*Article*

# Dynamic programming for computing power indices for weighted voting games with precoalitions

**Jochen Staudacher** [1]*, **Felix Wagner** [1] **and Jan Filipp** [1]

[1] Fakultät Informatik, Hochschule Kempten, 87435 Kempten, Germany
* Correspondence: jochen.staudacher@hs-kempten.de; Tel.: +49-831-2523-513

**Abstract:** We study the efficient computation of power indices for weighted voting games with precoalitions amongst subsets of players (reflecting, e.g., ideological proximity) using the paradigm of dynamic programming. Starting from the state-of-the-art algorithms for computing the Banzhaf and Shapley-Shubik indices for weighted voting games we present a framework for fast algorithms for the three most common power indices with precoalitions, i.e. the Owen index, the Banzhaf-Owen index and the Symmetric Coalitional Banzhaf index, and point out why our new algorithms are applicable for large numbers of players. We discuss implementations of our algorithms for the three power indices with precoalitions in C++ and review computing times as well as storage requirements.

**Keywords:** cooperative game theory; power indices; weighted voting games; dynamic programming; precoalitions; Shapley value; Owen value; Banzhaf index.

**JEL Classification:** C71

## 1. Introduction

Weighted voting games (also known as weighted majority games) are a common model for decision-making and voting in committees, panels or boards. There are $n$ players and each player $i$ is assigned a positive weight $w_i$ which in some situations may reflect the number of votes of a voting bloc. The decision-making body accepts a measure or motion if a certain quota $q$, normally more than 50 percent of the sum of all weights, is reached or exceeded, i.e. if and only if there is a winning coalition $S$ with $\sum_{i\in S} w_i \geq q$. In cooperative game theory, weighted voting games can be regarded as a special class of simple games, i.e. games in which coalitions are either winning with value 1 or losing with value 0. Power indices are applied to simple games and provide measures for the players' abilities to influence results in voting situations. On the basis of different bargaining models and different axiomatic assumptions numerous power indices have been introduced, see e.g. [1] for an overview, with the two most-widely used power indices being the Banzhaf index [2,3] and the Shapley-Shubik index [4].

Weighted voting games and power indices are applicable well beyond classical voting situations in politics, described e.g. in [5–7]. For example, power indices can also be used to analyze genetic networks and rank genes which may be responsible for genetic diseases [8], to solve reliability problems in the maintenance of computer networks [9], to understand indirect control power in corporate shareholding structures [10–12] or to analyze social networks [13,14]. In social, economic or corporate networks and in voting settings alike it is common for agents to agree beforehand to form a union and act as a voting bloc.

This article deals with the special case of a partition of the player set into disjoint precoalitions (also known as a priori unions), i.e. we assume that either all or none of the members of a precoalition join a certain coalition. Precoalitions can model strict party discipline in a legislative chamber, but they can also be interpreted as a way to reduce negotiation costs with each precoalition sending a delegate as its negotiator. As pointed out in the book by Owen [15], pp. 303, the players in such a union have agreed to keep together, but, even though they will do so in most situations, they are not forced to comply. Given that individual players may threaten to defect from their precoalition, we need to evaluate their influence in a two-stage process. In an external stage the power of the precoalition is evaluated, in an internal stage the results for the members of the precoalition are determined. Using the language of cooperative game theory, this two-stage process translates into



**Citation:** Staudacher, J.; Wagner, F.; Filipp, J. Dynamic programming for computing power indices for weighted voting games with precoalitions. *Preprints* **2021**, *1*, 0. https://doi.org/

Received:
Accepted:
Published:

an external game (also known as the quotient game) between the precoalitions and an internal game within each precoalition.

Owen generalized the Shapley value from cooperative game theory to the framework with a priori unions amongst subsets of players in his seminal 1977 paper [16]. However, the internal game within each precoalition is no longer a simple game if the external game is simple. Malawski [17] pointed out how this problem can be overcome by computing more than one simple game on the internal stage. For our study this implies that, in general, the solution of more than one weighted voting game will be necessary on the internal level. Following an article by Alonso-Meijide et al. [18], this paper studies the efficient computation of the three power indices for weighted voting games with precoalitions discussed therein: the Owen index [16] employing the Shapley-Shubik index on both the external and the internal level, the Symmetric Coalitional Banzhaf index [19] employing the Banzhaf index on the external and the Shapley-Shubik index on the internal level and the Banzhaf-Owen index [20] employing the Banzhaf index on both the external and the internal level. Computing these three power indices for weighted voting games with precoalitions is challenging as we are dealing with NP-hard problems [21] and as the voting and network applications outlined in the first two paragraphs [5–14] may involve large numbers of players.

Power indices for simple games can be computed using generating functions, see e.g. [5,22,23], and the work by Alonso-Meijide and Bowles [24] for the case of a priori unions. If the subsets of players reach only very few different weight sums, this method is favoured [7] and fast-access data structures for polynomials with few coefficients in computer algebra systems like Mathematica [25] can be used. In this paper we use the strongly related, though mathematically less sophisticated, paradigm of dynamic programming for power index computation [26–28]. The recent article [28] presents a new algorithm for the Johnston index [29], discusses algorithms for power indices based on minimal winning coalitions [30–32], surveys the state-of-the-art and introduces publicly available software. The work by Molinero and Blasco [33] studies algorithms for the Banzhaf-Owen and Owen indices and, to our knowledge, it is the only paper on dynamic programming algorithms for weighted voting games with precoalitions. This paper presents new algorithms for the Banzhaf-Owen, Owen and Symmetric Coalitional Banzhaf indices with more favourable pseudopolynomial complexities. Our algorithms are based on a generalization of the concepts for computing the Banzhaf and Shapley-Shubik indices from the papers by Uno (2012) [27] and Kurz (2016) [7].

In Section 2 we introduce the basic concepts from cooperative game theory, including simple games, power indices and games with precoalitions along the lines of [18,34]. Section 3 explains how dynamic programming is used to count coalitions efficiently for weighted voting games and presents the state-of-the-art algorithms for computing the Banzhaf and Shapley-Shubik indices following the paper by Kurz [7]. Section 4 forms the heart of this paper and presents the new algorithms for the three power indices with precoalitions. We point out how our algorithms reflect both the definition of the indices as well as the internal games described by Malawski [17]. Section 5 discusses numerical experiments, reports supportive computing times and storage requirements for our methods and provides the reader with links to our C++ software and test problems. We end with some concluding remarks and an outlook to open problems in Section 6.

## 2. Preliminaries from cooperative game theory

Cooperative game theory [35] deals with the outcomes and benefits which players can gain by forming coalitions. In the following, we briefly review some terminology for cooperative games and precoalitions along the lines of the papers by Alonso-Meijide et al. (2008) [18] and Mercik and Ramsey (2017) [34].

### *2.1. Cooperative games and simple games*

Let $N = \{1,...,n\}$ denote a finite set of $n$ players. A group of players $S \subseteq N$ is called a coalition, whereas $2^N$ stands for the set of all subsets of $N$. $\emptyset$ symbolizes the *empty coalition* and $N$ is called the *grand coalition*. By $|S|$ we express the cardinality of a coalition $S$, i.e. the number of its members, hence $|N| = n$. An $n$-person cooperative game can be characterized as a pair $(N, v)$ where $v : 2^N \to \mathbb{R}$ is referred to as the *characteristic function* assigning a real value to all coalitions $S \in 2^N$,

with $v(\emptyset) = 0$. A cooperative game is *monotone* if for all coalitions $S, T \in 2^N$ the relation $S \subseteq T$ implies $v(S) \leq v(T)$.

