O P E R A T I O N S R E S E A R C H A N D D E C I S I O N S
No. 2 2021
DOI: 10.37190/ord210206

COMPUTING POWER INDICES FOR WEIGHTED VOTING GAMES VIA DYNAMIC PROGRAMMING

JOCHEN STAUDACHER[1*], LÁSZLÓ Á. KÓCZY[2,3], IZABELLA STACH[4], JAN FILIPP[1], MARCUS KRAMER[1], TILL NOFFKE[1], LINUS OLSSON[1], JONAS PICHLER[1], TOBIAS SINGER[1]

[1]Fakultät Informatik, Hochschule Kempten, Bahnhofstr. 61, 87435 Kempten, Germany

[2]Institute of Economics, Centre for Economic and Regional Studies, Tóth Kálmán u. 4, H-1097 Budapest, Hungary

[3]Department of Finance, Budapest University of Technology and Economics, Magyar tudósok körútja 2, H-1112 Budapest, Hungary

[4]AGH University of Science and Technology, al. Mickiewicza 30, 30-059 Kraków, Poland

We study the efficient computation of power indices for weighted voting games using the paradigm of dynamic programming. We survey the state-of-the-art algorithms for computing the Banzhaf and Shapley–Shubik indices and point out how these approaches carry over to related power indices. Within a unified framework, we present new efficient algorithms for the Public Good index and a recently proposed power index based on minimal winning coalitions of the smallest size, as well as a very first method for computing the Johnston indices for weighted voting games efficiently. We introduce a software package providing fast C++ implementations of all the power indices mentioned in this article, discuss computing times, as well as storage requirements.

Keywords: *cooperative game theory, power indices, weighted voting games, dynamic programming, minimal winning coalitions*

1. Introduction

Decision-making and voting in committees are frequently modelled using weighted voting games (also known as weighted majority games) for n players. Each player i is assigned a positive weight w_i which in some situations may come from the number of votes of a voting block. A law or motion gets passed in the committee if a certain quota q, normally more than 50% of the sum of all weights, is reached or exceeded. Power indices provide measures for evaluating the power or influence of individual players. As

*Corresponding author, email address: jochen.staudacher@hs-kempten.de
Received 1 November 2020, accepted 19 April 2021

based on different bargaining models and different axiomatic assumptions, various power indices have been introduced in the literature, see, e.g., [9].

Weighted voting games together with power indices as a tool for their analysis enjoy a wide range of applications well beyond classical voting situations in politics, see, e.g., [2, 24, 26] for the latter. For example, power indices can also be used to rank genes that may be responsible for genetic diseases [27], to understand indirect control power in corporate shareholding structures [11, 30, 31], or to analyse social networks [21]. The fact that the number n of players in these applications may be large and that the computation of any power index presented in this article is NP-hard [29] make the efficient computation of power indices challenging. Plenty of attention has been devoted to generating functions for computing power indices, see, e.g., [1, 2, 12]. This approach benefits strongly if the subsets of players attain only very few different weight sums [26]. In the latter case, one can exploit fast-access data structures for polynomials with few coefficients in computer algebra systems like Mathematica [35]. This article studies the strongly related, though mathematically less sophisticated, paradigm of dynamic programming for the computation of power indices [28, 36]. In Section 2, we introduce 12 power indices investigated in this paper together with a small example. Section 3 explains how coalitions are counted via dynamic programming. In Section 4, we survey the state-of-the-art algorithms for computing the Banzhaf index [4] and the Shapley–Shubik index [33], along the lines of Kurz [26], and point out how these ideas carry over to other power indices. In Section 5, we study power indices based on minimal winning coalitions and present new recursions. We devise new efficient algorithms for computing the Public Good indices [20] of all players in $O(qn)$ time, as well as for the power index based on minimal winning coalitions of minimal size recently proposed by Felsenthal [18]. We present an $O(qn^2)$ algorithm for the Deegan–Packel index [16], devised by Uno [36], within our unified framework. In Section 6, we propose a very first efficient $O(qn^3)$ method for computing the Johnston index [22] and provide new valuable insight into how to count critical players within coalitions via dynamic programming. In Section 7, we introduce a powerful software package providing C++ implementations for all the power indices mentioned in this article, and discuss challenges like the necessity to process very large integers. Numerical experiments confirm our classification of power indices according to the different pseudopolynomial complexities of their computation.

2. Weighted voting games and power indices

A cooperative n-person game is defined by a set of n players $N = \{1, ..., n\}$ and a characteristic function v: $2^N \to \mathbb{R}$ assigning each subset $S \in 2^N$ a real value with

$v(\varnothing)=0$. N is called the grand coalition and $|S|$ denotes the cardinality of the set *S*. Weighted voting games are defined by *n* non-negative real weights $w_i, i=1,...,n,$ and a non-negative real quota *q*. They are simple games, meaning that the characteristic function *v* takes only the values 0 or 1, i.e., v: $2^N \to \{0,1\}$. Specifically, $v(S)=1$ if a coalition *S* is winning, i.e., $w(S)=\sum_{i\in S} w_i \geq q,$ and $v(S)=0$ otherwise, meaning coalition *S* is losing. A player *i* is called a critical player (also known as a decisive player or swing player) for a coalition *S* if $v(S)=1$ and $v(S\setminus\{i\})=0,$ i.e., by leaving the coalition *S* player *i* turns a winning coalition into a losing one. $Cr(S)$ is the set of critical players in a coalition *S*. A player *i* who is not critical for any coalition $S\in 2^N$, is called a null player. A coalition *S* containing at least one critical player is referred to as a vulnerable coalition. A coalition *S* is called a minimal winning coalition if each member of *S* is a critical player. A power index *f* is a function mapping a unique vector $f(v)=(f_1(v),...,f_N(v))$ to a given simple cooperative game specified by the player set *N* and the characteristic function *v*. We call *f* efficient if $\sum_{i=1}^{n} f_i(v)=v(N)=1.$

Definition 1. Let *v* be a simple *n*-player game, let W, W^{np}, W^m, and W^s denote the sets of winning coalitions, null player free winning coalitions, minimal winning coalitions and minimal winning coalitions of smallest cardinality, respectively, and W_i, W_i^{np}, W_i^m, and W_i^s the corresponding subsets containing player *i*. Further, let *VC* denote the set of vulnerable coalitions, $\eta_i(v)$ the number of coalitions for which *i* is a critical player, and $\eta_i(v,c)$ the number of coalitions of cardinality *c* for which *i* is a critical player.

A. The (absolute) Banzhaf index [4, 8] of player *i* is defined as

$$\beta_i=\frac{\eta_i(v)}{2^{n-1}}$$

The relative Banzhaf index [8] defined as $\beta_i'=\frac{\eta_i(v)}{\sum_{k=1}^{n}\eta_k(v)}$ is frequently used as an efficient counterpart, i.e., $\sum_{i=1}^{n}\beta_i'=1.$

B. The Deegan–Packel index [16] of a player i is defined as

$$DP_i = \frac{1}{|W^m|} \sum_{S \in W_i^m} \frac{1}{|S|}$$

C. The Felsenthal index [18] of a player i is defined as

$$PI_i = \frac{1}{|W^s|} \sum_{S \in W_i^s} \frac{1}{|S|}$$

D. The Johnston index [22] of a player i is defined as

$$\gamma_i = \frac{\sum_{S \in VC,\ i \in Cr(S)} \frac{1}{|Cr(S)|}}{\sum_{k=1}^{n} \sum_{S \in VC,\ k \in Cr(S)} \frac{1}{|Cr(S)|}}$$

if i is not a null player and $\gamma_i = 0$ otherwise.

