

混合相補性問題は非線形方程式や数理計画問題，種々の均衡問題を含む幅広いクラスの問題であり，オペレーションズ・リサーチの重要な問題の 1 つである．これまでに混合相補性問題に対して様々なアルゴリズムが提案されてきた．それらのアルゴリズムのほとんどは，扱う関数にある種の単調性が成り立つときに解を求めることができる．しかし，双行列ゲームなど現実の問題では，単調性が成り立たない場合が多い．どのような関数でも解を求めることができるアルゴリズムはほとんど提案されていない．また混合相補性問題の解をすべて列挙する効率のよいアルゴリズムはない．オペレーションズ・リサーチにおいては，解を求めるだけではなく，解の性質を分析したり，ある特定の解を必要とする場合が多い．そのため，解が唯一であることが保証されない場合でも，すべての解を列挙することができるアルゴリズムを構築することは重要である．

本報告書では，扱う関数が線形である線形混合相補性問題において解を全列挙するアルゴリズムを提案する．線形混合相補性問題の解集合は複数の凸多面体集合の和集合として表される．この凸多面体集合を列挙する方法として分枝限定法に基づいたアルゴリズムがある．しかし，凸多面体の候補となるものは問題のサイズの指数オーダ存在するため，単純に列挙する手法では大規模な問題に適用することができない．そこで，次の 2 つの工夫を提案する．1 つ目は分枝選択の工夫であり，2 つ目は凸緩和のアイディアに基づいた分枝の限定操作の工夫である．次に双行列ゲームの Nash 均衡解集合を求める問題への応用を考える．双行列ゲーム固有の性質を利用した提案アルゴリズムの実装方法を説明する．双行列のパラメータを乱数で生成した様々なサイズの問題に対して数値実験を行い，提案したアルゴリズムの有効性を調べた．双行列ゲームの均衡解を列挙する既存の手法に比べて，分枝の数が 1 割から 2 割程度改善ができることがわかった．特に 1 人のプレイヤーの戦略の数が，他のプレイヤーの戦略の数よりも少ないときには改善の幅が大きかった．

目 次

1 序論

混合相補性問題 (Mixed Complementarity Problem, MCP) とは，$l_i \in [-\infty, +\infty)$, $u_i \in (-\infty, +\infty]$, $l_i < u_i, i \in \mathcal{N} := \{1, \cdots, n\}$ とベクトル値写像 $F : \Re^n \to \Re^n$ が与えられたとき，

$$\begin{cases} x_i = l_i \Rightarrow F_i(x) \geq 0, \\ l_i < x_i < u_i \Rightarrow F_i(x) = 0, \\ x_i = u_i \Rightarrow F_i(x) \leq 0, \end{cases} \tag{1.1}$$

を満たすベクトル $x \in [l, u]$ を求める問題である．ここで，$l = (l_1, \ldots, l_n)^T$, $u = (u_1, \ldots, u_n)^T$ である．MCP は数理計画問題，種々の均衡問題，非線形方程式など幅広い応用分野を持つ [6]．そのため，現在まで MCP に関する様々な研究がなされている [5]．本報告書では線形混合相補性問題 (Linear Mixed Complementarity Problem, LMCP)，つまり写像 F が $F(x) := Mx + q$, $M \in \Re^{n \times n}$, $q \in \Re^n$ で与えられる場合を考える．さらに，表記を簡単にするため，本報告書では次の LMCP を扱うことにする．

$$\begin{cases} \text{Find} \quad x \in \Re^n, \\ \text{s.t.} \quad F_i(x) = 0, \ i \in \mathcal{N}_1, \\ \qquad x_i \geq 0, \ F_i(x) \geq 0, \ x_i F_i(x) = 0, \ i \in \mathcal{N}_2. \end{cases} \tag{1.2}$$

ここで，$\mathcal{N}_1 \cup \mathcal{N}_2 = \mathcal{N}$, $\mathcal{N}_1 \cap \mathcal{N}_2 = \emptyset$ である．問題 (1.1) と問題 (1.2) の等価性は付録で示す．本報告書の目的は行列 M に何も仮定しないときに，LMCP(1.2) のすべての解を効率よく列挙する手法を提案することである．

これまでに MCP の解法として様々なアルゴリズムが提案されている．最もよく使われるアプローチは，MCP を等価な制約無し最小化問題に定式化し，最小化問題に対する既存のアルゴリズムを用いて解を求めるものである．しかし，既存のアルゴリズムの多くは停留点を求めるものであり，大域的最適解を求めるものではない．停留点が大域的最適解となるためには，つまり MCP の解を求めるためには，F が単調になる (M が半正定値になる) など，何らかの仮定が必要となる．関数 F に何も仮定しなくても MCP の解を求めることができる手法として，ホモトピー法に基づいたものがある [3, 15, 16, 17]．しかし，ホモトピーパスを追跡するには，莫大な計算時間を必要とする．さらに，ホモトピー法で解を全列挙することは難しい．Kostreva と Zheng は MCP を整数計画法に再定式化して解く手法を提案している [11]．整数計画問題は問題の規模が大きくなると，取り扱いが難しくなり，その結果，効率よく列挙することができない．

Al-Khayyal[1] は LMCP の解を求めるアルゴリズムを提案している．このアルゴリズムは，列挙木を用いることによって解を求めることを保証しつつ，局所的探査を組み合わせることによって，効率よく 1 つの解を求めることに成功している．しかし，LMCP の解を全列挙すると，莫大な計算時間を必要とする．双行列ゲームの均衡解を求める問題は LMCP として定式化できることが知られている [9]．Audet ら [2] は分枝限定法を用いて双行列ゲームの均衡解をすべて列挙するアルゴリズムを提案している．このアルゴリズムは双行列ゲームの特性を利用することによって，効率よく解を全列挙している．本報告書で提案するアルゴリズムはこの Audet らの提案するアルゴリズム EEE(Enumeration of all Extreme Equilibria) を LMCP に拡張し，さらにいつくかの工夫を加えたものである．

本研究の目的は双行列ゲームに対するアルゴリズム EEE を拡張し，LMCP の解をすべて列挙するアルゴリズムを提案することである．さらに拡張したアルゴリズムを効率化するために，分枝する添字を選ぶ基準の工夫と凸緩和のアイディアに基づいた分枝の限定操作を厳しくする工夫を提案する．さらに双行列ゲームに対する具体的な実装方法を提案し，数値実験を通してその実用性を考察する．

本報告書の以降の構成は次のようになっている．2 節では LMCP の解集合の表現方法を説明し，分枝限定法の考えを利用して，その表現法で表された解集合を求めるアルゴリズムを提案する．また，分枝をできるだけカットするための 2 つの工夫を提案する．3 節では双行列ゲームの均衡点を求める問題を LMCP として定式化し，2 節で提案したアルゴリズムを適用方法の説明をする．特に，双行列ゲームの性質を利用して，分枝の限定操作に関する工夫の実装方法を示す．4 節では数値実験の結果を示す．5 節は本報告書の結論を述べる．

本報告書では以下の表記を用いる．ベクトルはすべて縦ベクトルとして表すものとする．$m \times n$ 行列 A に対して，$A_{i.},\ i \in \{1,\ldots,m\}$ は行列 A の第 i 行を表し，$A_{.j},\ j \in \{1,\ldots,n\}$ は行列 A の第 j 列を表すものとする．n 次元ベクトル q に対して，q_i はベクトル q の第 i 成分を表すものとする．ベクトル $(x^T, y^T, \alpha, \beta)^T \in \Re^{m+n+2}$ を (x, y, α, β) で表す．

2　混合相補性問題に対する分枝限定法

この節では LMCP:

$$\left\{\begin{array}{ll} \text{Find} & x \in \Re^n, \\ \text{s.t.} & M_{i.}x + q_i = 0,\ i \in \mathcal{N}_1, \\ & x_i \geq 0,\ M_{i.}x + q_i \geq 0,\ x_i(M_{i.}x + q_i) = 0\ i \in \mathcal{N}_2, \end{array}\right. \tag{2.1}$$