We call a cooperative game *simple* if it is monotone and there holds $v(N) = 1$ and $v(S) = 0$ or $v(S) = 1$ for each coalition $S \subset N$. Coalitions for which $v(S) = 1$ are classified as *winning coalitions* in simple games, whereas coalitions for which $v(S) = 0$ are called *losing coalitions*. A player $i$ is a *critical player* (also known as a decisive player or swing player) in a winning coalition $S$ if $v(S\backslash\{i\}) = 0$, i.e. the winning coalition $S$ turns into a losing one if player $i$ leaves $S$. We call a coalition $S$ with at least one critical player *vulnerable*.

*Weighted voting games* (also known as weighted majority games) are possibly the most important subclass of simple games. They are useful for a large number of practical applications [5–14]. Weighted voting games are specified by $n$ non-negative real weights $w_i, i = 1,\ldots,n$, and a non-negative real quota $q$, normally $q > \frac{1}{2}\sum_{i=1}^{n} w_i$. Its characteristic function $v : 2^N \to \{0,1\}$ takes the value $v(S) = 1$ in case of a winning coalition $S$, i.e. $w(S) = \sum_{i\in S} w_i \geq q$, and $v(S) = 0$ otherwise, meaning that coalition $S$ is losing.

### 2.2. Power indices and marginal contributions

Let $S \subset N$ denote a coalition and $i \notin S$ a player not contained in $S$. We call the difference $v(S\cup\{i\}) - v(S)$ the *marginal contribution* of player $i$ to coalition $S$. For simple games, the marginal contribution of player $i$ to coalition $S$ is 1 if and only if $i$ is a critical player in the winning coalition $S\cup\{i\}$; in any other case the marginal contribution is 0.

A *power index* is a function $f$ which obtains an $n$-person simple game specified by its player set $N$ and its characteristic function $v$ as its input and provides a unique vector $f(N,v) = (f_1(N,v),\ldots,f_n(N,v))$ as its output. We only introduce the Banzhaf index [2,3] and the Shapley-Shubik index [4] following the notation of the paper [18]. For a deeper discussion of the subject we refer to the overview article [1] by Bertini, Freixas, Gambarelli and Stach.

Let $v$ be a simple $n$-player game. The *(absolute) Banzhaf index* [3] of player $i$ and the *Shapley-Shubik index* [4] of player $i$ can be written in the form

$$f_i(N,v) = \sum_{S\subseteq N\backslash\{i\}} \alpha_S^i \cdot (v(S\cup\{i\}) - v(S)) \tag{1}$$

with the coefficients

$$\alpha_S^i = \frac{1}{2^{n-1}}$$

for the Banzhaf index and

$$\alpha_S^i = \frac{|S|!(n-|S|-1)!}{n!}$$

for the Shapley-Shubik index. We note that for the coefficients $\alpha_S^i$ the cardinality of a coalition $S$ (which is turned from a losing coalition to a winning coalition by the entrance of player $i$) plays a crucial role for the Shapley-Shubik index whereas it can be ignored for the Banzhaf index. We note that the Shapley-Shubik index is *efficient*, i.e. $\sum_{i=1}^{n} f_i(N,v) = v(N) = 1$ while the (absolute) Banzhaf index is not.

### 2.3. Simple games with precoalitions and coalitional power indices

In this subsection we introduce simple games with precoalitions and coalitional power indices borrowing some notation from [18,34].

We are establishing an external division of our set of players $N = \{1,...,n\}$ into precoalitions (also known as a priori unions) whose members will either enter a coalition together or not at all. Let $\mathscr{P}(N)$ stand for the set of all partitions of $N$, a partition being a set of non-empty subsets of $N$ satisfying the condition that $N$ is a disjoint union of these subsets. We refer to an element $P \in \mathscr{P}(N)$ as a *coalition structure* (also known as a system of unions) of the set $N$. A simple game with a coalition structure can be identified as a triplet $(N,v,P)$. Following [18] we write our coalition structure in the form $P = \{P_1,\ldots,P_m\}$, i.e. we have $m$ precoalitions $P_1,\ldots,P_m$ and the set $M = \{1,\ldots,m\}$ serves as the index set of the partition $P$. For a weighted voting game with a coalition structure $P$ the *external game* (also known as the quotient game) is defined as the weighted

voting game played between the $m$ precoalitions [17,18]. The external game can be represented by the (unaltered) quota $q$ and the $m$ weights $w(P_1),\ldots,w(P_m)$ where $w(P_k)=\sum_{i\in P_k} w_i$ with $k\in M$, $M=\{1,\ldots,m\}$.

A *coalitional power index* $g$ is a function which obtains an $n$-person simple game with a coalition structure specified by its player set $N$, its characteristic function $v$ and a partition $P$ as its input and provides a unique vector $g(N,v,P)=(g_1(N,v,P),\ldots,g_n(N,v,P))$ as its output. As in [18] we define three coalitional power indices, i.e. the *Banzhaf-Owen index* (hereafter BO index) [20], the *Symmetric Coalitional Banzhaf index* (SCB index) [19], and the *Owen index* (OW index) [16], within a unified framework.

Let $P=\{P_1,\ldots,P_m\}$ stand for a set of $m$ precoalitions $P_1,\ldots,P_m$ and the set $M=\{1,\ldots,m\}$ serve as the index set of $P$. Let further $Q_R=\cup_{l\in R}P_l$ denote the subset of players belonging to any precoalition referred to by the index subset $R\subseteq M$. For any player $i\in N$ contained in precoalition $k$, i.e. $i\in P_k$, we write a coalitional power index as follows (cf. [34]):

$$g_i(N,v,P)=\sum_{R\subseteq M\setminus\{k\}}\alpha_R^k\sum_{T\subseteq P_k\setminus\{i\}}\alpha_T^i\cdot(v(Q_R\cup T\cup\{i\})-v(Q_R\cup T)) \tag{2}$$

As in [34], $\alpha_R^k$ is an appropriately defined weight for the external game, while $\alpha_T^i$ corresponds to internal games within a precoalition. For the BO index we employ the weighting coefficients

$$\alpha_R^k=\frac{1}{2^{m-1}},\quad \alpha_T^i=\frac{1}{2^{|P_k|-1}},$$

for the SCB index

$$\alpha_R^k=\frac{1}{2^{m-1}},\quad \alpha_T^i=\frac{|T|!(|P_k|-|T|-1)!}{|P_k|!},$$

and for the OW index

$$\alpha_R^k=\frac{|R|!(m-|R|-1)!}{m!},\quad \alpha_T^i=\frac{|T|!(|P_k|-|T|-1)!}{|P_k|!}.$$

We note that the OW index is *efficient*, i.e. $\sum_{i=1}^n g_i(N,v,P)=v(N)=1$ whereas the BO and SCB indices are not.

Let us emphasize a crucial difference between formulas (1) and (2). While both expressions rely on marginal contributions of player $i$, in (2) we assume that precoalitions other than $P_k$ (which contains player $i$) have either joined with all their members or not at all. As pointed out in [34] the power of player $i\in P_k$ comes from scenarios in which both $P_k$ is critical in the external game and $i$ is a critical player in its precoalition in an appropriately defined internal voting game.