E. The König–Bräuninger index [23] of a player i is defined as

$$KB_i = \frac{|W_i|}{|W|}$$

F. The Nevison index [32] of a player i is defined as

$$Z_i = \frac{|W_i|}{2^n}$$

G. The null player free index f^{np} [3] of a player i is defined as

$$f_i^{np} = \frac{1}{|W^{np}|} \sum_{S \in W_i^{np}} \frac{1}{|S|}$$

H. The null player free index g^{np} [3] of a player i is defined as

$$g_i^{np} = \frac{|W_i^{np}|}{\sum_{k=1}^{n} |W_k^{np}|}$$

I. The Public Good index (also known as the Holler index or PGI) [20] of a player i is defined as

$$h_i = \frac{\left|W_i^m\right|}{\sum_{k=1}^{n} |W_k^m|}$$

J. The Public Help index θ (also known as PHI θ) [7] of a player i is defined as

$$\theta_i = \frac{\left|W_i\right|}{\sum_{k=1}^{n} |W_k|}$$

K. The Public Help index ξ (also known as PHI ξ) [10] of a player i is defined as

$$\xi_i = \frac{1}{\sum_{T \in W} \frac{1}{T}} \sum_{S \in W_i} \frac{1}{|S|^2}$$

L. The Shapley–Shubik index [33, 34] of a player i is defined as

$$\phi_i = \sum_{c=1}^{n} \frac{\eta_i(v, c)}{c\binom{n}{c}}$$

Example 1. We study the 5-player weighted voting game with weights $w_1 = 4,\ w_2 = 3,$ $w_3 = w_4 = w_5 = 1,$ and quota $q = 6$. We derive the values of the 12 power indices from Definition 1 in the following. With the coalition $\{1, 2\}$, there exists only one minimal winning coalition of the smallest cardinality leading to the Felsenthal indices $PI_1 = PI_2 = \frac{1}{2}$, $PI_3 = PI_4 = PI_5 = 0$. With the additional minimal winning coalitions $\{1, 3, 4\}$, $\{1, 3, 5\}$, $\{1, 4, 5\}$, and $\{2, 3, 4, 5\}$ we find the Deegan–Packel indices $DP_1 = 0.3,\ DP_2 = 0.15,$ $DP_3 = DP_4 = DP_5 = \frac{11}{60}$, and the Public Good indices $h_1 = \frac{4}{15},\ h_2 = \frac{2}{15},\ h_3 = h_4 = h_5 = 0.2.$ We need to include the additional 8 winning coalitions to derive the other indices. Players 1 and 2 are critical in $\{1, 2, 3\}$, $\{1, 2, 4\}$, and $\{1, 2, 5\}$, player 1 is the only critical player

in $\{1, 2, 3, 4\}$, $\{1, 2, 3, 5\}$, $\{1, 2, 4, 5\}, \{1, 3, 4, 5\}$, and there is no critical player in $N = \{1, 2, 3, 4, 5\}$. With this information, we find absolute Banzhaf indices: $\beta_1 = \frac{11}{16}$, $\beta_2 = \frac{5}{16}$, $\beta_3 = \beta_4 = \beta_5 = \frac{3}{16}$, Johnston indices: $\gamma_1 = \frac{7}{12}$, $\gamma_2 = \frac{3}{16}$, $\gamma_3 = \gamma_4 = \gamma_5 = \frac{11}{144}$, König–Bräuninger indices: $KB_1 = \frac{12}{13}$, $KB_2 = \frac{9}{13}$, $KB_3 = KB_4 = KB_5 = \frac{8}{13}$, Nevison indices: $Z_1 = \frac{3}{4}$, $Z_2 = \frac{9}{16}$, $Z_3 = Z_4 = Z_5 = \frac{1}{2}$, null player free indices f^{np}: $f_1^{np} = \frac{37}{130}$, $f_2^{np} = \frac{27}{130}$, $f_3^{np} = f_4^{np} = f_5^{np} = \frac{11}{65}$, null player free indices g^{np}: $g_1^{np} = \frac{4}{15}$, $g_2^{np} = 0.2$, $g_3^{np} = g_4^{np} = g_5^{np} = \frac{8}{45}$, Public Help indices θ: $\theta_1 = \frac{4}{15}$, $\theta_2 = \frac{1}{5}$, $\theta_3 = \theta_4 = \theta_5 = \frac{8}{45}$, Public Help indices ξ: $\xi_1 = \frac{362}{1185}$, $\xi_2 = \frac{262}{1185}$, $\xi_3 = \xi_4 = \xi_5 = \frac{187}{1185}$, Shapley–Shubik indices: $\phi_1 = \frac{9}{20}$, $\phi_2 = \frac{1}{5}$, $\phi_3 = \phi_4 = \phi_5 = \frac{7}{60}$. Note that the Public Help index θ and the null player free index g^{np} need to coincide because our example does not contain any null players.

3. Counting coalitions via dynamic programming

Every weighted voting game allows an integer representation [26]. Hence, we assume that the weights w_i of n players in our weighted voting game as well as the quota q are positive integers. We set $\tilde{w} = w(N) = \sum_{i=1}^{n} w_i$ and assume $q > \frac{\tilde{w}}{2}$.

The idea of dynamic programming is to solve a problem algorithmically, by dividing it into subproblems and storing intermediate results efficiently. We can employ this paradigm to solve the following question: How many subsets $S \subseteq \{1, 2, ..., n\}$ are there with weight x, i.e., $w(S) = \sum_{i \in S} w_i = x$ for $x \in \{0, 1, ..., \tilde{w}\}$?

Theorem 1 [15], p. 229. Let $T(i, x)$ be the number of possibilities to write the integer x as a sum of the first i weights $w_1, ..., w_i$. For all $i \in \{0, ..., n\}$ and all $x \in \{0,1 , ..., \tilde{w}\}$ there holds

$$T(i, 0) = 1 \text{ for } 0 \le i \le n$$
$$T(0, x) = 0 \text{ for } x > 0$$
$$T(i, x) = T(i-1, x) + T(i-1, x - w_i) \text{ otherwise} \square$$

Note that the above equations come down to a boundary condition, stating that we can obtain the sum 0 in exactly one way, i.e., via the empty set, to another boundary condition, stating that we cannot obtain any sum $x > 0$ without any term and an actual recursion reflecting that the first i weights can deliver a sum $x > 0$ either with or without player *i*.

Example 2. Let us compute $T(i, x)$ for the three weights $w_1 = 4,\ w_2 = 2,\ w_3 = 1$. We list only the non-zero values for weights $x > 0$ in every step. $i = 1$ leads to $T(1, 4) = 1$. $i = 2$ leads to $T(2, 2) = T(2, 4) = T(2, 6) = 1,\ i = 3$ leads to $T(3, 1) = T(3, 2) = T(3, 3) = T(3, 4) = T(3, 5) = T(3, 6) = T(3, 7) = 1$.

