

# Computing Power Indices in Weighted Majority Games with Formal Power Series

Naonori Kakimura<sup>1</sup>[0000–0002–3918–3479] and Yoshihiko Terai<sup>1</sup>

Keio University,  
Yokohama 223-8522, Japan.  
kakimura@math.keio.ac.jp, 11ty24@keio.jp

**Abstract.** In this paper, we propose fast pseudo-polynomial-time algorithms for computing power indices in weighted majority games. We show that we can compute the Banzhaf index for all players in  $O(n + q \log(q))$  time, where  $n$  is the number of players and  $q$  is a given quota. Moreover, we prove that the Shapley–Shubik index for all players can be computed in  $O(nq \log(q))$  time. Our algorithms are faster than existing algorithms when  $q = 2^{o(n)}$ . Our algorithms exploit efficient computation techniques for formal power series.

**Keywords:** Power index · Weighted majority game · Formal power series · Generating function

## 1 Introduction

In this paper, we study measuring the power of players in weighted majority games. The weighted majority game (also known as the weighted voting game) is a mathematical model of voting in which each player has a certain number of votes. In the game, each player votes for or against a decision, and the decision is accepted if the sum of players' voting is greater or equal to a fixed quota.

In 1964, Shapley and Shubik [24] proposed how to measure a voting power of each player, called the *Shapley–Shubik index*. The Shapley–Shubik index is a specialization of the Shapley value [23] for cooperative games. Another concept for measuring voting power was introduced by Banzhaf [5], called the *Banzhaf index*. These power indices are used to evaluate political systems such as the European Constitution and the IMF [1,13,16].

It is known that computing the Banzhaf and Shapley–Shubik indices are both known to be NP-hard [10,22]. This paper is thus concerned with calculating these power indices in pseudo polynomial time. To devise fast algorithms, there are various approaches in the literature, that include using dynamic programming [26], generating functions [8,6,19], binary decision diagrams [7], Monte-Carlo methods [4,18,27], and so on. The current best time complexity [16,26] is  $O(nq)$  for computing the Banzhaf index of all  $n$  players with a given quota  $q$ , and  $O(n^2q)$  for the Shapley–Shubik index of all  $n$  players. It should be remarked that we here assume, as previous papers, that the arithmetic operations of  $n$ -bit numbers can be performed in constant time. See also a remark in Section 5.

The main contribution of this paper is to design efficient algorithms for calculating the Banzhaf and Shapley–Shubik indices for all players, respectively, using techniques in theory of formal power series. Our proposed algorithms run in  $O(n + q \log(q))$  time for the Banzhaf index, and  $O(nq \log(q))$  time for the Shapley–Shubik index. Our algorithms are faster than existing algorithms when  $q = 2^{o(n)}$ .

In our algorithms, we first represent power indices with generating functions (i.e., polynomials). In the previous work such as [6,8,19], generating functions are computed by dynamic programming. In this paper, we regard a generating function as a formal power series, which is an infinite sum of monomials, and exploit the fast Fourier transform (FFT) for efficient computation. We remark that similar technique was recently used to solve the subset sum problem [12].

### 1.1 Related work

We here describe only algorithmic aspects of power indices in weighted majority games. See also a survey [20]. Practical applications of power indices may be found in, e.g., [1,13,16].

As mentioned before, it is NP-hard to calculate the Banzhaf and Shapley–Shubik indices. In fact, Deng and Papadimitriou [10] showed the problem of computing the Shapley–Shubik index is  $\#P$ -complete. Prasad and Kelly [22] proved that computing the Banzhaf index is  $\#P$ -complete, and that a number of decision problems around the Banzhaf and the Shapley–Shubik indices belong to the class of NP-complete problems. See also [21] for other hardness results. It is known that even approximating the Shapley–Shubik index within a constant factor is intractable, unless  $P = NP$  [11].

Cantor (as mentioned by [19]) represented the Shapley–Shubik index with a generating function (a polynomial), which was used for calculation [17,19]. See Brams and Affuso [8] for the Banzhaf index. Bilbao et al. [6] proposed an algorithm based on generating functions, running in  $O(nq)$  time to compute the Banzhaf index of one fixed player.

Matsui and Matsui [20] designed dynamic programming algorithms for the Banzhaf and Shapley–Shubik indices, respectively, which is a similar approach to the one for counting the number of knapsack solutions. Uno [26] later improved their algorithms, which run in  $O(nq)$  and  $O(n^2q)$  time, respectively, for calculating the Banzhaf and Shapley–Shubik indices of all players. See also [16] for slight improvement with other parameters. Klinz and Woeginger [15] proposed an  $O(n^22^{n/2})$ -time algorithm based on the enumeration technique. There is another line of research to calculate the power indices approximately, see, e.g., [4,18,27].

## 2 Preliminaries

### 2.1 Power Index

Let us formally introduce our setting. In a weighted majority game, there are  $n$  players. Let  $N = \{1, 2, \dots, n\}$  be a set of  $n$  players. Each player  $p$  in  $N$  has a non-negative integer weight  $w_p$ . We are also given a non-negative integer  $q$ , which is called a *quota*. Then each player votes for or against a decision. A *coalition* is a set  $S \subseteq N$  of players who vote for the decision. A coalition  $S$  is called *winning* if  $w(S) \geq q$ , where we define  $w(S) = \sum_{p \in S} w_p$ , and *losing* otherwise.

For a player  $p$ , the *Banzhaf index*  $Bz_p$  is defined as

$$Bz_p := \frac{\#\{S \subseteq N \mid w(S) \geq q, w(S \setminus \{p\}) < q\}}{2^n}.$$

Intuitively, for a winning coalition  $S$  with  $p \in S$ , the player  $p$  is considered to have a power in  $S$  if  $S \setminus \{p\}$  is a losing coalition. The Banzhaf index  $Bz_p$  is equal to the probability that the player  $p$  belongs to a coalition with having a power under the assumption that every coalition occurs uniformly at random.

We next define the Shapley–Shubik index. Consider the situation where, at first, a coalition  $S$  is the empty set, and players join the coalition  $S$  one by one in an order. Then, at the beginning of this situation,  $S$  is a losing coalition, and  $S$  becomes a winning one at some point. Let  $p$  be the player such that  $S$  has changed from a losing coalition to a winning one when  $p$  has been added to  $S$ . We can naturally think that the player  $p$  has a power.

More specifically, let  $\Pi$  be the set of permutations with length  $n$ . We note that  $|\Pi| = n!$ . A permutation  $\pi \in \Pi$  indicates an ordering of players to join a coalition, that is, players make coalitions  $S_1(\pi), S_2(\pi), \dots, S_n(\pi)$  in the order, where  $S_i(\pi) = \{\pi_1, \pi_2, \dots, \pi_i\}$  for  $i = 1, 2, \dots, n$ . Then there exists a unique player  $\pi_k$  such that  $w(S_{k-1}(\pi)) < q$  and  $w(S_k(\pi)) \geq q$ . The *Shapley–Shubik index*  $SS_p$  of player  $p$  is defined as

$$SS_p := \frac{\#\{\pi \in \Pi \mid \exists k \text{ s.t. } w(S_{k-1}(\pi)) < q, w(S_k(\pi)) \geq q, p = \pi_k\}}{n!}.$$

The Shapley–Shubik index is equal to the probability that the player  $p$  has a power under the assumption that every permutation occurs uniformly at random.

