授業を補助するプログラム (2)

— 協力ゲーム —

社会情報学科　行　方　常　幸

1．はじめに…… 51
2．「協力ゲーム」の解の計算方法…… 52
3．「協力ゲーム」プログラム…… 59
4．プログラムの構成…… 64
5．ツリーの利用について…… 68
6．補遺…… 70
7．おわりに…… 74
参考文献…… 75

1．はじめに

私が担当する数理的な科目（例えば，ゲーム理論を扱う「計画科学」等）において，その内容を理解するためには，多くの数値例を実際に解くことが必要であるものが多い。しかしながら，人間の手計算で解ける問題は，小規模なものに限られる。また，手計算で解ける問題でも，自分で求めた答えが正解であるかをチェックすることも容易ではない。

これに対処するために，協力ゲーム理論で有名な解である，シャープレイ値，τ-値，仁，団結値，最小二乗準仁などを計算するプログラムを作成したので本稿で紹介する。中規模程度以下の問題に対して，データを入力し，該当するボタンを押せば，これらの解を計算するプログラムである。このプログラムを有効に利用することにより，計算の手間に必要以上にとらわれず，解そのものの性質に注意の焦点を集めることが可能となる。

2.「協力ゲーム」の解の計算方法

この節では本稿で扱う解の定義と計算方法に焦点を当てて，ゲーム理論を説明する。本稿で扱うのは譲渡可能効用を持つ提携形ゲームである。この譲渡可能効用を持つ提携形ゲームとは（プレイヤーと呼ばれる）参加者全員の集合 $N=\{1,\ldots,n\}$ と特性関数 v の組 (N,v) である。v は（提携と呼ばれる）N の各部分集合 $S(\subset N)$ に対して，提携値と呼ばれる実数 $v(S)$ を対応させる関数であり，提携 S のメンバーが協力して得られる利益の総和を表す。

譲渡可能効用を持つ提携形ゲームの主な関心ごとは，全体提携が形成される時に得られる利益 $v(N)$ を，全体提携以外の提携値 $v(S)$ $(S\subset N)$ を参考にして，各プレイヤーに公平に分けるにはいかにすべきか？　である。この問題に対する答えとして，従来から多くの解（値とも呼ばれる）が提唱されている。本稿ではこれらの中から，シャープレイ値，τ-値，仁，団結値，最小二乗準仁，ν-値を扱う。以下では $v(N)\geq\sum_{j\in N}v(\{j\})$ と仮定しておく。

シャープレイ値，団結値，最小二乗準仁，ν-値は最小二乗値と呼ばれるグループに属する解で，次のように定義されている。

$m_{n,s}(s=1,\ldots,n-1)$ を各 $n(=2,\ldots)$ に対して少なくとも1項が正である，非負の重みとする。この重み m に対する最小二乗値 LS^m は次の最小化問題の最適解として定義される：

各ゲーム (N,v) に対して

$$\min\sum_{\substack{S:S\subset N\\ S\neq N}}m_{n,s}\left(v(S)-\sum_{j\in S}x_j\right)^2$$

$$\text{sub. to}\ \sum_{j\in N}x_j=v(N)$$

この最適解は陽に求めることができ，次のようになることが知られている。

$$LS_i^m(N,v) := \frac{v(N)}{n} + \frac{1}{\sum_{s=1}^{n-1}\binom{n-2}{s-1}m_{n,s}}\left[\sum_{\substack{S:i\in S\subset N\\ S\neq N}} m_{n,s}\, v(S) - \sum_{\substack{S:S\subset N\\ S\neq\phi,N}} \frac{s}{n}\, m_{n,s}\, v(S)\right]$$

$$(\forall\, i \in N) \tag{1}$$

(i)シャープレイ値，(ii)団結値，(iii)最小二乗準仁，(iv)ν-値は重み m が各々

(i) $m_{n,s} = \frac{1}{n-s}\binom{n-1}{s-1}^{-1}$ $(n=2,\ldots,;s=1,\ldots,n-1)$

(ii) $m_{n,s} = \frac{1}{(s+1)(n-\mathrm{s})}\binom{n-1}{s-1}^{-1}$ $(n=2,\ldots,;s=1,\ldots,n-1)$

(iii) $m_{n,s} = \frac{1}{2^{n-2}}$ $(n=2,\ldots,;s=1,\ldots,n-1)$

(iv) $m_{n,s} = \frac{2s}{n(n-s)}\binom{n-1}{s-1}^{-1}$ $(n=2,\ldots,;s=1,\ldots,n-1)$

で与えられる時の最小二乗値であることが知られている。これら4つの重み m は $\sum_{s=1}^{n-1}\binom{n-2}{s-1}m_{n,s}=1$ を満足する。これら4つの解（値）を計算するには公式(1)を直接計算すればよい。

τ-値は次のように計算される。まず，上界ベクトル b とギャップ関数 g と譲歩ベクトル λ を次のように定義する：

$$b_j := v(N) - v(N-\{j\})$$

$$g(S) := \sum_{j\in S} b_j - v(S) \tag{2}$$

$$\lambda_j := \min\{g(S) \mid j \in S \subset N\}$$

ゲーム (N,v) が準平衡ゲームならば，すなわち，

$$g(S) \geq 0 \ (\forall S \subset N)$$

$$\sum_{j \in N} \lambda_j \geq g(N)$$

を満足するならば，τ-値は

$$\tau(N,v) := \begin{cases} b - \dfrac{g(N)}{\sum_{j \in N} \lambda_j} \lambda & \left(\sum_{j \in N} \lambda_j > 0\right) \\ b & \left(\sum_{j \in N} \lambda_j = 0\right) \end{cases} \tag{3}$$

