解説

離散最適化とその応用

第2回　アルゴリズムの高速化，投票力指数に関して

宇野　毅明

1. はじめに

本連載では，ゲーム理論や分配問題への離散最適化の応用を解説しているが，今回は，ゲーム理論へのアルゴリズム的な研究という切り口で解説をしたいと思う．アルゴリズムは，およそ計算と名のつく事柄すべてに関わっている．特に複雑なシステムに関わる計算は，単純な方法では多くの時間を必要とするため，アルゴリズム的な研究の必要性が高い．ゲーム理論でも様々なアルゴリズム的研究が行われているが，今回は重み付き多数決ゲームの投票力指数計算アルゴリズムについて解説をする．

ある議会を考えよう．この議会ではn党の政党がそれぞれ議席をw_i席持っている．この議会ではq人の賛成があれば法案が通るとする．このような状況で，各政党をプレイヤーとみなし，その構造をモデル化したものが重み付き多数決ゲームである．投票力指数は，ゲーム内でのプレイヤーの力の指標である．

一般的に，政党の強さは議席数で測られることが多い．これは，政党の議席数の推移など，数自体を調べる目的には合っているが，法案を通すための力が大きいか小さいかを測るには，少々モデルとして合わない部分がある．例えば，政党A，B，C，D，Eがあり，それぞれ議席を40，24，12，12，12有するとする．法案成立に必要な票数qを51，つまり過半数とする(図1)．ここで，政党Aが法案を提案するとしよう．法案を成立させるためには，政党Aは政党B, C, D, Eいずれか1党の協力を得ればよい．そうすれば，過半数である51票を確保できるからである．さてここで選挙があり，Aは2，Bは13，議席を増やし，C, D, Eはそれぞれ5議席失ったとしよう．議席数はそれぞれ42，37，7，7，7になる．この状況では，Aが法案を通すためには，Bの協力か，C, D, Eのうち2党の協力が必要になった．つまり，Aは相変わらず第一党であり，かつ議席数も増えたにもかかわらず，法案を通す，という観点からは状況が悪くなっているのである．

このような，各政党が自分の法案を通すにはどの程度苦労するか，という観点から指標をモデル化するには，どのような政党の組合せ（提携と呼ぶ）が，q以上の議席数を確保できるか，という組合せ的な要因を考慮して指数化するのが良い．そのような指数がいくつか研究されているが，ここでは代表的な3指数，**バンザフ**（Banzhaf）**指数**，**シャープレイ・シュービック**（Shapley-Shubik）**指数**，**ディーガン・パックル**（Deegan-Packel）**指数**の紹介をしよう．以後プレイヤーp_iの各指数を$Bz(p_i), Ss(p_i), Dp(p_i)$と表記する．

ここでもう一度，用語の定義を行おう．ゲームのプレイヤー集合を$N=\{p_1, \cdots, p_n\}$とし，プレイヤーp_iの**重み**をw_iとする．プレイヤーの重みと定数q（法案を通すために必要な票数）が与えられると，**重み付き多数決ゲーム**が一つ定義される．プレイヤーの集合を**提携**と呼び，提携Sの重み$w(S)$をSに属するプレイヤーの重みの総和とする．空集合と全体集合も提携と呼ぶ．重みがqと同じかそれより大きい提携を**勝利提携**と呼び，そうでない提携を**敗北提携**と呼ぶ．

重み付き多数決ゲームは，**投票システム**と呼ばれる，より一般的なシステムの特殊ケースである．投票システムとは，プレイヤー集合Nと勝利提携集合Wの組(N, W)で定義される．重み付き多数決ゲームは，勝利提携集合Wが，重みがq以上の提携の集合であるような投票システムである．

1.1 バンザフ指数

勝利提携Sからプレイヤーp_iが抜けることによってSが敗北提携になるとき，p_iはSの**スウィング**であるという．p_iがSのスウィングであれば，p_iはある種の発言力を持つと考えられるだろう．ここで，全