## 2.2 Formal Power Series

Let  $R$  be a commutative ring where the zero element and the identity are denoted by 0 and 1, respectively. A *formal power series*  $f$  over  $R$  is an infinite sum of the form

$$f = \sum_{i=0}^{\infty} a_i x^i = a_0 + a_1 x + a_2 x^2 + \dots,$$

where  $a_i \in R$  for every non-negative integer  $i$ . We call  $a_i$  the  *$i$ -th coefficient of  $f$* . The  $i$ -th coefficient of  $f$  is denoted by  $[x^i]f$ .

We denote the set of all formal power series over  $R$  by  $R[[x]]$ , that is,  $R[[x]] = \{\sum_{i=0}^{\infty} a_i x^i \mid a_i \in R\}$ . Then  $R[[x]]$  forms a ring. Specifically, the ring  $R[[x]]$  defines the following operations for two formal power series  $f = \sum_{i=0}^{\infty} a_i x^i$  and  $g = \sum_{i=0}^{\infty} b_i x^i$ :

1. (Sum)

$$f + g = \sum_{i=0}^{\infty} (a_i + b_i) x^i.$$

2. (Product)

$$f \cdot g = \sum_{k=0}^{\infty} \left( \sum_{i+j=k} a_i \cdot b_j \right) x^k.$$

The (*multiplicative*) *inverse* of a formal power series  $f$  is a formal power series  $g$  such that  $f \cdot g = 1$ . The inverse of  $f$ , which is known to be unique (if exists), is denoted by  $1/f$  or  $f^{-1}$ . For example, the inverse of  $1 - x$  is equal to  $1 + x + x^2 + \dots = \sum_{i=0}^{\infty} x^i$ , as  $(1 - x)(1 + x + x^2 + \dots) = 1$ . Note that  $f$  may not necessarily have the inverse. An element having the inverse is called a *unit*.

We observe that a formal power series  $f \in R[[x]]$  is a unit if and only if  $[x^0]f \in R$  has the inverse in a ring  $R$ . In particular, when  $R$  is a field,  $f \in R[[x]]$  is a unit if and only if  $[x^0]f \in R$  is not the zero element 0.

A polynomial is a formal power series with only finitely many non-zero terms. It is well-known that the product of two polynomials over the real field  $\mathbb{R}$  can be computed efficiently by the fast Fourier transform (FFT). See e.g., [14].

**Lemma 1.** *Suppose that we are given two polynomials  $f = \sum_{i=0}^d a_i x^i$  and  $g = \sum_{j=0}^d b_j x^j$  over the real field  $\mathbb{R}$ . Then their product  $f \cdot g$  can be computed in  $O(d \log(d))$  time. That is, we can compute, in  $O(d \log(d))$  time, all the  $k$ -th coefficients*

$$[x^k](f \cdot g) = \sum_{i+j=k} a_i \cdot b_j$$

for non-negative integers  $k$  with  $k \leq 2d$ .

By the above lemma, we can efficiently compute the product of two formal power series in the following sense. For two formal power series  $f$  and  $g$ , if we are given the first  $t$  coefficients  $[x^i]f$  and  $[x^i]g$  for  $i = 0, 1, \dots, t-1$ , then we can obtain the first  $t$  coefficients  $[x^i](f \cdot g)$  of  $f \cdot g$  for  $i = 0, 1, \dots, t-1$ , in  $O(t \log(t))$  time. Note that, since a formal power series is an infinite sum, we set a parameter  $t$  indicating the number of coefficients we want to compute.

### 3 Computing the Banzhaf index

In this section, we present a fast algorithm for computing the Banzhaf index in weighted majority games. Let  $N = \{1, 2, \dots, n\}$  be a set of players where player  $p$  has a non-negative integer weight  $w_p$ , and let  $q$  be a quota. We may assume that  $1 \leq w_p < q$  for any player  $p$ , and that there exists at least one winning coalition in the given game, that is,  $w(N) \geq q$ .

The main result of this section is the following.

**Theorem 1.** *For a weighted majority game with  $n$  players and a quota  $q$ , we can compute the Banzhaf index for all players in  $O(n + q \log(q))$  time.*

It is known in [8] that the Banzhaf index  $Bz_p$  for a player  $p$  can be represented with a polynomial over the real field  $\mathbb{R}$ . For a player  $p$ , we define  $f_p \in \mathbb{R}[[x]]$  as

$$f_p = \prod_{\ell \neq p} (1 + x^{w_\ell}).$$

Then we observe that, for every non-negative integer  $i$ , the  $i$ -th coefficient  $[x^i]f_p$  is equal to the number of subsets  $S \subseteq N \setminus \{p\}$  with  $w(S) = i$ .

**Lemma 2 (Brams and Affuso [8]).** *For a player  $p$ , it holds that*

$$2^n \cdot \text{Bz}_p = \sum_{j=q-w_p}^{q-1} [x^j] f_p.$$

In Lemma 3 below, we shall represent the Banzhaf index of all players with one formal power series over the real field  $\mathbb{R}$ . Define  $\hat{f} \in \mathbb{R}[[x]]$  as

$$\hat{f} = \prod_{p=1}^n (1 + x^{w_p}).$$

Then, for each player  $p$ , we have

$$f_p = \frac{\hat{f}}{1 + x^{w_p}}.$$

**Lemma 3.** *For each player  $p$ , it holds that*

$$\begin{aligned} 2^n \cdot \text{Bz}_p &= [x^{q-1}] \frac{f_p}{1-x} - [x^{q-w_p-1}] \frac{f_p}{1-x} \\ &= [x^{q-1}] \frac{\hat{f}}{(1-x)(1+x^{w_p})} - [x^{q-w_p-1}] \frac{\hat{f}}{(1-x)(1+x^{w_p})}. \end{aligned} \quad (1)$$

*Proof.* Since it holds that  $\frac{1}{1-x} = \sum_{i=0}^{\infty} x^i$ , we have

$$\frac{f_p}{1-x} = f_p \cdot \left( \sum_{i=0}^{\infty} x^i \right) = \sum_{i=0}^{\infty} \left( \sum_{j=0}^i a_{pj} \right) x^i,$$

where we denote  $a_{pj} = [x^j] f_p$  for every non-negative integer  $j$ . Therefore, it holds that

$$[x^{q-1}] \frac{f_p}{1-x} - [x^{q-w_p-1}] \frac{f_p}{1-x} = \sum_{j=0}^{q-1} a_{pj} - \sum_{j=0}^{q-w_p-1} a_{pj} = \sum_{j=q-w_p}^{q-1} a_{pj},$$

which is equal to  $2^n \cdot \text{Bz}_p$  by Lemma 2. Thus the lemma holds.  $\square$

We compute the Banzhaf index for all players using the formal power series in (1). The proposed algorithm is described as follows.

