Sociedad de Estadística e Investigación Operativa
Top (2000) *Vol.* 8, *No.* 2, pp. 191–213

Generating functions for computing power indices efficiently

J. M. Bilbao, J. R. Fernández, A. Jiménez Losada, and J. J. López
Departamento de Matemática Aplicada II, Escuela Superior de Ingenieros
Camino de los Descubrimientos s/n, 41092 Sevilla, Spain
http://www.esi2.us.es/~mbilbao/
e-mail: mbilbao@cica.es

Abstract

The *Shapley-Shubik* power index in a voting situation depends on the number of orderings in which each player is pivotal. The *Banzhaf* power index depends on the number of ways in which each voter can effect a swing. We introduce a combinatorial method based in *generating functions* for computing these power indices efficiently and we study the *time complexity* of the algorithms. We also analyze the meet of two weighted voting games. Finally, we compute the voting power in the Council of Ministers of the European Union with the generating functions algorithms and we present its implementation in the system Mathematica.

Key Words: Power index, generating function, computational complexity

AMS subject classification: 91A12

1 Introduction

The analysis of power is central in political science. In general, it is difficult to define the idea of power, but for the special case of voting power there are mathematical power indices that have been used. The first such power index was proposed by Shapley and Shubik (1954) who apply the Shapley value (1953) to the case of simple games. Another concept for measuring voting power was introduced by Banzhaf (1965), a lawyer, whose work has appeared mainly in law journals, and whose index has been used in arguments in various legal proceedings.

A *cooperative game* is a function $v : 2^N \to \mathbb{R}$, with $v(\emptyset) = 0$. The players are the elements of N and the coalitions are the subsets $S \subseteq N$ of players.

Received: January 1999; Accepted: November 2000

This work has been partially supported by the Spanish Ministery of Science and Technology under grant SEC2000-1243.

A *simple game* is a cooperative game $v : 2^N \to \{0,1\}$, such that $v(N) = 1$ and v is nondecreasing, i.e., $v(S) \subseteq v(T)$ whenever $S \subseteq T \subseteq N$. A coalition is *winning* if $v(S) = 1$, and *losing* if $v(S) = 0$. The collection of all winning coalitions is denoted by $\mathcal{W}$. We will use a shorthand notation and write $S \cup i$ for the set $S \cup \{i\}$. The *Shapley value* for the player $i \in N$ is defined by

$$\Phi_i(v) = \sum_{\{S \subseteq N : i \notin S\}} \frac{s!(n-s-1)!}{n!} \left(v(S \cup i) - v(S)\right),$$

where $n = |N|$, $s = |S|$. This value is an average of *marginal contributions* $v(S \cup i) - v(S)$ of the player i to all coalitions $S \subseteq N \setminus i$. In this value, the sets S of different size get different weight. For simple games, Shapley and Shubik (1954) introduced the following power index, which is a specialization of the Shapley value.

Definition 1.1. The Shapley-Shubik index for the simple game (N, v) is the vector $\Phi(v) = (\Phi_1(v), \ldots, \Phi_n(v))$, given by

$$\Phi_i(v) = \sum_{\{S \notin \mathcal{W} : S \cup i \in \mathcal{W}\}} \frac{s!(n-s-1)!}{n!}.$$

We now define the normalized Banzhaf index. A *swing* for player i is a pair of coalitions $(S \cup i, S)$ such that $S \cup i$ is winning and S is not. For each $i \in N$, we denote by $\eta_i(v)$ the number of swings for i in the game v, and the total number of swings is

$$\overline{\eta}(v) = \sum_{i \in N} \eta_i(v).$$

Definition 1.2. The normalized Banzhaf index is the vector

$$\beta(v) = (\beta_1(v), \ldots, \beta_n(v)), \text{ where } \beta_i(v) = \frac{\eta_i(v)}{\overline{\eta}(v)}.$$

Coleman (1973) considered two indices to measure the power to prevent action and the power to initiate action. In the above notation, these two Coleman indices are

$$\gamma_i(v) = \frac{\eta_i(v)}{\omega}, \quad \gamma_i^*(v) = \frac{\eta_i(v)}{\lambda},$$

where ω and λ are the total number of winning and losing coalitions, respectively. For a comprehensive work on the problem of measuring voting power, see Felsenthal and Machover (1998).

We introduce a special class of simple games called *weighted voting games*. The symbol $[q;\ w_1, w_2, \dots, w_n]$ will be used, where the quota q and the weights $w_1, w_2, \dots, w_n$ are positive integers with $0 < q \leq \sum_{i=1}^{n} w_i$. Here there are n players, w_i is the number of votes of player i, and q is the quota needed for a coalition to win. Then, the above symbol represents

$$v(S) = \begin{cases} 1, & \text{if } w(S) \geq q \\ 0, & \text{if } w(S) < q, \end{cases}$$

where $w(S) = \sum_{i \in S} w_i$.