のすべての解を列挙する手法を説明する．ここで，M, q はそれぞれ $M \in \Re^{n\times n},\ q \in \Re^n$ となる行列とベクトルである．

2.1　解集合の表現

LMCP の解は一般には無限個存在するため，すべてを列挙することは不可能である．そこで解集合を有限個の凸多面体の和集合で表すことを考える．I, J を添字集合 $\mathcal{N}_2$ の部分集合とする．この添字集合の組 (I, J) を用いて集合 $S(I, J) \subseteq \Re^n$ を

$$S(I,J) := \left\{ x \in \Re^n \;\middle|\; \begin{array}{l} M_{i.}x + q_i = 0,\ i \in \mathcal{N}_1 \\ M_{j.}x + q_j = 0,\ j \in J \\ M_{j.}x + q_j \geq 0,\ j \in \mathcal{N}_2 \setminus J \\ x_i = 0,\ i \in I \\ x_i \geq 0\ i \in \mathcal{N}_2 \setminus I \end{array} \right\}$$

と定義する．$S(I, J)$ は凸多面集合である．さらに次の集合 Φ_0, Φ_S を定義する．

$$\begin{aligned} \Phi_0 &:= \{(I,J) \mid I \cup J = \mathcal{N}_2,\ I \cap J = \emptyset\}, \\ \Phi_S &:= \{(I,J) \in \Phi_0 \mid S(I,J) \neq \emptyset\}. \end{aligned}$$

$(I, J) \in \Phi_S$ であれば，$I \cup J = \mathcal{N}_2$ であることと $S(I, J)$ の定義より，$x \in S(I, J)$ は LMCP(2.1) の解である．逆に LMCP の任意の解 x に対して，$x \in S(I, J)$ となる $(I, J) \in \Phi_S$ が存在する．したがって，LMCP(2.1) の解集合 E は以下のように表すことができる，

$$E = \bigcup_{(I,J)\in\Phi_S} S(I,J).$$

つまり LMCP(2.1) の解集合は，図 1 のように有限個の凸多面体の和集合として表すことができる．

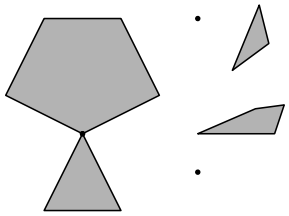

図 1: LMCP の解集合のイメージ

以下では解集合 E を表現する Φ_S の要素をすべて列挙するアルゴリズムを提案する．

2.2 解の列挙法

Al-Khayyal[1] のアイディアに基づいて，Φ_S の要素の列挙は列挙木によっておこなう．図 2 は $|\mathcal{N}_2| = 2$ の場合の列挙木の例である．列挙木の各ノードには添字集合の組 (I, J) が対応する．列挙木の初期ノードす

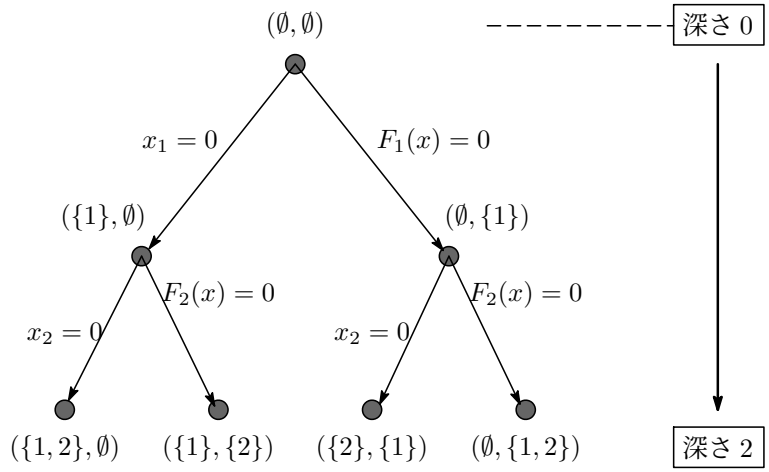

図 2: $|\mathcal{N}_2| = 2$ の場合の列挙木の例

なわち根には $(\emptyset, \emptyset)$ を与える．根に与える集合 $S(\emptyset, \emptyset)$ は LMCP(2.1) から相補性条件を取り除いたものであり，

$$S(\emptyset,\emptyset) := \left\{ x \in \Re^n \;\middle|\; \begin{array}{c} M_{i\cdot}x + q_i = 0,\ i \in \mathcal{N}_1 \\ M_{i\cdot}x + q_i \geq 0,\ i \in \mathcal{N}_2 \\ x_i \geq 0,\ i \in \mathcal{N}_2 \end{array} \right\}$$

である．各ノードの分枝は以下のように与えられる．

分枝操作　現在のノード (I, J) において添字集合 $\mathcal{N}_2 \setminus (I \cap J)$ から添字を 1 つ選び (その添字を $\tilde{\imath}$ とする)．次の 2 つのノードに分枝する (図 3)．

1. 等式制約 $x_{\tilde{\imath}} = 0$ を加えたノード $(I \cup \{\tilde{\imath}\}, J)$,
2. 等式制約 $M_{\tilde{\imath}\cdot}x + q_{\tilde{\imath}} = 0$ を加えたノード $(I, J \cup \{\tilde{\imath}\})$.

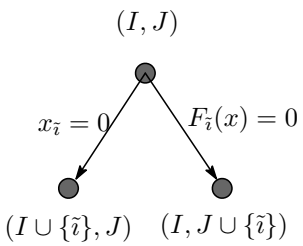

図 3: 分枝の様子

この分枝を深さが $|\mathcal{N}_2|$ になるまですべてのノードに対しておこなう．

列挙木において，深さ $|\mathcal{N}_2|$ のノードに対応した (I, J) の集合が Φ_0 に対応する．$\Phi_S \subseteq \Phi_0$ であるから，列挙木によって Φ_S の要素がすべて列挙できることがわかる．

2.3 列挙の高速化

列挙木のノードは $2^{|\mathcal{N}_2|} - 1$ 個存在する．したがって，ノード数は $|\mathcal{N}_2|$ に対して指数的に増加してしまう．そこで，列挙するノード数を減らすために分枝操作を限定することを考える．

分枝カット 列挙木において，現在のノードに対応する添字集合の組を (I, J) とする．現在のノードにおいて $S(I, J) = \emptyset$ であれば，ノード (I, J) の子孫のノード (I', J') でも $S(I', J') = \emptyset$ である．つまり，このノードの子孫には Φ_S の要素となるものは存在しない．

したがって，$S(I, J)$ が空集合でなく，$I \neq \mathcal{N}_2$ または $J \neq \mathcal{N}_2$ であれば現在のノードを分枝し，さもなければそのノードをカット，すなわちそれ以上の分枝操作を行わないようにする．

添字の選び方 次に分枝する添字の選び方を考える．分枝操作に関して，列挙木の根に近いところで分枝カットできるほど，生成されるノード数は少なくなる．そこで，分枝する添字 $\tilde{\imath}$ を効果的に選ぶために，相補積 $x^T(Mx + q)$ を線形近似した目的関数を持つ以下の線形計画問題 $R(I, J)$ を考える．

$$R(I, J): \begin{cases} \min & x^T(M\hat{x} + q), \\ \text{s.t.} & x \in S(I, J). \end{cases}$$

ここで $\hat{x} \in \Re^n$ は現在のノードの親に対応する副問題で得られた解とする．現在のノードにおける制約領域 $S(I, J)$ が空でなければ，シンプレックス法によって副問題 $R(I, J)$ の解 $x^k \in S(I, J)$ を求めることができる．この x^k は LMCP(2.1) の近似解となる．近似解 x^k で最も相補性条件が満たされていないような添字 $\tilde{\imath}$ において分枝することを考える．すなわち，

$$\tilde{\imath} = \arg \max_{i \in \mathcal{N}_2 \setminus (I \cup J)} \{x_i^k (M_{i\cdot} x^k + q_i)\}$$