と定義される。

ゲーム (N,v) が準平衡ゲームでない場合は，まず，ダミープレイヤーを求める。ダミープレイヤーとは

$$v(S \cup \{i\}) - v(S) = v(\{i\}) \ (\forall S \subset N - \{i\})$$

が成り立つプレイヤー i のことである。ダミープレイヤーの集合を D とし $M := N - D$ とおく。ダミープレイヤーの τ-値 $\tau(v)$ は次のように定義される：

$$\tau_j(N,v) := v(\{j\}) \ (j \in D)$$

M に属するプレイヤーの τ-値は，少し面倒であるが，次のように定義される。まず，$0 \leq \varepsilon \leq 1$ とし，乗法的 ε-税ゲーム (M, v^ε) を次のように定義する：

$$v^\varepsilon(S) := \begin{cases} v(M) & S = M \\ (1-\varepsilon) v(S) + \varepsilon \sum_{j \in S} v(\{j\}) & S \subsetneq M \end{cases}$$

$\varepsilon=0$ の時，(M,v^{ε}) は (M,v) と一致するが，準平衡ゲームではない。$\varepsilon=1$ の時，(M,v^{ε}) は (M,v) と一致しないが，準平衡ゲームである。そこで，(M,v^{ε}) が準平衡ゲームとなる最小の ε を ε^* とおき，M に属するプレイヤーの τ-値を (M,v^{ε^*}) の τ-値と定義する，すなわち，

$$\varepsilon^*=\min\{\varepsilon\,|\,0\leq\varepsilon\leq 1,(M,v^{\varepsilon})\text{ は準平衡ゲーム}\} \tag{4}$$

の時，

$$\tau_j(N,v):=\tau_j(M,v^{\varepsilon^*})\ (j\in M)$$

これを実際に計算するには以下のようにする（詳しくは「補遺」を参照)。まず，ε^* とその時の上界ベクトル b^*，譲歩ベクトル λ^* を求める。

$$\begin{aligned} b_j^{\varepsilon}&:=v^{\varepsilon}(M)-v^{\varepsilon}(M-\{j\})\ (j\in M)\\ g^{\varepsilon}(S)&:=\sum_{j\in S}b_j^{\varepsilon}-v^{\varepsilon}(S)\\ \lambda_j^{\varepsilon}&:=\min\{g^{\varepsilon}(S)\mid j\in S\subset M\}\end{aligned} \tag{5}$$

とおくと，

$$\varepsilon^*:=\min\varepsilon$$

$$\text{sub. to}\begin{cases} g^{\varepsilon}(S)\geq 0\ (\forall S\subset M)\\ \sum_{j\in M}\lambda_j^{\varepsilon}\geq g^{\varepsilon}(M)\end{cases}$$

を解けばよい。

$$g^{\varepsilon}(S)=\begin{cases} g(M) & \\ \quad +\varepsilon\left[\sum_{j\in M} v(\{j\})+\sum_{j\in M} v(M-\{j\})-|M|\sum_{j\in M} v(\{j\})\right] & (S=M) \\ g(S) & \\ \quad +\varepsilon\left[v(S)+\sum_{j\in S} v(M-\{j\})-|S|\sum_{j\in M} v(\{j\})\right] & (S\subsetneq M) \end{cases}$$

となることに注意すれば，制約式は ε の1次式である。ここで $g(S)$ は(2)で与えられるものである。従って，次の線形計画問題を解くことに帰着される：

$$\varepsilon^{*}:=\min \varepsilon$$

$$\text{sub. to}\begin{cases} -\left[v(S)+\sum_{j\in S} v(M-\{j\})-|S|\sum_{j\in M} v(\{j\})\right]\varepsilon & \\ \qquad +\lambda_i\leq g(S) & (\forall i, S: i\in S\subsetneq M) \\ -\left[\sum_{j\in M} v(\{j\})+\sum_{j\in M} v(M-\{j\})-|M|\sum_{j\in M} v(\{j\})\right]\varepsilon & \\ \qquad +\lambda_i\leq g(M) & (\forall i\in M) \\ -\left[\sum_{j\in M} v(\{j\})+\sum_{j\in M} v(M-\{j\})-|M|\sum_{j\in M} v(\{j\})\right]\varepsilon & \\ \qquad +\sum_{j\in M}\lambda_j\geq g(M) & \\ \varepsilon\leq 1 & \\ \varepsilon\geq 0 & \\ \lambda_i\geq 0 & (\forall i\in M) \end{cases}$$

変数は ε と $\lambda_j\,(j\in M)$ であり，この線形計画問題を解いた時の目的関数の値が(4)の ε^{*} と一致し，(5)から上界ベクトル $b^{*}:=b^{\varepsilon^{*}}$ と譲歩ベクトル $\lambda^{*}:=\lambda^{\varepsilon^{*}}$ を求めこれらを(3)に代入すれば M に属するプレイヤーの τ-値が計算できる。以上により，準平衡ゲームではないゲームの τ-値は線形計画問題を解くことによ

って計算できることが分かった。

仁は次のように計算される。まず，(x に対する提携 S の) 不満 $e(S,x) := v(S) - \sum_{j\in S} x_j$ を定義する。そして，最大不満を最小にする，次の線形計画問題を解く：

$$\varepsilon_1 := \min \varepsilon$$
$$\text{sub. to} \begin{cases} \sum_{j\in N} x_j = v(N) \\ x_j \geq v(\{j\}) \ (\forall j \in N) \\ v(S) - \sum_{j\in S} x_j \leq \varepsilon \ (\forall S \subset N, S \neq \phi, N) \end{cases}$$