To be self-contained section 2 recalls the main results on generating functions to obtain Shapley-Shubik and Banzhaf indices. If the input size of the problem is n, then the function which measures the worst case running time for computing the indices is in $O(2^n)$. Section 3 introduces the computational complexity and the algorithms based in the generating functions to obtain these power indices. In sections 4 and 5, we present several algorithms for computing the power indices with the system Mathematica. The paper concludes with some remarks on the complexity of generating functions methods for computing power indices in weighted voting games.

2 Generating functions

In order to obtain the power indices *exactly*, we present a combinatorial method based in the generating functions. The most useful method for counting the number of elements $f(k)$ of a finite set is to obtain its generating function. The *ordinary generating function* of $f(k)$ is the formal power series

$$\sum_{k \geq 0} f(k) x^k.$$

This power series is called formal because we ignore the evaluation on particular values and problems on convergence (see Stanley, 1986). We can work with generating functions of several variables

$$\sum_{k \geq 0} \sum_{j \geq 0} \sum_{l \geq 0} f(k, j, l) x^k x^j x^l.$$

For each $n \in \mathbb{N}$, the number of subsets of k elements of the set $N = \{1, 2, \ldots, n\}$ is given by the explicit formula of the binomial coefficients

$$\binom{n}{k} = \frac{n(n-1)\cdots(n-k+1)}{k!}.$$

A generating function approach to binomial coefficients may be obtained as follows. Let $S = \{x_1, x_2, \ldots, x_n\}$ be an n-element set. Regard the elements $x_1, x_2, \ldots, x_n$ as independent indeterminates. It is an immediate consequence of the process of multiplication (one could also give a proof by induction) that

$$(1+x_1)(1+x_2)\cdots(1+x_n) = \sum_{T \subseteq S} \prod_{x_i \in T} x_i.$$

Note that if $T = \emptyset$ then we obtain 1. If we put each $x_i = x$, we obtain

$$(1+x)^n = \sum_{T \subseteq S} \prod_{x \in T} x = \sum_{T \subseteq S} x^{|T|} = \sum_{k \geq 0} \binom{n}{k} x^k.$$

We now present generating functions for computing the Shapley-Shubik and the Banzhaf power indices in weighted voting games, defined by

$$[q;\ w_1, w_2, \ldots, w_n].$$

David G. Cantor used generating functions for computing exactly the Shapley-Shubik index for large voting games. As related by Mann and Shapley (1962), Cantor's contribution was the following result (see Lucas, 1975, pp. 214–216). The Shapley-Shubik index of the player $i \in N$, satisfies

$$\begin{aligned} \Phi_i(v) &= \sum_{\{S \notin \mathcal{W}:\, S \cup i \in \mathcal{W}\}} \frac{s!(n-s-1)!}{n!} \\ &= \sum_{j=0}^{n-1} \frac{j!(n-j-1)!}{n!} \left(\sum_{k=q-w_i}^{q-1} A^i(k,j) \right), \end{aligned}$$

where $A^i(k,j)$ is the number of ways in which j players, other than i, can have a sum of weights equal to k.

Proposition 2.1. *(Cantor) Let $[q;\ w_1, w_2, \dots, w_n]$ be a weighted voting game. Then the generating function of the number $A^i(k,j)$ of coalitions S of j players with $i \notin S$ and $w(S) = k$, is given by*

$$ShG_i(x,z) = \prod_{j \neq i} (1 + z\, x^{w_j}) .$$

Proof. Let $W = \{w_1, w_2, \dots, w_n\}$ be the set of the weights of all the players. We consider the following generating function

$$\begin{aligned}(1 + z\, x^{w_1}) \cdots (1 + z\, x^{w_n}) &= \sum_{T \subseteq W} \left(z^{|T|}\, x^{\sum_{w_i \in T} w_i} \right) \\ &= \sum_{k \geq 0} \sum_{j \geq 0} A(k,j)\, x^k z^j ,\end{aligned}$$

where the coefficient $A(k,j)$ is the number of coalitions of weight k and size j. To obtain the numbers $A^i(k,j)$, we drop the factor $(1 + z\, x^{w_i})$. □

The above approach was applied by Brams and Affuso (1976) for computing the normalized Banzhaf index. The number of swings for the player i satisfies

$$\begin{aligned}\eta_i(v) &= |\{S \notin \mathcal{W} : S \cup i \in \mathcal{W}\}| \\ &= \sum_{k=q-w_i}^{q-1} b^i(k),\end{aligned}$$

where $b^i(k)$ is the number of coalitions that do not include i with weight k.