を分枝する添字に選ぶ．最も相補積が大きい添字 $\tilde{\imath}$ では，等式 $x_{\tilde{\imath}} = 0$ または $M_{\tilde{\imath}\cdot} x + q_{\tilde{\imath}} = 0$ が成り立ちにくいと考えられる．そのため，$\tilde{\imath}$ で分枝したときの子ノードが実行不可能となりやすいからである．

なお，以下で示すように，任意の副問題 $R(I, J)$ は $S(I, J) \neq \emptyset$ ならば必ず解を持つ．$\hat{x}$ が親ノードの副問題において実行可能であることより，

$$M_{i\cdot} \hat{x} + q_i \geq 0, \ i \in \mathcal{N}_2,$$

が成り立つ．さらに，制約領域 $S(I, J)$ の定義より，

$$x_i \geq 0, \ i \in \mathcal{N}_2$$

であるので，$R(I,J)$ の目的関数は $x^T(M\hat{x}+q) \geq 0$ となり下に有界となる．そのため，$S(I,J) \neq \emptyset$ であれば，$R(I,J)$ は解を持つ．

以上のことをまとめると，アルゴリズムは以下のように記述できる．

アルゴリズム 1: LMCP(2.1) の解集合 E を表現する Φ_S の要素をすべて列挙する手法

STEP1:初期化

列挙木 T に初期ノード $(\emptyset,\emptyset)$ を与える．
もし $S(\emptyset,\emptyset)=\emptyset$ であれば，解集合 $E=\emptyset$ として終了．
さもなければ，初期ノードの親の近似解を $S(\emptyset,\emptyset)$ の任意の実行可能解として STEP2 へ．

STEP2:ノードの選択

T が空であれば終了．
さもなければ，ノード (I,J) を T から深さ優先探索で選び T から削除する．

STEP3:実行可能性のチェック

もし，$S(I,J)=\emptyset$ であれば STEP2 へ戻る．
$S(I,J)\neq\emptyset$ であり現在のノードの深さが $|\mathcal{N}_2|$ であれば (I,J) を解として記録し STEP2 へ戻る．
いずれでもなければ，副問題 $R(I,J)$ の解 x^k(LMCP の近似解) を求め，STEP4 へ．

STEP4:分枝操作

現在のノードの近似解 x^k に対して，

$$\tau_i := \begin{cases} x_i^k(M_{i.}x^k+q_i), & i \in \mathcal{N}_2 \setminus (I \cup J) \\ -1, & i \in (I \cup J) \end{cases} \tag{2.2}$$

を計算する．$\tau_i,\ i\in\mathcal{N}_2$ を最大とする添字 $\tilde{\imath}$ を求める．
そして，ノード $(I\cup\{\tilde{\imath}\},J)$ とノード $(I,J\cup\{\tilde{\imath}\})$ を列挙木 T に加える．STEP2 へ戻る．

このアルゴリズム 1 は Audet らが提案した双行列ゲームに対するアルゴリズムの LMCP への自然な拡張となっている．次にこのアルゴリズム 1 をより高速にする 2 つの工夫を提案する．

分枝選択の工夫 現在のノードにおける LMCP の近似解 x^k がもし LMCP の解となっている場合には，式 (2.2) の相補積 τ_i はすべての $i\in\mathcal{N}_2\setminus(I\cup J)$ に対してゼロとなる．この場合には上記の添字の選び方が機能しなくなる．そこで，x^k が LMCP の解となっている場合には

$$\tau_i = \max\{x_i^k,\ M_{i.}x^k+q_i\},\ i\in\mathcal{N}_2\setminus(I\cup J)$$

が最大値となる添字を $\tilde{\imath}$ とする．このようにする理由は，τ_i が大きい i においては，x_i または $M_{i.}x+q_i$ が 0 になりにくく，そのため，$x_i=0$ または $M_{i.}x+q_i=0$ と固定した子ノードがカットされやすくなるからである．

凸緩和のアイディアに基づいた分枝の限定操作 アルゴリズムの基本的な動作は，相補性問題の相補性条件を除いた領域 $S(\emptyset,\emptyset)$ から始め，1 つの不等式を等式に変換しながら 2 つのノードに分枝することである．それぞれのノードで集合 $S(I,J)$ が空かどうかをチェックし，$S(I,J)\neq\emptyset$ であれば現在のノードをさらに分枝し，$S(I,J)=\emptyset$ であればノードをカットする．ここでは各ノードで以下の仮定が成り立つとき，各ノードにおける実行可能性のチェックをさらに厳しくする工夫を提案する．

仮定 1: 現在のノードに対応する集合 $S(I,J)$ において，添字 $i \in \mathcal{N}_2 \setminus (I \cup J)$ に対して

$$0 \leq x_i \leq x_i^{\max},\ 0 \leq M_{i\cdot}x + q_i \leq F_i^{\max},\ i \in \mathcal{N}_2 \setminus (I \cup J) \tag{2.3}$$

となる上限 $x_i^{\max}, F_i^{\max}$ が存在する．

任意のある添字 $i \in \mathcal{N}_2 \setminus (I \cup J)$ に対して，相補性条件

$$\begin{cases} 0 \leq x_i \leq x_i^{\max}, \\ 0 \leq M_{i\cdot}x + q_i \leq F_i^{\max}, \\ x_i(M_{i\cdot}x + q_i) = 0 \end{cases}$$

を満たす領域は図 4(a) のようになる．

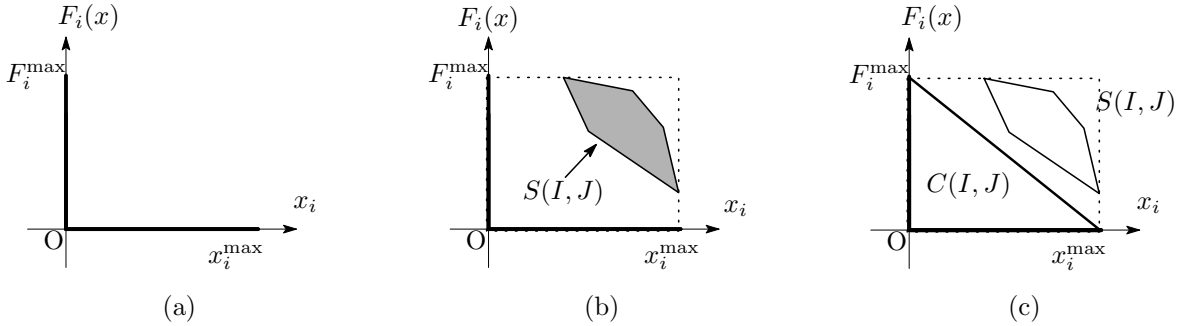

図 4: 工夫の効果のイメージ

アルゴリズム 1 では集合 $S(I,J)$ が空集合かどうかで実行可能性のチェックを判定していた．しかし，図 4(b) のような領域に $S(I,J)$ がある場合，$S(I,J) \neq \emptyset$ であるが，すでに解となる領域を含んでいない．しかし，$S(I,J) \neq \emptyset$ であるので，分枝操作がおこなわれてしまう．このような場合でも，相補性条件を満たす領域 (図 4(a)) と $S(I,J)$ が共通部分がないことをチェックできれば，分枝カットできる．しかし，相補性条件を満たす領域は凸でないために，共通部分の有無を調べることは難しい．そこで，図 4(a) の領域を凸緩和した制約領域 $C(I,J)$ を

$$C(I,J) := \{x \in \Re^n \mid \frac{x_i}{x_i^{\max}} + \frac{M_{i\cdot}x + q_i}{F_i^{\max}} \leq 1,\ i \in \mathcal{N}_2 \setminus (I \cup J)\}$$

を考える (図 4(c))．この $C(I,J)$ を用いて，副問題 $R_c(I,J)$ を次のように定義する．

$$R_c(I,J): \left\{ \begin{array}{ll} \min & x^T(M\hat{x} + q) \\ \text{s.t.} & x \in S(I,J) \cap C(I,J) \end{array} \right.$$