この線形計画問題を解いた時の制約条件を満たす x の集合を X_1 とする。すなわち，

$$X_1 := \left\{ x \;\middle|\; \begin{array}{c} \sum_{j\in N} x_j = v(N), \\ x_j \geq v(\{j\}) \ (\forall j \in N), \\ v(S) - \sum_{j\in S} x_j \leq \varepsilon_1 \ (\forall S \subset N, S \neq \phi, N) \end{array} \right\}$$

この X_1 が1点からなる集合ならば，この要素が仁である。そうでない場合，以下のように進む。

$$U_1 := \left\{ S \subset N \;\middle|\; \begin{array}{c} S \neq \phi, N \\ v(S) - \sum_{j\in S} x_j = \varepsilon_1 \ (\forall x \in X_1) \end{array} \right\}$$

U_1 に属している提携の不満を ε_1 未満にすることは不可能である。そこで，次の線形計画問題を解く：

$$\varepsilon_2 := \min \varepsilon$$

$$\text{sub. to} \begin{cases} \sum_{j\in N} x_j = v(N) \\ x_j \geq v(\{j\}) \ (\forall j \in N) \\ v(S) - \sum_{j\in S} x_j = \varepsilon_1 \ (\forall S \in U_1) \\ v(S) - \sum_{j\in S} x_j \leq \varepsilon \ (\forall S \notin U_1, S \neq \phi, N) \end{cases}$$

この問題を解いた時の制約条件を満たす x の集合 X_2 が1点からなる時はその要素が仁である。複数個の点からなる時は，更に以下のように進む。

$$U_k := \left\{ S \subset U \;\middle|\; \begin{array}{c} S \neq \phi, N \\ S \notin U_1 \cup \cdots \cup U_{k-1}, \\ v(S) - \sum_{j\in S} x_j = \varepsilon_k (\forall x \in X_{k-1}) \end{array} \right\}$$

とおき，次の線形計画問題を解く：

$$\varepsilon_{k+1} := \min \varepsilon$$

$$\text{sub. to} \begin{cases} \sum_{j\in N} x_j = v(N) \\ x_j \geq v(\{j\}) \ (\forall j \in N) \\ v(S) - \sum_{j\in S} x_j = \varepsilon_1 \ (\forall S \in U_1) \\ \vdots \\ v(S) - \sum_{j\in S} x_j = \varepsilon_k \ (\forall S \in U_k) \\ v(S) - \sum_{j\in S} x_j \leq \varepsilon \ (\forall S \notin U_1 \cup \cdots \cup U_k, S \neq \phi, N) \end{cases}$$

この問題を解いた時の制約条件を満たす x の集合 X_{k+1} が1点からなるまで繰返す。この要素が仁である。

最小値 ε_k を求めた線形計画問題から U_k を求めるには次のように行う。最小値 ε_k を求めた線形計画問題の不等号制約式

$$v(S)-\sum_{j\in S}x_j\leq\varepsilon(\forall S\not\in U_1\cup\cdots\cup U_{k-1},S\neq\phi,N)$$

の中で,

$$v(S)-\sum_{j\in S}x_j=\varepsilon_k \tag{6}$$

となった制約式に対応する提携 S は U_k の要素となる可能性がある。そこで,S が U_k の要素であるか否かをチェックするために,(6)が成立する S 以外提携に対応する制約式の右辺の変数 ε を値 ε_k におきかえ,S に対応する制約式はそのままでもう一度 ε を最小にする。新たな最小値が元の最小値と一致する時のみ S は U_k の要素である。

以上のことから線形計画問題を繰返し解くことにより仁を求めることができる。

3.「協力ゲーム」プログラム

作成した「協力ゲーム」プログラムを起動したのが図1である。この図が示すようにグラフィカルなインターフェースを持つアプリケーションである。

「特性関数の入力」タブに3人ゲームのデータが入力できるようになっている。この図から容易に想像できるように,必要ならばプレイヤーの総数を変更し,提携値を入力し,「値の計算」タブで解の計算を行う。この「協力ゲーム」プログラムがどのように動作するかを見てみる。