- Step 1.** Compute the  $i$ -th coefficients of  $\hat{f}$  for all non-negative integers  $i$  with  $i < q$ .
- Step 2.** Compute the  $i$ -th coefficients of  $g = \frac{\hat{f}}{1-x}$  for all non-negative integers  $i$  with  $i < q$ .
- Step 3.** For each player  $p$  with different weights, compute the Banzhaf index  $\text{Bz}_p$  by (1).

We next evaluate the computational complexity of each step in the proposed algorithm.

Jin and Wu [12] showed as below that a polynomial  $f$  in the form of  $f = \prod_{i=1}^n (1 + x^{s_i})$  can be calculated efficiently with the aid of Lemma 1. Thus Step 1 can be performed in  $O(n + q \log(q))$  time.

**Theorem 2 (Jin and Wu [12]).** For  $n$  positive integers  $s_1, s_2, \dots, s_n$ , define a polynomial  $f$  over  $\mathbb{R}$  to be  $f = \prod_{i=1}^n (1 + x^{s_i})$ . Then, for a positive integer  $t$ , the first  $t$  coefficients of  $f$  can be computed in  $O(n + t \log(t))$  time.

For Step 2, since

$$g = \hat{f} \cdot \left( \frac{1}{1-x} \right) = \sum_{i=0}^{\infty} \left( \sum_{j=0}^i \hat{a}_j \right) x^i,$$

where  $\hat{a}_j = [x^j] \hat{f}$  for every non-negative integer  $j$ , we have  $[x^i]g = \sum_{j=0}^i \hat{a}_j$  for  $i = 0, 1, \dots, q-1$ . Hence we can compute  $[x^0]g, \dots, [x^{q-1}]g$  sequentially in  $O(q)$  time.

The following lemma evaluates the time complexity of Step 3.

**Lemma 4.** Suppose that we are given the first  $q$  coefficients of  $g$ , that is, we are given  $[x^0]g, [x^1]g, \dots, [x^{q-1}]g$ . For a player  $p$  with weight  $w_p$ , the Banzhaf index  $Bz_p$  can be computed in  $O\left(\frac{q}{w_p}\right)$  time.

*Proof.* We use (1) to compute the Banzhaf index  $Bz_p$ . It holds that

$$\frac{\hat{f}}{(1-x)(1+x^{w_p})} = \frac{g}{1+x^{w_p}} = \left( \sum_{j=0}^{\infty} (-1)^j x^{jw_p} \right) \cdot g.$$

Then, for  $k \leq q-1$ , its  $k$ -th coefficient is equal to

$$\sum_{i+jw_p=k} (-1)^j \cdot [x^i]g = \sum_{j=0}^{\lfloor \frac{k}{w_p} \rfloor} (-1)^j \cdot [x^{k-jw_p}]g.$$

Since the right-hand side has at most  $\frac{k}{w_p} + 1 = O\left(\frac{q}{w_p}\right)$  terms, we can compute the  $k$ -th coefficient of  $\frac{g}{1+x^{w_p}}$  in  $O\left(\frac{q}{w_p}\right)$  time, provided  $[x^0]g, [x^1]g, \dots, [x^{q-1}]g$ . Therefore, since the right-hand side of (1) consists of two coefficients of  $\frac{g}{1+x^{w_p}}$ ,  $Bz_p$  can be calculated in  $O\left(\frac{q}{w_p}\right)$  time.  $\square$

We are now ready to prove Theorem 1.

*Proof (Proof of Theorem 1).* As discussed above, Step 1 in the proposed algorithm takes  $O(n + q \log(q))$  time by Theorem 2, and Step 2 takes  $O(q)$  time. It follows from Lemma 4 that  $Bz_p$  of a player  $p$  can be computed in  $O\left(\frac{q}{w_p}\right)$  time. Since two players  $p, p'$  with  $w_p = w_{p'}$  satisfy  $Bz_p = Bz_{p'}$ , we can compute the Banzhaf index of all players by computing those of players with different weights. Since weights are integers between 1 and  $q-1$ , the complexity for Step 3 can be bounded by

$$O\left(\sum_{w=1}^{q-1} \frac{q}{w}\right) = O(q \log(q)).$$

Therefore, the total complexity of the proposed algorithm is  $O(n + q \log(q))$ .  $\square$

## 4 Computing the Shapley–Shubik index

In this section, we show that the Shapley–Shubik index can be computed efficiently.

**Theorem 3.** *For a weighted majority game with  $n$  players and a quota  $q$ , we can compute the Shapley–Shubik index of all players in  $O(n + \tilde{n}q \log(q))$  time, where  $\tilde{n} = \min\{n, q\}$ .*

To prove the theorem, we introduce a polynomial with two variables. For a polynomial  $f = \sum_{k=0}^d \sum_{j=0}^{d'} b_{jk} x^j y^k$  with two variables, we denote  $[y^k x^j] f_p = b_{jk}$  for non-negative integers  $k, j$ . We also denote  $[x^j] f_p = \sum_{k=0}^d b_{jk} y^k$  and  $[y^k] f_p = \sum_{j=0}^{d'} b_{jk} x^j$ .

### 4.1 Algorithm

As shown by Cantor (cf. [17,19]), the Shapley–Shubik index can be represented with a polynomial with two variables. For a player  $p$ , we define a polynomial  $f_p$  in two variables  $x, y$  over the real field  $\mathbb{R}$  as

$$f_p = \prod_{\ell \neq p} (1 + yx^{w_\ell}).$$

Then  $[y^k x^j] f_p$  is equal to the number of subsets  $S \subseteq N \setminus \{p\}$  such that  $|S| = k$  and  $w(S) = j$ .

**Lemma 5 (Cantor (cf. [17,19])).** *For a player  $p$ , it holds that*

$$n! \cdot \text{SS}_p = \sum_{k=1}^{n-1} \left( k! \cdot (n-1-k)! \sum_{j=q-w_p-1}^{q-1} [y^k x^j] f_p \right).$$

Since  $w_p \geq 1$  for every player  $p$ , we see that, if  $[y^k x^j] f_p \neq 0$ , then  $k \leq j$  holds. Hence it suffices to consider  $[y^k x^j] f_p$  for  $k \leq j$  in the right-hand side, implying that

$$n! \cdot \text{SS}_p = \sum_{k=1}^{\tilde{n}-1} \left( k! \cdot (n-1-k)! \sum_{j=q-w_p-1}^{q-1} [y^k x^j] f_p \right), \quad (2)$$

where  $\tilde{n} = \min\{n, q\}$ .