うの　たけあき
国立情報学研究所
〒101-8430 東京都千代田区一ツ橋2-1-2

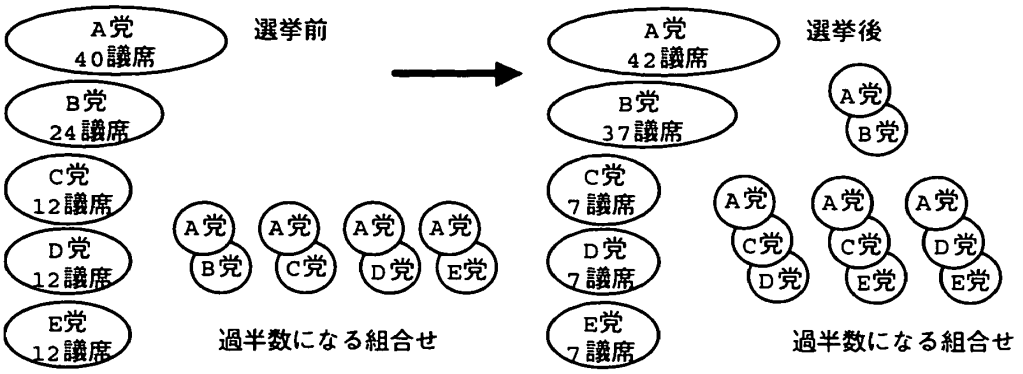

図1　選挙後，A党の議席は増えたが，過半数を取れる組合せは減った

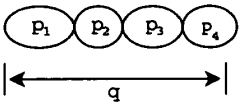

図2　極小勝利提携．各プレイヤーの幅が重みに対応する．どのプレイヤーが抜けても重みが q を下回る

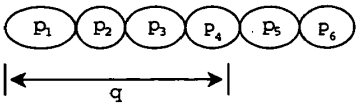

図3　$p_1, \cdots, p_6$ の順で提携に加わる．p_4 が加わった時点で勝利提携になり，この順列のピボットは p_4 になる

ての提携が一様に発生すると仮定すると，プレイヤー p_i がスウィングである確率は

$$|\{S \subseteq N | p_i \in S, q \leq w(S) < q + w_i\}| / 2^n$$

となる．$Bz(p_i)$ はこの確率により定義される[1]．

1.2　ディーガン・パックル指数

提携 S に対し，S から最小重みのプレイヤーを除いた提携を $D(S)$ とする．勝利提携 S から，どのプレイヤー p_i が抜けても敗北提携になる，つまり，$w(D(S)) < q$ であるとき，S は**極小勝利提携**であるという（図2）．プレイヤー達が勝利提携を組むとき，大人数で合意を取るのは手間がかかるので，余分なプレイヤーはメンバーになっていないだろうと考えられる．つまり，発生するのは極小勝利提携のみであるとみなせる．極小勝利提携 S に属するプレイヤーが発言力 $1/|S|$ を持つとし，極小勝利提携が等確率で発生すると仮定すると，プレイヤー p_i の発言力の期待値は，X を極小勝利提携の集合とすると，

$$\sum_{p_i \in S,\, q \leq w(S),\, w(D(S)) < q} 1 / |S| |X|$$

となる．これが $Dp(p_i)$ の定義である[3]．

1.3　シャープレイ・シュービック指数

勝利提携が形成されるとき，その勝利提携は一瞬にして発生するのではなく，徐々に大きくなっていくと考えられる．つまり，最初に空の提携 S があり，ある順列 π の順にプレイヤーが参加していくと考えられる．最初 S は敗北提携であるが，順々に重みが増し，あるプレイヤー p_i が参加したところで勝利提携に変化する．このとき，プレイヤー p_i を順列 π の**ピボット**と呼ぶ（図3）．ピボットは，何らかの意味で発言力を持つと考えられるので，全ての順列が一様に発生すると，p_i が発言力を持つ確率は