Looking at the extreme coalition structures $P^n=\{\{1\},\{2\},\ldots,\{n\}\}$ and $P^N=\{N\}$, we recognize that the BO index reduces to the Banzhaf index in both cases whereas the OW index reduces to the Shapley-Shubik index in both cases. As for the SCB index, it becomes the Banzhaf index for $P^n=\{\{1\},\{2\},\ldots,\{n\}\}$ and the Shapley-Shubik index in case $P^N=\{N\}$. The BO index [20] applies the Banzhaf index both in the external game and in the internal process between unions. We mention an interpretation of the BO index from the paper by Laruelle and Valenciano [36]. The BO index assesses the a priori decisiveness of every voter within her bloc in the context of the coalitional structure, i.e. assuming that all the other precoalitions will act as voting blocs. The SCB index by Alonso-Meijide and Fiestras-Janeiro [19] employs the Banzhaf index for the external game and the Shapley-Shubik index for the internal processes, whereas the OW index [16] invokes the Shapley-Shubik index on both levels of a two-level decision-making process. Taking a closer look at these two-level decision-making processes will play a crucial role in Section 4 when we derive efficient dynamic programming algorithms for the BO, SCB and OW indices for weighted voting games with precoalitions.

## 3. Dynamic programming for the Banzhaf and Shapley-Shubik indices

In this section we introduce the technique of dynamic programming for counting coalitions in weighted voting games efficiently, discuss the state-of-the-art algorithms for the Banzhaf and

Shapley-Shubik indices [7,27,28] and present generalizations of existing algorithms that will come out to be useful when we deal with precoalitions and internal games within precoalitions in Section 4. In other words: We are not computing any power indices with precoalitions in this section, but lay the groundwork for doing so in Section 4.

### *3.1. Counting winning and losing coalitions via dynamic programming*

It is well known that every weighted voting game allows for an integer representation [7]. Hence, we assume that the weights $w_i$ of the $n$ players in our weighted voting game as well as the quota $q$ are positive integers for the rest of the article. We set $\tilde{w} = w(N) = \sum_{i=1}^{n} w_i$ and assume $q > \frac{\tilde{w}}{2}$. Players with integer weight 0 are null players, i.e. they are never critical players, and for all power indices discussed in this paper they can be handled in a preprocessing step and assigned the value 0 without any need to include them in any further computation.

When we want to compute the Banzhaf index of player $i$, we need to be able to answer one of the following two questions: How many losing coalitions are turned into winning coalitions when player $i$ is joining? Alternatively, in how many winning coalitions is player $i$ critical? For the Shapley-Shubik index of player $i$ the same two questions are relevant, but we also need to take account of cardinalities of coalitions.

Dynamic programming is an algorithmic paradigm resting on two pillars: dividing a given problem into subproblems and storing intermediate results efficiently. We will use dynamic programming to count losing and winning coalitions and present algorithms following the work by Kurz [7]. For a more general introduction to counting coalitions via dynamic programming (including examples) we refer to the recent article [28] or the textbook by Chakravarty, Mitra and Sarkar [35], chapter 12.

Algorithm 1 is a generalization of the first algorithm in the article by Kurz [7]. Equipped with the number $n$ of players, the quota $q$, a vector $w = (w_1, \ldots, w_n)$ of positive integer weights and an integer vector $T$ with $T(0) = 1$ and $T(x) = 0$ for $x = 1, \ldots, q-1$, as its four input arguments, it outputs a vector $T$ with $T(x)$ listing the number of losing coalitions attaining weight $0 \leq x \leq q-1$. Algorithm 1 needs $O(nq)$ time and $O(q)$ space.

**Algorithm 1** Generalized Counting of Number of Losing Coalitions per Weight

```
1:  procedure LosingCoalitionsByWeight(n, q, w, T)
2:      for i from 1 to n do
3:          if (q > w(i)) then
4:              for x from q − 1 to w(i) do
5:                  if (x-w(i) == 0) then
6:                      T(x) ← T(x) + T(0)              ▷ initialize T(0)=1 for empty set
7:                  else
8:                      T(x) ← T(x) + T(x − w(i))
9:                  end if
10:             end for
11:         end if
12:     end for
13:     return T                                  ▷ number of losing coalitions of weight x ∈ [0, q − 1]
14: end procedure
```

Note that Algorithm 1 echoes a boundary condition stating that we can obtain the sum 0 in exactly one way, i.e. via the empty set, reflected in $T(0) = 1$. We admit that the if-else statement in lines 5 to 8 in Algorithm 1 is superfluous as $T(x - w(i)) = T(0)$ in case $x = w(i)$, but we wanted to highlight the role of $T(0)$. The actual recursion in line 8 in Algorithm 1 mirrors the fact that the first $i$ weights can deliver a sum $x > 0$ either with or without player $i$. We will find providing an initial vector $T$ as an input useful in the following section when we wish to incorporate results from the external game in computations of internal games.

Kurz [7] pointed out that rather than counting losing coalitions from below, we can also count winning coalitions from above making use of the fact that we can write the sum of all weights in exactly one way, i.e. $\tilde{w} = w(N) = \sum_{i=1}^{n} w_i$.

Algorithm 2 is a generalization of the second algorithm in the article by Kurz [7]. Provided with the number $n$ of players, the quota $q$, the sum of all weights $\tilde{w} = w(N)$, a vector $w = (w_1, \ldots, w_n)$ of positive integer weights and an integer vector $T$ with $T(\tilde{w}) = 1$ and $T(x) = 0$ for $x = q, \ldots, \tilde{w}-1$, as its five input arguments, it outputs a vector $T$ with $T(x)$ listing the number of winning coalitions attaining weight $q \leq x \leq \tilde{w}$. Algorithm 2 needs $O(n(\tilde{w}-q+1))$ time and $O(\tilde{w}-q+1)$ space.

**Algorithm 2** Generalized Counting of Number of Winning Coalitions per Weight

```
1:  procedure WinningCoalitionsByWeight(n, q, w̃, w, T)
2:      for i from 1 to n do
3:          if (q + w(i) ≤ w̃) then
4:              for x from q + w(i) to w̃ do
5:                  T(x − w(i)) ← T(x) + T(x − w(i))
6:              end for
7:          end if
8:      end for
9:      return T                          ▷ number of winning coalitions of weight x ∈ [q, w̃]
10: end procedure
```

Algorithms 1 and 2 can be generalized for the distribution of cardinalities of coalitions in a straightforward manner [7,28]. Given that we normally assume $q > \frac{\tilde{w}}{2}$ (and hence $\tilde{w}-q+1 \leq q$), we prefer to work with the winning coalitions from above. Therefore we provide only the corresponding algorithm for winning coalitions and cardinalities from above in Algorithm 3. Provided with the number $n$ of players, the quota $q$, the sum of all weights $\tilde{w} = w(N)$, a vector $w = (w_1, \ldots, w_n)$ of positive integer weights and a 2-dimensional array $C$ with $C(\tilde{w}, n) = 1$ and $C(x,s) = 0$ for all other combinations of weights $x = q, \ldots, \tilde{w}$, and cardinalities $s = 1, \ldots, n$, as its five input arguments, it outputs a 2-dimensional array $C$ with $C(x,s)$ listing the number of winning coalitions attaining weight $q \leq x \leq \tilde{w}$ and cardinalities $s = 1, \ldots, n$. Algorithm 3 needs $O(n^2(\tilde{w}-q+1))$ time and $O(n(\tilde{w}-q+1))$ space.