Similarly to Lemma 3 in Section 3, the Shapley–Shubik index  $\text{SS}_p$  for a player  $p$  can be represented as follows. Define

$$\hat{f} = \prod_{p=1}^n (1 + yx^{w_p}) \quad \text{and} \quad g = \frac{\hat{f}}{1-x}.$$

**Lemma 6.** *For each player  $p$ , it holds that*

$$n! \cdot \text{SS}_p = \sum_{k=1}^{\tilde{n}-1} k! \cdot (n-1-k)! \left( [y^k x^{q-1}] \frac{g}{1+yx^{w_p}} - [y^k x^{q-w_p-1}] \frac{g}{1+yx^{w_p}} \right). \quad (3)$$

*Proof.* It holds that

$$\frac{g}{1+yx^{w_p}} = \frac{f_p}{1-x} = f_p \cdot \left( \sum_{i=0}^{\infty} x^i \right) = \sum_{i=0}^{\infty} \left( \sum_{j=0}^i a_{pj} \right) x^i,$$

where we denote  $a_{pj} = [x^j]f_p$  for every non-negative integer  $j$ . We note that  $a_{pj}$  is a polynomial in  $y$ . The coefficient of  $y^k x^i$  in  $f_p/(1-x)$  is equal to

$$[y^k] \left( \sum_{j=0}^i a_{pj} \right) = \sum_{j=0}^i [y^k] a_{pj} = \sum_{j=0}^i [y^k x^j] f_p.$$

Hence we have

$$\begin{aligned} [y^k x^{q-1}] \frac{g}{1+yx^{w_p}} - [y^k x^{q-w_p-1}] \frac{g}{1+yx^{w_p}} &= \sum_{j=0}^{q-1} [y^k x^j] f_p - \sum_{j=0}^{q-w_p-1} [y^k x^j] f_p \\ &= \sum_{j=q-w_p}^{q-1} [y^k x^j] f_p, \end{aligned}$$

implying that the right-hand side of (3) is equal to  $n! \cdot \text{SS}_p$  by (2).  $\square$

Our proposed algorithm computes the right-hand side of (3) for a player  $p$ . By (3), it suffices to compute  $[y^k x^j] \frac{g}{1+yx^{w_p}}$  for all  $k < \tilde{n}$  and  $j < q$ . To this end, we regard a two-variable rational function as a formal power series (w.r.t.  $x$ ) over the formal power series ring (w.r.t.  $y$ ). Specifically, let  $R = \mathbb{R}[[y]]/(y^{\tilde{n}})$ , which is the quotient ring of  $\mathbb{R}[[y]]$  by the ideal generated by  $y^{\tilde{n}}$ . That is, an element in  $R$  can be written as  $\sum_{i=0}^{\tilde{n}-1} b_i y^i$  for  $b_i \in \mathbb{R}$ . Note that  $R$  is isomorphic to the quotient ring  $\mathbb{R}[y]/(y^{\tilde{n}})$  of polynomials. Consider the formal power series ring  $R[[x]]$ . Then, a formal power series  $f$  in  $R[[x]]$  can be written as

$$f = \sum_{i=0}^{\infty} \left( \sum_{j=0}^{\tilde{n}-1} a_{ij} y^j \right) x^i,$$

where  $a_{ij} \in \mathbb{R}$ . The  $i$ -th coefficient  $[x^i]f$  is equal to  $\sum_{j=0}^{\tilde{n}-1} a_{ij} y^j$ . We also denote  $[y^j x^i]f = a_{ij}$  for every non-negative integers  $i$  and  $j < \tilde{n}$ .

The proposed algorithm is presented as below.

- Step 1.** Compute the  $i$ -th coefficients of  $\hat{f} \pmod{y^{\tilde{n}}}$  for all non-negative integers  $i$  with  $i < q$ .
- Step 2.** Compute the  $i$ -th coefficients of  $g \pmod{y^{\tilde{n}}}$  for all non-negative integers  $i$  with  $i < q$ .
- Step 3.** For each player  $p$  with different weights, compute the Shapley–Shubik index  $\text{SS}_p$  by (3).

To evaluate the time complexity of the proposed algorithm, we first show a variant of Theorem 2 as follows, where the proof is deferred to Section 4.2. The theorem implies that Step 1 can be executed in  $O(n + \tilde{n}q \log(\tilde{n}q))$  time.

**Theorem 4.** *For  $n$  positive integers  $s_1, s_2, \dots, s_n$ , define a polynomial  $f = \prod_{i=1}^n (1 + yx^{s_i})$  over the real field  $\mathbb{R}$ . Then, for two positive integers  $m, t$ , we can compute all the coefficients  $[y^k x^j] f$  for  $j < t$  and  $k < m$  in  $O(n + mt \log(mt))$  time.*

Moreover, we have the following lemma for Step 3.

**Lemma 7.** *Suppose that we are given the first  $q$  coefficients of  $g$  modulo  $y^{\tilde{n}}$ , denoted by  $a_0, a_1, \dots, a_{q-1}$ . For a player  $p$  with weight  $w_p$ , the Shapley–Shubik index  $\text{SS}_p$  can be computed in  $O\left(\frac{\tilde{n}q}{w_p}\right)$  time.*

*Proof.* We use (3) to compute the Shapley–Shubik index  $\text{SS}_p$ . It holds that

$$\frac{g}{1 + yx^{w_p}} = \left( \sum_{j=0}^{\infty} (-1)^j y^j x^{jw_p} \right) \cdot g.$$

Then, for  $k \leq q - 1$ , its  $k$ -th coefficient is

$$[x^k] \frac{g}{1 + yx^{w_p}} = \sum_{i+jw_p=k} (-1)^j y^j \cdot [x^i] g = \sum_{j=0}^{\lfloor \frac{k}{w_p} \rfloor} (-1)^j y^j \cdot [x^{k-jw_p}] g.$$

Hence, for  $\ell = 1, 2, \dots, \tilde{n}$ , we have

$$[y^\ell x^k] \frac{g}{1 + yx^{w_p}} = \sum_{j=0}^{\lfloor \frac{k}{w_p} \rfloor} (-1)^j \cdot [y^{\ell-j} x^{k-jw_p}] g.$$

Since the right-hand side has at most  $\frac{k}{w_p} + 1$  terms, we can compute it in  $O\left(\frac{q}{w_p}\right)$  time, provided  $a_0, a_1, \dots, a_{q-1}$ . Therefore, since the right-hand side of (3) can be calculated from  $2(\tilde{n} - 1)$  coefficients of  $\frac{g}{1 + yx^{w_p}}$ ,  $\text{SS}_p$  can be obtained in  $O\left(\frac{\tilde{n}q}{w_p}\right)$  time.  $\square$

We now prove Theorem 3.