$$|\{\pi | \pi \text{ は順列}, p_i \text{ は } \pi \text{ のピボット}\}| / n!$$

となる．これが $Ss(p_i)$ の定義である[7]．

バンザフ指数は Banzhaf により 1965 年[1]に，シャープレイ・シュービック指数は Shapley と Shubik により 1953 年[7]に，ディーガン・パックル指数は Deegan と Packel により 1978 年[3]に提唱された．バンザフ指数，シャープレイ・シュービック指数に関しては公理系からの指数の導出や，性質・構造の解析など，多種多様な研究が行われてきている．例えば，$w_i \leq w_j$ であれば必ず $Ss(p_i) \leq Ss(p_j)$, $Bz(p_i) \leq Bz(p_j)$ が成り立つ，つまり議席数に対する指数の逆転は起きない，ということが知られている[8]．ディーガン・パックル指数に関しても，前者ほどではないが，研究がある．この他にもいくつかの指数が提唱されているが，これら3指数ほどの深い研究は行われていないようである．

2.　単純な指数計算アルゴリズム

前節で定義した3つの指数を計算する最も単純な方法は，定義式通りに計算を行う方法である．シャープレイ・シュービック指数は，大きさ n の全ての順列を発生させ，それらのピボットを見つけて以下のように計算できる．アルゴリズム実行前，任意の i につい

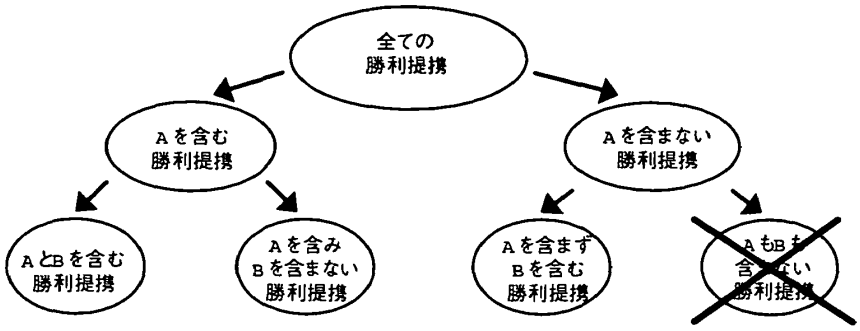

図4　問題を分枝して勝利提携を列挙する．勝利提携が含まれない子問題（×印）は探索しない

て $s[i]=0$ であるとする．

(Ss) 全ての大きさ n の順列のピボット p_i に対して
$s[i]=s[i]+1/n!$ とする

アルゴリズム実行後，各 i に対して $Ss(p_i)=s[i]$ となる．計算時間は $O(n\times n!)$ である．バンザフ指数，ディーガン・パックル指数も，以下のように全ての提携を発生させて計算できる．

(Bz) 全ての提携 S と $p_i\in S$ に対し，
$q\leq w(S)<q+w_i$ なら $s[i]=s[i]+1/2^n$

(Dp) 全ての提携 S と $p_i\in S$ に対し，
S が極小勝利提携なら $s[i]=s[i]+1/|X|$

これらの計算時間は $O(n\times 2^n)$ である．これらは，いわゆるナイーブなアルゴリズムであり，利用している指数の性質は，定義に用いられる集合族の大きさが有限である，というものだけである．「有限時間で計算できるかどうか」という問いに答えているにすぎない．そこで次に，指数の性質をより深く利用したアルゴリズムを解説しよう．

3.　指数計算アルゴリズムの高速化

シャープレイ・シュービック指数の計算では，全ての順列を発生させている，という部分が計算上のボトルネックになっている．そこで，

・p_i がピボットであり，p_i が加わる前の提携が S であるような順列は $|S|!(n-|S|-1)!$ 個存在する（このとき p_i は $S\cup\{p_i\}$ のスウィングである）．