Theorem 1 can easily be generalised for the distribution of cardinalities of coalitions.

Theorem 2 [15], p. 231. Let $C(i, x, c)$ be the number of possibilities to write the integer x as a sum of weights of a coalition $S \subseteq \{1, ..., i\}$ with cardinality c. For all i, $c \in \{0, ..., n\}$ and all $x \in \{0, 1, ..., \tilde{w}\}$ there holds

$$C(i, 0, c) = 1 \text{ for } 0 \le i \le n \text{ and } 0 \le c \le n$$
$$C(0, x, c) = 0 \text{ for } 1 \le x \le \tilde{w} \text{ and } 1 \le c \le n$$
$$C(i, x, 0) = 0 \text{ for } 1 \le i \le n \text{ and } 1 \le x \le \tilde{w}$$
$$C(i, x, c) = C(i-1, x, c) \text{ for } 1 \le i \le n,\ 1 \le x < w_i \text{ and } 1 \le c \le n$$
$$C(i, x, c) = C(i-1, x, c) + C(i-1, x - w_i, c-1) \text{ otherwise} \square$$

Example 3. Let us revisit Example 2 and compute $C(i, x, c)$ for the three weights $w_1 = 4,\ w_2 = 2,\ w_3 = 1$ listing the non-zero values for weights $x > 0$ and cardinalities $c > 0$. $i = 1$ leads to $C(1, 4, 1) = 1,\ i = 2$ leads to $C(2, 2, 1) = C(2, 4, 1) = 1$, and $C(2, 6, 2) = 1$, $i = 3$ leads to $C(3, 1, 1) = C(3, 2, 1) = C(3, 4, 1) = 1$ as well as to $C(3, 3, 2) = C(3, 5, 2) = C(3, 6, 2) = 1$ and $C(3, 7, 3) = 1$, the latter reflecting the sum of all three weights.

In practice, none of the following algorithms for computing power indices will require us to store *T* from Theorem 1 as a two-dimensional table. Instead, we shall always be able to work efficiently with a vector that we can update for *i*. Likewise, we never need to store *C*

from Theorem 2 as a three-dimensional array, allowing us to work with a matrix instead. It is the merit of Kurz [26] to have pointed out that rather than computing the quantities $T(n,x)$ and $C(n,x,c)$ from below we can equivalently compute them from above using $T(n,\tilde{w}) = C(n,\tilde{w},n) = 1$. As we assume $q > \frac{\tilde{w}}{2}$, this brings down computing times from $O(qn)$ and $O(qn^2)$ to $O(\Delta n)$ and $O(\Delta n^2)$ with $\Delta = \min(q, \tilde{w} - q + 1)$, respectively, in the following section.

4. The approach for power index computation by Kurz

The Banzhaf index [4] is one of the most important power indices. Uno [36] presents the first algorithm for computing the Banzhaf indices of all players in a weighted voting game in $O(qn)$ time. This result is improved by Kurz [26] to $\Delta = \min(q, \tilde{w} - q + 1)$. We summarise this result and point out that it almost trivially carries over to four other power indices from Definition 1.

Theorem 3. Let v be an n-player weighted voting game with positive integer weights. The quantities $\eta_i(v)$, $|W_i|$ and $|W_i^{np}|$ can be computed in $O(\Delta n)$ time and $O(\Delta)$ memory space for all the players. Hence, the following indices can be computed in $O(\Delta n)$ time and $O(\Delta)$ space for all players: Banzhaf, König–Bräuninger, Nevison, null player free index g^{np}, and Public Help θ.

Proof. We start with the proof for the Banzhaf indices – hence $\eta_i(v)$ – following Kurz [26]. It provides algorithmic insight into how $\eta_i(v)$ can be computed from either below or above. In the conceptually simpler case $q \le (\tilde{w} - q + 1)$, we compute the vector $T(n,x)$ for all weights $0 \le x \le q-1$ from below, according to Theorem 1. This can be done in $O(qn)$ time and $O(q)$ space. Next, we need to work out how frequently a player i is critical, i.e., how often i turns a losing coalition into a winning one. We set $T_{-i}(n,x) = 0$ for all $x \in \{0, 1, ..., w_i - 1\}$ and loop for x from w_i to $q-1$ to find

$$T_{-i}(n,x) = T(n,x) - T_{-i}(n, x - w_i)$$

Then, $T_{-i}(n,x)$ tells us how frequently a player *i* does not belong to a losing coalition with weight *x* and we obtain

$$\eta_i(v) = \sum_{x=q-w_i}^{q-1} T_{-i}(n,x)$$

In case $(\tilde{w}-q+1)<q$, we compute the vector $T(n,x)$ for $q \le x \le \tilde{w}$ from above, using the corresponding algorithm in the paper by Kurz [26] in $O((\tilde{w}-q+1)n)$ time and $O(\tilde{w}-q+1)$ space. We set $T_{+i}(n,x)=T(n,x)$ for all $x \in \{w-w_i+1,...,\tilde{w}\}$, and then loop for *x* from $\tilde{w}-w_i$ down to *q* to find

$$T_{+i}(n,x) = T(n,x) - T_{+i}(n,x+w_i)$$

Now, $T_{+i}(n,x)$ tells us how often player *i* belongs to a winning coalition with weight *x* and we obtain

$$\eta_i(v) = \sum_{x=q}^{q+w_i-1} T_{+i}(n,x)$$

We know $|W| = \sum_{x=q}^{\tilde{w}} T(n,x)$ (as stated in [26]). We need $|W_i|$ for the König–Bräuninger, Nevison, and Public Help θ indices. We can compute $|W_i| = \sum_{x=q}^{\tilde{w}} T_{+i}(n,x)$ or, alternatively, use the identity by Dubey and Shapley [17] and get $|W_i| = 0.5(|W| + \eta_i(v))$ from $\eta_i(v)$. For the computation of null-player free indices g^{np}, let us assume the *n* non-negative integer weights in descending order. Computing $\eta_i(v)$, we can find the number np of non-null players in a preprocessing step. Setting $\tilde{w}^{np} = \sum_{i=1}^{np} w_i$, we get

$$|W_i^{np}| = \sum_{x=q}^{\tilde{w}^{np}} T_{+i}(np,x)$$

for the non-null players and hence the null-player free indices g^{np} in $O(\Delta n)$ time. □

We mention in passing another fast algorithm from the paper by Matsui and Matsui [28] for finding all null players in $O(qn)$ time, see also the textbook [15], p. 234–235.

Next to the Banzhaf index, the Shapley–Shubik index [33] is arguably the most widely used power index. Uno [36] devises the first algorithm for computing the Shapley–Shubik indices of all players in a weighted voting game in $O(qn^2)$ time. It is improved by Kurz [26] to $O(\Delta n^2)$. We summarise this result and carry it over to two other power indices from Definition 1.

Theorem 4. Let v be an n-player weighted voting game with positive integer weights. The Shapley–Shubik index, the Public Help index ξ, and the null player free index f^{np} can be computed in $O(\Delta n^2)$ time and $O(\Delta n)$ memory space for all players.