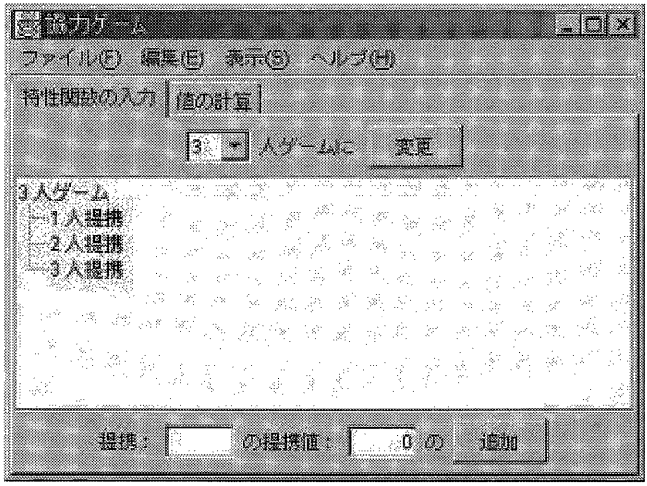

図1．「協力ゲーム」を起動したところ

次の4人ゲームの種々の値を求める（図2以下参照）。

図2．4人ゲームに変更したところ

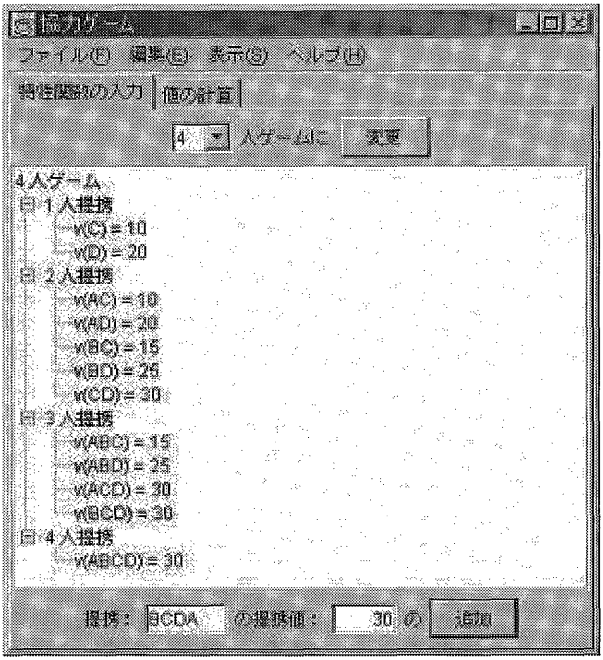

図3．データ入力後のウィンドウ

$v(\mathrm{C})=10$, $v(\mathrm{D})=20$, $v(\mathrm{AC})=10$, $v(\mathrm{AD})=20$, $v(\mathrm{BC})=15$, $v(\mathrm{BD})=25$, $v(\mathrm{CD})=30$, $v(\mathrm{ABC})=15$, $v(\mathrm{ABD})=25$, $v(\mathrm{ACD})=30$, $v(\mathrm{BCD})=30$, $v(\mathrm{ABCD})=30$。その他の $v(S)$ は0。

まず，注意することはプレイヤーがA，B，C，...となっている点である。3人ゲームならば，プレイヤーはA，B，Cであり，4人ゲームならば，プレイヤーはA，B，C，Dである。4人ゲームに変更するために，ウィンドウ上方中央部のコンボボックスの右の下三角をクリックし，「4」を選び，「変更」ボタンを押す。

ウィンドウ下方のテキストボックスに提携とその提携値を入力し「追加」ボタンを押す。例えば，$v(\mathrm{AC})=10$を入力するには，提携を入力するテキストボックスに，「AC」(小文字でも可，入力する順序は問わない，「cA」でもよい)

と入力し，提携値を入力するテキストボックスに「10」と入力する（非ゼロの有理数が入力できる）。$v(S)=0$ である提携 S と提携値 0 は入力しない。すなわち，入力されていないデータは 0 とみなす。上記のデータを入力したのが，図 3 である。

「特性関数の入力」タブのユーザーインターフェースについて少し説明する。提携値がツリー構造で表示されている。このツリーでは「＋」を押せば，そのノードが「展開」し，「-」を押せば，そのノードが「縮む」。入力済みの提携値を削除するには，その提携値をツリー上で選択し，「Delete」キーを押す（または，マウスを右クリックし，ポップアップメニューから「削除」を選ぶ）。入力済みのすべての提携値を一度に削除するには，「編集」メニューから「全データの削除」を選ぶ。同じ提携に違う提携値を「追加」しようとすると「提携値を変更しますか？」と表示され，「はい」と答えれば，変更される。入力済みの提携値を変更する他の方法は，その変更したい提携値をツリー上で選択し，マウスを右クリックし，ポップアップメニューから「変更」を選ぶ。「提携：」及び「提携値：」テキストフィールドに変更したい値が入力され，「追加」ボタンの名前が「修正」に変わる。提携値を修正し，「修正」ボタンを押せばよい。「表示」メニューから「分数」，「小数」，「帯分数」のいずれかを選択することにより，数値の表示を変更できる。

次に，「値の計算」タブに移る。先ほど入力した非ゼロの提携値が出力されている（図 4 参照）。

「シャープレイ値」等の 6 つのボタンが「値の計算」タブにあり，該当するボタンを押せばその値が計算される。ここではすべてのボタンを左上から右下の順に押す。得られた値は

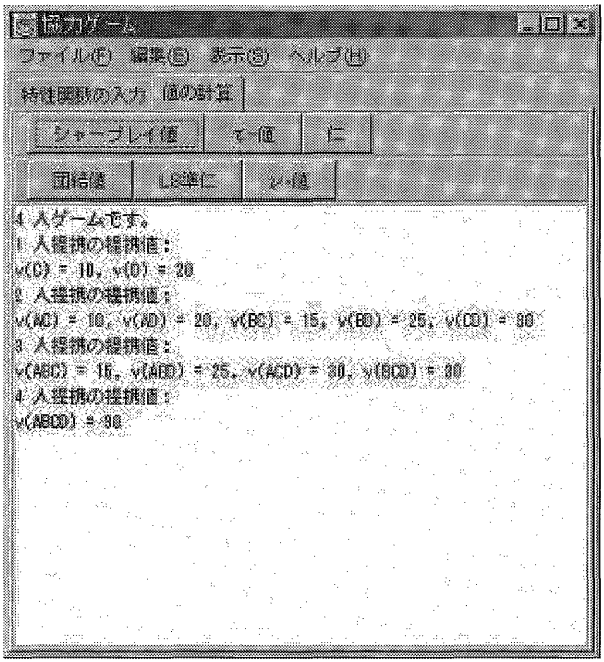

図 4.「値の計算」タブ

シャープレイ値＝(0, 1＋2/3, 9＋1/6, 19＋1/6), τ-値＝(0, 0, 10, 20),
仁＝(0, 0, 10, 20), 団結値＝(4＋13/18, 5＋5/18, 8＋7/36, 11＋29/36),
最小二乗準仁＝(－5/8, 1＋7/8, 9＋3/8, 19＋3/8),
ν-値＝(4＋31/36, 5＋5/12, 8＋7/36, 11＋19/36)
となる。