Proposition 2.2. *(Brams-Affuso) Let $[q;\ w_1, w_2, \dots, w_n]$ be a weighted voting game. Then the generating function of the number $b^i(k)$ of coalitions S such that $i \notin S$, and $w(S) = k$, is given by*

$$BG_i(x) = \prod_{j \neq i} (1 + x^{w_j}) .$$

Proof. For the weights $W = \{w_1, w_2, \dots, w_n\}$, we consider the generating function

$$\begin{aligned}
(1+x^{w_1})\cdots(1+x^{w_n}) &= \sum_{V\subseteq W}\prod_{w_i\in V} x^{w_i} \\
&= \sum_{V\subseteq W}\left(x^{\sum_{w_i\in V} w_i}\right) \\
&= \sum_{k\geq 0} b(k)\, x^k,
\end{aligned}$$

where $b(k)$ is the number of coalitions with weight k. To obtain the numbers $b^i(k)$, we delete the factor $(1+\, x^{w_i})$. □

On the collection of simple games, we define the operation meet $\wedge$ by

$$(v_1 \wedge v_2)(S) = \min\{v_1(S), v_2(S)\}.$$

Let $v_1 = [q;\ w_1, w_2, \ldots, w_n]$, $v_2 = [p;\ p_1, p_2, \ldots, p_n]$ be weighted voting games. Then the meet game satisfies

$$(v_1 \wedge v_2)(S) = \begin{cases} 1, & \text{if } w(S) \geq q \text{ and } p(S) \geq p \\ 0, & \text{if } w(S) < q \text{ or } p(S) < p. \end{cases}$$

Proposition 2.3. *Let $v_1 = [q;\ w_1, w_2, \ldots, w_n]$, $v_2 = [p;\ p_1, p_2, \ldots, p_n]$ be weighted voting games. Then the generating function of the number $b^i(k,r)$ of coalitions S such that $i \notin S$, and $w(S) = k$, $p(S) = r$, is given by*

$$BG_i(x,y) = \prod_{j\neq i}(1+\, x^{w_j} y^{p_j}).$$

Proof. For the weights $W = \{w_1, w_2, \ldots, w_n\}$ and $P = \{p_1, p_2, \ldots, p_n\}$, we consider the generating function

$$\begin{aligned}
(1+x^{w_1}y^{p_1})\cdots(1+x^{w_n}y^{p_n}) &= \sum_{V\subseteq W}\sum_{R\subseteq P}\prod_{w_i\in V} x^{w_i}\prod_{p_j\in R} y^{p_j} \\
&= \sum_{V\subseteq W}\sum_{R\subseteq P}\left(x^{\sum_{w_i\in V} w_i}\right)\left(y^{\sum_{p_j\in R} p_j}\right) \\
&= \sum_{k\geq 0}\sum_{r\geq 0} b(k,r)\, x^k y^r,
\end{aligned}$$

where $b(k,r)$ is the number of coalitions $S \subseteq N$ such that $w(S) = k$, and $p(S) = r$. For obtaining the numbers $b^i(k,r)$, we delete the factor $(1+\, x^{w_i} y^{p_i})$. □

Proposition 2.4. *Let $v_1 = [q;\, w_1, w_2, \dots, w_n]$, $v_2 = [p;\, p_1, p_2, \dots, p_n]$ be weighted voting games. Then the generating function of the number $A^i(k,r,j)$ of coalitions S of j players such that $i \notin S$, $w(S) = k$, and $p(S) = r$, is given by*

$$ShG_i(x,y,z) = \prod_{j \neq i} \left(1 + z\, x^{w_j} y^{p_j}\right).$$

3 Computational Complexity

The classical procedures for computing the power indices are based in the enumeration of all coalitions. Thus, if the input size of the problem is n, then the function which measures the worst case running time for computing the indices is in $O(2^n)$. In this section, we will give the algorithms based in the generating functions to obtain these power indices and we study its computational complexity. Throughout the remainder of this section, we will assume the logarithmic cost model. In this model, if we perform only a polynomial number of operations on numbers with at most a polynomial number of digits, then the algorithm will be polynomial (see Gács and Lovász, 1999).

Let $f(n)$ be a function from $\mathbb{Z}_+$ to $\mathbb{Z}_+$. Recall that we denote $O(f(n))$ for the set of all functions g such that $f(n) \leq cg(n)$ for all $n \geq n_0$. With this definition a polynomial $\sum_{i=0}^{d} a_i n^i$ is in $O(n^d)$ and this means that only the asymptotic behavior of the function as $n \to \infty$ is being considered. The programs of our language contain only *assignments* and a **for**-*loop construct*. We use the symbol $\leftarrow$ for assignments, for example, $g(x) \leftarrow 1$ denotes setting the value of $g(x)$ to 1. A **for**-loop to calculate $\sum_{i \in I} a_i$, can be defined by

$h \leftarrow 0$

for $i \in I$ **do**

  $h \leftarrow h + a_i$

endfor

We denote by DTIME$(f(n))$ the class of languages whose time complexity is at most $f(n)$. We say that a language has *space complexity* at most $f(n)$, if it can be decided by a Turing machine with space demand (cells and tapes) at most $f(n)$. This class is denoted by DSPACE($f(n)$).