Proof. The reasoning is very similar to the proof of Theorem 3. Again, we follow the proof for the Shapley–Shubik indices from the paper by Kurz [26], but strive to provide more algorithmic details in our presentation while focussing on the concepts rather than implementation issues. For the Shapley–Shubik indices, the challenge is to find $\eta_i(v,c)$. In the conceptually simpler case $q \le (\tilde{w}-q+1)$ we compute the matrix $C(n,x,c)$ for all weights $0 \le x \le q-1$ and all cardinalities $0 \le c \le n$ from below, according to Theorem 2. This can be done in $O(qn^2)$ time and $O(qn)$ space. Next, we need to work out how frequently player *i* is critical in a coalition of cardinality *c*, i.e., how often *i* turns a losing coalition of cardinality $c-1$ into a winning one. We set $C_{-i}(n,x,c)=0$ for all $x \in \{0,1,...,w_i-1\}$ and all cardinalities $c \in \{1,...,n\}$. We loop for *x* from w_i to $q-1$ in an outer loop and for *c* from *n* down to 1 in an inner loop in order to find

$$C_{-i}(n,x,c-1) = C(n,x,c) - C_{-i}(n,x-w_i,c)$$

Then, $C_{-i}(n,x,c-1)$ tells us how frequently player *i* does not belong to a losing coalition with weight *x* and cardinality *c* – 1. We obtain

$$\eta_i(v,c) = \sum_{x=q-w_i}^{q-1} C_{-i}(n,x,c-1)$$

In case $(\tilde{w}-q+1)<q,$ we compute the matrix $C(n, x, c)$ for $q \le x \le \tilde{w}$ and all cardinalities $1 \le c \le n$ from above using the algorithmic concepts from the paper by Kurz [26] in $O\left((\tilde{w}-q+1)n^2\right)$ time and $O\left((\tilde{w}-q+1)n\right)$ space. We set $C_{+i}(n, x, c) = C(n, x, c)$ for all weights $x \in \{w-w_i+1, ..., \tilde{w}\}$ and all cardinalities $c \in \{1, ..., n\}$. We loop for x from $\tilde{w}-w_i$ down to q in an outer loop and for c from 0 to $n-1$ in an inner loop to find

$$C_{+i}(n, x, c) = C(n, x, c) - C_{+i}(n, x+w_i, c+1)$$

Now, $C_{+i}(n, x, c)$ tells us how often player i belongs to a winning coalition with weight x and cardinality c and we obtain

$$\eta_i(v, c) = \sum_{x=q}^{q+w_i-1} C_{+i}(n, x, c)$$

We can compute the Public Help index ξ of player i from above in $O\left(\Delta n^2\right)$ time and $O(\Delta n)$ space using the quantity $\sum_{x=q}^{\tilde{w}} C_{+i}(n, x, c)$, i.e., the number of times player i is a part of a winning coalition of cardinality c. For the computation of null-player free indices f^{np}, let us assume the n non-negative integer weights in descending order. We find the number np of non-null players in a preprocessing step as in the proof of Theorem 3. Setting $\tilde{w}^{np} = \sum_{i=1}^{np} w_i,$ we get

$$\left|W^{np}\right| = \sum_{x=q}^{\tilde{w}^{np}} \sum_{c=1}^{np} C(np, x, c)$$

The number of times a non-null player i is part of a null-player free winning coalition of cardinality c is given as

$$\sum_{x=q}^{\tilde{w}^{np}} C_{+i}(np, x, c)$$

and we can compute the indices f^{np} for all players in $O\left(\Delta n^2\right)$ time and $O(\Delta n)$ space. □

5. Computing power indices based on minimal winning coalitions

Holler [20] argues that in many situations, only minimal winning coalitions will form, Felsenthal [18] even argues for minimal winning coalitions of least possible size. As for computing the Public Good index [20] for all players, the best dynamic programming algorithm in the literature would be $O(qn^2)$, finding $|W_i^m|$, according to the textbook [15], p. 235–238, whereas we note that $|W^m| = \sum_{i=1}^{n} \sum_{x=q-w_i}^{q-1} T(i-1, x)$ is computable in $O(qn)$ time. We suggest a new $O(qn)$ approach for the Public Good index which we carry over to the Felsenthal index [18], and put our new results in the context of an existing $O(qn^2)$ algorithm for the Deegan–Packel index [16] by Uno [36]. In this and the following section, we assume the positive integer weights of our n players to be in descending order, i.e., $w_1 \geq \ldots \geq w_n$.

Definition 2. Let v be an n-player weighted voting game with its positive integer weights sorted in descending order. For a coalition S, let the operator $d(S)$ remove the player with the largest index from S. Further, let $R_i = \{i, \ldots, n\}$ for all $1 \leq i \leq n$. Finally, let

$$B(i, x) = |\{S \in 2^N \mid S \subseteq R_i, i \in S, w(d(S)) < x \leq w(S)\}|$$

for all $1 \leq i \leq n$ and all weights $0 \leq x \leq q$. $B(i, x)$ is the number of coalitions S with i as the player with the smallest index such that any coalition S has a weight greater or equal x whereas S without its player with the largest index has the weight less than x.

We observe that $|W_i^m| = B(i, q)$ for all players. We can compute B by looping for i from n to 1, as follows.

Theorem 5. For all $1 \leq i \leq n$ and all weights $0 \leq x \leq q$ there holds

$$B(i, x) = \begin{cases} 1 \text{ for } 1 \leq x \leq w_i \\ B(i+1, x - w_i + w_{i+1}) + B(i+1, x - w_i) \text{ for } x > w_i, i < n \\ 0 \text{ otherwise} \end{cases}$$

Proof. The statement is true for $i = n$ and $x \le w_i$. In all other cases the recurrence relation states that either player $i+1$ is part of a coalition counted in $B(i, x)$ (second term) or player $i+1$ is not part of a coalition counted in $B(i, x)$ (first term).

$$\begin{aligned} B(i,x) &= |\{S \in 2^N \mid S \subseteq R_{i+1}, (i+1) \notin S, w(d(S)) < x - w_i \le w(S)\}| \\ &\quad + |\{S \in 2^N \mid S \subseteq R_{i+1}, (i+1) \in S, w(d(S)) < x - w_i \le w(S)\}| \\ &= |\{S \in 2^N \mid S \subseteq R_{i+1}, (i+1) \in S, w(d(S)) < x - w_i + w_{i+1} \le w(S)\}| \\ &\quad + |\{S \in 2^N \mid S \subseteq R_{i+1}, (i+1) \in S, w(d(S)) < x - w_i \le w(S)\}| \\ &= B(i+1, x - w_i + w_{i+1}) + B(i+1, x - w_i) \ \square \end{aligned}$$

Theorem 5 can easily be extended for cardinalities of coalitions.