また, τ-値を計算する際に, 準平衡ゲームであるか否かをチェックするので, どちらであるかが出力されている。このゲームは準平衡ゲームではないことが分かる (図 5 参照)。

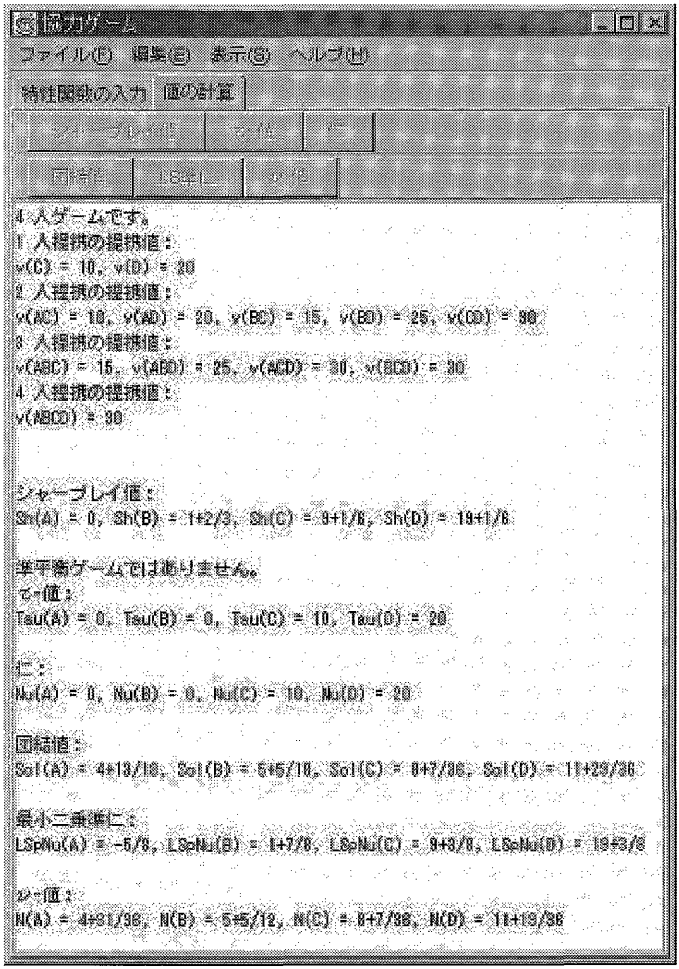

図5．種々の値

4．プログラムの構成

この節では，「協力ゲーム」プログラムの構成の概略を述べる。

まず，私が意図し最終的に作成したUI（ユーザーインターフェース）は次

の通りである。

(A) 提携値を表示するために，Java にあらかじめ定義されているツリー構造である JTree クラスを用いる。

(B) 近似計算ではなく可能な限り正確な計算をさせるために，扱える数値は分数とする。数値の表示は，分数，小数，帯分数の 3 つが指定できるようにする。

(C) この分数の入力を処理する際に，明らかな入力ミスは受け付けない。すなわち，「0」から「9」までの数字，小数点「.」，マイナス記号「-」，分数の「/」以外は入力として受け付けない。更に，「.」や「/」は 2 個以上入力できない。「-」は第 2 文字目以降として受け付けない。分数として解釈できない入力に対してはその旨表示し，「0」と解釈する。

(D) 提携を入力する時に，入力ミスは受け付けない。例えば，3 人ゲームの時，可能な入力は「A, a, B, b, C, c」であるのでそれ以外は受け付けない。

次に，これらをどのように実現したかについて述べる。表 1 に作成した主なパブリック・クラスをまとめてある。各パブリック・クラス内では多くのプライベート・クラスを利用しているが，その説明は省略する。

(A)に関してはツリーの利用法に焦点を当てて，次節で少し詳しく述べる。

(B)の分数を扱う部分が Frac クラスである。このクラスでは，分数を，既約分数として，その分子と分母を保持し，加減乗除最大最小の計算をし，入力文字列から分数を求め，出力として表示用の文字列を作成する。ここで工夫した点は分子と分母を保持するために BigInteger クラスを使用したこと，Frac クラスに保持されている分数を文字列として出力する際に分数表示，小数表示，帯分数表示出来るようにしたことである。分子と分母を保持するために long を利用することも考えられるが，分数の累乗をすればすぐに桁あふれが生じるため，BigInteger クラスを利用した。BigInteger クラスは Java にあらかじめ定義されている整数を扱うクラスで，記憶すべき数値を正確に保持するためにその都度必要な桁数を確保する。

(C)を実現するのが FracField クラスである。このクラスは Java にあらかじ

表1．作成したおもなクラス

クラス名	内　　容
ApplicationCooperativeGame	main 関数を含む
CharacteristicFunction	「特性関数の入力」タブの実体である JPanel クラスを派生したもの
CharField	提携の入力用テキストフィールド
CooperativeGameFrame	「協力ゲーム」のメインウィンドウ
Frac	分数
FracField	Frac の入力用テキストフィールド
LinearProgramming	線形計画問題を解くクラス
MyAboutDialog	バージョン情報を表示するダイアログボックス
MyComboBox	プレイヤーの総数を設定するコンボボックス