The storage demand of a k-tape Turing machine is at most k times its time demand (in one step, at most k cells will be written). Therefore, DTIME($f(n)$) $\subset$ DSPACE($f(n)$).

Theorem 3.1. *Let $[q;\ w_1, \ldots, w_n]$ be a weighted voting game. If C is the number of nonzero coefficients of the generating function $BG(x)$, then the time complexity of the generating algorithm for the Banzhaf indices is $O(n^2C)$.*

Proof. Let i be a player, the function $BG_i(x) = \prod_{j \neq i} (1 + x^{w_j})$ is given by

$BG(x) \leftarrow 1$

for $j \in \{1, \ldots, n\}$ **with** $j \neq i$ **do**

$BG(x) \leftarrow BG(x) + BG(x)\, x^{w_j}$

endfor

The time to compute the line in the loop is in $O(C)$ for every player. Thus the time to compute this function is $O(nC)$. We take $BG_i(x) = \sum_{k \geq 0} b^i(k)\, x^k$, for every player $i \in N$, and consider the **for** loop

$w \leftarrow w_i$

$s \leftarrow 0$

for $k \in \{q - w, \ldots, q - 1\}$ **do**

  $s \leftarrow s + b^i(k)$

endfor

The time spent in the above loop is $O(C)$, since in the sum we only consider the nonzero coefficients and the total time in the procedure is $O(nC)$. If this procedure is executed n times, we obtain the indices of the n players. □

Corollary 3.1. *Let $v_1 = [q;\ w_1, \ldots, w_n]$, $v_2 = [p;\ p_1, \ldots, p_n]$ be weighted voting games. If C is the number of nonzero coefficients of the generating function $BG(x, y)$, then the time complexity of the generating algorithm for the Banzhaf indices of the meet game is $O(n^2C)$.*

Remark 3.1. If the weighted n-voting game satisfies $w_i = w$, for every player $i \in N$ then the number of nonzero coefficients of $BG(x)$ is $n+1$. For weighted n-voting games such that all the sums of the weights are different, the number of nonzero coefficients of $BG(x)$ is 2^n.

Theorem 3.2. *Let $[q;\, w_1, \ldots, w_n]$ be a weighted voting game. If C is the number of nonzero coefficients of of the generating function $ShG(x,z)$, then the time complexity of the generating algorithm for the Shapley-Shubik indices is $O(n^2C)$.*

Proof. The time to compute the function

$$ShG_i(x,z) = \prod_{j \neq i} (1 + z\, x^{w_j}),$$

with a **for** loop is $O(nC)$, for every player. Also, there are two independents **for** loops:

$w \leftarrow w_i$

$gg(z) \leftarrow 0$

for $k \in \{q-w, \ldots, q-1\}$ **do**

$\quad gg(z) \leftarrow gg(z) + A^i(k,j)\, z^j$

endfor

Thus we obtain the polynomial $gg(z) = \sum_{j=0}^{n-1} b_j z^j$, whose coefficients appears in the next sum,

$t \leftarrow 0$

for $j \in \mathbb{Z}$ **with** $0 \leq j \leq n-1$ **do**

$\quad t \leftarrow t + b_j(n-j-1)!j!$

endfor

$t/n!$

Note that the factorial function takes $O(n)$, and $n \leq C$. Thus we can calculate the index for player i in time $O(nC)$. For the n players, this procedure is executed n times. □

Corollary 3.2. *Let $v_1 = [q;\ w_1, \cdots, w_n]$, $v_2 = [p;\ p_1, \cdots, p_n]$ be weighted voting games. If C is the number of nonzero coefficients of the generating function $ShG(x, y, z)$, then the time complexity of the generating algorithm for the Shapley-Shubik indices of the meet game is $O(n^2C)$.*

Remark 3.2. Since DTIME($f(n)$) $\subset$ DSPACE($f(n)$), the generating algorithms described above have polynomial space complexity.

4 Algorithms with Mathematica

We present several algorithms for computing the power indices with the system Mathematica. The procedure is similar to present by Tannenbaum (1997), but in our algorithms we delete Expand in the definition of the generating function and we use Apply[Plus,]. Furthermore, we compute the voting power in the Council of Ministers of the European Union with the generating functions algorithms.