Theorem 6. Let

$$\tilde{B}(i, x, c) = |\{S \in 2^N \mid S \subseteq R_i, i \in S, |S| = c, w(d(S)) < x \le w(S)\}|$$

for all $1 \le i \le n$, all weights $0 \le x \le q$, and all cardinalities $0 \le c \le n$. There holds

$$\tilde{B}(i, x, c) = \begin{cases} 1 \text{ for } 1 \le x \le w_i,\ c = 1 \\ \tilde{B}(i+1, x - w_i + w_{i+1}, c) + \tilde{B}(i+1, x - w_i, c-1) \text{ for } x > w_i, i < n \\ 0 \text{ otherwise} \end{cases}$$

Proof. The proof is almost identical to the proof of Theorem 5. □

Theorem 7. Let *v* be an *n*-player weighted voting game with positive integer weights sorted in descending order. Using *T* from Theorem 1 and *B* from Theorem 5, the Public Good index can be computed in $O(qn)$ time and $O(q)$ memory space for all players as there holds

$$|W_i^m| = \sum_{x=0}^{q} T(i-1, x) B(i, q-x)$$

Proof. The claim holds for $i = 1$. $|W_1^m| = T(0, 0) B(1, q) = B(1, q)$. In the case $i \ge 2$, we abbreviate $\overline{R_i} = N \setminus R_i = \{1, ..., i-1\}$ and observe

$$
\begin{aligned}
|W_i^m| &= |\{S \in 2^N \mid i \in S,\ w(d(S)) < q \le w(S)\}| \\
&= |\{S \in 2^N \mid S = S_1 \cup S_2,\ S_1 \subseteq \overline{R_i},\ S_2 \subseteq R_i,\ i \in S_2,\ w(d(S_2)) < q - w(S_1) \le w(S_2)\}| \\
&= \sum_{x=0}^{q} |\{S_1 \subseteq \overline{R_i} \mid w(S_1) = x\}| \, |\{S_2 \subseteq R_i \mid w(d(S_2)) < q - x \le w(S_2)\}| \\
&= \sum_{x=0}^{q} T(i-1, x)\ B(i, q-x)
\end{aligned}
$$

Algorithmically, we compute $T(n-1, x)$ for all weights according to Theorem 1. We find our way to $B(n-1, x)$ from $B(n, x)$, to $T(n-2, x)$ from $T(n-1, x)$, and so on, looping backwards over *i* until all $|W_i|$ are computed. This can be done in $O(qn)$ time and $O(q)$ space. □

Theorem 8. Let *v* be an *n*-player weighted voting game with positive integer weights sorted in a descending order. Let $\tilde{m}$ be the size of the smallest minimal winning coalition. Making use of *C* from Theorem 2 and $\tilde{B}$ from Theorem 6, the Felsenthal index can be computed in $O(q\tilde{m}n)$ time and $O(q\tilde{m})$ memory space for all players as there holds

$$
|W_i^s| = \sum_{x=0}^{q} \sum_{c=0}^{\tilde{m}} C(i-1, x, c)\, B(i, q-x, \tilde{m}-c)
$$

Proof. We find $\tilde{m} = \min\{i \in \{1, ..., n\} \mid \sum_{k=1}^{i} w_k \ge q\}$ in $O(n)$ time and $O(1)$ memory space. In working with *C* and $\tilde{B}$, we can restrict our efforts to cardinalities less or equal $\tilde{m}$. The rest of the proof is almost identical to the proof of Theorem 7. □

We observe that working out $\sum_{x=0}^{q} \sum_{c=0}^{s} C(i-1, x, c)\, \tilde{B}(i, q-x, s-c)$ for all possible cardinalities $\tilde{m} \le s \le n$ will not lead to finding the corresponding numbers of minimal winning coalitions of all players in $O(qn^2)$ time. Uno [36] suggests a recursion related to $\tilde{B}$ based on reciprocal values. We summarise the algorithm for the Deegan–Packel index.

Theorem 9 [36]. Let *v* be an *n*-player weighted voting game with positive integer weights sorted in descending order and let

$$\hat{B}(i,x,c)=\sum_{\{S\in 2^N \mid S\subseteq R_i,\, i\in S,\, w(d(S))<x\le w(S)\}}\frac{1}{|S|+c}$$

for all $1\le i\le n,$ all weights $0\le x\le q$ and all cardinalities $0\le c\le i.$ There holds

$$\hat{B}(i,x,c)=\begin{cases}1 \text{ for } 1\le x\le w_i\\ \hat{B}(i+1,x-w_i+w_{i+1},c)+\hat{B}(i+1,x-w_i,c+1) \text{ for } x>w_i,\, i<n\\ 0 \text{ otherwise}\end{cases}$$

Using *C* from Theorem 2, the following equality

$$DP_i\left|W^m\right|=\sum_{x=0}^{q}\sum_{c=0}^{i-1}C(i-1,x,c)\,\hat{B}(i,q-x,c)$$

guarantees that the Deegan–Packel index can be computed in $O(qn^2)$ time and $O(qn)$ memory space for all players. □

6. Computing the Johnston index

As stated in Definition 1, the Johnston index [22] relies on the numbers of critical players within a vulnerable coalition, i.e., the cardinalities of vulnerable coalitions are not sufficient to find $\gamma_i^{\text{raw}}=\sum_{S\in VC,\, i\in Cr(S)}\frac{1}{|Cr(S)|}$, often called the raw Johnston index, for each player *i*. The Johnston index is useful in practice, e.g., in indirect control of corporations [11, 30, 31] where problems are typically large and brute force calculation is not feasible. However, there exist neither dynamic programming algorithms nor has the Johnston index been tackled via quasi-ordered binary decision diagrams in [14]. Also, we are not aware of approximation algorithms. Our new approach for the Johnston index hinges on the critical player *i* with the largest index in a vulnerable coalition *S*. Once we remove all players with larger indices from *S*, we either end up with a minimal winning coalition with a surplus $\sigma=w(S)-q$ or with a losing coalition with a deficiency $\delta=q-w(S)$. We would now like to constructively work out how dummy players, i.e., non-critical players (with indices larger than *i*), can be added to the coalition *S*.

Definition 3. We assume n positive integer weights in descending order, i.e., $w_1 \geq \ldots \geq w_n$. We input these weights in reverse order into the recursion from Theorem 1 to obtain $T^*(i, x)$ for all $i \in \{0, \ldots, n-1\}$ and all $x \in \{0, \ldots, q-1\}$ (with the notation T^* supposed to indicate that we simply obtain T from Theorem 1 but input the weights for T in reverse order). We define $I: \{0, \ldots, q\} \to \{0, \ldots, n\}$ with

$$I(k) = n - \min\{i \in \{1, \ldots, n\} \mid w_i \geq k\} + 1$$

for $1 \leq k \leq n$ and $I(0) = 0$.

Let us construct vulnerable coalitions S such that i is the critical player with the largest index, i.e., there is no critical player in S with a weight less than w_i. If S is minimal winning, then there is a surplus $\sigma = w(S) - q$ and we have $\sum_{l=0}^{w_i - \sigma - 1} T^*(I(l),\ l)$ ways to obtain vulnerable coalitions. If S is losing with a deficiency $\delta = q - w(S)$, there are $\sum_{l=\delta+1}^{w_i + \delta - 1} T^*(I(l-\delta),\ l)$ ways to construct vulnerable coalitions. In the latter formula, the upper bound $w_i + \delta - 1$ guarantees that i remains critical. An example appears to be appropriate.