制約領域 $C(I,J)$ は線形不等式で表されるから，副問題 $R_c(I,J)$ も線形計画問題である．元の問題 $R(I,J)$ と比べて，より厳しい制約 $S(I,J) \cap C(I,J)$ のもとで実行可能性を判定することになる．図 4(c) では，$S(I,J) \cap C(I,J) = \emptyset$ であるので，このノードをカットできることがわかる．したがって，分枝カットされるノード数がさらに増えることが期待できる．

上限 $x_i^{\max}, F_i^{\max},\ i \in \mathcal{N}_2 \setminus (I \cup J)$ は次の最大化問題

$$\left.\begin{array}{ll} \max & x_i \\ \text{s.t.} & x \in S(I,J) \end{array}\right\} (i \in \mathcal{N}_2 \setminus (I \cup J))$$

$$\left.\begin{array}{ll} \max & M_{i\cdot}x + q_i \\ \text{s.t.} & x \in S(I,J) \end{array}\right\} (i \in \mathcal{N}_2 \setminus (I \cup J))$$

を解くことによって求めることができる．しかし，この単純な方法では各ノード (I,J) に対して $2|\mathcal{N}_2\setminus(I\cup J))|$ 回，すなわちまだ選ばれていない添字の数の 2 倍の回数も線形計画問題を解かなければならない．一方，2 節で説明するように，双行列ゲームにおいては上限の近似をうまく求めることができる．以上のアイディアをまとめたアルゴリズム 2 を以下に記述する．

アルゴリズム 2: 2 つの工夫を加えたアルゴリズム 1

STEP1:初期化

列挙木 T に初期ノード $(\emptyset,\emptyset)$ を与える．
もし $S(\emptyset,\emptyset)=\emptyset$ であれば，解集合 $E=\emptyset$ として終了．
さもなければ，初期ノードの親の近似解を $S(\emptyset,\emptyset)$ の任意の実行可能解として STEP2 へ．

STEP2:ノードの選択

T が空であれば終了．
さもなければ，ノード (I,J) を T から深さ優先探索で選び T から削除する．
$S(I,J)$ が有界でなければ STEP3 へ，有界であれば STEP4 へ．

STEP3:$S(I,J)$ による実行可能性のチェック

もし，$S(I,J)=\emptyset$ であれば STEP2 へ戻る．
$S(I,J)\neq\emptyset$ であり現在のノードの深さが $|\mathcal{N}_2|$ であれば (I,J) を解として記録し STEP2 へ戻る．
いずれでもなければ，副問題 $R(I,J)$ の解 x^k(LMCP の近似解) を計算し STEP5 へ．

STEP4:$S(I,J)\cap C(I,J)$ による実行可能性のチェック

仮定 1 が満たされていれば，$C(I,J)$ を求める．そうでなければ，$C(I,J)=\Re^n$ とする．
もし，$S(I,J)\cap C(I,J)=\emptyset$ であれば STEP2 へ戻る．
$S(I,J)\cap C(I,J)\neq\emptyset$ であり現在のノードの深さが $|\mathcal{N}_2|$ であれば (I,J) を解として記録し STEP2 へ戻る．
いずれでもなければ，副問題 $R_c(I,J)$ を解いて近似解 x^k を計算し STEP5 へ．

STEP5:分枝操作

現在のノードの近似解 x^k に対して，

$$\tau_i:=\begin{cases}x_i^k(M_{i.}x^k+q_i), & i\in\mathcal{N}_2\setminus(I\cup J)\\ -1, & i\in(I\cup J)\end{cases}$$

を計算する．もし，すべての $i\in\mathcal{N}_2\setminus(I\cup J)$ に対して $\tau_i=0$ となっていたら，

$$\tau_i:=\begin{cases}\max(x_i^k,\ M_{i.}x^k+q_i), & i\in\mathcal{N}_2\setminus(I\cup J)\\ -1, & i\in(I\cup J)\end{cases}$$

を計算する．$\tau_i,\ i\in\mathcal{N}_2$ を最大とする添字 $\tilde{\imath}$ を求める．そして，ノード $(I\cup\{\tilde{\imath}\},J)$ とノード $(I,J\cup\{\tilde{\imath}\})$ を列挙木 T に加える．STEP2 へ戻る．

3　双行列ゲームへの適用

この節では，双行列ゲームが LMCP に定式化できることを説明する．さらに，定式化された LMCP では仮定 1 が成立することを示し，(2.3) 式の上界値を効率よく求める方法を提案する．

3.1　双行列ゲーム

双行列ゲーム (Bimatrix Game) とは以下のように定式化される非ゼロ和非協力ゲームである．双行列ゲームは 2 人のプレイヤー（プレイヤー 1, プレイヤー 2）で行われるゲームであり，プレイヤー 1 は m 個の戦略を, プレイヤー 2 は n 個の戦略から自分の戦略を選択するものとする．プレイヤー 1 が戦略 i を選択するときに $x_i = 1$，選択しないときに $x_i = 0$ とする決定変数 $x \in \Re^m$ を導入する．ここで x_i が 0 から 1 の間で確率的な値を取る混合戦略を考える．このとき x は $x \geq 0, \sum_{i=1}^{m} x_i = 1$ を満たさなければならない．プレイヤー 2 に対しても同様に混合戦略 $y \in \Re^n$ を選ぶものとする．このときプレイヤー 1，2 の利得の期待値はそれぞれ，$x^T Ay, x^T By$ と表すことができる．ここで $A, B \in \Re^{m \times n}$ はプレイヤー 1，2 の利得行列である．各プレイヤーは自分が得る利得の期待値を最大化する戦略を選ぶ．すなわち，プレイヤー 1,2 は他のプレイヤーの戦略 y, x をパラメータとした以下の最大化問題の解 x^*, y^* を自分の戦略とする．

$$\mathrm{P1}(y): \begin{cases} \max_x & x^T Ay \\ \text{s.t.} & x \geq 0 \\ & e_m^T x = 1, \end{cases} \qquad \mathrm{P2}(x): \begin{cases} \max_y & y^T B^T x \\ \text{s.t.} & y \geq 0 \\ & e_n^T y = 1, \end{cases}$$

ここで，$e_m \in \Re^m,\ e_n \in \Re^n$ はすべての要素が 1 であるベクトルである．

今，混合戦略 (x^*, y^*) が次の関係

$$\begin{aligned} (x^*)^T Ay^* &\geq x^T Ay^* \quad \text{for all } x \geq 0,\ e_m^T x = 1, \\ (x^*)^T By^* &\geq (x^*)^T By \quad \text{for all } y \geq 0,\ e_n^T y = 1, \end{aligned}$$

を満たすとき，双行列ゲームの Nash 均衡 (Nash equilibrium) であるといい，(x^*, y^*) を Nash 均衡解と呼ぶ．ゲーム理論の議論より Nash 均衡解は常に少なくとも 1 つ存在する [14].

次に双行列ゲームの Nash 均衡解を求める問題を LMCP として定式化する．今, 双行列ゲームにおいて混合戦略 (x^*, y^*) が Nash 均衡点であれば，(x^*, y^*) は以下の線形計画問題

$$\mathrm{P1}: \begin{cases} \max_x & x^T Ay^* \\ \text{s.t.} & x \geq 0 \\ & e_m^T x = 1, \end{cases} \qquad \mathrm{P2}: \begin{cases} \max_y & y^T B^T x^* \\ \text{s.t.} & y \geq 0 \\ & e_n^T y = 1, \end{cases}$$

の最適解となっている．P1,P2 の KKT 条件は次のようになる．

$$\mathrm{P1}: \begin{cases} Ay^* - \lambda_1 e_m - \mu^1 = 0, \\ e_m^T x^* = 1, \\ x^* \geq 0,\ \mu^1 \geq 0,\ (x^*)^T \mu^1 = 0, \end{cases} \qquad \mathrm{P2}: \begin{cases} B^T x^* - \lambda_2 e_n - \mu^2 = 0, \\ e_n^T y^* = 1, \\ y^* \geq 0,\ \mu^2 \geq 0,\ (y^*)^T \mu^2 = 0. \end{cases} \tag{3.1}$$