The game of the power of the countries in the EU Council is defined by

$$\begin{aligned} N &= \{\text{GE, UK, FR, IT, SP, NE, GR, BE, PO, SW, AU, DE, FI, IR, LU}\}, \\ v &= [q;\ 10, 10, 10, 10, 8, 5, 5, 5, 5, 4, 4, 3, 3, 3, 2], \end{aligned}$$

where $q = 62$ or $q = 65$. Of course the classical procedures runs in time exponential 2^n, where n is the number of players. In general, we cannot hope for a polynomial time complexity for the generating functions algorithms, but in many problems we obtain polynomial time whenever the number of coefficients and the maximum of the weights are polynomial in n.

The notebook of Mathematica for computing the classical index power is the following.

In[1]:=

```
votesUE={10,10,10,10,8,5,5,5,5,4,4,3,3,3,2};
```

In[2]:=

```
VUE=N[%/Plus @@ %,3]
```

Out[2]=

```
{0.115, 0.115, 0.115, 0.115, 0.092, 0.0575, 0.0575, 0.0575,
 0.0575, 0.046, 0.046, 0.0345, 0.0345, 0.0345, 0.023}
```

The function BanzhafG computes the generating function for computing the Banzhaf index of a weighted voting game, given by a list of integer weights.

In[3]:=

```
BanzhafG[weights_List]:=Times @@ (1+x^weights)
```

We can find the complexity bound C for the function BanzhafG in the EU-game, as follows:

In[4]:=

```
Length[BanzhafG[votesUE]//Expand]
```

Out[4]=

```
86
```

The function BanzhafIndexPlus computes the normalized Banzhaf index of player i by summing the appropriate coefficients in this generating function. Dividing the index of each player by the sum of all the indices gives the BanzhafPowerPlus distribution.

In[5]:=

```
BanzhafIndexPlus[i_,weights_List,q_Integer]:=
Module[{delw,sw,g,coefi},
delw=Delete[weights,i];
sw=Apply[Plus,delw];
g=BanzhafG[delw];
coefi=CoefficientList[g,x];
Apply[Plus,coefi[[Range[Max[1,q-weights[[i]]+1],Min[q,sw]]]]]]
```

In[6]:=

```
BanzhafPowerPlus[weights_List,q_Integer]:= # /(Plus @@ #)& @
Table[BanzhafIndexPlus[i,weights,q],{i,Length[weights]}]
```

In[7]:=

```
Timing[BanzhafPowerPlus[votesUE,62]]
```

Out[7]=

```
{0.8*Second, {1849/16565, 1849/16565, 1849/16565, 1849/16565,
 1531/16565, 973/16565, 973/16565, 973/16565, 973/16565,
 793/16565, 793/16565, 119/3313, 119/3313, 119/3313, 75/3313}}
```

In[8]:=

```
Ban62=N[%[[2]],3]
```

Out[8]=

```
{0.112, 0.112, 0.112, 0.112, 0.0924, 0.0587, 0.0587, 0.0587,
 0.0587, 0.0479, 0.0479, 0.0359, 0.0359, 0.0359, 0.0226}
```

In[9]:=

```
Timing[BanzhafPowerPlus[votesUE,65]]
```

Out[9]=

```
{0.8*Second, {1227/11149, 1227/11149, 1227/11149, 1227/11149,
 1033/11149, 671/11149, 671/11149, 671/11149, 671/11149,
 507/11149, 507/11149, 411/11149, 411/11149, 411/11149,
 277/11149}}
```

In[10]:=

```
Ban65=N[%[[2]],3]
```

Out[10]=

```
{0.11, 0.11, 0.11, 0.11, 0.0927, 0.0602, 0.0602, 0.0602,
 0.0602, 0.0455, 0.0455, 0.0369, 0.0369, 0.0369, 0.0248}
```

The number of coalitions of weight k and size j is the coefficient of $x^k z^j$ in the generating function ShG for the Shapley-Shubik index. The function ShPowerPlus computes the Shapley-Shubik power distribution with the implementation in Mathematica of Tannenbaum and the modifications mentioned above.

In[11]:=

```
ShG[weights_List]:=Times @@ (1+z x^weights)
```

The complexity bound C for the function ShG in the EU-game is

In[12]:=

```
Length[ShG[votesUE]//Expand]
```

Out[12]=

```
338
```

In[13]:=

```
ShPowerPlus[weights_List,q_Integer]:=
Module[{n=Length[weights],delw,sw,g,coefi,gg},
Table[delw=Delete[weights,i];
sw=Apply[Plus,delw]+1;
g=ShG[delw];
coefi=CoefficientList[g,x];
gg=Apply[Plus,coefi[[
Range[Max[1,q-weights[[i]]+1],Min[q,sw]]]]];
Sum[Coefficient[gg,z,j] j! (n-j-1)!,{j,n-1}],{i,n}]/n!]
```