**Algorithm 3** Generalized Counting of Number of Winning Coalitions per Weight and Cardinality

```
1:  procedure WinningCoalitionsByWeightCardinality(n, q, w̃, w, C)
2:      for i from 1 to n do
3:          if q + w(i) ≤ w̃ then
4:              for x from q + w(i) to w̃ do
5:                  for s from 2 to n do
6:                      C(x − w(i), s − 1) ← C(x, s) + C(x − w(i), s − 1)
7:                  end for
8:              end for
9:          end if
10:     end for
11:     return C      ▷ number of winning coalitions of weight x ∈ [q, w̃] and cardinality s ∈ [1, n]
12: end procedure
```

### *3.2. Computing the Banzhaf and Shapley-Shubik indices*

The paper by Uno (2012) [27] presented the first algorithm for computing the Banzhaf and Shapley-Shubik indices of all players in a weighted voting game in $O(qn)$ and $O(qn^2)$ time, respectively. By working from above as in Algorithms 2 and 3, Kurz (2016) [7] improved this result to $O(\Delta n)$ and $O(\Delta n^2)$ with $\Delta = \min(q, \tilde{w}-q+1)$ for the Banzhaf and Shapley-Shubik indices, respectively. It is worthwhile summarizing these results and their proofs together with some new generalizations of algorithms which we will use later in Section 4.

**Theorem 1.** *For an n-player weighted voting game with positive integer weights the Banzhaf indices for all players can be computed in $O(\Delta n)$ time and $O(\Delta)$ memory space.*

**Proof.** Let us focus on the case $(\tilde{w}-q+1) \leq q$ and compute the Banzhaf indices from above. We obtain the vector $T(x)$ for $q \leq x \leq \tilde{w}$ via Algorithm 2 in $O((\tilde{w}-q+1)n)$ time and $O(\tilde{w}-q+1)$ space by calling

$$T = WinningCoalitionsByWeight(n,q,\tilde{w},w,T)$$

using an integer vector $T$ with $T(\tilde{w}) = 1$ and $T(x) = 0$ for $x = q,\ldots,\tilde{w}-1$, as our input. We set $T_{+i}(x) = T(x)$ for all $x \in \{\tilde{w}-w_i+1,\ldots,\tilde{w}\}$ and then loop for $x$ from $\tilde{w}-w_i$ down to $q$ in order to find

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

$T_{+i}(x)$ states how frequently player $i$ belongs to a winning coalition with weight $x$ and

$$f_i^{BZ}(N,v) = \frac{1}{2^{n-1}} \sum_{x=q}^{q+w_i-1} T_{+i}(x)$$

gives us the Banzhaf index of player $i$. □

Algorithm 4 is a generalization of this idea which we will use in the next section. It counts the number of coalitions containing a player with a specified weight.

**Algorithm 4** Count coalitions containing a player specified by its weight

```
1:  procedure CoalitionsWithPlayer(q, w̃, weight, T)
2:      for x from q to w̃ do
3:          Tw(x) ← T(x)
4:      end for
5:      if (w̃ − weight ≥ q) then
6:          for x from w̃ − weight to q do
7:              Tw(x) ← T(x) − Tw(x + weight)
8:          end for
9:      end if
10:     return Tw
11: end procedure
```

Theorem 1 and Algorithm 4 can be carried over to versions with cardinalities and thus to the computation of Shapley-Shubik indices.

**Theorem 2.** *For an n-player weighted voting game with positive integer weights the Shapley-Shubik indices for all players can be computed in $O(\Delta n^2)$ time and $O(\Delta n)$ memory space.*

**Proof.** Again, we concentrate on the case $(\tilde{w}-q+1) \leq q$ and compute the Shapley-Shubik indices from above. We obtain $C(x,s)$ for $q \leq x \leq \tilde{w}$ and all cardinalities $1 \leq s \leq n$ from above via Algorithm 3 in $O((\tilde{w}-q+1)n^2)$ time and $O((\tilde{w}-q+1)n)$ space by calling

$$C = WinningCoalitionsByWeightCardinality(n,q,\tilde{w},w,C)$$

using a 2-dimensional array $C$ with $C(\tilde{w},n) = 1$ and $C(x,s) = 0$ for all other combinations of weights $x = q,\ldots,\tilde{w}$, and cardinalities $s = 1,\ldots,n$, as our input. We set $C_{+i}(x,c) = C(x,c)$ for all weights $x \in \{\tilde{w}-w_i+1,\ldots,\tilde{w}\}$ and all cardinalities $s \in \{1,\ldots,n\}$. We loop for $x$ from $\tilde{w}-w_i$ down to $q$ in an outer loop and for $s$ from 0 to $n-1$ in an inner loop in order to find

$$C_{+i}(x,s) = C(x,s) - C_{+i}(x+w_i,s+1).$$

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

$$f_i^{SH}(N,v) = \frac{1}{n!} \sum_{s=0}^{n-1} s!(n-s-1)! \sum_{x=q}^{q+w_i-1} C_{+i}(x,s+1)$$

we obtain the Shapley-Shubik index of player $i$. □

Algorithm 5 is a generalization of this idea which we will need in the next section. It counts the number of coalitions and cardinalities containing a player with a specified weight.

**Algorithm 5** Count coalitions and cardinalities containing a player specified by its weight

```
1:  procedure CoalitionsCardinalityWithPlayer(n, q, w̃, weight, C)
2:      for x from q to w̃ do
3:          for s from 1 to n do
4:              Cw(x,s) ← C(x,s)
5:          end for
6:      end for
7:      if w̃ − weight ≥ q then
8:          for x from w̃ − weight to q do
9:              for s from 1 to n − 1 do
10:                 Cw(x,s) ← C(x,s) − Cw(x + weight, s + 1)
11:             end for
12:         end for
13:     end if
14:     return Cw
15: end procedure
```

## 4. Computing power indices with precoalitions via dynamic programming

This section forms the heart of the article. It presents new dynamic programming algorithms for the BO, SCB and OW indices. As we outlined in Subsection 2.3, we assume that there are $m$ precoalitions $P_1,\ldots,P_m$. The *external game* (also known as the quotient game) is defined as the weighted voting game played between the precoalitions [17,18], i.e. a weighted voting game represented by the (unaltered) quota $q$ and the $m$ weights $w(P_1),\ldots,w(P_m)$ where $w(P_k)=\sum_{i\in P_k} w_i$ with $k\in M$, $M=\{1,\ldots,m\}$. Furthermore, we define $p$ as the maximal size of a precoalition, i.e. $p=\max_{k\in M}|P_k|$.

To our knowledge, the paper by Molinero and Blasco [33] is the only work on dynamic programming algorithms for weighted voting games with a precoalition structure. Their algorithms for the BO and OW indices combine backtracking with the algorithms from subsection 5.2 of the survey paper by Matsui and Matsui (2000) [26] for power index computation. For computing the BO or OW index of a single player $i$ with $i\in P_k$, i.e. player $i$ contained in precoalition $P_k$, their algorithm needs $O((n'|P_k|)^2 q)$ computing time and $O(n'q)$ space with $n'=m+|P_k|-2$.

In the following, we point out how the concepts and algorithms from the previous section translate into external and internal weighted voting games and algorithms for the BO, SCB and OW indices with favourable pseudopolynomial computing times and storage requirements. Each subsection will be devoted to one of the three indices with the proof of each theorem serving as a description of an algorithm for computing the index. We follow the convention that the superscript $+k$ indicates that we count only those winning coalitions containing precoalition $k$ on the external level, whereas the subscript $+i$ indicates that we look at only those coalitions containing player $i$ on the level of individual players within precoalition $P_k$. Furthermore, the superscript *ext* symbolizes the external level, i.e. the level of the precoalitions, whereas the superscript *int* stands for the internal level, i.e. the level of individual players within a precoalition.

### *4.1. Efficient computation of the Banzhaf-Owen (BO) index*

For the BO index, our idea is to reflect the external game and one internal game per precoalition via dynamic programming.