*Proof (Proof of Theorem 3).* As discussed above, Step 1 in the proposed algorithm takes  $O(n + \tilde{n}q \log(\tilde{n}q))$  time by Theorem 4. Since  $\tilde{n} = \min\{n, q\} \leq q$ , the time complexity for Step 1 is  $O(n + \tilde{n}q \log(q))$ . Moreover, Step 2 takes  $O(\tilde{n}q)$  time, similarly to Section 3.

It follows from Lemma 4 that the Shapley–Shubik index  $\text{SS}_p$  of player  $p$  can be computed in  $O\left(\frac{\tilde{n}q}{w_p}\right)$  time. Since two players  $p, p'$  with  $w_p = w_{p'}$  satisfy  $\text{SS}_p = \text{SS}_{p'}$ , we can compute

the Shapley–Shubik index of all players by computing those of players with different weights. The complexity for Step 3 can be bounded by

$$O\left(\sum_{w=1}^{q-1} \frac{\tilde{n}q}{w}\right) = O(\tilde{n}q \log(q)).$$

Therefore, the total complexity is  $O(n + \tilde{n}q \log(q))$ , which completes the proof of the theorem.  $\square$

## 4.2 Two-variable Formal Power Series

This section is devoted to proving Theorem 4. The proof extends the one of Theorem 2 by [12] for the real field  $\mathbb{R}$  to the case when  $R = \mathbb{R}[[y]]/(y^m)$ , where  $m$  is a non-negative integer.

We first show that the product of two-variable polynomials over  $\mathbb{R}$  can be computed by Lemma 1.

**Lemma 8.** *Suppose that we are given two polynomials  $f = \sum_{i=0}^{d_x} \sum_{j=0}^{d_y} a_{ij}x^i y^j$  and  $g = \sum_{i=0}^{d_x} \sum_{j=0}^{d_y} b_{ij}x^i y^j$ . Then the product  $f \cdot g$  can be computed in  $O(d_x d_y \log(d_x d_y))$  time. That is, we can compute, in  $O(d_x d_y \log(d_x d_y))$  time, all the coefficients*

$$[y^h x^k](f \cdot g) = \sum_{i_1+i_2=k} \sum_{j_1+j_2=h} f_{i_1 j_1} \cdot g_{i_2 j_2}$$

for two non-negative integers  $k, h$  satisfying  $k \leq 2d_x$  and  $h \leq 2d_y$ .

*Proof.* We transform  $f$  and  $g$  to one-variable polynomials by substituting  $y = x^{2d_x+1}$ . The resulting polynomials are denoted by  $F$  and  $G$ , respectively, that is,

$$F = \sum_{i=0}^{d_x} \sum_{j=0}^{d_y} a_{ij} x^{i+(2d_x+1)j} \quad \text{and} \quad G = \sum_{i=0}^{d_x} \sum_{j=0}^{d_y} b_{ij} x^{i+(2d_x+1)j}.$$

Then, since  $F$  and  $G$  are polynomials of degree  $O(d_x d_y)$ , their product  $F \cdot G$  can be computed in  $O(d_x d_y \log(d_x d_y))$  time by Lemma 1. Since it holds that

$$[x^{k+(2d_x+1)h}]F \cdot G = [y^h x^k]f \cdot g$$

for every integers  $k, h$  with  $k \leq 2d_x, h \leq 2d_y$ , we can obtain the coefficients of the product  $f \cdot g$ .  $\square$

For a formal power series  $f \in R[[x]]$  such that  $[x^0]f$  is nilpotent with respect to  $R$ , define

$$\exp(f) = \sum_{i=0}^{\infty} \frac{f^i}{i!}.$$

We recall that an element  $a \in R$  is *nilpotent* if there exists a positive integer  $k$  such that  $a^k = 0$ . Also, for a formal power series  $f \in R[[x]]$  such that  $[x^0](f - 1)$  is nilpotent with respect to  $R$ , define

$$\log(f) = - \sum_{i=1}^{\infty} \frac{(1-f)^i}{i}.$$

When  $R = \mathbb{R}$ , it is known that  $1/f$ ,  $\exp(f)$ , and  $\log(f)$  can be computed efficiently with the aid of Lemma 1 and the Newton method as below.

**Lemma 9 (Brent [9]).** *Let  $R = \mathbb{R}$ , and let  $f$  be a formal power series over  $R[[x]]$  such that the first  $t$  coefficients  $a_0, a_1, \dots, a_{t-1}$ , where  $a_i = [x^i]f$ , are given. Then the following statements hold.*

1. If  $f$  is a unit, the first  $t$  coefficients of  $1/f$  can be computed in  $O(t \log(t))$  time.
2. If  $a_0 = 1$ , then the first  $t$  coefficients of  $\log(f)$  can be computed in  $O(t \log(t))$  time.
3. If  $a_0 = 0$ , then the first  $t$  coefficients of  $\exp(f)$  can be computed in  $O(t \log(t))$  time.

Using the above lemma, together with the fact that  $f = \exp(\log(f))$ , Theorem 4 follows. See [12] for the details.

In this section, we show analogous results to Lemma 9 when  $R = \mathbb{R}[[y]]/(y^m)$ . The proof idea is similar to that of Lemma 9, but the difference is in computing the 0-th coefficients of  $\log(f)$  and  $\exp(f)$ , while it is trivial for the case when  $R = \mathbb{R}$ . We here provide the proof for completeness.

**Lemma 10.** *Let  $R = \mathbb{R}[[y]]/(y^m)$  for a positive integer  $m$ , and let  $f$  be a formal power series over  $R[[x]]$  such that the first  $t$  coefficients  $a_0, a_1, \dots, a_{t-1}$ , where  $a_i = [x^i]f$  in  $R$ , are given. Then the following statements hold.*

1. If  $f$  is a unit, the first  $t$  coefficients of  $1/f$  can be computed in  $O(mt \log(mt))$  time.
2. If  $a_0 - 1$  is nilpotent, then the first  $t$  coefficients of  $\log(f)$  can be computed in  $O(mt \log(mt))$  time.
3. If  $a_0$  is nilpotent, then the first  $t$  coefficients of  $\exp(f)$  can be computed in  $O(mt \log(mt))$  time.

*Proof.* (1) Let  $g^* = 1/f$ . Since  $f \cdot g^* = 1$ , we see that  $[x^0]g^* = 1/a_0$ . Since  $a_0$  is a unit in  $R = \mathbb{R}[[y]]/(y^m)$ , the inverse  $1/a_0$  can be computed in  $O(m \log(m))$  time by Lemma 9. Let  $g_1 = 1/a_0$ . Then  $g_1 \equiv g^* \pmod{x^1}$ .

Suppose that, for some integer  $k$ , we have  $g_k$  in  $R[[x]]$  such that  $[x^i]g_k = [x^i]g^*$  for any  $i < k$  and  $[x^i]g_k = 0$  for any  $i > k$ . This means that  $g_k \equiv g^* \pmod{x^k}$ . We also observe that, since  $[x^i](f \cdot g_k) = [x^i](f \cdot g^*)$  for any  $i < k$ , we have  $f \cdot g_k \equiv 1 \pmod{x^k}$ .