ここで，$\lambda_1 \in \Re, \lambda_2 \in \Re, \mu^1 \in \Re^m, \mu^2 \in \Re^n$ はそれぞれの制約式に対する Lagrange 乗数である．式 (3.1) から μ_1, μ_2 を消去すると，

$$\begin{cases} x^* \geq 0,\ Ay^* - \lambda_1 e_m \geq 0,\ (x^*)^T (Ay^* - \lambda_1 e_m) = 0,\ e_m^T x^* = 1 \\ y^* \geq 0,\ B^T x^* - \lambda_2 e_n \geq 0,\ (y^*)^T (B^T x^* - \lambda_2 e_n) = 0,\ e_n^T y^* = 1 \end{cases} \tag{3.2}$$

を得る．つまり，双行列ゲームの Nash 均衡解を求める問題は LMCP

$$\begin{cases} \text{find} & (x,y,\alpha,\beta)^T \in \Re^{m+n+2}, \\ \text{s.t.} & x \geq 0,\ \alpha e_m - Ay \geq 0,\ \ x^T(\alpha e_m - Ay) = 0,\ \ e_m^T x = 1, \\ & y \geq 0,\ \beta e_n - B^T x \geq 0,\ y^T(\beta e_n - B^T x) = 0,\ e_n^T y = 1, \end{cases} \tag{3.3}$$

と等価である．実際,

$$M := \begin{bmatrix} 0 & -A & e_m & 0 \\ -B^T & 0 & 0 & e_n \\ e_m^T & 0 & 0 & 0 \\ 0 & e_n^T & 0 & 0 \end{bmatrix},\ q := \begin{bmatrix} 0 \\ 0 \\ -1 \\ -1 \end{bmatrix}$$

と定義すれば，問題 (3.3) を LMCP(2.1) の形に書くことができる．M は半正定値行列ではないので，既存の手法で解くことは難しい．以降の議論における表記の簡単のため，添字集合 $\mathcal{M} := \{1,\ldots,m\}$, $\mathcal{N} := \{1,\ldots,n\}$ を定義しておく．

3.2 均衡解集合の表現

今，双行列ゲームの均衡解集合 E は，LMCP(3.3) として表されることからもわかるように，有限個の凸多面体の和集合として表される．さらに，均衡解集合を構成するそれぞれの多面体は有界であることが示されている [13].

ここで添字集合 I_x, I_A を $\mathcal{M} := \{1,\ldots,m\}$ の部分集合，添字集合 J_y, J_B を $\mathcal{N} := \{1,\ldots,n\}$ の部分集合とする．2 節における I には (I_x, J_y) が，J には (I_A, J_B) が対応することに注意する．さらに，集合 $S(I_x, I_A, J_y, J_B) \subseteq \Re^{m+n+2}$ を

$$S(I_x,I_A,J_y,J_B) := \left\{ \begin{pmatrix} x \\ y \\ \alpha \\ \beta \end{pmatrix} \in \Re^{m+n+2} \;\middle|\; \begin{array}{l} e_m^T x = 1,\ e_n^T y = 1 \\ x_i = 0,\ i \in I_x \\ x_i \geq 0,\ i \in \mathcal{M} \setminus I_x \\ \alpha - A_{i\cdot} y = 0,\ i \in I_A \\ \alpha - A_{i\cdot} y \geq 0,\ i \in \mathcal{M} \setminus I_A \\ y_j = 0,\ j \in J_y \\ y_j \geq 0,\ j \in \mathcal{N} \setminus J_y \\ \beta - B_{\cdot j}^T x = 0,\ j \in J_B \\ \beta - B_{\cdot j}^T x \geq 0,\ j \in \mathcal{N} \setminus J_B \end{array} \right\}$$

と定義する．さらに，集合 Ψ_0, Ψ_S を

$$\Psi_0 := \left\{ (I_x,I_A,J_y,J_B) \;\middle|\; \begin{array}{l} I_x \cup I_A = \mathcal{M},\ I_x \cap I_A = \emptyset \\ J_y \cup J_B = \mathcal{N},\ J_y \cap J_B = \emptyset \end{array} \right\}$$

$$\Psi_S := \{(I_x,I_A,J_y,J_B) \in \Psi_0 \mid S(I_x,I_A,J_y,J_B) \neq \emptyset\}$$

と定義する．均衡解集合 E_B はこれらを用いて,

$$E_B = \bigcup_{(I_x,I_A,J_y,J_B)\in\Psi_S} S(I_x,I_A,J_y,J_B)$$

と表現することができる．したがって双行列ゲームの均衡解をすべて表すには，Ψ_S の要素をすべて列挙すればよい.

3.3 アルゴリズム 2 の具体化

双行列ゲームに 2 節で提案したアルゴリズム 2 を適用する際の具体的な実装方法を説明する．列挙木の各ノードには (I_x, I_A, J_y, J_B) が対応する．列挙木の初期ノードすなわち根には $(\emptyset, \emptyset, \emptyset, \emptyset)$ を与える．そして，制約領域 $S(I_x, I_A, J_y, J_B) \subset \Re^{m+n+2}$ を

$$S(I_x, I_A, J_y, J_B) := \left\{ \left(\begin{array}{c} x \\ y \\ \alpha \\ \beta \end{array} \right) \in \Re^{m+n+2} \left| \begin{array}{c} e_m^T x = 1,\ e_n^T y = 1 \\ x_i = 0,\ i \in I_x \\ x_i \geq 0,\ i \in \mathcal{M} \setminus I_x \\ \alpha - A_{i\cdot} y = 0,\ i \in I_A \\ \alpha - A_{i\cdot} y \geq 0,\ i \in \mathcal{M} \setminus I_A \\ y_j = 0,\ j \in J_y \\ y_j \geq 0,\ j \in \mathcal{N} \setminus J_y \\ \beta - B_{\cdot j}^T x = 0,\ j \in J_B \\ \beta - B_{\cdot j}^T x \geq 0,\ j \in \mathcal{N} \setminus J_B \end{array} \right. \right\}$$

と定義する．根に与える初期制約領域 $S(\emptyset, \emptyset, \emptyset, \emptyset)$ は相補性条件を除いた領域

$$S(\emptyset, \emptyset, \emptyset, \emptyset) := \left\{ \left(\begin{array}{c} x \\ y \\ \alpha \\ \beta \end{array} \right) \in \Re^{m+n+2} \left| \begin{array}{c} e_m^T x = 1,\ e_n^T y = 1 \\ x_i \geq 0,\ i \in \mathcal{M} \\ \alpha - A_{i\cdot} y \geq 0,\ i \in \mathcal{M} \\ y_j \geq 0,\ j \in \mathcal{N} \\ \beta - B_{\cdot j}^T x \geq 0,\ j \in \mathcal{N} \end{array} \right. \right\}$$

である．

分枝操作　相補性条件が $x_i(\alpha - A_{i\cdot} y) = 0,\ i \in \mathcal{M}$ と $y_j(\beta - B_{\cdot j}^T x) = 0,\ j \in \mathcal{N}$ の 2 つの形に分離できるので，双行列ゲームの分枝操作は 2 つのパターンで表される．現在のノード (I_x, I_A, J_y, J_B) に対して，

1. 分枝する添字として $\tilde{\imath} \in \mathcal{M}$ が選ばれたら，次の 2 つのノードに分枝する．
   - 等式制約 $x_{\tilde{\imath}} = 0$ を加えたノード $(I_x \cup \{\tilde{\imath}\}, I_A, J_y, J_B)$,
   - 等式制約 $\alpha - A_{\tilde{\imath}\cdot} y = 0$ を加えたノード $(I_x, I_A \cup \{\tilde{\imath}\}, J_y, J_B)$.
2. 分枝する添字として $\tilde{\jmath} \in \mathcal{N}$ が選ばれたら，次の 2 つのノードに分枝する．
   - 等式制約 $y_{\tilde{\jmath}} = 0$ を加えたノード $(I_x, I_A, J_y \cup \{\tilde{\jmath}\}, J_B)$,
   - 等式制約 $\beta - B_{\cdot\tilde{\jmath}}^T x = 0$ を加えたノード $S(I_x, I_A, J_y, J_B \cup \{\tilde{\jmath}\})$.