**Theorem 3.** *For a weighted voting game with positive integer weights and a precoalition structure $P=\{P_1,\ldots,P_m\}$, the Banzhaf-Owen (BO) indices for all players can be computed in $O(\Delta(m+p))$ time and $O(\Delta)$ memory space with $\Delta=\min(q,\tilde{w}-q+1)$. For computing the BO index of an individual player $i\in P_k$, i.e. player $i$ is contained in precoalition $P_k$, we need $O(\Delta(m+|P_k|))$ time and $O(\Delta)$ space.*

**Proof.** Let us focus on the case $(\tilde{w}-q+1) \leq q$ and compute the BO indices from above. Let $w^{ext} = (w(P_1),\ldots,w(P_m))$ be the vector of the weights of the $m$ precoalitions. We obtain the vector $T^{ext}(x)$ for $q \leq x \leq \tilde{w}$ with the number of winning coalitions on the level of the precoalitions for weights $q \leq x \leq \tilde{w}$ via Algorithm 2 in $O((\tilde{w}-q+1)m)$ time and $O(\tilde{w}-q+1)$ space by computing

$$T^{ext} = WinningCoalitionsByWeight(m,q,\tilde{w},w^{ext},T)$$

using an integer vector $T$ with $T(\tilde{w}) = 1$ and $T(x) = 0$ for $x = q,\ldots,\tilde{w}-1$, as our input. Next, we deal with precoalition $P_k$. Algorithm 4 gives us the number of those winning coalitions containing precoalition $k$ with weight $w_k^{ext}$ via

$$T^{ext,+k} = CoalitionsWithPlayer(q,\tilde{w},w_k^{ext},T^{ext})$$

Now we switch to the level of precoalition $P_k$. We incorporate the individual weights $w^{int} = (w_1,\ldots,w_{|P_k|})$ of the players in precoalition $P_k$ into our vector $T^{ext,+k}$ using Algorithm 2 by computing

$$T^{int,+k} = WinningCoalitionsByWeight(|P_k|,q,\tilde{w},w^{int},T^{ext,+k})$$

The vector $T^{int,+k}(x)$ states the number of winning coalitions for weights $q \leq x \leq \tilde{w}$ when players from precoalition $P_k$ join on individual level whereas the other precoalitions join as unions. From the vector $T^{int,+k}(x)$ we can work out how frequently an individual player $i \in P_k$ with weight $w_i^{int}$ belongs to a winning coalition with weight $q \leq x \leq \tilde{w}$ using Algorithm 4 via

$$T_{+i}^{int,+k} = CoalitionsWithPlayer(q,\tilde{w},w_i^{int},T^{int,+k})$$

From $T_{+i}^{int,+k}$ we can assess how frequently player $i$ is a swing player within precoalition $P_k$, i.e. assuming that all the other precoalitions are acting as unions. The expression

$$f_i^{BO}(N,v,P) = \frac{1}{2^{m-1}}\frac{1}{2^{|P_k|-1}}\sum_{x=q}^{q+w_i-1} T_{+i}^{int,+k}(x)$$

yields the BO index of player $i$. □

*4.2. Efficient computation of the Symmetric Coalitional Banzhaf (SCB) index*

As for the SCB index, our approach for the external game will be identical as for the BO index. However, in the internal game of each precoalition we will need to include cardinalities.

**Theorem 4.** *For a weighted voting game with positive integer weights and a precoalition structure $P = \{P_1,\ldots,P_m\}$, the Symmetric Coalitional Banzhaf (SCB) indices for all players can be computed in $O(\Delta m + \Delta p^2)$ time and $O(\Delta p)$ memory space with $\Delta = \min(q,\tilde{w}-q+1)$. For computing the SCB index of an individual player $i \in P_k$, i.e. player $i$ is contained in precoalition $P_k$, we need $O(\Delta m + \Delta|P_k|^2)$ time and $O(\Delta|P_k|)$ space.*

**Proof.** Again, we discuss the case $(\tilde{w}-q+1) \leq q$ and compute the SCB indices from above. On the external level, we perform the identical computations as for the BO indices. Let $w^{ext} = (w(P_1),\ldots,w(P_m))$ denote the vector of the weights of the $m$ precoalitions. We obtain the vector $T^{ext}(x)$ with the number of winning coalitions on the level of the precoalitions for weights $q \leq x \leq \tilde{w}$ via Algorithm 2 by computing

$$T^{ext} = WinningCoalitionsByWeight(m,q,\tilde{w},w^{ext},T)$$

using an integer vector $T$ with $T(\tilde{w}) = 1$ and $T(x) = 0$ for $x = q,\ldots,\tilde{w}-1$, as our input. As before, Algorithm 4 gives us the number of those winning coalitions containing precoalition $k$ with weight $w_k^{ext}$ via

$$T^{ext,+k} = CoalitionsWithPlayer(q,\tilde{w},w_k^{ext},T^{ext})$$

Now we switch to the level of precoalition $P_k$ with the individual weights $w^{int} = (w_1, \ldots, w_{|P_k|})$. We create a 2-dimensional array $C^{int,+k}$ with

$$C^{int,+k}(x, |P_k|) = T^{ext,+k}(x) \tag{3}$$

for all weights $q \leq x \leq \tilde{w}$ and $C^{int,+k}(x,s) = 0$ for all other combinations of weights $x = q, \ldots, \tilde{w}$, and cardinalities $s = 1, \ldots, |P_k| - 1$. We note that equation (3) states how many winning coalitions were joined by $P_k$ as a union. We incorporate the individual weights $w^{int}$ and their cardinalities using Algorithm 3 by computing

$$C^{int,+k} = WinningCoalitionsByWeightCardinality(|P_k|, q, \tilde{w}, w^{int}, C^{int,+k})$$

From our updated array $C^{int,+k}(x,s)$ we learn the numbers of winning coalitions with weights $x \in [q, \tilde{w}]$ with $s$ players from $P_k$, assuming that players from precoalition $P_k$ join on individual level whereas the other precoalitions join as unions. We can work out how frequently an individual player $i \in P_k$ with weight $w_i^{int}$ belongs to a winning coalition with $s$ players from $P_k$ via Algorithm 5 by computing

$$C_{+i}^{int,+k} = CoalitionsCardinalityWithPlayer(|P_k|, q, \tilde{w}, w_i^{int}, C^{int,+k})$$

$C_{+i}^{int,+k}$ states how frequently player $i$ is a part of a winning coalition containing $s$ players from $P_k$, i.e. assuming that all the other precoalitions are acting as unions. From

$$f_i^{SCB}(N, v, P) = \frac{1}{2^{m-1}} \frac{1}{|P_k|!} \sum_{s=0}^{|P_k|-1} s!(|P_k| - s - 1)! \sum_{x=q}^{q+w_i-1} C_{+i}^{int,+k}(x, s+1)$$

we obtain the SCB index of player $i$. □

*4.3. Efficient computation of the Owen (OW) index*

The computation of the OW index is slightly more sophisticated. As observed by Malawski (2004) [17], more than one weighted voting game is needed on the internal level. These internal games reflect how many precoalitions have joined a vulnerable coalition for which a certain player is critical. We translate the observations from [17] into a dynamic programming algorithm.