We will show that, using  $g_k$ , we can obtain  $g_{2k}$  in  $R[[x]]$  such that  $g_{2k} \equiv g^* \pmod{x^{2k}}$ . Specifically, given  $g_k$  in  $R[[x]]$ , we define  $g_{2k}$  as

$$g_{2k} \equiv 2g_k - fg_k^2 \pmod{x^{2k}}. \quad (4)$$

We show that  $g_{2k}$  defined above satisfies  $g_{2k} \equiv g^* \pmod{x^{2k}}$ . Since  $f \cdot g_k \equiv 1 \pmod{x^k}$ , (4) implies that  $g_{2k} \equiv 2g_k - g_k = g_k \pmod{x^k}$ . Hence it holds that  $(g_{2k} - g_k)^2 \equiv 0 \pmod{x^{2k}}$ , which is equivalent to

$$g_k^2 \equiv -g_{2k}^2 + 2g_{2k}g_k \pmod{x^{2k}}.$$

Using this to (4), we obtain

$$\begin{aligned} g_{2k} &\equiv 2g_k + f \cdot (g_{2k}^2 - 2g_{2k}g_k) \pmod{x^{2k}} \\ \iff (g_{2k} - 2g_k) &\equiv f \cdot g_{2k} \cdot (g_{2k} - 2g_k) \pmod{x^{2k}}. \end{aligned} \quad (5)$$

Recall that  $g^*$  has the inverse, and hence  $[x^0]g^*$  has the inverse in  $R$ . Since  $[x^0](g_{2k} - 2g_k) = -[x^0]g_k = -[x^0]g^*$ , we see that  $(g_{2k} - 2g_k)$  also has the inverse. Therefore, (5) implies that  $f \cdot g_{2k} \equiv 1 \pmod{x^{2k}}$ . This means that  $g_{2k} \equiv g^* \pmod{x^{2k}}$ .

Thus we can compute  $g_{2k}$  from  $g_k$  by (4), which takes  $O(mk \log(mk))$  time by Lemma 8. We repeat this procedure until the integer  $k$  is larger than or equal to  $t$ . In the end, we can obtain a desired one  $g_t$  in  $R[[x]]$  such that  $[x^i]g_t = [x^i]g^*$  for any  $i < t$ .

Let  $T(t)$  be the time complexity to compute  $g_t$ . Then we have

$$T(t) = T(t/2) + O(mt \log(mt)) = O(mt \log(mt)).$$

Thus the lemma (1) holds.

**(2)** It is known that  $\log(f) = \int f'/f$ , where  $f'$  is the differential of  $f$  and  $\int$  is an integral (See e.g., [3] for the details). By definition, it holds that  $f' = \sum_{i=1}^{\infty} ia_i x^{i-1}$ . Hence we know the first  $t-1$  coefficients of  $f'$ ,  $[x^0]f'$ ,  $[x]f'$ ,  $\dots$ ,  $[x^{t-2}]f'$ , from  $a_1, \dots, a_{t-1}$ . Also, the first  $t$  coefficients of  $1/f$  can be computed in  $O(mt \log(mt))$  time by the first statement of this lemma.

Let  $f' \cdot (1/f) = \sum_{i=0}^{\infty} b_i x^i$ . Then it follows from Lemma 8 that the first  $t-1$  coefficients  $b_0, \dots, b_{t-2}$  can be computed in  $O(mt \log(mt))$  time. We see that

$$\int \frac{f'}{f} = \sum_{i=1}^{\infty} \frac{b_{i-1}}{i} x^i + [x^0] \log(f).$$

Therefore, since  $\log(f) = \int f'/f$ , we can obtain  $[x^1] \log(f), \dots, [x^{t-1}] \log(f)$  from  $b_0, \dots, b_{t-2}$ . The time complexity of this step is  $O(mt \log(mt))$ .

It remains to compute  $[x^0] \log(f)$ . We observe that  $[x^0] \log(f) = \log([x^0]f)$ . In fact, it holds by definition that

$$[x^0] \log(f) = [x^0] \left( - \sum_{i=1}^{\infty} \frac{(1-f)^i}{i} \right) = - \sum_{i=1}^{\infty} [x^0] \frac{(1-f)^i}{i} = - \sum_{i=1}^{\infty} \frac{(1-a_0)^i}{i},$$

where the last equation follows from  $[x^0](1-f)^i = (1-a_0)^i$ . The most right-hand side, which is  $\log(a_0)$ , is an element of  $\mathbb{R}[[y]]/(y^m)$ , and hence it can be computed in  $O(m \log(m))$  time by Lemma 9. Thus,  $[x^0] \log(f)$  can be obtained in  $O(m \log(m))$  time.

Therefore, all the first  $t$  coefficients of  $\log(f)$  can be calculated in  $O(mt \log(mt))$  time in total, which completes the proof of the lemma (2).

(3) Let  $g^* = \exp(f)$ . Then  $\log(g^*) = f$ . Also, define  $G(h) = \log(h) - f$ .

We first observe that  $[x^0] \exp(f) = \exp([x^0]f)$ , since we have by definition

$$[x^0] \exp(f) = [x^0] \left( \sum_{i=0}^{\infty} \frac{f^i}{i!} \right) = \sum_{i=0}^{\infty} \frac{[x^0]f^i}{i!} = \sum_{i=0}^{\infty} \frac{a_0^i}{i!} = \exp([x^0]f).$$

Since  $[x^0]f$  is an element of  $R = \mathbb{R}[[y]]/(y^m)$ ,  $\exp([x^0]f)$  can be computed in  $O(m \log(m))$  time by Lemma 9 applied to  $\mathbb{R}[[y]]$ . We set  $g_1 = \exp([x^0]f)$ .

Assume that, for some integer  $k$ , we are given a formal power series  $g_k$  in  $R[[x]]$  that satisfies  $g_k \equiv g^*$  (mod  $x^k$ ). In other words,  $g_k$  satisfies that  $G(g_k) \equiv 0$  (mod  $x^k$ ). Then we define  $g_{2k}$  in  $R[[x]]$  as

$$g_{2k} \equiv g_k - \frac{G(g_k)}{G'(g_k)} \pmod{x^{2k}}, \quad (6)$$

where  $G'$  is the differential of  $G$ .