という性質に着目しよう．これは，p_i がピボットであれば，p_i の前，および後ろのプレイヤー同士の順番を入れ替えても，p_i はピボットになっているということである．これより，全ての提携 S を発生させれば，p_i がピボットとなる順列の数がわかる．そこで，スウィングが p_i である全ての提携 S について，$|S|!(n-|S|-1)!$ の総和をとる，という方法で $Ss(p_i)$ が計算できる．

(Ss 2) 全ての提携 S と $p_i\in S$ に対し，$q\leq w(S)<q+w_i$ なら $s[i]=s[i]+(|S|-1)!(n-|S|)!/n!$

計算時間は $O(n\times 2^n)$ に高速化される．

さらに，計算に実際に必要な提携は，ディーガン・パックル指数では極小勝利提携，バンザフ，シャープレイ・シュービック指数ではスウィングを持つ勝利提携だけであることから，以下の提携列挙アルゴリズムを用いて計算時間の短縮ができる．ここでは，プレイヤーはその重みの降順でソートされている，つまり任意の i に対して $w_i\geq w_{i+1}$ であるとする．また，$i\leq j$ に対して $N_{ij}=\{p_i,\cdots,p_j\}$ とする．初期入力を $S=\phi, i=1$ とする．

極小勝利提携列挙（提携 S，添え字 i）

1. $w(N_{ij})+w(S)\geq q$ となる最小の j を見つけ，$S\cup N_{ij}$ を出力
2. $w(N_{in})+w(S)-w_{h+1}\geq q, h+1\leq j$ なる h について，$h\leq n-2$ ならば**極小勝利提携列挙**$(S\cup N_{ih}, h+2)$ を再帰呼び出し

スウィング勝利提携列挙（提携 S，添え字 i）

1. $w(N_{ij})+w(S)\leq q+w_k$ となる最大の j を見つける（p_k は $S\cup N_{ij}$ の最小添字プレイヤー）
2. $q\leq w(N_{ih})+w(S), h\leq j$ となる各 h について，$S\cup N_{ih}$ を出力
3. $w(N_{in})+w(S)-w_{h+1}\geq q, h+1\leq j$ なる各 h について，$h\leq n-2$ ならば**スウィング勝利提携列挙**$(S\cup N_{ih}, h+2)$ を再帰呼び出し

これらのアルゴリズムは分割法と呼ばれる列挙法を用いている．分割法は，図4で示すように，元の問題を二つの非空な問題に分割して再帰的に列挙する方法である．両アルゴリズムともに，出力する提携一つあたりに必要な計算時間は $O(n)$ であるので，計算時間は（必要な勝利提携数）$\times n$ となる．スウィングを持

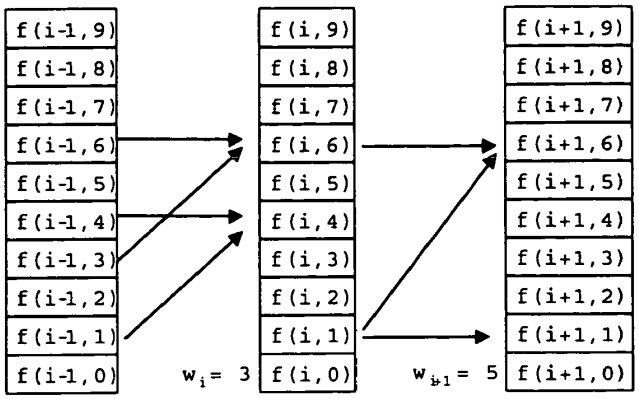

図5　関数値を $i=1,2,\cdots$ の順番で計算．矢印は，$f(i,6)$ は $f(i-1,6)$ と $f(i-1,3)$，$f(i+1,6)$ は $f(i,6)$ と $f(i,1)$ から計算されることを表す

つ勝利提携数，極小勝利提携数ともに，最悪の場合 $\Theta(2^n)$ となるので計算量の意味での改善はないが，実際の計算時間の減少が期待できる．極小勝利提携の列挙に関しては松井・松井[5]に解説がある．