この分枝を相補性条件が満たされるまでおこなう．つまり，ノードの深さが $m+n$ になるまですべてのノードに対しておこなう．

添字の選び方　現在のノードにおける制約領域 $S(I_x, I_A, J_y, J_B)$ に対して，線形な副問題 $R(I_x, I_A, J_y, J_B)$

$$R(I_x, I_A, J_y, J_B) : \left\{ \begin{array}{ll} \min & x^T(\hat{\alpha} e_m - A\hat{y}) + y^T(\hat{\beta} e_n - B^T \hat{y}) \\ \text{s.t.} & (x, y, \alpha, \beta) \in S(I_x, I_A, J_y, J_B) \end{array} \right.$$

を解いて近似解 $(x^k, y^k, \alpha_k, \beta_k) \in \Re^{m+n+2}$ を得る．ここで $(\hat{x}, \hat{y}, \hat{\alpha}, \hat{\beta}) \in \Re^{m+n+2}$ は現在のノードに対する親ノードの近似解である．そして，相補積

$$\tau_i = x_i^k(\alpha_k - A_{i\cdot} y^k),\ i \in \mathcal{M} \setminus (I_x \cup I_A)$$

の中で最大となるものを $\tau_{\tilde{i}}$ とし，

$$\pi_j = y_j^k(\beta - B_{\cdot j}^T x^k),\ j \in \mathcal{N} \setminus (J_y \cup J_B)$$

の中で最大となるものを $\pi_{\tilde{j}}$ とする．$\tau_{\tilde{i}} \geq \pi_{\tilde{j}}$ ならば分枝する添字として $\tilde{i}$ を採用し，反対に $\tau_{\tilde{i}} < \pi_{\tilde{j}}$ ならば分枝する添字として $\tilde{j}$ を採用する．

近似解 $(x^k, y^k, \alpha_k, \beta_k) \in \Re^{m+n+2}$ が均衡解になっている場合は，

$$\tau_i := \max(x_i^k,\ \alpha_k - A_{i\cdot} y^k),\ i \in \mathcal{M} \setminus (I_x \cup I_A),$$
$$\pi_j := \max(y_j^k,\ \beta - B_{\cdot j}^T x^k),\ j \in \mathcal{N} \setminus (J_y \cup J_B),$$

とする．

$C(I_x, I_A, J_y, J_B)$ の求め方　各ノード (I_x, I_A, J_y, J_B) に対する $C(I_x, I_A, J_y, J_B)$ の求め方を説明する．問題の構造上，α と β は互いに依存しないので，$C(I_x, I_A, J_y, J_B)$ は α に対応した集合 $C_\alpha(I_x, I_A, J_y, J_B)$ と β に対応した集合 $C_\beta(I_x, I_A, J_y, J_B)$ の和集合として表すことができる．そこで以下では，α に対応した $C_\alpha(I_x, I_A, J_y, J_B)$ の求め方を中心に説明する．

まず，制約

$$x \geq 0,\ e_m^T x = 1$$
$$y \geq 0,\ e_n^T y = 1$$

より，$x_i^{\max} \leq 1, \forall i \in \mathcal{M},\ y_j^{\max} \leq 1, \forall j \in \mathcal{N}$ である．

次に，各ノードに対する $\alpha - A_{i\cdot} y$ の上界値の求め方を説明する．変数 α に関して，$I_A = \{i \mid \alpha - A_{i\cdot} y = 0\}$ より，$I_A \neq \emptyset$ ならば有界となる．$I_A = \emptyset$ ならば α は有界でないので，$C_\alpha(I_x, I_A, J_y, J_B) = \Re^{m+n+2}$ とする．現在の制約領域 $S(I_x, I_A, J_y, J_B)$ に対して，α の取りうる最大値と $A_{i\cdot} y$ の取りうる最小値を求めれば $\alpha - A_{i\cdot} y$ の上界値を求めることができる．$(x, y, \alpha, \beta) \in S(I_x, I_A, J_y, J_B)$ より

$$\alpha - A_{i\cdot} y \geq 0,\ i \in \mathcal{M} \setminus I_A$$
$$\alpha - A_{i\cdot} y = 0,\ i \in I_A \tag{3.4}$$
$$y_j \geq 0,\ j \in \mathcal{N} \setminus J_y \tag{3.5}$$
$$y_j = 0,\ j \in J_y \tag{3.6}$$
$$e_n^T y = 1 \tag{3.7}$$

である．まず $A_{i\cdot} y$ の上界値と下界値を求める．各 $i \in \mathcal{M}$ に対して $A_{i\cdot} y$ は

$$A_{i\cdot} y = \sum_{j \in \mathcal{N}} a_{ij} y_j = \sum_{j \in \mathcal{N} \setminus J_y} a_{ij} y_j$$

と表すことができる．制約式 (3.5)-(3.7) に注意すれば，要素 $\{a_{ij}\},\ j \in \mathcal{N} \setminus J_y$ の最大値と最小値に対応する y_j をそれぞれ 1 とすることで，

$$\min_{j \in \mathcal{N} \setminus J_y} \{a_{ij}\} \leq A_{i\cdot} y \leq \max_{j \in \mathcal{N} \setminus J_y} \{a_{ij}\},\ i \in \mathcal{M} \tag{3.8}$$

を得る．次に α の上界値を求める．$I_A \neq \emptyset$ の場合は式 (3.4) より，すべての $i \in I_A$ に対して，

$$\alpha = A_{i\cdot} y,\ i \in I_A$$

が成立する．したがって，各 $i \in I_A$ に対して

$$\alpha \leq \max_{j \in \mathcal{N} \setminus J_y} \{a_{ij}\},\ i \in I_A$$

が成り立つから，α の上界値は

$$\alpha \leq \min_{i \in I_A}(\max_{j \in \mathcal{N} \setminus J_y}\{a_{ij}\}) \tag{3.9}$$

となる．式 (3.8)(3.9) より，$I_A \neq \emptyset$ のとき，各 $i \in \mathcal{M}$ に対して $\alpha - A_{i\cdot}y$ の上界値 ΔA_i は

$$\Delta A_i := \min_{k \in I_A}(\max_{j \in \mathcal{N} \setminus J_y}\{a_{kj}\}) - \min_{j \in \mathcal{N} \setminus J_y}\{a_{ij}\}$$

となる．よって，$C_\alpha(I_x, I_A, J_y, J_B)$ は以下のように表される．

$$C_\alpha(I_x, I_A, J_y, J_B) := \left\{(x, y, \alpha, \beta)^T \in \Re^{m+n+2} \;\middle|\; x_i + \tfrac{\alpha - A_{i\cdot}y}{\Delta A_i} \leq 1,\ i \in \mathcal{M} \setminus (I_x \cup I_A)\ \right\}$$

これと同様の議論が $\beta - B_{\cdot j}^T x,\ j \in \mathcal{N}$ にも成り立ち，$J_B \neq \emptyset$ のとき，各 $j \in \mathcal{N}$ に対して，$\beta - B_{\cdot j}^T x$ の上界値 ΔB_j は

$$\Delta B_j := \min_{k \in J_B}(\max_{i \in \mathcal{M} \setminus I_x}\{b_{ik}\}) - \min_{i \in \mathcal{M} \setminus I_x}\{b_{ij}\}$$

となる．そのため，$J_B \neq \emptyset$ のときの相補性条件の凸緩和領域 $C_\beta(I_x, I_A, J_y, J_B)$ は

$$C_\beta(I_x, I_A, J_y, J_B) := \left\{(x, y, \alpha, \beta)^T \in \Re^{m+n+2} \;\middle|\; y_j + \tfrac{\beta - B_j^T x}{\Delta B_j} \leq 1,\ j \in \mathcal{N} \setminus (J_y \cup J_B)\ \right\},$$

となる．以上をまとめると，$I_A \neq \emptyset, J_B \neq \emptyset$ のときの $C(I_x, I_A, J_y, J_B)$ は

$$C(I_x, I_A, J_y, J_B) := C_\alpha(I_x, I_A, J_y, J_B) \cup C_\beta(I_x, I_A, J_y, J_B)$$