Example 4. We revisit the 5-player weighted voting game with weights $w_1 = 4,\ w_2 = 3,$ $w_3 = w_4 = w_5 = 1$, and quota $q = 6$ from Example 1. One easily confirms that $T^*(0, 0) = 1,$ $T^*(1,\ 1) = 1,\ T^*(2,\ 1) = 2,\ T^*(2,\ 2) = 1,\ T^*(3,\ 1) = T^*(3,\ 2) = 3,\ T^*(3,\ 3) = 1,\ T^*(4,\ 1)$ $= T^*(4, 2) = 3, T^*(4, 3) = 2, T^*(4, 4) = T^*(4, 5) = 3$, and $T^*(i, x) = 0$ for all other $i \in \{0, \ldots, n-1\}$ and $x \in \{0, \ldots, q-1\}$. For the computation of the raw Johnston indices of players 3, 4, and 5, only the minimal winning coalitions $\{1, 3, 4\}$, $\{1, 3, 5\}$, $\{1, 4, 5\}$, and $\{2, 3, 4, 5\}$ matter and we obtain $\gamma_3^{\text{raw}} = \gamma_4^{\text{raw}} = \gamma_5^{\text{raw}} = 2 \times \frac{1}{3} + 1 \times \frac{1}{4} = \frac{11}{12}$. As for the minimal winning coalition $\{1, 2\}$, its surplus is $\sigma = w(\{1, 2\}) - 6 = 1,$ and we obtain

$$\sum_{l=0}^{3-1-1} T^*(I(l), l) = T^*(0, 0) + T^*(3, 1) = 1 + 3 = 4$$

corresponding vulnerable coalitions, reflecting $\{1, 2\}$ itself as well as $\{1, 2, 3\}$, $\{1, 2, 4\}$, and $\{1, 2, 5\}$. As for the losing coalition $\{1\}$, its deficiency is $\delta = 6 - w(\{1\}) = 2,$ pointing

to $\sum_{l=2+1}^{4+2-1} T^*\left(I\left(l-2\right),l\right) = T^*\left(3,3\right) + T^*\left(3,4\right) + T^*\left(4,5\right) = 1+0+3 = 4$ vulnerable coalitions with a player 1 as the only critical player, with $T^*\left(3,3\right) = 1$ reflecting the coalition $\left\{1, 3, 4, 5\right\}$ and $T^*\left(4,5\right) = 3,$ reflecting the coalitions $\left\{1, 2, 3, 4\right\}$, $\left\{1, 2, 3, 5\right\}$, and $\left\{1, 2, 4, 5\right\}$. Our calculations confirm $\gamma_1^{\text{raw}} = 4\times 1 + 4\times\frac{1}{2} + 3\times\frac{1}{3} = 7$ and $\gamma_2^{\text{raw}} = 4\times\frac{1}{2} + 1\times\frac{1}{4} = 2.25.$ □

This insight enables us to formulate a practical method for the Johnston index by proving:

Theorem 10. For a weighted voting game *v* with *n* players, the Johnston index can be computed in $O\left(qn^3\right)$ time and $O\left(qn\right)$ memory space for all players.

Proof. Let the weights be sorted in descending order, and $w_1 < q$ for otherwise $\gamma_1 = 1,\ \gamma_2 = ... = \gamma_n = 0.$ Let us work out $E_i\left(c\right),$ the number of vulnerable coalitions *S* with *c* critical players with $i \in Cr\left(S\right).$ Let $E_{i,j}\left(c\right)$ abbreviate the number of vulnerable coalitions *S* with *c* critical players with $i, j \in Cr\left(S\right)$ and j the critical player with the largest index. By $\hat{E}_{i,j}\left(c\right)$ and $\tilde{E}_{i,j}\left(c\right),$ we distinguish those coalitions that remain winning or become losing coalitions after the removal of all non-critical players, respectively. Using the values T^* from Definition 3, we precompute

$$D\left(j,\delta\right) = \sum_{l=\delta+1}^{w_j+\delta-1} T^*\left(I\left(l-\delta\right),l\right)$$

for all players $1 \le j \le n$ and all deficiencies $\delta \in \left\{1, ..., q-w_1-1\right\}$ and

$$U\left(x\right) = \sum_{l=0}^{x} T^*\left(I\left(l\right),l\right)$$

for all $x \in \left\{1, ..., w_1-1\right\}.$ We make use of $C\left(i-1, x, c\right)$ from Theorem 2, and find $E_{i,i}\left(c\right) = \tilde{E}_{i,i}\left(c\right) + \hat{E}_{i,i}\left(c\right)$ via

$$\tilde{E}_{i,i}\left(c\right) = \sum_{x=0}^{q-w_i-1} C\left(i-1, x, c-1\right) D\left(i, q-x-w_i\right)$$

as a coalition of weight $x + w_i < q$ has deficiency $\delta = q - x - w_i$, and

$$\hat{E}_{i,i}(c) = \sum_{x=q-w_i}^{q-1} C(i-1, x, c-1) U(q-x-1)$$

as a coalition of weight $x + w_i \geq q$ has surplus $\sigma = x + w_i - q$ making our argument for U above $w_i - \sigma - 1 = q - x - 1$. To find $E_{i,j}(c)$ for all $i < j \leq n$ we need the values $C_{-i}(j, x, \cdot)$ obtained as in Theorem 2 from the weights $w_1, ..., w_{i-1}, w_{i+1}, ..., w_n$. We can compute these values iteratively, starting from $C(i-1, x)$, updating a corresponding matrix in a loop for j from $i+1$ to n which we use to work out $E_{i,j}(c) = \tilde{E}_{i,j}(c) + \hat{E}_{i,j}(c)$. As before, we find

$$\tilde{E}_{i,j}(c) = \sum_{x=0}^{q-w_i-w_j-1} C_{-i}(j-1, x, c-2) D(j, q-x-w_i-w_j)$$

as a coalition of weight $x + w_i + w_j < q$ has deficiency $\delta = q - x - w_i - w_j$, and

$$\hat{E}_{i,j}(c) = \sum_{x=q-w_i-w_j}^{q-w_i-1} C_{-i}(j-1, x, c-2) U(q-x-w_i-1)$$

as a coalition of weight $x + w_i + w_j \geq q$ has surplus $\sigma = x + w_i + w_j - q$, making our argument for U above $w_j - \sigma - 1 = q - x - w_i - 1$. We compute $E_i(c) = \sum_{j=i}^{n} E_{i,j}(c)$ for player i in $O(qn^2)$ time and $O(qn)$ space. For all players we need $O(qn^3)$ time and $O(qn)$ space. □

7. The software EPIC and numerical results

We introduce a powerful software package named Efficient Power Index Computation (EPIC) providing efficient C++ implementations of all 12 power indices discussed in this article. In addition, scaled or unscaled variants of these indices, e.g., the relative Banzhaf, the raw Johnston index, and quantities like $|W|$, $|W^{np}|$, $|W^m|$, $|W^s|$, and $\eta_i(v)$, $|W_i|$, $|W_i^{np}|$, $|W_i^m|$, $|W_i^s|$ for all players, are available. The software was

tested extensively under MS Windows and Ubuntu Linux. The software is publicly available on GitHub at https://github.com/jhstaudacher/EPIC.