We shall show that  $G(g_{2k}) \equiv 0$  (mod  $x^{2k}$ ). Since  $G(g_k) \equiv 0$  (mod  $x^k$ ), it holds that  $g_{2k} \equiv g_k$  (mod  $x^k$ ) by (6). Hence we have  $(g_{2k} - g_k)^2 \equiv 0$  (mod  $x^{2k}$ ). Take the Taylor expansion of  $G$  at  $g_k$ , that is,

$$G(f_0) = G(g_k) + G'(g_k)(f_0 - g_k) + O((f_0 - g_k)^2)$$

for a formal power series  $f_0$ . This implies that

$$\begin{aligned} G(g_{2k}) &= G(g_k) + G'(g_k)(g_{2k} - g_k) + O((g_{2k} - g_k)^2) \\ &\equiv G(g_k) + G'(g_k)(g_{2k} - g_k) \pmod{x^{2k}}, \end{aligned}$$

since  $(g_{2k} - g_k)^2 \equiv 0$  (mod  $x^{2k}$ ). Substituting (6) for  $g_{2k}$  in the equation, we have  $G(g_{2k}) \equiv 0$  (mod  $x^{2k}$ ).

Since  $G'(g_k) = 1/g_k$ , (6) is equal to

$$g_{2k} \equiv g_k - g_k(\log(g_k) - f) = g_k(1 - \log(g_k) + f) \pmod{x^{2k}}.$$

Therefore,  $g_{2k}$  can be computed from  $g_k$  in  $O(mk \log(mk))$  time by Lemma 8.

We repeat the above procedure until the integer  $k$  is larger than or equal to  $t$ . Then we can compute  $g_t$  such that  $G(g_t) \equiv 0$  (mod  $x^t$ ), which is a desired one, as  $G(g_t) \equiv 0$  (mod  $x^t$ ) means that  $\log(g_t) \equiv f$  (mod  $x^t$ ). Let  $T(t)$  be the time complexity to compute  $g_t$ . Then we have

$$T(t) = T(t/2) + O(mt \log(mt)) = O(mt \log(mt)).$$

Thus the total complexity is  $O(mt \log(mt))$  time.  $\square$

We note that the above lemma does not hold for a general commutative ring  $R$ , as it might happen that  $[x^i]f$  is an infinite sum, and moreover, it is not clear how to compute the 0-th coefficients.

We are now ready to prove Theorem 4.

*Proof (Proof of Theorem 4).* Recall that  $f = \prod_{i=1}^n (1 + yx^{s_i})$  is a polynomial over the real field  $\mathbb{R}$ . We consider  $f$  as an element of the formal power series ring  $R[[x]]$  where  $R = \mathbb{R}[[y]]/(y^m)$ , and compute the first  $t$  coefficients of  $f \pmod{y^m}$  using the relationship  $f = \exp(\log(f))$ . We may assume that  $s_i \leq t$  for every  $i$ , as otherwise we may remove  $(1 + yx^{s_i})$  for  $s_i > t$  from  $f$ .

We first compute the first  $t$  coefficients of  $\log(f)$  as follows. It holds that

$$\log(f) = \log \prod_{i=1}^n (1 + yx^{s_i}) = \sum_{i=1}^n \log(1 + yx^{s_i}). \quad (7)$$

Thus the first  $t$  coefficients of  $\log(f)$  can be computed from those of  $\log(1 + yx^{s_i})$  with different exponent  $s_i$ . By the definition of the logarithm, we have

$$\log(1 + yx^{s_i}) = \sum_{j=1}^{\infty} (-1)^j \frac{(yx^{s_i})^j}{j} = \sum_{j=1}^{\infty} \frac{(-1)^j y^j}{j} x^{s_i j}.$$

Then  $\log(1 + yx^{s_i})$  has at most  $\frac{t}{s_i}$  non-zero coefficients in the first  $t$  coefficients, where each coefficient has only one monomial in  $y$ . Hence the first  $t$  coefficients of  $\log(1 + yx^{s_i})$  can be computed in  $O\left(\frac{t}{s_i}\right)$  time. Therefore, since  $s_i \leq t$  for every  $i$ , it follows from (7) that the first  $t$  coefficients of  $\log(f)$  can be computed in time

$$O\left(\sum_{s=1}^t \frac{t}{s}\right) = O(t \log(t)).$$

Finally, since  $f = \exp(\log(f))$ , we can compute the first  $t$  coefficients of  $f$  in  $O(mt \log(mt))$  time by Lemma 10. Thus the theorem holds, as reading the input takes  $O(n)$  time.  $\square$

## 5 Conclusion

In this paper, we proposed fast pseudo-polynomial-time algorithms for computing power indices in weighted majority games. Our algorithms calculate the Banzhaf index and the Shapley–Shubik index for all players in  $O(n + q \log(q))$  time and  $O(nq \log(q))$  time, respectively, where  $n$  is the number of players and  $q$  is a given quota. Our algorithms are faster than existing algorithms when  $q = 2^{o(n)}$ .

We remark that our algorithms ignore the time complexity to deal with high-precision integers, following the literature such as [20,26]. As the numerators of the Banzhaf and Shapley–Shubik indices may be  $2^{\Omega(n)}$ , the time complexity in the standard computation model would increase by a factor of  $n$ . On the other hand, we can avoid arithmetic operations of large integers by applying our algorithm  $O(n)$  times as follows. Let  $\alpha_1, \dots, \alpha_\ell$  be prime numbers such that  $q < \alpha_i \leq n$  for each  $i$  and  $2^n < \prod_i \alpha_i$ , where  $\ell \leq n$ . Then, for each  $i$ , we perform our algorithm over a prime field  $\mathbb{F}_{\alpha_i}$  to obtain the output modulo  $\alpha_i$ . It follows from the Chinese Remainder Theorem that we can uniquely recover the exact value of the output from these remainders.

In weighted majority games, other indices are proposed to measure the voting power of each player, such as the Deegan–Packel index and the Public Good index. See e.g., [2,16,25] and references therein. It is known that most of the existing approaches for computing the Banzhaf and Shapley–Shubik indices can be adapted to these indices. It would be interesting if our approach can be used to calculate these power indices.

**Acknowledgments.** This work was supported by JSPS KAKENHI Grant Numbers 22H05001, 23K21646, and JST ERATO Grant Number JPMJER2301, Japan.