**Theorem 5.** *For a weighted voting game with positive integer weights and a precoalition structure $P = \{P_1, \ldots, P_m\}$, the Owen (OW) indices for all players can be computed in $O(\Delta(mp)^2)$ time and $O(\Delta(m+p))$ memory space with $\Delta = \min(q, \tilde{w} - q + 1)$. For computing the OW index of an individual player $i \in P_k$, i.e. player $i$ is contained in precoalition $P_k$, we need $O(\Delta(m|P_k|)^2)$ time and $O(\Delta(m + |P_k|))$ space.*

**Proof.** We discuss the case $(\tilde{w} - q + 1) \leq q$ and compute the OW indices from above. On the external level, we need to reflect cardinalities of precoalitions. As before, the vector $w^{ext} = (w(P_1), \ldots, w(P_m))$ stands for the weights of the $m$ precoalitions. We obtain the 2-dimensional array $C^{ext}(x,s)$ with the number of winning coalitions on the level of the precoalitions via Algorithm 3 for weights $x \in [q, \tilde{w}]$ and cardinalities $s = 1, \ldots, m$, in $O(m^2(\tilde{w} - q + 1))$ time and $O(m(\tilde{w} - q + 1))$ space by computing

$$C^{ext} = WinningCoalitionsByWeightCardinality(m, q, \tilde{w}, w^{ext}, C)$$

using a 2-dimensional array $C$ with $C(\tilde{w}, m) = 1$ and $C(x,s) = 0$ for all other combinations of weights $x = q, \ldots, \tilde{w}$, and cardinalities $s = 1, \ldots, m$, as our input. Algorithm 5 gives us the number of those winning coalitions containing precoalition $k$ with weight $w_k^{ext}$ together with their cardinalities via

$$C^{ext,+k} = CoalitionsCardinalityWithPlayer(m, q, \tilde{w}, w_k^{ext}, C^{ext}).$$

Now we need to loop for $r$ from 0 to $m-1$ and work out the internal games on the level of precoalition $P_k$ with the individual weights $w^{int} = (w_1, \ldots, w_{|P_k|})$. We create a 2-dimensional array $C^{int,+k}$ with

$$C^{int,r+1,+k}(x,|P_k|) = C^{ext,+k}(x,r+1) \tag{4}$$

for all weights $q \le x \le \tilde{w}$ and $C^{int,r+1,+k}(x,s) = 0$ for all other combinations of weights $x = q,\ldots,\tilde{w}$, and cardinalities $s = 1,\ldots,|P_k|-1$. We note that equation (4) states how many winning coalitions of cardinality $r+1$ joined by $P_k$ as a union there are on the external level. We incorporate the individual weights $w^{int}$ and their cardinalities using Algorithm 3 by computing

$$C^{int,r+1,+k} = WinningCoalitionsByWeightCardinality(|P_k|,q,\tilde{w},w^{int},C^{int,r+1,+k}).$$

From our updated array $C^{int,r+1,+k}(x,s)$ we learn the numbers of winning coalitions with weights $x \in [q,\tilde{w}]$ with $s$ players from $P_k$, assuming that players from precoalition $P_k$ join on individual level whereas the other $r$ precoalitions join as unions. We can work out how frequently an individual player $i \in P_k$ with weight $w_i^{int}$ belongs to a winning coalition with $s$ players from $P_k$ via Algorithm 5 by computing

$$C_{+i}^{int,r+1,+k} = CoalitionsCardinalityWithPlayer(q,\tilde{w},w_i^{int},C^{int,r+1,+k}).$$

$C_{+i}^{int,r+1,+k}$ states how frequently player $i$ is a part of a winning coalition containing $s$ players from $P_k$, i.e. assuming that all the other $r$ precoalitions are acting as unions. From

$$f_i^{OW}(N,v,P) = \frac{1}{m!}\frac{1}{|P_k|!}\sum_{r=0}^{m-1} r!(m-r-1)!\sum_{s=0}^{|P_k|-1} s!(|P_k|-s-1)!\sum_{x=q}^{q+w_i-1} C_{+i}^{int,r+1,+k}(x,s+1)$$

we obtain the OW index of player $i$ in $O(\Delta(m+|P_k|)^2)$ time and $O(\Delta(m+|P_k|))$ space. □

## 5. Numerical results and software

We implemented our new algorithms for the BO, SCB and OW indices presented in the previous section and tested our software thoroughly under MS Windows and Ubuntu Linux. Our implementations are freely available at *link and further information to be included after double-blind review.*

Users can specify weighted voting games as CSV (Comma Separated Values) files, i.e. EPIC supports a simple and widely used data format which allows users to edit input files using notepad, MS Excel and many other text editors. An input file contains one precoalition per line and weights of the players in the same precoalition are specified in the same line separated by commas, see *link to be included after double-blind review* for more details.

When computing power indices for large weighted voting games counting coalitions may force us to handle very large integers. For arbitrary-precision arithmetic EPIC uses GMP [37], i.e. the GNU Multiple Precision Arithmetic Library. GMP [37] is a very widely used, established and well-tested C++ library for working with very large integers. We are using GMP in the same vein as described in [28].

In order to test our new algorithms, we created a number of test problems and made them available at *link to be included after double-blind review.*

In the following, we discuss computing times and memory requirements for three of these example problems. The numerical results in Tables 1, 2 and 3 were 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 all three Tables 1, 2 and 3 we report computing times and storage requirements for a quota of 50 % plus 1 vote and compare them to a quota of 75 % of the votes. By default, we compute power indices from above since normally $q > \frac{1}{2}\tilde{w}$. As expected from our theoretical considerations, computing times and storage are approximately cut in half for the larger quota of 75 %. All three Tables 1, 2 and 3 investigate the BO, SCB and OW indices using the new algorithms for handling precoalition structures introduced in Section 4 and compare the Banzhaf (BZ) index (without precoalition structure).

Table 1 is the only instance where we additionally compare the Shapley-Shubik (SH) index (without precoalition structure). The results in Table 1 illustrate the pseudopolynomial time and space complexities reported in Theorem 2 and Theorem 5. The presence of a coalition structure makes computing the OW index slower rather than faster as compared to the SH index. However, the OW index can be computed with significantly smaller storage requirements than the SH index. For the problems in Tables 2 and 3, each containing more than 3000 players, we chose to omit the SH indices for storage reasons.

**Table 1.** Computing times and memory requirements for an example with 741 players, $\tilde{w} = 37064$, 40 precoalitions, a maximum coalition size of 40 and an average coalition size of 18.

| | $q = 18533$ | (50% plus 1 vote) | $q = 27799$ | (75% of votes) |
|---|---|---|---|---|
| Index | time (sec) | memory (MB) | time (sec) | memory (MB) |
| BZ | 0.619 | 11.172 | 0.253 | 8.78 |
| SH | 594.522 | 2008.62 | 248.898 | 933.676 |
| SCB | 22.77 | 88.396 | 11.11 | 47.628 |
| BO | 0.539 | 10.956 | 0.26 | 8.86 |
| OW | 777.103 | 165.812 | 370.018 | 86.34 |

In Tables 1, 2 and 3, the computations of both the BO indices and the BZ index (without a coalition structure) never need more than one minute indicating that our algorithms could handle much larger problems. As predicted from the pseudopolynomial complexities in Theorem 1 and Theorem 3, computing times and storage needs are lower for the BO indices than for the BZ indices for large problem instances, as exemplified in the test problems in Tables 2 and 3. In the smaller test problem in Table 1 with 741 players, this effect is not yet visible as the implementation of the BO index is more sophisticated and more small arrays need to be allocated (in order to deal with the precoalition structure) than for the BZ index. As predicted in Section 4, the SCB indices need more storage and computing time than the BO indices due to the Shapley-Shubik computations for the internal games.