となる．さらに，副問題 $R(I_x, I_A, J_y, J_B)$ の制約領域を $S(I_x, I_A, J_y, J_B) \cap C_\alpha(I_x, I_A, J_y, J_B)$ とした副問題を $R_\alpha(I_x, I_A, J_y, J_B)$ と呼び，制約領域を $S(I_x, I_A, J_y, J_B) \cap C_\beta(I_x, I_A, J_y, J_B)$ とした副問題を $R_\beta(I_x, I_A, J_y, J_B)$，制約領域を $S(I_x, I_A, J_y, J_B) \cap C(I_x, I_A, J_y, J_B)$ とした副問題を $R_c(I_x, I_A, J_y, J_B)$ と呼ぶことにする．

3.4 アルゴリズム EAI(Enumeration of All Indices which express all equilibria of bimatrix games)

STEP1:初期化

列挙木 T に初期ノード $(\emptyset, \emptyset, \emptyset, \emptyset)$ を与える．初期ノードの親の解は，$S(\emptyset, \emptyset, \emptyset, \emptyset)$ の任意の実行可能解とする．

STEP2:ノードの選択

T が空であれば終了．

さもなければ，ノード (I_x, I_A, J_y, J_B) を T から深さ優先探索で選び T から削除する．

$I_A = J_B = \emptyset$ ならば，$R(I_x, I_A, J_y, J_B)$ を副問題とする．

$I_A \neq \emptyset, J_B = \emptyset$ ならば，$R_\alpha(I_x, I_A, J_y, J_B)$ を副問題とする．

$I_A = \emptyset, J_B \neq \emptyset$ ならば，$R_\beta(I_x, I_A, J_y, J_B)$ を副問題とする．

$I_A \neq \emptyset, J_B \neq \emptyset$ ならば，$R_c(I_x, I_A, J_y, J_B)$ を副問題とする．

STEP3:実行可能性のチェック

副問題が実行不可能であれば STEP2 へ戻る．

副問題が実行可能かつ現在のノードの深さが $m+n$ であれば，添字集合の組を解として記録し STEP2 へ戻る．

いずれでもなければ，副問題を解いて暫定解 $(x^k, y^k, \alpha_k, \beta_k) \in \Re^{m+n+2}$ を計算し，STEP4 へ．

STEP4:分枝操作

ノードの暫定解 $(x^k, y^k, \alpha_k, \beta_k) \in \Re^{m+n+2}$ に対して，

$$\tau_i := \begin{cases} x_i^k(\alpha_k - A_{i.}y^k), & i \in \mathcal{M} \setminus (I_x \cup I_A) \\ -1, & i \in (I_x \cup I_A) \end{cases},$$

$$\pi_j := \begin{cases} y_j^k(\beta_k - B_{.j}^T x^k), & j \in \mathcal{N} \setminus (J_y \cup J_B) \\ -1, & j \in (J_y \cup J_B) \end{cases}$$

を計算する．もし，すべての $i \in \mathcal{M},\ j \in \mathcal{N}$ に対して $\tau_i = 0,\ \pi_j = 0$ となっていたら，

$$\tau_i := \begin{cases} \max\{x_i^k,\ (\alpha_k - A_{i.}y^k)\}, & i \in \mathcal{M} \setminus (I_x \cup I_A) \\ -1, & i \in (I_x \cup I_A) \end{cases}, \tag{3.10}$$

$$\pi_j := \begin{cases} \max\{y_j^k,\ (\beta_k - B_{.j}^T x^k)\}, & j \in \mathcal{N} \setminus (J_y \cup J_B) \\ -1, & j \in (J_y \cup J_B) \end{cases} \tag{3.11}$$

を計算する．$\tau_i,\ i \in \mathcal{M}$ を最大とする添字 $\tilde{\imath}$，$\pi_j,\ j \in \mathcal{N}$ を最大する添字 $\tilde{\jmath}$ を求める．$\tau_{\tilde{\imath}} \geq \pi_{\tilde{\jmath}}$ であれば，ノード $(I_x \cup \{\tilde{\imath}\}, I_A, J_y, J_B)$ とノード $(I_x, I_A \cup \{\tilde{\imath}\}, J_y, J_B)$ を列挙木 T に加える．$\tau_{\tilde{\imath}} < \pi_{\tilde{\jmath}}$ であれば，ノード $(I_x, I_A, J_y \cup \{\tilde{\imath}\}, J_B)$ とノード $(I_x, I_A, J_y, J_B \cup \{\tilde{\imath}\})$ を列挙木 T に加える．STEP2 へ戻る．

4 数値実験

アルゴリズム EAI を評価するために数値実験を行った．本実験は CPU が 3.2GHz の Pentium4 上で，MATLAB7.0 を用いておこなった．線形計画問題は，option の'Largescale' を'off' とした組み込み関数'linprog' で解いた．

問題例として双行列ゲームの均衡解を求める問題と等価な LMCP(3.3) を用いた．双行列ゲームにおける利得行列 $A, B \in \Re^{m \times n}$ の要素はすべて $[0, 1]$ の乱数で生成した．利得行列 A, B のサイズ m, n に関して以下の 3 種類のパターンを考えた．

- $m = n$ として n の値を大きくしていった場合;
- $m = 3$ と固定して n の値を大きくしていった場合;
- $m = 5$ と固定して n の値を大きくしていった場合.

各サイズで 10 個の問題例を解き，アルゴリズムが終了したときの列挙木のノード数の平均と標準偏差を表 1 から表 3 にまとめる．表中の EEE,EAI1,EAI2,EAI3 はそれぞれ以下のアルゴリズムである，

- EEE : Audet らによって提案された双行列ゲームの均衡解を求めるアルゴリズム，
- EAI1: アルゴリズム EAI で実行可能性のチェックを強化する工夫を行わない，つまり，$C(I_x, I_A, J_y, J_B) = \Re^{m+n+2}$ としたアルゴリズム，
- EAI2: アルゴリズム EAI で分枝選択の工夫を行わない，つまり，式 (3.10) の計算をおこなわないアルゴリズム，
- EAI3: アルゴリズム EAI で工夫を 2 つとも行わないアルゴリズム，つまり，アルゴリズム 1.

表 1: $m = n$ の場合の総ノード数

m=n		均衡解数	EAI	EEE	EAI1	EAI2	EAI3
n=4	平均	2.80	41.80	45.00	48.40	41.40	45.00
	標準偏差	1.75	18.65	19.93	20.44	20.08	21.60
n=7	平均	5.80	181.20	203.00	191.00	194.00	205.00
	標準偏差	3.16	73.28	79.84	73.20	81.34	82.18
n=10	平均	10.20	630.40	736.00	645.40	695.60	712.20
	標準偏差	6.68	308.69	376.77	309.41	376.10	377.25
n=13	平均	23.20	2074.60	2424.60	2099.00	2331.80	2358.20
	標準偏差	14.56	873.83	1203.47	869.31	1011.64	1011.55
n=16	平均	55.80	9071.20	10229.40	9101.80	9953.80	9999.80
	標準偏差	20.37	4367.87	5138.27	4368.57	4738.87	4743.88

表 2: $m = 3$ の場合の総ノード数

m=3		均衡解数	EAI	EEE	EAI1	EAI2	EAI3
n=5	平均	2.20	33.80	34.20	39.00	33.20	35.00
	標準偏差	1.40	13.93	16.17	15.38	14.74	17.13
n=10	平均	2.60	78.40	95.60	94.80	78.20	93.60
	標準偏差	1.58	26.33	29.96	24.61	28.43	25.42
n=20	平均	3.60	182.20	221.00	207.20	188.60	223.80
	標準偏差	2.32	81.72	117.35	76.32	102.07	117.75
n=100	平均	5.00	1484.20	1889.80	1709.00	1766.80	1980.60
	標準偏差	1.89	625.50	713.17	589.93	672.47	745.29
n=130	平均	5.80	2049.80	2814.20	2436.40	2400.20	2828.80
	標準偏差	1.69	664.29	1056.63	815.51	743.01	1029.40