さらに，両列挙アルゴリズムで，j,h の発見に2分探索を用いると，提携一つ当たりの計算時間を $O(\log n)$ に減少できる．$s[i]$ への数値の足しこみに，1反復で $O(n)$ かかっている部分を，区間木を用いた加算を用いることで $O(\log n)$ に減少すれば，指数の計算時間を（必要な提携数）$\times O(\log n)$ まで減少できる．

4．動的計画法とその高速化

投票力指数計算に対しては，上述のアルゴリズムとは異なる観点に着目して設計された，動的計画法も提案されている[2, 4]．バンザフ指数計算を例にして解説しよう．まず，以下の関数を定義する．

$$f_{Bz}(i,x)=|\{S|S\subseteq N_{1i}, w(S)=x\}|$$

この関数を用いると，

$$Bz(p_n)\times 2^n=\sum_{q-w_n\le x<q} f_{Bz}(n-1,x)$$

となる．$f_{Bz}(i,x)$ は

$$f_{Bz}(i,x)=f_{Bz}(i-1,x)+f_{Bz}(i-1,x-w_i)$$

という再帰式を満たすので，全ての $0\le i\le n-1, 0\le x<q$ に対する f_{Bz} の関数値を $i=1,2,\cdots,n-1$ の順に動的計画法を用いて計算できる（図5）．$f_{Bz}(i,x)>0$ となるような i,x の組合せは，最悪で 2^n 個存在するので，この方法での計算量の減少は，そのままでは見込めない．そこで，プレイヤー重みの整数性を仮定しよう．議会などの現実問題ではプレイヤーの重みはだいたいが整数であるので，この仮定はさほど一般性を失わない．これより，x が整数であるときのみ $f_{Bz}(i,x)>0$ となるので，$O(qn)$ 時間で $Bz(p_n)$ が計算できる．$i\neq n$ であるような $Bz(p_i)$ を計算するには，p_i と p_n の添え字を入れ替えればよい．シャープレイ・シュービック指数に関しても，

$$f_{Ss}(i,k,x)=|\{S|S\subseteq N_{1i}, |S|=k, w(S)=x\}|$$

と関数を定義すると，

$$Ss(p_n)\times n!=\sum_{q-w_n\le x<q,\ k\le n} f_{Ss}(n-1,k,x)$$

となり，重みの整数性を仮定すると $O(n^2q)$ 時間で計算できる．ディーガン・パックル指数についても同じような動的計画法が提案されているが[5]，ここでは割愛させていただく．

空間計算量，つまり上述の動的計画法アルゴリズムのメモリ使用量についても言及しておこう．アルゴリズム実行中，全ての関数値をメモリに保持すると，それぞれ $\Theta(nq), \Theta(n^2q)$ のメモリを使用する．しかし，よく観察すると，引数 $i=j$ なる関数値を計算する際には，$i=j-1$ なる関数値のみが参照されている．そこで，$i<j-1$ なる関数値は破棄しても差し支えないことがわかる．これより，メモリ使用量はそれぞれ $\Theta(q), \Theta(nq)$ ですむ．

上述の動的計画法では，全ての引数の組合せに対して関数値の計算を行うと，それぞれちょうど $\Theta(nq)$, $\Theta(n^2q)$ 時間を消費する．しかし，実際計算に必要な関数値は非ゼロなものだけである．そこで，引数 $i=j-1$ である関数値の中で非ゼロのものだけを，引数 x の大きさ順に連結リストを用いて保持しよう．すると，図6のように二つのポインタでリストをなぞることにより，引数 $i=j$ である全ての関数値を，効率良く計算できる[9]．これにより，関数値がゼロであるような引数に対しては計算を省略できる．