In[14]:=

```
Timing[ShPowerPlus[votesUE,62]]
```

Out[14]=

```
{4.2*Second, {7/60, 7/60, 7/60, 7/60, 860/9009, 19883/360360,
 19883/360360, 19883/360360, 19883/360360, 743/16380,
 743/16380, 1588/45045, 1588/45045, 1588/45045, 932/45045}}
```

In[15]:=

```
Sh62=N[%[[2]],3]
```

Out[15]=

```
{0.117, 0.117, 0.117, 0.117, 0.0955, 0.0552, 0.0552, 0.0552,
 0.0552, 0.0454, 0.0454, 0.0353, 0.0353, 0.0353, 0.0207}
```

In[16]:=

```
Timing[ShPowerPlus[votesUE,65]]
```

Out[16]=

```
{4.2*Second, {21733/180180, 21733/180180, 21733/180180,
 21733/180180, 4216/45045, 2039/36036, 2039/36036,
 2039/36036, 2039/36036, 3587/90090, 3587/90090,
 2987/90090, 2987/90090, 2987/90090, 1667/90090}}
```

In[17]:=

```
Sh65=N[%[[2]],3]
```

Out[17]=

```
{0.121, 0.121, 0.121, 0.121, 0.0936, 0.0566, 0.0566, 0.0566,
 0.0566, 0.0398, 0.0398, 0.0332, 0.0332, 0.0332, 0.0185}
```

In[18]:=

```
TableForm[Transpose[{VUE,Ban62,Ban65,Sh62,Sh65}],
TableHeadings->{countries,{''VUE'',''Ban 62'',''Ban 65'',
''Sh 62'',''Sh 65''}}]
```

Out[18]=

Country	VUE	Ban 62	Ban 65	Sh 62	Sh 65
Germany	.115	.112	.11	.117	.121
U. Kingdom	.115	.112	.11	.117	.121
France	.115	.112	.11	.117	.121
Italy	.115	.112	.11	.117	.121
Spain	.092	.0924	.0927	.0955	.0936
Netherlands	.0575	.0587	.0602	.0552	.0566
Greece	.0575	.0587	.0602	.0552	.0566
Belgium	.0575	.0587	.0602	.0552	.0566
Portugal	.0575	.0587	.0602	.0552	.0566
Sweden	.046	.0479	.0455	.0454	.0398
Austria	.046	.0479	.0455	.0454	.0398
Denmark	.0345	.0359	.0369	.0353	.0332
Finland	.0345	.0359	.0369	.0353	.0332
Ireland	.0345	.0359	.0369	.0353	.0332
Luxembourg	.023	.0226	.0248	.0207	.0185

Table 1

The next table shows the *time in seconds* for the new functions and the classical algorithms BanzhafIndex and ShapleyValue3, based in the potential of Hart and Mas-Colell which is implemented by Carter (1993).

q	BanzhafIndex	BanzhafPowerPlus	ShapleyValue3	ShPowerPlus
62	6285.24	0.824	442.15	4.174
65	6050.04	0.824	464.07	4.229

Table 2

5 Power in 2-weighted voting games

To study the meet of the weighted voting games given by the votes in the EU Council and the population of the EU countries, we introduce the index PUE and the integer weights (obtained by roundoff) according to this population. The notebook of Mathematica to calculate power indices in *2-weighted voting games* is the following.

In[1]:=

```
votesUE={10,10,10,10,8,5,5,5,5,4,4,3,3,3,2};
```

In[2]:=

```
population={80.61,57.96,57.53,56.93,39.11,15.24,10.35,
10.07,9.86,8.69,7.91,5.18,5.06,3.56,0.4};
```

In[3]:=

```
PUE=N[%/Plus @@ %,3];
```

Out[3]=

```
{0.219, 0.157, 0.156, 0.155, 0.106, 0.0414, 0.0281, 0.0273,
 0.0268, 0.0236, 0.0215, 0.0141, 0.0137, 0.00966, 0.00109}
```

The fuction Round[x] gives the integer closest to x. For numbers such that x.5 the round is x.

In[4]:=

```
popUE=Round[PUE*100]
```

Out[4]=

```
{22, 16, 16, 15, 11, 4, 3, 3, 3, 2, 2, 1, 1, 1, 0}
```

In[5]:=

```
Apply[Plus,%]
```

Out[5]=

```
100
```

First, the meet game $v_1 \wedge v_2$ in the EU Council is defined by

$$\begin{aligned} v_1 &= [62;\ 10,10,10,10,8,5,5,5,5,4,4,3,3,3,2]\,, \\ v_2 &= [p;\ 22,16,16,15,11,4,3,3,3,2,2,1,1,1,0]\,, \end{aligned}$$

where $p \in \{51, 75\}$. Next, we define the generating function BanzhafTwoG for the meet of two weighted voting games. To obtain the normalized Banzhaf index of player i, we define the function BanzhafTwoIndex and BanzhafTwoPower computes the vector of these indices for all players.