Comparing the results from Tables 2 and 3 for the SCB indices, we confirm the theoretical findings from Theorem 4. For computing SCB indices, it is more favourable to have more precoalitions rather than large average coalition sizes (both in terms of speed and storage). For the computation of the OW indices, no such statement can be made as the OW index uses the Shapley-Shubik index both for the external and internal decision processes. This is reflected in the pseudopolynomial time and storage complexities from Theorem 5 and confirmed in the numerical results in Tables 2 and 3.

**Table 2.** Computing times and memory requirements for an example with 3034 players, $\tilde{w} = 152098$, 60 precoalitions, a maximum coalition size of 78 and an average coalition size of 50.

| | $q = 76050$ | (50% plus 1 vote) | $q = 114074$ | (75% of votes) |
|---|---|---|---|---|
| Index | time (sec) | memory (MB) | time (sec) | memory (MB) |
| BZ | 52.001 | 59.216 | 20.318 | 28.388 |
| SCB | 806.655 | 665.86 | 401.751 | 336.364 |
| BO | 9.193 | 23.612 | 4.469 | 15.28 |
| OW | 38181.746 | 1146.04 | 18638.011 | 574.436 |

**Table 3.** Computing times and memory requirements for an example with 3434 players, $\tilde{w} = 72068$, 200 precoalitions, a maximum coalition size of 54 and an average coalition size of 17.

| | $q = 36035$ | (50% plus 1 vote) | $q = 54051$ | (75% of votes) |
|---|---|---|---|---|
| Index | time (sec) | memory (MB) | time (sec) | memory (MB) |
| BZ | 27.694 | 35.768 | 8.959 | 19.324 |
| SCB | 578.455 | 316.324 | 259.954 | 144.224 |
| BO | 6.067 | 17.028 | 2.879 | 11.896 |
| OW | 86139.148 | 1154.708 | 39411.552 | 559.856 |

## 6. Outlook and conclusions

In this paper we present new dynamic programming algorithms for the three most common power indices with precoalitions, i.e. the Owen (OW), Banzhaf-Owen (BO) and Symmetric Coalitional Banzhaf (SCB) indices. Our algorithms reflect the external and internal decision processes between and within precoalitions used to specify these indices. We provide efficient C++ implementations of our new algorithms and point out that our methods can be applied for large numbers of players.

Still, there are many open questions in connection with the computation of power indices with precoalitions. Certain power indices with a priori unions, like e.g. [38,39], do not fit the framework used in this paper and it will be interesting to see how dynamic programming can be applied efficiently in these cases. In this context we would like to stress that the very general framework for computing power indices via quasi-ordered binary decision diagrams [40–43], i.e. a recent approach based on relational algebra, has not yet been extended to power indices with a coalition structure. Furthermore, the mathematical relations between dynamic programming and generating functions [5,22–24] for power index computation justify further research, both in the cases with and without precoalitions. We expect these two paradigms to be very fruitful for each other and such investigations have the potential to lead to even faster algorithms.

Another challenge is the parallel computation of power indices. To our knowledge, there are no publications on parallel computations of any of the power indices (without precoalitions) discussed in [28] or the dynamic programming algorithms presented therein. The new algorithms for power indices with precoalitions presented in the proofs of Theorems 3, 4 and 5 appear to be particularly well suited for parallel processing, given that the internal games can be computed independently.

In terms of practical calculations for cooperative games with a coalition structure, we regret a lack of software offering more than the computation of the Owen value [16], i.e. the most widely used solution concept. In our view, even a prototypical implementation of various solution concepts for games with a partition structure, e.g. similar to the freely available R package CoopGame [44] for cooperative games, might be helpful in popularizing these ideas for analyzing real-world problems. A very concrete real-world application we wish to study using our new algorithms is the problem of indirect control in complex shareholding structures [10–12]. These networks are usually large and, to our knowledge, precoalitions of companies and investors within corporate networks have yet to be investigated.

**Author Contributions:** Author 1 conceived the presented algorithms and performed the formal and theoretical analysis. Author 2 and Author 3 validated the algorithms and performed the numerical simulations. All authors discussed the results and contributed to the final manuscript. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research was partially funded by the Bavarian Ministry of Science and Arts.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Acknowledgments:** The authors would like to thank Sascha Kurz (University of Bayreuth) for having made the code he used for [7] publicly available via ResearchGate and having allowed them to use it as a sample for their software.