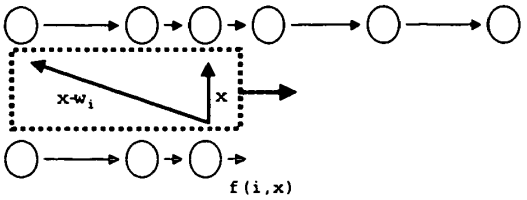

図6　非ゼロな関数値のリスト．x と $x-w_i$ を指すポインタをずらし，どちらかのポインタが非ゼロ要素にぶつかると，$f(i,x)$ の次の非ゼロ関数値が見つかる

次に，この動的計画法の高速化を考えよう．とはいえ，プレイヤー1人の指数を計算する動的計画法には無駄な部分は少なく，根本的に新しいアイディアがなければ計算量の改善は見込めない．現在のところ，そ

のような新しいアイディアは発見されていない．そこで少々視点を変え，全プレイヤーの指数計算を高速化する，という方法を考えよう．上記動的計画法は，全プレイヤーの指数計算をする際には，同じような問題を何回も解くことになり，ここには大いなる無駄があるので，そこを省略する方法を考える．

まず簡単に思いつく方法が，図7のような分割統治法的なアプローチである．以下では，プレイヤーの添え字の小数部分は切り捨ててあるとする．上述の動的計画法は，プレイヤー $p_{n/2+1}, \cdots, p_n$ の指数計算を行うとき，アルゴリズム中のループの前半，$i=n/2$ となるまではまったく同じ計算を行う．つまり，計算した関数値（f と書く）は再利用できるのである．プレイヤー $p_1, \cdots, p_{n/2}$ に関しては，プレイヤーの添え字を逆順に付け直したとみなして動的計画法を適用すると，やはりループの前半でまったく同じ計算を行うので，計算した関数値を再利用できる．

この方法を再帰的に適用して，関数値の計算を省略する方法を考えよう．今，引数 $i=n/2$ である f の関数値が求まっていて，プレイヤー $p_{n/2+1}, \cdots, p_n$ の指数計算を行うとしよう．このとき，プレイヤー $p_{3n/4+1}, \cdots, p_n$ の指数計算は，$i=n/2+1, \cdots, 3n/4$ までまったく同じ計算を行うので，関数値の再利用ができる．プレイヤー $p_{n/2+1}, \cdots, p_{3n/4}$ に関しても，プレイヤー $p_{n/2+1}, \cdots, p_n$ の添え字のみ逆順にすれば，同じく前半部分の計算結果を再利用できる．このようにして再帰的にプレイヤーを分割し，添え字を入れ替えながら計算を行えば，計算のかなりの部分を再利用できる．この方法でアルゴリズムを記述すると，以下のようになる．

指数計算分割統治 ($f(j-1, *)$ の関数値, j, j')

1. **if** $j=j'$ **then** p_j の指数を計算し終了
2. 引数 i が j から $(j+j')/2$ までの f の関数値を逐次計算し，**指数計算分割統治** ($f((j+j')/2, *)$ の関数値, $\lceil (j+j')/2+1 \rceil, j'$) を再帰呼び出し
3. j から j' までの添え字を逆順に付け直す
4. 引数 i が j から $(j+j')/2$ までの f の関数値を逐次計算し，**指数計算分割統治** ($f((j+j')/2, *)$ の関数値, $\lfloor (j+j')/2+1 \rfloor, j'$) を再帰呼び出し

このアルゴリズムは，最初に全ての $f(0, *)$ の値，$j=1$，$j=n$ を引数として呼び出す．バンザフ指数，シャープレイ・シュービック指数どちらにも適用できる．1回再帰を行うと，着目するプレイヤー数は半分になるので，再帰の深さはたかだか $\log_2 n$ となる．再帰の各レベルでの計算時間の総和は，もとのアルゴリズムのループ n 回り分に相当する．よって，全プレイヤーの指数計算は，バンザフ指数では $O(qn^2)$ から $O(qn \log n)$，シャープレイ・シュービック指数では $O(qn^3)$ から $O(qn^2 \log n)$ に減少される．