## References

1. Algaba, E., Bilbao, J.M., Fernández, J.R.: The distribution of power in the European Constitution. *European Journal of Operational Research* **176**(3), 1752–1766 (2007). <https://doi.org/https://doi.org/10.1016/j.ejor.2005.12.002>, <https://www.sciencedirect.com/science/article/pii/S0377221705008805>
2. Alonso-Mejide, J.M., Freixas, J., Molinero, X.: Computation of several power indices by generating functions. *Applied Mathematics and Computation* **219**(8), 3395–3402 (2012). <https://doi.org/https://doi.org/10.1016/j.amc.2012.10.021>, <https://www.sciencedirect.com/science/article/pii/S0096300312010089>
3. Ardila, F.: Algebraic and geometric methods in enumerative combinatorics. *Handbook of Enumerative Combinatorics* pp. 3–172 (2015)
4. Bachrach, Y., Markakis, E., Resnick, E., Procaccia, A.D., Rosenschein, J.S., Saberi, A.: Approximating power indices: theoretical and empirical analysis. *Auton. Agents Multi Agent Syst.* **20**(2), 105–122 (2010). <https://doi.org/10.1007/S10458-009-9078-9>, <https://doi.org/10.1007/s10458-009-9078-9>
5. Banzhaf, J.F.: Weighted voting doesn't work: A mathematical analysis. *Rutgers Law Review* **19**(2), 317–343 (1965)
6. Bilbao, J.M., Fernandez, J.R., Jiménez-Losada, A., López, J.: Generating functions for computing power indices efficiently. *Top* **8**(2), 191–213 (2000)
7. Bolus, S.: Power indices of simple games and vector-weighted majority games by means of binary decision diagrams. *European Journal of Operational Research* **210**(2), 258–272 (2011). <https://doi.org/https://doi.org/10.1016/j.ejor.2010.09.020>, <https://www.sciencedirect.com/science/article/pii/S0377221710006181>
8. Brams, S.J., Affuso, P.J.: Power and size: A new paradox. *Theory and Decision* **7**(1), 29–56 (1976)
9. Brent, R.P.: Multiple-precision zero-finding methods and the complexity of elementary function evaluation. In: Traub, J. (ed.) *Analytic Computational Complexity*, pp. 151–176. Academic Press (1976). <https://doi.org/https://doi.org/10.1016/B978-0-12-697560-4.50014-9>, <https://www.sciencedirect.com/science/article/pii/B9780126975604500149>
10. Deng, X., Papadimitriou, C.H.: On the complexity of cooperative solution concepts. *Mathematics of Operations Research* **19**(2), 257–266 (1994). <https://doi.org/10.1287/MOOR.19.2.257>, <https://doi.org/10.1287/moor.19.2.257>
11. Elkind, E., Goldberg, L.A., Goldberg, P.W., Wooldridge, M.J.: Computational complexity of weighted threshold games. In: Proceedings of the Twenty-Second AAAI Conference on Artificial Intelligence. pp. 718–723. AAAI Press (2007), <http://www.aaai.org/Library/AAAI/2007/aaai07-114.php>
12. Jin, C., Wu, H.: A simple near-linear pseudopolynomial time randomized algorithm for subset sum. In: Fineman, J.T., Mitzenmacher, M. (eds.) 2nd Symposium on Simplicity in Algorithms, SOSA 2019. OASIcs, vol. 69, pp. 17:1–17:6. Schloss Dagstuhl - Leibniz-Zentrum für Informatik (2019). <https://doi.org/10.4230/OASIcs.SOSA.2019.17>, <https://doi.org/10.4230/OASIcs.SOSA.2019.17>
13. Á. Kóczy, L.: Beyond lisbon: Demographic trends and voting power in the European Union Council of Ministers. *Mathematical Social Sciences* **63**(2), 152–158 (2012). <https://doi.org/https://doi.org/10.1016/j.mathsocsci.2011.08.005>, <https://www.sciencedirect.com/science/article/pii/S0165489611000916>, around the Cambridge Compromise: Apportionment in Theory and Practice
14. Kleinberg, J., Tardos, É.: Algorithm Design. Pearson Education India (2006)
15. Klinz, B., Woeginger, G.J.: Faster algorithms for computing power indices in weighted voting games. *Mathematical Social Sciences* **49**(1), 111–116 (2005). <https://doi.org/https://doi.org/10.1016/j.mathsocsci.2004.06.002>, <https://www.sciencedirect.com/science/article/pii/S016548960400068X>

16. Kurz, S.: Computing the power distribution in the IMF. arXiv preprint arXiv:1603.01443 (2016)
17. Lucas, W.F.: Measuring Power in Weighted Voting Systems, pp. 183–238. Springer New York, New York, NY (1983). [https://doi.org/10.1007/978-1-4612-5430-0\\_9](https://doi.org/10.1007/978-1-4612-5430-0_9), [https://doi.org/10.1007/978-1-4612-5430-0\\_9](https://doi.org/10.1007/978-1-4612-5430-0_9)
18. Mann, I., Shapley, L.S.: Values of Large Games, IV: Evaluating the Electoral College by Montecarlo Techniques. RAND Corporation, Santa Monica, CA (1960)
19. Mann, I., Shapley, L.S.: Values of Large Games, VI: Evaluating the Electoral College Exactly. RAND Corporation, Santa Monica, CA (1962)
20. Matsui, T., Matsui, Y.: A survey of algorithms for calculating power indices of weighted majority games. Journal of the Operations Research Society of Japan **43**(1), 71–86 (2000). <https://doi.org/10.15807/jorsj.43.71>
21. Matsui, Y., Matsui, T.: NP-completeness for calculating power indices of weighted majority games. Theor. Comput. Sci. **263**(1-2), 305–310 (2001). [https://doi.org/10.1016/S0304-3975\(00\)00251-6](https://doi.org/10.1016/S0304-3975(00)00251-6), [https://doi.org/10.1016/S0304-3975\(00\)00251-6](https://doi.org/10.1016/S0304-3975(00)00251-6)
22. Prasad, K., Kelly, J.S.: NP-completeness of some problems concerning voting games. International Journal of Game Theory **19**(1), 1–9 (Mar 1990). <https://doi.org/10.1007/BF01753703>, <https://doi.org/10.1007/BF01753703>
23. Shapley, L.S.: A value for n-person games. In: Kuhn, H.W., Tucker, A.W. (eds.) Contributions to the Theory of Games II, pp. 307–317. Princeton University Press, Princeton (1953)
24. Shapley, L.S., Shubik, M.: A method for evaluating the distribution of power in a committee system. The American Political Science Review **48**(3), 787–792 (1954), <http://www.jstor.org/stable/1951053>
25. Staudacher, J., Kóczy, L.Á., Stach, I., Philipp, J., Kramer, M., Noffke, T., Olsson, L., Pichler, J., Singer, T.: Computing power indices for weighted voting games via dynamic programming. Operations Research and Decisions **31**(2) (2021). <https://doi.org/10.37190/ORD210206>, <https://doi.org/10.37190/ord210206>
26. Uno, T.: Efficient computation of power indices for weighted majority games. In: Chao, K., Hsu, T., Lee, D. (eds.) 23rd International Symposium on Algorithms and Computation, ISAAC 2012. Lecture Notes in Computer Science, vol. 7676, pp. 679–689. Springer (2012). [https://doi.org/10.1007/978-3-642-35261-4\\_70](https://doi.org/10.1007/978-3-642-35261-4_70), [https://doi.org/10.1007/978-3-642-35261-4\\_70](https://doi.org/10.1007/978-3-642-35261-4_70)
27. Ushioda, Y., Tanaka, M., Matsui, T.: Monte carlo methods for the shapley-shubik power index. Games **13**(3), 44 (2022). <https://doi.org/10.3390/G13030044>, <https://doi.org/10.3390/g13030044>