め定義されている1行のテキストを入力するための JTextField クラスを派生させ，(C)で述べた機能を持たせたものである。

(D)を実現するのが CharField クラスである。このクラスは FracField クラスと同様に JTextField クラスを派生させ，入力された1文字をチェックし，不適切なものは受け付けないようにしたものである。

表1にある他のクラスについて，簡単に説明する。ApplicationCooperativeGame クラスにはこのプログラムの最初の実行ポイントである main 関数が定義されている。この main 関数が最終的に図1のメインウィンドウ「協力ゲーム」を表示する。

CharacteristicFunction クラスは図1のメインウィンドウ「協力ゲーム」の「特性関数の入力」タブの実体（JPanel を派生したクラス）である。このクラスでゲームのデータの入力，保持，各値の計算を行う。上方にあるゲームのプレイヤーの総数「変更」ボタンが押されると，変更後のプレイヤーの総数が現在の値と違う場合にのみツリー構造を変更する。この時，変更前のデータで変更後のデータとして妥当なものだけ残すようにする。すなわち，4人ゲーム

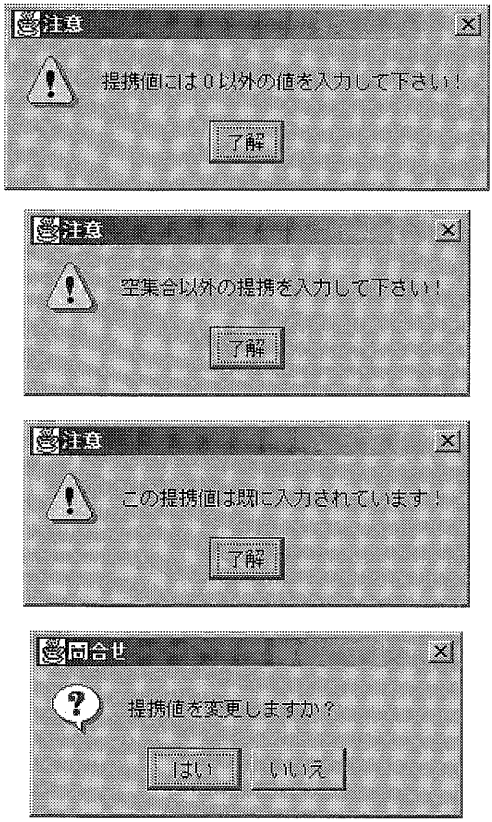

図6．注意と問合せ

を3人ゲームに変更する場合，$v(\mathrm{D})=5$，$v(\mathrm{AD})=10$等のデータは削除するが，$v(\mathrm{AC})=15$，$v(\mathrm{ABC})=30$等のデータは残す。下方にある「追加」ボタンが押された場合，提携と提携値のデータが妥当であるかチェックする。提携値が0である場合，提携が空文字列の場合，同じデータを入力しようと試みた場合，提携値のみ違った値を入力した場合，図6のような注意が表示される。入力されたデータが妥当な場合，ツリーにデータを追加する。ツリーの利用法は次節で述べる。

CooperativeGameFrameクラスは図1のメインウィンドウ「協力ゲーム」

である。メニューと「特性関数の入力」と「値の計算」タブを表示する。「値の計算」タブにあるボタンが押された場合，CharacteristicFunction クラスに対応する値を計算し結果を返すように指示する。返された結果を「値の結果」タブにあるテキストエリアに表示する。

LinearProgramming クラスは τ-値と仁を計算する時に利用される線形計画問題を解くクラスである。このクラス（を主に利用しているアプリケーション）に関しては別の機会に紹介する。

MyAboutDialog クラスは「ヘルプ」メニューから「バージョン情報…」を選んだ時に，表示される図7のダイアログボックスである。

図7．バージョン情報

5．ツリーの利用について

この節では，「特性関数の入力」タブの実体である CharacteristicFunction クラスにおけるツリーの利用について説明する。便宜のため，図1を図8として再掲する。

CharacteristicFunction クラスのユーザーインターフェースの中央部に配置

してあるのがツリー JTree である。JTree は Java にあらかじめ準備されているツリーである。このツリーのルートノードに「n 人ゲーム」と表示し，ルートノードに n 個の子ノードを追加し「？人提携」と表示させる。入力データがあれば，対応する「？人提携」ノードの子として新たなノードを追加し，「$v(??)=??$」と表示する。

まず，ルート用のノード rootnode として DefaultMutableTreeNode クラスのオブジェクトを準備し，これを引数として JTree のコンストラクタを呼び出しておく。DefaultMutableTreeNode クラスは Java にあらかじめ準備されているクラスである。親ノードに追加する（子）ノードも DefaultMutableTreeNode クラスのオブジェクトである。これらのノードにはデータとして任意のオブジェクトを保持できる。これを利用して，rootnode には文字列「n 人ゲーム」を，その子ノードには文字列「？人提携」を，その子の提携値のデータを扱うノードにはそのデータ（CoalitionWorth クラスのオブジェクト）を保持させる。ノードに保持されているデータと表示内容は分離されており，ツリーセルレンダラーというものを介して，保持オブジェクトを適切に表示する。