4.1 考察

まず，EAI と EEE を比較する．すべての問題例に関して，EAI で生成されたノード数の平均値は EEE のノード数よりも少ないことがわかる．実際に EAI は EEE と比べてノード数が平均 15 %少なくなっている．特に，$m = 3, n = 130$ では EAI のノード数は EEE のノード数に対して 27.2 %も少ないように $m = n$ の場合よりも $m \ll n$ の場合の方が改善の幅が大きくなることがわかる．また，EEE は双行列ゲームに特化したアルゴリズムであるのに対して，EAI は一般の LMCP に適用することができる．そのことを考慮すると，十分満足のいく結果である．

次に，実行可能性のチェックの強化の工夫，つまり $C(I_x, I_A, J_y, J_B)$ の効果を見るために EAI2 と EAI3 を比較する．$m = n$ の場合はあまり効果がなかった．この例では解の数が問題のサイズに対して指数的に増加している．そのため，実行可能であるようなノードが増えてしまい，制約の限定をする工夫の効果があまり得られていないと考えられる．$m \ll n$ の場合は $m = n$ の場合よりも効果があることが見て取れる．これは，問題のサイズに対して解の数が少ないので，$C(I_x, I_A, J_y, J_B)$ によって実行不可能となるノードが多くなるためである．

次に，分枝選択 (3.10) の工夫の効果を見るため，EAI1 と EAI3 を比較する．分枝選択の工夫は問題のサイズが小さいうちは，逆に工夫を加えない場合よりもノード数が増加するが，問題のサイズが大きくなればなるほど効果が出てくる傾向にある．

最後に，2 つの工夫を合わせた効果を見るため，EAI,EAI3 で比較する．$m = n$ の場合は，どのサイズの問題例も平均 10 %程度でノード数が減少している．$m \neq n$ の場合は，問題のサイズが大きくなるにつれて，工夫の効果が増加している．また，問題のサイズが大きくなるにつれて 2 つの工夫の相乗効果が安定

表 3: $m = 5$ の場合の総ノード数

m=5		均衡解数	EAI	EEE	EAI1	EAI2	EAI3
n=8	平均	3.80	104.40	116.20	118.00	109.00	117.40
	標準偏差	2.86	68.90	73.76	70.35	77.66	79.88
n=10	平均	4.00	153.40	172.00	171.00	157.20	178.60
	標準偏差	2.54	76.46	81.07	71.85	96.50	86.73
n=20	平均	7.40	533.00	622.60	571.80	583.80	629.20
	標準偏差	3.95	234.00	272.71	240.48	249.13	249.22
n=50	平均	11.80	2491.40	2998.60	2606.80	3033.80	3182.20
	標準偏差	6.68	1203.63	1463.42	1247.18	1601.03	1637.61

することがわかった.

5 結論

本報告書では，LMCP の解集合を構成する凸多面体をすべて列挙するアルゴリズムを提案した．提案したアルゴリズムを双行列ゲームの Nash 均衡解を求める問題に適用して数値実験を行った．数値実験の結果から，アルゴリズムが既存の手法よりもよい結果が得られることを確認した．また，双行列ゲームにおける 2 人のプレイヤーの戦略の数に大きな差があるほど改善の幅が大きいこともわかった.

今後の課題として，今回の数値実験の実装において各ノードでの副問題を MATLAB の組み込み関数で解き，凸緩和の工夫を加えるための変数と関数の上限を求める部分は自分でコーディングしている．したがって，上限を求める操作が副問題を解く時間に比べて，どれだけ時間がかかるのが議論できていない．また，一般の LMCP に対して実際に数値実験を行っていない．これらの点を含めて，より綿密に数値実験をおこなうことが今後の研究課題である.

謝辞

日頃から御教授下さり，本研究に対しても熱心な御指導を賜った福嶋雅夫教授，ならびに本報告書作成にあたり細部に至るまで貴重な御指摘と御指導を頂いた山下信雄助教授に深く感謝の意を表します．また，日頃からお世話になっている林俊介特任助手をはじめとする福嶋研究室の皆様に厚く御礼申し上げます.

参考文献

[1] Al-khayyal, F.A. : *An implicit enumeration procedure for the general linear complementarity problem*, Mathematical programming study 31, 1-21, 1987.

[2] Audet, C., Hansen, P., Jaumard, B. and Savard, G. : *Enumeration of all extreme equilibria of bimatrix games*, SIAM Journal on Science and Computer, 23, No.1, 323-338, 2001.

[3] Billups, S.C. : *A homotopy based algorithm for mixed complementarity problems*, Technical report, Department of mathematics, University of Colorado, Denver, 1998.

[4] Cottle, R.W., Pang, J.-S. and Stone, R.E. : *The linear complementarity problem*, Academic press INC,Boston, 1992.

[5] Facchinei, F. and Pang, J.-S. : *Finite-Dimensional variational inequalities and complementarity problems vol.1 vol.2*, Springer INC, New York, 2003.

[6] Ferris, M.C. and Pang, J.-S. : *Engineering and economic applications of complementarity problems*, SIAM Review 39, 669-713, 1997.

[7] 福島雅夫: 非線形最適化の基礎, 朝倉書店, 2001.

[8] Horst, R. and Pardalos, P.M. : *Handbook of global optimization*, Kluwer academic publishers, Boston, 1995.

[9] Horst, R., Pardalos, P.M. and Thoai, N.V. : *Introduction to global optimization*, Kluwer academic publisher, Boston, 1995.

[10] Kanzow, C. : *Global optimization techniques for mixed complementarity problems*, Journal of Global Optimization 16, 1-21, 2000.

[11] Kostreva, M.M. and Zheng, Q. : *Integral global optimization method for solution of nonlinear complementarity problems*, Journal of Global Optimization 5, 181-193, 1994.

[12] Mangasarian, O.L. and Solodov, M.V. : *Nonlinear complementarity as unconstrained and constrained minimizaton*, Mathematical Programing, 62, 277-297, 1993.

[13] Millham, C.B. : *On nash subsets of bimatrix games*, Nabal Reserch Logistics, 74, 307-317, 1974.

[14] 岡田　章 : ゲーム理論, 有斐閣, 1996.

[15] Sellami, H. and Robinson, S.M. : *Implementation of a continuation method for normal maps*, Mathematical Programming 76, 563-578, 1997.

[16] Sellami, H. and Robinson, S.M. : *Homotopies based on nonsmooth equations for optimization and applications*, Plenum press, New York, 1996.

[17] Watson, L.T. : *Solving the nonlinear complementarity problem by homotopy method*, SIAM journal on Control and Optimization 17, 36-46, 1979.

A 付録

A.1 LMCP(1.1) から等式制約付き相補性問題 (1.2) への変換

LMCP(1.1) は直方体上の変分不等式問題

$$\begin{aligned} &\text{find} \quad x^* \in [l,u]\,, \\ &\text{s.t.} \quad \langle F(x^*), x - x^* \rangle \geq 0,\ \forall x \in [l,u] \end{aligned} \tag{A.1}$$

と等価である [7]．式 (A.1) の KKT 条件は

$$\begin{cases} F(x) - \mu + \lambda = 0 \\ \mu \geq 0,\ x - l \geq 0,\ \mu^T (x - l) = 0 \\ \lambda \geq 0,\ u - x \geq 0,\ \lambda^T (u - x) = 0 \end{cases} \tag{A.2}$$

と書ける．ここで，$y := (x, \mu, \lambda)^T \in \Re^{3n}$，$F' : \Re^{3n} \to \Re^{3n}$ を

$$F'(y) := \begin{bmatrix} F(x) - \mu + \lambda \\ x - l \\ u - x \end{bmatrix} \tag{A.3}$$

とおくと，LMCP(1.1) は以下の等価な等式条件と相補性条件を満たすベクトルを求める問題

$$F_i'(y) = 0, \qquad (i = 1, \cdots, n), \tag{A.4}$$

$$F_i'(y) \geq 0,\ y_i \geq 0,\ y_i F_i'(y) = 0, \qquad (i = n+1, \cdots, 3n) \tag{A.5}$$

に書くことができる．