**Conflicts of Interest:** The authors declare no conflict of interest.

1. Bertini, C.; Freixas, J.; Gambarelli, G.; Stach, I. Comparing power indices. *Int. Game Theory Rev.* **2013**, *15*, 1340004. DOI: 10.1142/S0219198913400045
2. Penrose, L.S. The elementary statistics of majority voting. *Journal of the Royal Statistical Society* **1946**, *109*, 53–57.
3. Banzhaf III, J.F. Weighted voting doesn't work: A mathematical analysis. *Rutgers L. Rev.* **1964**, *19*, 317.
4. Shapley, L.S.; Shubik, M. A method for evaluating the distribution of power in a committee system. *Amer. Pol. Sci. Rev.* **1954**, *48*, 787–792.
5. Algaba, E.; Bilbao, J.M.; Fernández-García, J.R. The distribution of power in the European Constitution. *Eur. J. Oper. Res.* **2007**, *176*, 1752–1766. DOI: 10.1016/j.ejor.2005.12.002
6. Kóczy, L.A. Beyond Lisbon. Demographic trends and voting power in the European Union Council of Ministers. *Math. Soc. Sci.* **2012**, *63*, 152–158. DOI: doi.org/10.1016/j.mathsocsci.2011.08.005
7. Kurz, S. Computing the power distribution in the IMF. arXiv Preprint **2016**, arXiv: 1603.01443.
8. Lucchetti R.; Radrizzani P. Microarray Data Analysis via Weighted Indices and Weighted Majority Games. In *Computational Intelligence Methods for Bioinformatics and Biostatistics. CIBB 2009. Lecture Notes in Computer Science*; Masulli, F., Peterson, L.E., Tagliaferri, R., Eds.; Springer: Berlin/Heidelberg, Germany, 2009; Volume 6160, pp. 179–190. DOI: 10.1007/978-3-642-14571-1_13
9. Bachrach, Y.; Rosenschein, J.S.; Porat, E. Power and stability in connectivity games. In *Proceedings of the 7th international joint conference on Autonomous agents and multiagent systems*, 2008, Volume 2, pp. 999–1006.
10. Stach, I.; Mercik, J.; Bertini, C. Some propositions of approaches for measuring indirect control power of firms and mutual connections in corporate shareholding structures. In *Transactions on Computational Collective Intelligence XXXV. Lecture Notes in Computer Science*, Nguyen, N.T., Kowalczyk, R., Mercik, J., Motylska-Kuźma, A., Eds.; Springer: Berlin/Heidelberg, Germany, 2020; Volume 12330, pp. 116–132. DOI: 10.1007/978-3-662-62245-2_8
11. Stach, I.; Mercik, J. Measurement of control power in corporate networks. *Oper. Res. Dec.* **2021**, *31*, 97–121. DOI: 10.37190/ord210106
12. Staudacher, J.; Olsson, L.; Stach, I. Implicit power indices for measuring indirect control in corporate structures. In *Transactions on Computational Collective Intelligence XXXVI. Lecture Notes in Computer Science*; Nguyen, N., Kowalczyk, R., Mercik, J., Motylska-Kuźma, A. Eds.; Springer: Berlin/Heidelberg, Germany, 2021; Volume 13010, 21 pages. In Press.
13. Holler, M.J.; Rupp, F. Power in Networks: A PGI Analysis of Krackhardt's Kite Network. In *In Transactions on Computational Collective Intelligence XXXIV. Lecture Notes in Computer Science*; Nguyen, N., Kowalczyk, R., Mercik, J., Motylska-Kuźma, A. Eds.; Springer: Berlin/Heidelberg, Germany, 2019; Volume 11890, pp. 21–34. DOI: 10.1007/978-3-662-60555-4_2
14. Holler, M.J.; Rupp, F. Power in Networks: The Medici. *Homo Oecon.* **2021**, *38*, 1–17. DOI: 10.1007/s41412-021-00108-1
15. Owen, G. *Game Theory*, 3rd ed.; Academic Press: London, United Kingdom, 1995.
16. Owen, G. Values of Games with a Priori Unions. In *Mathematical Economics and Game Theory. Lecture Notes in Economics and Mathematical Systems*; Henn R., Moeschlin, O., Eds.; Springer: Berlin/Heidelberg, Germany, 1977; Volume 141, pp. 76-88. DOI: 10.1007/978-3-642-45494-3_7
17. Malawski, M. "Counting" Power Indices for Games with a Priori Unions. In *Essays in Cooperative Games. Theory and Decision Library*; Gambarelli, G., Eds.; Springer: Boston, MA, 2004; Volume 36, pp. 125–140. DOI: 10.1007/978-1-4020-2936-3_10
18. Alonso-Meijide, J.M.; Bowles, C.; Holler, M.J.; Napel, S. Monotonicity of power in games with a priori unions. *Theory and Decision* **2009**, *66*, 17–37. DOI: 10.1007/s11238-008-9114-2
19. Alonso-Meijide, J.M.; Fiestras-Janeiro, M.G. Modification of the Banzhaf value for games with a coalition structure. *Ann. Oper. Res.* **2002**, *109*, 213–227. DOI: 10.1023/A:1016356303622
20. Owen, G. Modification of the Banzhaf-Coleman Index for Games with a Priori Unions. In *Power, Voting, and Voting Power.* Holler, M.J., Eds.; Physica: Heidelberg, Germany, 1981; pp. 232–238. DOI: 10.1007/978-3-662-00411-1_17
21. Matsui, Y.; Matsui, T. NP-completeness for calculating power indices of weighted majority games. *Theor. Comp. Sci.* **2001**, *263*, 305–310.
22. Algaba, E.; Bilbao, J.M.; Fernández-García, J.F.; López, J.J. Computing power indices in weighted multiple majority games. *Math. Soc. Sci.* **2003**, *46*, 63–80. DOI: 10.1016/S0165-4896(02)00086-0
23. Bilbao, J.M.; Fernández-García, J.F.; Jiménez Losada, A.; López, J.J. Generating functions for computing power indices efficiently. *TOP* **2000**, *8*, 191–213. DOI: 10.1007/BF02628555
24. Alonso-Meijide, J.M.; Bowles, C. Generating Functions for Coalitional Power Indices: An Application to the IMF. *Ann. Oper. Res.* **2005**, *137*, 21–44. DOI: 10.1007/s10479-005-2242-y
25. Tanenbaum, P. Power in weighted voting games. *Mathematica Journal* **1997**, *7*, 58–63.
26. Matsui, T.; Matsui, Y. A survey of algorithms for calculating power indices of weighted majority games. *J. Oper. Res. Soc. Japan* **2000**, *43*, 71–86.
27. Uno, T. Efficient Computation of Power Indices for Weighted Majority Games. In *International Symposium on Algorithms and Computation. ISAAC 2012. Lecture Notes in Computer Science*; Chao, K.M., Hsu, T., Lee, D.T., Eds.; Springer: Berlin/Heidelberg, Germany, 2020; Volume 7676, pp. 679–689. DOI: 10.1007/978-3-642-35261-4_70
28. Staudacher, J.; Kóczy, L.Á.; Stach, I.;Filipp, J.; Kramer, M.; Noffke, T.; Olsson, L.; Pichler, J.; Singer, T. Computing power indices for weighted voting games via dynamic programming. *Oper. Res. Dec.* **2021**, *31*, 123–145. DOI: 10.37190/ord210206
29. Johnston, R.J. On the measurement of power. Some reactions to Laver. *Environ. Plan. A* **1978**, *10*, 907–914.
30. Deegan, J.; Packel, E.W. A new index of power for simple n-person games. *Int. J. Game Theory* **1978**, *7*, 113–123.
31. Holler, M.J. Forming coalitions and measuring voting power. *Pol. Studies* **1982**, *30*, 262–271.

32. Felsenthal, D.S. A well-behaved index of a priori p-power for simple n-person games. *Homo Oecon.* **2016**, *33*, 367–381. DOI: 10.1007/s41412-016-0031-2
33. Molinero, X.; Blasco, J. Coalitional power indices applied to voting systems. In *Proceedings of the 9th International Conference on Operations Research and Enterprise Systems*, 2020, pp. 372–376. DOI: 10.5220/0009166803720376
34. Mercik, J.; Ramsey, D.M. The effect of Brexit on the balance of power in the European Union Council: An approach based on pre-coalitions. In *Transactions on Computational Collective Intelligence XXVII. Lecture Notes in Computer Science*; Nguyen, N., Kowalczyk, R., Mercik, J., Eds.; Springer: Cham, Switzerland, 2017; Volume 10480, pp. 87–107. DOI: 10.1007/978-3-319-70647-4_7
35. Chakravarty, S.R.; Mitra, M.; Sarkar, P. *A Course on Cooperative Game Theory*, Cambridge University Press: Cambridge, United Kingdom, 2015.
36. Laruelle, A.; Valenciano, F. On the meaning of Owen–Banzhaf coalitional value in voting situations. *Theory and Decision* **2004**, *56*, 113–123. DOI: 10.1007/s11238-004-5639-1
37. The GNU Multiple Precision Arithmetic Library. Available online: https://gmplib.org/ (accessed on 17 November 2021).
38. Alonso-Meijide, J.M.; Casas-Méndez, B.; Fiestras-Janeiro, M.G.; Holler, M.J. Two variations of the Public Good Index for games with a priori unions. *Control. Cybern.* **2010**, *39*, 839–855.
39. Alonso-Meijide, J.M.; Casas-Méndez, B.; Fiestras-Janeiro, M.G.; Holler, M. J. The Deegan-Packel index for simple games with a priori unions. *Qual. Quant.* **2011**, *45*, 425–439. DOI: 10.1007/s11135-009-9306-z
40. Berghammer, R.; Bolus, S.; Rusinowska, A.; De Swart, H. A relation-algebraic approach to simple games. *Eur. J. Oper. Res.* **2011**, *210*, 68–80. DOI: 10.1016/j.ejor.2010.09.006
41. Berghammer, R.; Bolus, S. On the use of binary decision diagrams for solving problems on simple games. *Eur. J. Oper. Res.* **2012**, *222*, 529–541. DOI: 10.1016/j.ejor.2010.09.020
42. Bolus, S. Power indices of simple games and vector-weighted majority games by means of binary decision diagrams. *Eur. J. Oper. Res.* **2011**, *210*, 258–272. DOI: 10.1016/j.ejor.2010.09.020
43. Bolus, S. A QOBDD-based approach to simple games. Ph.D. thesis, Christian-Albrechts Universität Kiel, Kiel, Germany, 2012.
44. Staudacher, J.; Anwander J. Using the R package CoopGame for the analysis, solution and visualization of cooperative games with transferable utility. Available online: https://cran.r-project.org/package=CoopGame (accessed on 17 November 2021), *R Vignette for package version 0.2.2*, 2021.