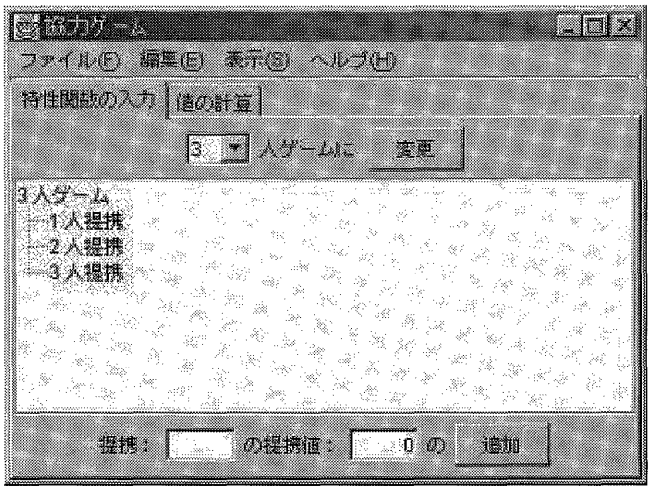

図 8．「特性関数の入力」ダブ（再掲）

文字列オブジェクトの場合はその文字列を表示し，CoalitionWorth オブジェクトの場合は，その内容から「$v(??)=??$」の文字列を生成して表示する。

6．補　　遺

この節では第2節の「協力ゲーム」の解の計算方法で残しておいた上界ベクトル，ギャップ関数，譲歩ベクトルの関係を導出する。

ゲーム (N,v) においてダミープレイヤーの集合を D とし，$M:=N-D$ とおく。この時，

$$v(S)=v(S\cap M)+\sum_{j\in S\cap D}v(\{j\})\ (\forall S\subset N)$$

が成立する。

(N,v) に対する上界ベクトル，ギャップ関数，譲歩ベクトルは次のように定義されている：

$$b_j:=v(N)-v(N-\{j\})$$

$$g(S):=\sum_{j\in S}b_j-v(S)$$

$$\lambda_j:=\min\{g(S)\mid j\in S\subset N\}$$

また，

$$v^{\varepsilon}(S):=\begin{cases}v(M) & S=M\\ (1-\varepsilon)v(S)+\varepsilon\sum_{j\in S}v(\{j\}) & S\subsetneqq M\end{cases}$$

で与えられる (M,v^{ε}) に対する上界ベクトル，ギャップ関数，譲歩ベクトルは次のように定義されている：

$$b_j^\varepsilon := v^\varepsilon(M) - v^\varepsilon(M-\{j\}) \ (j\in M)$$

$$g^\varepsilon(S) := \sum_{j\in S} b_j^\varepsilon - v^\varepsilon(S)$$

$$\lambda_j^\varepsilon := \min\{g^\varepsilon(S) \mid j\in S\subset M\}$$

さらに，ゲーム (N,v) に対しベクトル $\tilde{b}^\varepsilon$ と $\tilde{\lambda}^\varepsilon$，関数 $\tilde{g}^\varepsilon$ を次のように定義する：

$$\tilde{b}_j^\varepsilon := \begin{cases} b_j^\varepsilon & (j\in M) \\ v(\{j\}) & (j\in D) \end{cases}$$

$$\tilde{g}^\varepsilon(S) := \sum_{j\in S} \tilde{b}^\varepsilon - \tilde{v}^\varepsilon(S) \ (\forall S\subset N)$$

$$\tilde{\lambda}_j^\varepsilon := \min\{\tilde{g}^\varepsilon(S) \mid j\in S\subset N\} \ (\forall j\in N)$$

ただし，

$$\tilde{v}^\varepsilon(S) := \begin{cases} v(N) & (S=N) \\ (1-\varepsilon)v(S) + \varepsilon\sum_{j\in S} v(\{j\}) & (S\subsetneqq N) \end{cases}$$

この時，次の命題が成り立つ。

命題：次のことが成立する：

$$\tilde{g}^\varepsilon(N) = g^\varepsilon(M)$$
$$= g(M) + \varepsilon\left[\sum_{j\in M} v(\{j\}) + \sum_{j\in M} v(M-\{j\}) - |M|\sum_{j\in M} v(\{j\})\right]$$

$$\tilde{g}^\varepsilon(S) = g^\varepsilon(S\cap M) \qquad (S\subsetneqq N, S\neq M)$$
$$= g(S\cap M)$$
$$+ \varepsilon\left[v(S\cap M) + \sum_{j\in S\cap M} v(M-\{j\}) - |S\cap M|\sum_{j\in M} v(\{j\})\right]$$

$g^{\varepsilon}(S)\geq 0(\forall S\subsetneqq M)$ ならば，$\tilde{\lambda}_j^{\varepsilon}=\begin{cases}\lambda_j^{\varepsilon} & (j\in M)\\ 0 & (j\in D)\end{cases}$

証 明：まず，$j\in M$ の時，

$$b_j=v(N)-v(N-\{j\})$$
$$=v(M)+\sum_{k\in D}v(\{k\})-v(M-\{j\})-\sum_{k\in D}v(\{k\})$$
$$=v(M)-v(M-\{j\})$$
$$b_j^{\varepsilon}=v(M)-(1-\varepsilon)v(M-\{j\})-\varepsilon\sum_{k\in M-\{j\}}v(\{k\})$$
$$=b_j+\varepsilon\left[v(M-\{j\})+v(\{j\})-\sum_{k\in M}v(\{k\})\right]$$
$$\tilde{g}^{\varepsilon}(N)=\sum_{j\in N}\tilde{b}_j^{\varepsilon}-\tilde{v}^{\varepsilon}(N)$$
$$=\sum_{j\in M}b_j^{\varepsilon}+\sum_{j\in D}v(\{j\})-v(N)$$
$$=\sum_{j\in M}b_j^{\varepsilon}-v(M)$$
$$=g^{\varepsilon}(M)$$
$$=\sum_{j\in M}b_j-v(M)$$
$$+\varepsilon\left[\sum_{j\in M}v(\{j\})+\sum_{j\in M}v(M-\{j\})-|M|\sum_{k\in M}v(\{k\})\right]$$
$$=g(M)$$
$$+\varepsilon\left[\sum_{j\in M}v(\{j\})+\sum_{j\in M}v(M-\{j\})-|M|\sum_{j\in M}v(\{j\})\right]$$

