ISSN 1346-5597

National Institute of Informatics

NII Technical Report

Efficient Computation of Power Indices for Weighted Majority Games

Takeaki Uno

NII-2003-006E
July 2003

URL: http://www.nii.ac.jp/ Tokyo 101-8430, JAPAN +81-3-4212-2000 FAX: +81-3-3556-1916

Efficient Computation of Power Indices for Weighted Majority Games

Takeaki UNO *

July 11, 2003

Abstract: Power indices of weighted majority games are measures of the effects of parties on the voting in a council. Among the many kinds of power indices, Banzhaf index, Shapley-Shubik index and Deegan-Packel index have been studied well. For computing these power indices, dynamic programming algorithms had been proposed. The time complexities of these algorithms are $O(n^2q)$, $O(n^3q)$, and $O(n^4q)$, respectively. We propose new algorithms for computing power indices, whose time complexities are $O(nq)$, $O(n^2q)$, and $O(n^2q)$, respectively.

Keywords: weighted majority game, power index, Banzhaf index, Shapley-Shubik index, Deegan-Packel index, dynamic programming

1 Introduction

Let $\{p_1, ..., p_n\}$ be a set of players, and w_i be the weight of player p_i. We consider a council composed of parties each of which corresponds to a player. Suppose that the weight of a party is the number of members of the party. For each proposal, each party (player) votes "yes" or "no". If the sum of the weights of the "yes" votes is larger than a constant q, the proposal is accepted. Constant q is called a *quota*. A *weighted majority game* is a game dealing with this situation, composed of these players, their weights, and the quota. Each player has a distinct weight for their vote, thus the effect of each player on the voting is different. Many kinds of power indices have been proposed for measuring the intensity of these effects.

Among these power indices proposed for weighted majority games, Banzhaf index, Shapley-Shubik index and Deegan-Packel index have been well studied. In general, computing exact values of Banzhaf index, Shapley-Shubik index and Deegan-Packel index take huge computation time if the number of players n is large, since existing algorithms for computing these indices take $O(2^n)$ time. Under the condition that all the weights of the players are integer, these indices are computed in polynomial time of n and q by dynamic programming algorithms [4, 2, 5]. For computing the power index of a player, the algorithm for computing Banzhaf index takes $O(nq)$ time [4, 2], the algorithm for Shapley-Shubik index takes $O(n^2q)$ time [4, 2], and the algorithm for Deegan-Packel index takes $O(n^3q)$ time [5]. For computing indices of all players, these algorithms take $O(n^2q)$ time, $O(n^3q)$ time, and $O(n^4q)$ time, respectively.

*National Institute of Informatics, 2-1-2, Hitotsubashi, Chiyoda-ku, Tokyo, Tokyo 101-8430, Japan. uno@nii.jp

Roughly speaking, an exponential time algorithm does not terminate in one hour when the input size is up to 30. In a practical sense, a computation should terminate at most in an hour, hence this is a limit for exponential algorithms. These days, a PC can execute about 10^{11} basic calculations in an hour. If qn^2, qn^3 or qn^4 are larger than this number, dynamic programming algorithms take more than one hour for computation. Usually, basic problems such as computing power indices are solved many times repeatedly, or solved as a sub-problem of another large complicated problem. Algorithms for solving these basic problems should be speeded up so that they can terminate in a short time.

In this paper, we propose new algorithms for computing these three power indices of all players. The framework of our algorithms is to compute the power indices of two players by existing dynamic programming algorithms, and compute indices of other players by using the computational results. By our algorithms, Banzhaf index and Shapley-Shubik of all players are computed in $\mathrm{O}(nq)$ time and $\mathrm{O}(n^2q)$ time, respectively. Our algorithm for Deegan-Packel index is based on a new dynamic programming taking $\mathrm{O}(n^2q)$ time for one player. Hence, the new algorithm takes only $\mathrm{O}(n^2q)$ time for all players. The constant factor of the time complexity increases about 3, i.e. the computing time of our algorithms are almost equal to that of original dynamic programming algorithms for three players. For example, if $n = 50$, the increase of speed is about 16 times.

These algorithms are described in the following sections. Section 2 explains our algorithm for Banzhaf index, which is the simplest one among our three algorithms. In this section, we explain the common basic idea of our algorithms. In Section 3, we explain the way to save unnecessary computation. Section 4 and Section 5 describe the algorithms for Shapley-Shubik index and Deegan-Packel index, respectively.

2 Algorithm for Banzhaf Index

We define several notations. A set of players is called a *coalition*. For a coalition S, we define the weight of S by $\sum_{p_i \in S} w_i$, and denote it by $w(S)$. In particular, we define $w(\emptyset) = 0$. If $w(S) \geq q$ holds, S is called a *winner*, and if $w(S) < q$ holds, S is called a *loser*. Let P_i be the set of players from p_1 through p_i, i.e., $P_i = \{p_1, p_2, ..., p_i\}$, and $\bar{P}_i$ be the set of players from p_i through p_n, i.e., $\bar{P}_i = \{p_i, p_{i+1}, ..., p_n\}$. For a function $g(p, a, b, ...)$, we define $V(g(p_i))$ be the set of the values of g for all possible combination of parameters $p_i, a, b, ...$

Let S be a winner including player p_i. If S becomes a loser when p_i exits from S, then p_i is considered to have a power in S. By assuming that every coalition occurs randomly at the same probability, then the probability that p_i belongs to the coalition and has a power in the coalition is

$$\mid \{S | S \subseteq \{p_1, ..., p_n\}, p_i \in S, w(S) \geq q, w(S \setminus \{p_i\}) < q\} \mid \ / \ 2^n.$$

This probability is the definition of the Banzhaf index [1] of player p_i. We denote it by $Bz(p_i)$.