In[6]:=

```
BanzhafTwoG[weights_List,pop_List]:=
Times @@ (1+x^weights y^pop)
```

The complexity bound C for the function BanzhafTwoG in the above two weighted voting game is given by

In[7]:=

```
Length[BanzhafTwoG[votesUE,popUE]//Expand]
```

Out[7]=

```
1644
```

In[8]:=

```
BanzhafTwoIndex[i_,weights_List,pop_List,q_Integer,p_Integer]
:=Module[{delwe,delpo,g,sw,sp,coefi,s1,s2},
delwe=Delete[weights,i]; delpo=Delete[pop,i];
g=BanzhafTwoG[delwe,delpo];
sw=Apply[Plus,delwe]+1; sp=Apply[Plus,delpo]+1;
coefi=CoefficientList[g,{x,y}]/.{} -> Table[0,{sp}];
s1=Apply[Plus,Flatten[coefi[[
Range[Max[1,q-weights[[i]]+1],sw],
Range[Max[1,p-pop[[i]]+1],sp]]]]];
s2=If[((q+1)>sw) || ((p+1)>sp),0,Apply[Plus,
Flatten[coefi[[Range[q+1,sw],Range[p+1,sp]]]]]];
s1-s2]
```

In[9]:=

```
BanzhafTwoPower[weights_List,pop_List,q_,p_]:=
# /(Plus @@ #)& @Table[BanzhafTwoIndex[i,weights,pop,q,p],
{i,Length[weights]}]
```

In[10]:=

```
Timing[BanzhafTwoPower[votesUE,popUE,62,51]]
```

Out[10]=

```
{14*Second, {1849/16565, 1849/16565, 1849/16565, 1849/16565,
 1531/16565, 973/16565, 973/16565, 973/16565, 973/16565,
 793/16565, 793/16565, 119/3313, 119/3313, 119/3313, 75/3313}}
```

In[11]:=

```
BanTwo51=N[%[[2]],3]
```

Out[11]=

```
{0.112, 0.112, 0.112, 0.112, 0.0924, 0.0587, 0.0587, 0.0587,
 0.0587, 0.0479, 0.0479, 0.0359, 0.0359, 0.0359, 0.0226}
```

In[12]:=

```
Timing[BanzhafTwoPower[votesUE,popUE,62,75]]
```

Out[12]=

```
{13.79*Second, {1013/6672, 775/6672, 775/6672, 193/1668,
 329/3336, 355/6672, 117/2224, 117/2224, 117/2224,
 23/556, 23/556, 33/1112, 33/1112, 33/1112, 125/6672}}
```

In[13]:=

```
BanTwo75=N[%[[2]],3]
```

Out[13]=

```
{0.152, 0.116, 0.116, 0.116, 0.0986, 0.0532, 0.0526, 0.0526,
 0.0526, 0.0414, 0.0414, 0.0297, 0.0297, 0.0297, 0.0187}
```

For computing the Shapley-Shubik index for 2-weighted voting games, the functions are denoted by ShTwoG and ShTwoPower. These functions are defined as follows.

In[14]:=

```
ShTwoG[weights_List,pop_List]:=Times @@ (1+x^weights y^pop z)
```

The complexity bound C for the function ShTwoG in the European Union two weighted voting game is

In[15]:=

```
Length[ShTwoG[votesUE,popUE]//Expand]
```

Out[15]=

```
2206
```

In[16]:=

```
ShTwoPower[weights_List,pop_List,q_Integer,p_Integer]:=
Module[{n=Length[weights],delwe,delpo,g,sw,sp,coefi,s1,s2,gg},
Table[delwe=Delete[weights,i]; delpo=Delete[pop,i];
g=ShTwoG[delwe,delpo];
sw=Apply[Plus,delwe]+1; sp=Apply[Plus,delpo]+1;
coefi=CoefficientList[g,{x,y}]/.{} -> Table[0,{sp}];
s1=Apply[Plus,Flatten[coefi[
[Range[Max[1,q-weights[[i]]+1],sw],
Range[Max[1,p-pop[[i]]+1],sp]]]]];
s2=If[((q+1)>sw) || ((p+1)>sp),0,
Apply[Plus,Flatten[coefi[
[Range[q+1,sw],Range[p+1,sp]]]]]];
gg=s1-s2;
Sum[Coefficient[gg,z,j] j! (n-j-1)!,{j,0,n-1}]/n!,{i,n}]]
```