EPIC carefully deals with plenty of issues excluded from the previous sections to make our presentation more readable. Players with weight 0 are handled in a preprocessing step, and so is the case of a single player with weight greater or equal q. EPIC will throw an error message for negative weights and for $q \leq \frac{\tilde{w}}{2}$. By setting a flag, users can find and pre-process null players efficiently, aiming to reduce the size of a power index computation. Currently, we do not use any sophisticated approach for finding minimum sum integer representations of games. We only shrink weights and quotas using the greatest common divisor. Games with non-negative floating point weights (or quotas) will be converted to integer representations, according to Kurz [26]. Weighted voting games can be specified as comma separated values (CSV) files, employing a simple and widely used data format that allows users to edit input files using notepad, MS Excel, and many other text editors. EPIC can be used interactively via an R interface available at https://github.com/jhstaudacher/EPIC-R.

For the arbitrary-precision arithmetic parts, EPIC uses the GMP-library [19], i.e., the GNU Multiple Precision Arithmetic Library. GMP [19] is a very widely used, established, and well-tested library for handling very large integers. Internally, large integers in EPIC are either represented as a 64-bit unsigned integer if our numbers are small enough or a pointer to a GMP object for large integers (allocating memory economically for very large problems) or an array of l integers representing a large integer through l coprimes. The latter follows the idea by Kurz [26] and employs the Chinese remainder theorem to perform the most frequently used basic operations using standard data types. EPIC analyses the problem, and can then adaptively choose l different coprimes $p_1, ..., p_l$ such that all integers to be processed lie between 0 and $-1+\prod_{j=1}^{l} p_j$. All basic operations can then be computed modulo p_j for all $1 \leq j \leq l$. In the end, we can recover the actual large integer computed via the Chinese remainder theorem. However, we find this approach faster as compared to simply using GMP only when no more than $l = 4$ coprimes are needed. (Note that our findings do not contradict the study by Kurz [26] who computes the Banzhaf and Shapley–Shubik indices for the IMF problem with 188 players using $l = 3$ primes.) On a 64-bit system using GMP within our software architecture requires at least 4 times the storage of a 64-bit unsigned integer (8 Byte) and will normally save memory once more than $l = 4$ coprimes would be needed. By default, EPIC switches to relying entirely on GMP in that case. Users can override the default by setting flags enforcing either the use of coprimes or GMP for processing large integers.

Table 1. Computing times and memory requirements for an example with 150 players with uniformly distributed weights, $q = 35\,951$, $\tilde{w} = 71\,901$

Index	Time [s]	Memory [MB]	No. of coprimes or GMP	Complexity
Banzhaf	0.14	9.04	3	$O(\Delta n)$
Deegan–Packel	51.22	217.73	3	$O(qn^2)$
Felsenthal	26.41	181.39	GMP	$O(q\tilde{m}n)$
Johnston	5942.84	636.29	GMP	$O(qn^3)$
König–Bräuninger	0.19	9.04	3	$O(\Delta n)$
Nevison	0.19	9.04	3	$O(\Delta n)$
Null-player free index f^{np}	29.93	346.80	3	$O(\Delta n^2)$
Null-player free index g^{np}	0.19	9.04	3	$O(\Delta n)$
Public Good	0.38	13.03	GMP	$O(qn)$
Public Help θ	0.19	9.04	3	$O(\Delta n)$
Public Help ξ	29.72	346.80	3	$O(\Delta n^2)$
Shapley–Shubik	19.77	346.80	3	$O(\Delta n^2)$

Data available at https://github.com/jhstaudacher/EPIC/blob/master/test_cases/uniform/uniform. n150. q35951.csv

Table 2. Computing times and memory requirements for "limit tests" for a series of problems with a fixed quota $q = 100\,000$ and $\tilde{w} = 199\,999$

Index	No. of players (n)	Time [s]	Memory [MB]	No. of coprimes or GMP	Complexity
Banzhaf	2000	33.48	54.63	GMP	$O(\Delta n)$
Deegan–Packel	900	2472.80	7495.88	GMP	$O(qn^2)$
Felsenthal	800	2994.23	3532.79	GMP	$O(q\tilde{m}n)$
Johnston	60	2572.48	332.62	2	$O(qn^3)$
Public Good	2000	72.37	82.43	GMP	$O(qn)$
Public Help ξ	600	2639.00	8187.70	GMP	$O(\Delta n^2)$
Shapley–Shubik	600	2097.35	8183.35	GMP	$O(\Delta n^2)$

We report the number of players (up to 2000) that can be processed within one hour for selected indices. Data and R-code available at https://github.com/jhstaudacher/EPIC/tree/master/test_cases/linear_range _test_cases

The numerical results in Table 1 and Table 2 are obtained under Ubuntu 20.04 focal (64-bit) on an AMD FX(tm)-4170 Quad-Core CPU with a clock speed of 4.20 GHz and 8 GB RAM, i.e., on a standard laptop PC. In both tables the column No. of coprimes or GMP reflects the default described in the previous paragraph. Only if no more than $l = 4$ coprimes are needed, then we use the Chinese remainder theorem. Banzhaf and Shapley–Shubik indices are by default computed from above, as explained in the proofs of Theorem 3 and Theorem 4, respectively. For König-Bräuninger, Nevison, g^{np}, and Public Help θ we employ the identity by Dubey and Shapley [17] for computing $|W_i| = 0.5\left(|W| + \eta_i(v)\right)$ by default, as we find it to need less arithmetic operations, and thus be faster than the alternative formula $|W_i| = \sum_{x=q}^{\tilde{w}} T_{+i}(n, x)$ (which can optionally be selected by the user) in most cases. For the problem in Table 1, i.e., for a weighted voting game with uniformly distributed weights, quota $q = 35951$, $\tilde{w} = 71901$, an average weight of 479.34 and a median weight of 500.5, the computing times for these four indices, though as expected slightly larger than for Banzhaf, are negligible, and the same is true for the Public Good index. The larger computing times for Public Help ξ and f^{np} as compared to Shapley–Shubik reflect the summations for x from q to $\tilde{w}$ rather than from q to $q + w_i - 1$, as pointed out at the end of the proof of Theorem 4. For the Public Good, Felsenthal, and Johnston indices we need the multiplication operation, and hence 5 coprimes would be necessary for our 150 player problem since the coprimes must be less or equal 2^{32} so that the product of the largest possible numbers still fits into an unsigned 64-bit integer variable. By default, EPIC switches to relying entirely on GMP in that case. For Table 2 we construct a series of problems with a fixed quota q = 100 000 and $\tilde{w}$ = 199 999. For selected indices, we approximate the number of players that can be processed within one hour in 10 steps for up to 100 players and 100 steps for up to 2000 players. Our new algorithm for the Public Good index can handle 2000 player problems in just over one minute.