分割統治法的なアイディアに基づかない改良も提案されている[9]．最初に説明した動的計画法は，プレイヤー p_i の指数計算では，プレイヤー p_i とプレイヤー p_n を入れ替えて再計算する．その結果，p_{i+1} から p_{n-1} の部分の関数値をすべて計算しなおすことになる．この計算を他の何らかの方法で共通化できれば，高速化が行えるだろう．この発想に基づいた改良を，まずはバンザフ指数の場合について説明する．

プレイヤー p_i 以外のプレイヤーを i より添え字が小さいプレイヤーと，そうでないプレイヤーに分けて考え，関数 $g_{Bz}(i, x)$ を

$$|\{S | p_i \in S, w(S \cap N_{1(i-1)}) = x, q \leq w(S) < q + w_i\}|$$

で定義する．この関数は，「添え字が i より小さいプレイヤーの重み和が x である勝利提携で，p_i がスウィングになっているものの数」である．この関数を用

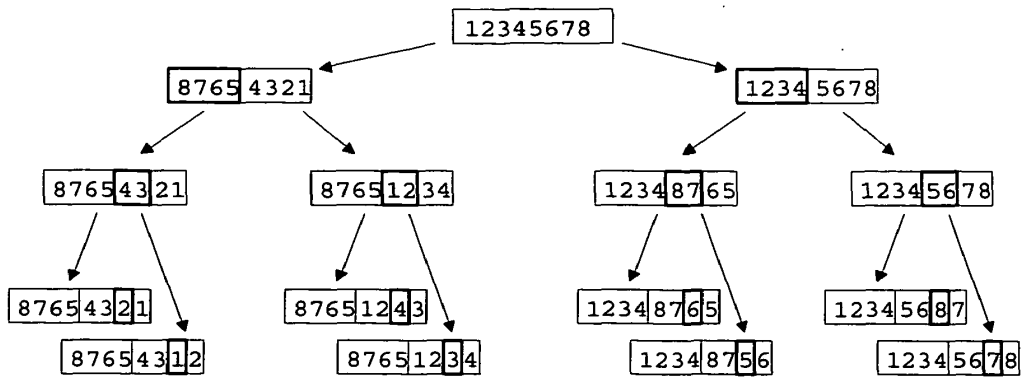

図7　分割統治の様子．各反復で太線で囲まれたところのみ計算する．灰色の部分は上位の反復で計算済み

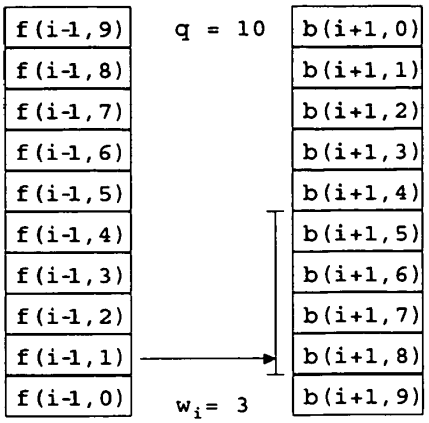

図8　$Bz(p_i)$の指数計算をfとbを使って行う．$w_i=3$より，$g(i,1)$は縦線区間の和と$f(i-1,1)$の積になる

いると，

$$Bz(p_i)\times 2^n=\sum_{0\le x<q} g_{Bz}(i,x)$$

と表される．関数g_{Bz}の効率の良い計算のために

$$b_{Bz}(i,x)=|\{S|S\subseteq N_{in},\ w(S)=x\}|$$

$$h_{Bz}(i,x)=\sum_{0\le y\le x} b_{Bz}(i,y)$$

と定義する．$b_{Bz}(i,x)$は，「添え字がi以上のプレイヤーの提携で，重みがxになるものの数」である．関数f_{Bz}とb_{Bz}は対称なので，b_{Bz}の関数値はf_{Bz}と同じ方法で計算できる．h_{Bz}はb_{Bz}より簡単に計算できる．これらの関数を用いると，

$$\begin{aligned} g_{Bz}(i,x)&=f_{Bz}(i-1,x)\times\sum_{q-w_i-x\le y<q-x} b_{Bz}(i+1,y)\\ &=f_{Bz}(i-1,x)\times(h_{Bz}(i+1,q-1-x)\\ &\quad -h_{Bz}(i+1,q-1-x-w_i)) \end{aligned}$$