In[17]:=

```
Timing[ShTwoPower[votesUE,popUE,62,51]]
```

Out[17]=

```
{28.5*Second, {7/60, 7/60, 7/60, 7/60, 860/9009,
 19883/360360, 19883/360360, 19883/360360, 19883/360360,
 743/16380, 743/16380, 1588/45045, 1588/45045,
 1588/45045, 932/45045}}
```

In[18]:=

```
ShTwo51=N[%[[2]],3]
```

Out[18]=

```
{0.117, 0.117, 0.117, 0.117, 0.0955, 0.0552, 0.0552, 0.0552,
 0.0552, 0.0454, 0.0454, 0.0353, 0.0353, 0.0353, 0.0207}
```

In[19]:=

```
Timing[ShTwoPower[votesUE,popUE,62,75]]
```

Out[19]=

```
{28*Second, {607/3003, 4835/36036, 4835/36036, 1202/9009,
 5561/45045, 1427/32760, 13207/360360, 13207/360360,
 13207/360360, 1597/60060, 1597/60060, 487/25740,
 487/25740, 487/25740, 829/90090}}
```

In[20]:=

```
ShTwo75=N[%[[2]],3]
```

Out[20]=

```
{0.202, 0.134, 0.134, 0.133, 0.123, 0.0436, 0.0366, 0.0366,
 0.0366, 0.0266, 0.0266, 0.0189, 0.0189, 0.0189, 0.0092}
```

In[21]:=

```
TableForm[Transpose[{PUE,BanTwo51,BanTwo75,ShTwo51,ShTwo75}],
TableHeadings->{countries,{''PUE'',''2Ban 51'',''2Ban 75'',
''2Sh 51'',''2Sh 75''}}]
```

Out[21]=

Country	PUE	2Ban 51	2Ban 75	2Sh 51	2Sh 75
Germany	.219	.112	.152	.117	.202
U. Kingdom	.157	.112	.116	.117	.134
France	.156	.112	.116	.117	.134
Italy	.155	.112	.116	.117	.133
Spain	.106	.0924	.0986	.0955	.123
Netherlands	.0414	.0587	.0532	.0552	.0436
Greece	.0281	.0587	.0526	.0552	.0366
Belgium	.0273	.0587	.0526	.0552	.0366
Portugal	.0268	.0587	.0526	.0552	.0366
Sweden	.0236	.0479	.0414	.0454	.0266
Austria	.0215	.0479	.0414	.0454	.0266
Denmark	.0141	.0359	.0297	.0353	.0189
Finland	.0137	.0359	.0297	.0353	.0189
Ireland	.00966	.0359	.0297	.0353	.0189
Luxembourg	.00109	.0226	.0187	.0207	.0092

Table 3

6 Concluding remarks

In the present paper we have considered weighted and 2-weighted voting games. We have shown that there exist polynomial time algorithms based in generating functions to compute the classical power indices. We have also obtained that the complexity bound for these algorithms is the number C of nonzero coefficients of the generating function.

References

Banzhaf, J. F. III (1965). Weighted Voting Doesn't Work: A Mathematical Analysis. *Rutgers Law Review* **19**, 317–343.

Brams, S. F. and P. J. Affuso (1976). Power and Size: A New Paradox. *Theory and Decision* **7**, 29–56.

Carter, M. (1993). Cooperative games. In H. R. Varian, Ed., *Economic and Financial Modeling with Mathematica*, Springer-Verlag, Berlin, 167–191.

Coleman, J. S. (1973). Loss of power. *American Sociological Review* **38**, 1–17.

Felsenthal, D. S. and M. Machover (1998). *The Measurement of Voting Power: Theory and Practice, Problems and Paradoxes.* Edward Elgar, Cheltenham.

Gács, P. and L. Lovász (1999). *Complexity of Algorithms.* Lecture Notes, Yale University, available at http://www.esi2.us.es/~mbilbao/pdffiles/complex.pdf

Lucas, W. F. (1983). Measuring Power in Weighted Voting Systems. In S. J. Brams, W. F. Lucas, P. D. Straffin, Eds., *Political and Related Models,* Springer-Verlag, New York, 183–238.

Mann, I. and L. S. Shapley (1962). Value of Large Games, VI: Evaluating the Electoral College Exactly. *RM-3158-PR, The Rand Corporation.*

Shapley, L. S. (1953). A value for n-person games. *Ann. Math.* **28**, 307–317.

Shapley, L. S., and M. Shubik (1954). A Method for Evaluating the Distribution of Power in a Committee System. *American Political Science Review* **48**, 787–792.

Stanley, R. P. (1986). *Enumerative Combinatorics I,* Wadsworth, Monterey.

Tannenbaum, P. (1997). Power in Weighted Voting Systems. *The Mathematica Journal* **7**, 58–63.