8. Conclusions

We provide efficient algorithms and a powerful software package for the exact computation of power indices. We assume the software to be helpful not only for analysing voting situations but also for broadening the appeal of power indices for network analysis. Our new $O(qn)$ algorithm for the Public Good index can easily handle more than 1000 players, allowing us to extend the recent study by Holler and Rupp [21] to much larger networks. Our new $O(qn^3)$ algorithm makes the Johnston index applicable for

analysing indirect control in larger corporate networks [11, 30, 31]. While the availability of such efficient algorithms is encouraging, the calculation of some indices, such as the strategic power indices by Kóczy [25], remains an open problem. In terms of algorithms, it will be interesting to compare dynamic programming to computing power indices via quasi-ordered binary decision diagrams, a recent approach based on relational algebra [5, 6, 13, 14]. Finally, the interplay between dynamic programming and generating functions deserves further attention as we expect these two areas to be very fruitful for each other, e.g., in developing even faster algorithms.

Acknowledgements

The authors would like to thank Sascha Kurz (University of Bayreuth) for the code he used for [26], publicly available via ResearchGate, and for allowing them to use it as a sample for their software. The first author thanks the funding of the Bavarian State Ministry of Science and Arts. The second author thanks the funding of the National Research, Development and Innovations Office (NKFIH, K-128573).

References

[1] ALGABA E., BILBAO J.M., GARCIA J.F., LÓPEZ J., *Computing power indices in weighted multiple majority games*, Math. Soc. Sci., 2003, 46 (1), 63–80.
[2] ALGABA E., BILBAO J.M., FERNÁNDEZ J.R., *The distribution of power in the European Constitution*, Eur. J. Oper. Res., 2007, 176 (3), 1752–1766.
[3] ÁLVAREZ-MOZOS M., FERREIRA F., ALONSO-MEIJIDE J.M., PINTO A.A., *Characterizations of power indices based on null player free winning coalitions*, Optimization, 2015, 64 (3), 675–686.
[4] BANZHAF J.F., *Weighted voting doesn't work: A mathematical analysis*, Rutgers Law Rev., 1965, 19, 317.
[5] BERGHAMMER R., BOLUS S., RUSINOWSKA A., DE SWART H., *A relation-algebraic approach to simple games*, Eur. J. Oper. Res., 2011, 210 (1), 68–80.
[6] BERGHAMMER R., BOLUS S., *On the use of binary decision diagrams for solving problems on simple games*, Eur. J. Oper. Res., 2012, 222 (3), 529–541.
[7] BERTINI C., GAMBARELLI G., STACH I., *A Public Help index*, [In:] M. Braham, F. Steffen (Eds.), *Power, freedom, and voting*, Springer, Berlin 2008, 83–98.
[8] BERTINI C., STACH I., *Banzhaf voting power measure*, [In:] K. Dowding (Ed.), *Encyclopedia of Power*, SAGE, Los Angeles 2011, 54.
[9] BERTINI C., FREIXAS J., GAMBARELLI G., STACH I., *Comparing power indices*, Int. Game Theory Rev., 2013, 15 (2), 1340004.
[10] BERTINI C., STACH I., *On public values and power indices*, Dec. Mak. Manuf. Serv., 2015, 9 (1), 9–25.
[11] BERTINI C., MERCIK J., STACH I., *Indirect control and power*, Oper. Res. Dec., 2016, 26 (2), 7–30.
[12] BILBAO J.M., FERNANDEZ J.R., JIMÉNEZ LOSADA A., LOPEZ J.J., *Generating functions for computing power indices efficiently*, Top, 2000, 8 (2), 191–213.
[13] BOLUS S., *Power indices of simple games and vector-weighted majority games by means of binary decision diagrams*, Eur. J. Oper. Res., 2011, 210 (2), 258–272.
[14] BOLUS S., *A QOBDD-based approach to simple games*, PhD thesis, Christian-Albrechts Universität Kiel, Kiel 2012.

[15] CHAKRAVARTY S.R., MITRA M., SARKAR P., *A Course on Cooperative Game Theory*, Cambridge University Press, Cambridge 2015.
[16] DEEGAN J., PACKEL E.W., *A new index of power for simple n-person games*, Int. J. Game Theory, 1978, 7 (2), 113–123.
[17] DUBEY P., SHAPLEY L.S., *Mathematical properties of the Banzhaf power index*, Math. Oper. Res., 1979, 4 (2), 99–131.
[18] FELSENTHAL D.S., *A well-behaved index of a priori p-power for simple n-person games*, Homo Oecon., 2016, 33 (4), 367–381.
[19] https://gmplib.org/ (URL consulted in November 2020).
[20] HOLLER M.J., *Forming coalitions and measuring voting power*, Pol. Studies, 1982, 30 (2), 262–271.
[21] HOLLER M.J., RUPP F., *Power in networks. A PGI analysis of Krackhardt's kite network*, Springer Lecture Notes in Computer Science 11890, 2019, 21–34.
[22] JOHNSTON R.J., *On the measurement of power. Some reactions to Laver*, Environ. Plan. A, 1978, 10 (8), 907–914.
[23] KÖNIG T., BRÄUNINGER T., *The inclusiveness of European decision rules*, J. Theor. Pol., 1998, 10 (1), 125–142.
[24] KÓCZY L.Á., *Beyond Lisbon. Demographic trends and voting power in the European Union Council of Ministers*, Math. Soc. Sci., 2012, 63 (2), 152–158.
[25] KÓCZY L.Á., *Power indices when players can commit to reject coalitions*, Homo Oecon., 2016, 33 (1–2), 77–91.
[26] KURZ S., *Computing the power distribution in the IMF*, arXiv preprint, Comp. Sci. Game Theeory, 2016, arXiv: 1603.01443.
[27] LUCCHETTI R., RADRIZZANI P., *Microarray data analysis via weighted indices and weighted majority games*, [In:] F. Masulli, L.E. Peterson, R. Tagliaferri (Eds.), *International meeting on computational intelligence methods for bioinformatics and biostatistics*, Springer, Berlin 2009, 179–190.
[28] MATSUI T., MATSUI Y., *A survey of algorithms for calculating power indices of weighted majority games*, J. Oper. Res. Soc. Japan, 2000, 43 (1), 71–86.
[29] MATSUI Y., MATSUI T., *NP-completeness for calculating power indices of weighted majority games*, Theor. Comp. Sci., 2001, 263 (1–2), 305–310.
[30] MERCIK J., LOBOS K., *Index of implicit power as a measure of reciprocal ownership*, Springer Lecture Notes in Computer Science 9760, 2016, 128–140.
[31] MERCIK J., STACH I., *On measurement of control in corporate structures*, Springer Lecture Notes in Computer Science 11290, 2018, 64–79.
[32] NEVISON H., *Structural power and satisfaction in simple games*, [In:] S.J. Brams, A. Schotter, G. Schwödiauer (Eds.), Appl. Game Theory, Phys., Heidelberg 1979, 39–57.
[33] SHAPLEY L.S., SHUBIK M., *A method for evaluating the distribution of power in a committee system*, Amer. Pol. Sci. Rev., 1954, 48 (3), 787–792.
[34] STACH I., *Shapley–Shubik index*, [In:] K. Dowding (Ed.), *Encyclopedia of Power*, SAGE, Los Angeles 2011, 603–606.
[35] TANENBAUM P., *Power in weighted voting games*, Math. J., 1997, 7, 58–63.
[36] UNO T., *Efficient computation of power indices for weighted majority games*, Springer Lecture Notes in Computer Science 7676, 2012, 679–689.