と表現される．よって，g_{Bz}の全ての関数値は$O(qn)$時間で計算でき，全プレイヤーのバンザフ指数は$O(qn)$時間で計算できる．

シャープレイ・シュービック指数，ディーガン・パックル指数に関しても，

$$bs_S(i,k,x)=\sum_{S\subseteq N_{in},\ w(S)=x}(k+|S|)!(n+1-k-|S|)!$$

$$b_{Dp}(i,k,x)=\sum_{q-x\le w(S)<q-x+z(S)} 1/(|S|+k)$$

と定義すると，$Ss(p_i)$, $Dp(p_i)$はそれぞれ

$$\sum_{0\le x\le q-1,\,0\le k\le n}(f_{Ss}(i-1,k,x)\times\sum_{q-w_i-x\le y\le q-1-x} bs_S(i+1,k,y))$$

$$\sum_{0\le x\le q-1,\,0\le k\le n}(f_{Ss}(i-1,k,x)\times b_{Dp}(i,k,x))$$

を用いて，$O(n^2q)$時間で計算できる．

5. 補遺

本稿で取り上げた投票力指数に関する詳しい解説が文献[5, 6]にある．特に文献[6]には，各指数について，公理系からの導出から実際問題への適用例まで詳しい解説がある．バンザフ，シャープレイ・シュービック指数計算を行う動的計画法は，文献[2, 4]で解説されている．また，ディーガン・パックル指数計算を行う動的計画法も含めた解説が文献[5]にある．全プレイヤーの指数計算を高速化する方法は文献[9]で提案されている．スウィングを持つ勝利提携の列挙アルゴリズム，および勝利提携列挙を一つ当たりO(log n)時間で列挙する方法，区間木を用いて足しこみをO(log n)時間で行う方法，分割統治法による動的計画法の改良は，本稿のために新しく考案したものであり，文献はない．しかし証明自体は簡単であるので試していただきたい．

参考文献

[1] J. F. Banzhaf III, "Weighted Voting doesn't work", Rutgers Law Review 19, pp. 317-343, 1965.

[2] S. J. Brams and P. J. Affuso, "Power and size: a new paradox", Theory and Decision 7, pp. 29-56, 1975.

[3] J. Deegan and E. W. Packel, "A New Index of Power for Simple n-person Games", Internatinal Journal of Game Theory 7, pp. 113-123, 1978.

[4] W. F. Lucas, "Measuring Power in Weighted Voting Systems", in S. J. Brams, W. F. Lucas and P. D. Straffin Eds., Political and related models, Springer-Verlag, pp. 183-238, 1983.

[5] T. Matsui and Y. Matsui, "A Survey of Algorithms for Calculating Power Indices of Weighted Majority Games", Journal of the Operations Research Society of Japan 43, pp. 71-86, 2000.

[6] 武藤滋夫, 小野理恵,「投票システムのゲーム分析」, 日科技連, 1998.

[7] L. S. Shapley and M. Shubik, "A Method for Evaluating the Distribution of Power in a Committee System", American Political Science Review 48, pp. 787-792, 1954.

[8] P. D. Straffin, "Power indices in politics", Political and Related Models, Springer-Verlag, pp. 256-321, 1983.

[9] 宇野毅明,「重み付き投票ゲームにおける投票力指数の計算の高速化」, 情報処理学会アルゴリズム研究会 80, 2001.