である。次に，$S\subsetneqq N$, $S\neq M$ の時，

$$\tilde{g}^{\varepsilon}(S)=\sum_{j\in S}\tilde{b}_j^{\varepsilon}-\tilde{v}^{\varepsilon}(S)$$

$$=\sum_{j\in S\cap M}b_j^{\varepsilon}+\sum_{j\in S\cap D}v(\{j\})$$

$$-(1-\varepsilon)\left[v(S\cap M)+\sum_{j\in S\cap D}v(\{j\})\right]-\varepsilon\sum_{j\in S}v(\{j\})$$

$$=\sum_{j\in S\cap M}b_j^{\varepsilon}-v^{\varepsilon}(S\cap M)$$

$$=g^{\varepsilon}(S\cap M)$$

$$=\sum_{j\in S\cap M}b_j^{\varepsilon}-v(S\cap M)+\varepsilon\left[v(S\cap M)-\sum_{j\in S\cap M}v(\{j\})\right]$$

ここで，$j\in S\cap M$の時，

$$b_j^{\varepsilon}=v^{\varepsilon}(M)-v^{\varepsilon}(M-\{j\})$$

$$=v(M)-(1-\varepsilon)v(M-\{j\})-\varepsilon\sum_{k\in M-\{j\}}v(\{k\})$$

$$=v(M)-v(M-\{j\})+\varepsilon\left[v(M-\{j\})+v(\{j\})-\sum_{k\in M}v(\{k\})\right]$$

$$=b_j+\varepsilon\left[v(M-\{j\})+v(\{j\})-\sum_{j\in M}v(\{j\})\right]$$

従って，

$$\tilde{g}^{\varepsilon}(S)=g^{\varepsilon}(S\cap M)$$

$$=\sum_{j\in S\cap M}b_j-v(S\cap M)$$

$$+\varepsilon\left[v(S\cap M)+\sum_{j\in S\cap M}v(M-\{j\})-|S\cap M|\sum_{j\in M}v(\{j\})\right]$$

$$=g(S\cap M)$$
$$+\varepsilon\left[v(S\cap M)+\sum_{j\in S\cap M}v(M-\{j\})-|S\cap M|\sum_{j\in M}v(\{j\})\right]$$

となる。

$\tilde{g}^{\varepsilon}$とg^{ε}の関係と$g^{\varepsilon}(S)\geq 0(\forall S\subsetneqq M)$であることに注意すれば，$\lambda$に関する主張は明らかである。 （証明終わり）

この命題の系として，次のことが明らかに成立する。

系：

$$\varepsilon^{*}:=\min\left\{\varepsilon\mid \tilde{g}^{\varepsilon}(N)\geq 0,\ \tilde{g}^{\varepsilon}(S)\geq 0\ (\forall S\subsetneqq M)\right\}$$

$$\sum_{j\in N}\tilde{\lambda}_{j}^{\varepsilon^{*}}\geq\tilde{g}^{\varepsilon}(N)$$

ならば，このε^{*}は(4)を満たし，$\tau_j(N,v)=\tau_j(M,v^{\varepsilon^{*}})\ (j\in M)$となる。

プログラム「協力ゲーム」においても，準平衡ゲームではないゲームのτ-値を求める時に，線形計画問題を解く前にこの系の適用可能性を調べ，可能ならば適用している。

7．おわりに

本稿では授業を補助するプログラム「協力ゲーム」を紹介した。譲渡可能効用を持つ提携形ゲームの6つの解を計算するプログラムである。今のところ，プレイヤーの総数として最大で（「A」から「Z」の）31人まで可能である。今後の発展として，(1)他の解（例えば，任意の重みmを持った最小二乗値）も計算できるようにすること，(2)入力として，特性関数ではなく，他の適切に定義された問題を扱えるようにし，その問題から特性関数を求めるようにすること，等が考えられる。

参考文献

[1] 行方常幸「授業を補助するプログラム (1) —行列の演算—」小樽商科大学商学討究　第52巻　第2・3合併号，2001。

[2] 行方常幸，行方洋子「はじめてのゲーム理論　—ゲーム理論と人間の繋がり—」エフ・コピント富士書院，1995。

[3] Theo S.H. Driessen : Cooperative Games, Solutions and Application. Kluwer Academic Publishers, 1988.

[4] ミッシェル・マニング著／玉川竜司訳「JBuilder で学ぶ Java」プレンティスホール，1998。

[5] 掌田津耶乃「JBuilder ではじめる Java プログラミング入門」秀和システム，2001。

[6] Mary Campione, Kathy Walrath, Alison Huml : The Java Tutorial, Third Edition - A Short Course on the Basics. Addison Wesley, 2001.

[7] Kathy Walrath, Mary Campione : The JFC Swing Tutorial - A Guide to Constructing GUIs. Addison Wesley, 2000.

[8] Campione, Walrath, Huml, Tutorial Team : The Java Tutorial Continued - The Rest of the JDK. Addison Wesley, 1998.