A dynamic programming algorithm for computing $Bz(p_n)$ was proposed in [4, 2]. This algorithm computes the index by computing all the values of a function, which we denote it by f in this paper. For any player p_i and any $y, 0 \leq y \leq q-1$, f is defined by the number of coalition $S \subseteq P_i$ satisfying $w(S) = y$, i.e.,

$$f(p_i, y) = \mid \{S | S \subseteq P_i, w(S) = y\} \mid.$$

Since

$$\begin{aligned} Bz(p_n) \times 2^n &= \mid \{S | S \subseteq P_n, p_n \in S, w(S) \geq q, w(S \setminus \{p_n\}) < q\} \mid \\ &= \mid \{S | S \subseteq P_{n-1}, q - w_n \leq w(S) \leq q - 1\} \mid \\ &= \sum_{y=q-w_n}^{q-1} f(p_{n-1}, y), \end{aligned}$$

we can compute $Bz(p_n)$ from $V(f(p_{n-1}))$ in $\mathrm{O}(q)$ time. To compute the values of f, the following lemma is used.

Property 1 *For any* $2 \leq i \leq n$ *and any* $0 \leq y \leq q-1$,

$$f(p_i, y) = \begin{cases} f(p_{i-1}, y) + f(p_{i-1}, y - w_i) & \text{if} \quad y \geq w_i \\ f(p_{i-1}, y) & \text{if} \quad y < w_i \end{cases}$$

holds. ∎

This property shows that $V(f(p_i))$ is computed from $V(f(p_{i-1}))$ in $\mathrm{O}(q)$ time. Since each value of $V(f(p_1))$ can be computed directly in $\mathrm{O}(1)$ time, $V(f(p_{n-1}))$ can be computed in $\mathrm{O}(nq)$ time. This is the basic idea of the existing dynamic programming algorithms of [4, 2]. For player $p_i, i < n$, we exchange the indices of players p_i and p_n, and compute $Bz(p_n)$. Therefore, the time complexity of the existing dynamic programming algorithm is $\mathrm{O}(n^2 q)$ for all players.

This dynamic programming algorithm seems to do no unnecessary operations to compute $Bz(p_n)$. Hence, reducing time complexity of the dynamic programming algorithm for computing $Bz(p_n)$ seems to be hard. However, the algorithm does quite similar operations for computing $Bz(p_n)$ and that of $Bz(p_i)$. In this point, we can see a possibility of improving. In the following, we define new functions b and h, and propose an algorithm for computing $Bz(p_1), Bz(p_2), ..., Bz(p_n)$ by using these functions, instead of solving n similar dynamic programming problems.

The functions are defined as follows:

$$b(p_i, y) = \mid \{S | S \subseteq \bar{P}_i, w(S) = y\} \mid, \quad \text{and}$$

$$h(p_i, z) = \sum_{y=0}^{z} b(p_i, y).$$

For $y = -1$, we define $h(p_i, y) = 0$. The function f is used to solve the dynamic programming in the forward direction, and b is that in the backward direction. Since f and b are symmetric, the following equation holds according to Property 1.

$$b(p_i, y) = \begin{cases} b(p_{i+1}, y) + b(p_{i+1}, y - w_i) & \text{if} \quad y \geq w_i \\ b(p_{i+1}, y) & \text{if} \quad y < w_i \end{cases}$$

This shows that $V(b(p_i))$ can be computed from $V(b(p_{i+1}))$ in $\mathrm{O}(q)$ time, in the same way as f. $V(h(p_i))$ can be computed from $V(b(p_i))$ in $\mathrm{O}(q)$ time.

Now we explain the way to compute $Bz(p_i)$ by using f, b and h. For each coalition S not including p_i, consider the partition of S, which are $S \cap P_{i-1}$ and $S \cap \bar{P}_{i+1}$. Thus, the condition

$$q - w_i \leq w(S) \leq q - 1$$

is equal to

$$q - w_i \leq w(S \cap P_{i-1}) + w(S \cap \bar{P}_{i+1}) \leq q - 1.$$

Hence,

$$\begin{aligned} Bz(p_i) \times 2^n &= |\{S | p_i \notin S, q - w_i \leq w(S) \leq q - 1\}| \\ &= |\{S | p_i \notin S, q - w_i \leq w(S \cap P_i) + w(S \cap \bar{P}_i) \leq q - 1\}| \\ &= |\{(S_1, S_2) | S_1 \subseteq P_{i-1}, S_2 \subseteq \bar{P}_{i+1}, q - w_i \leq w(S_1) + w(S_2) \leq q - 1\}| \\ &= \sum_{z=0}^{q-1} |\{S_1 | S_1 \subseteq P_{i-1}, w(S_1) = z\}| \times \sum_{y=\max\{q-w_i-z,0\}}^{q-1-z} |\{S_2 | S_2 \subseteq \bar{P}_{i+1}, w(S_2) = y\}|. \end{aligned}$$

In the last line, the left side and right side can be computed by using f and b, respectively. Hence,

$$\begin{aligned} Bz(p_i) \times 2^n &= \sum_{z=0}^{q-1} f(p_{i-1}, z) \times \sum_{y=\max\{q-w_i-z,0\}}^{q-1-z} b(p_{i+1}, y) \\ &= \sum_{z=0}^{q-1} f(p_{i-1}, z) \times \left(h(p_i, q-1-z) - h(p_i, \max\{q - w_i - z, 0\} - 1)\right). \end{aligned}$$

Since the last line takes $\mathrm{O}(q)$ time to be computed, we have the following theorem.

Theorem 1 *For a weighted majority game with n players, weights $w_1, ..., w_n$ of players and a quota q, Banzhaf indices for all players can be computed in $O(nq)$ time and $O(nq)$ space.* ∎

3 Reducing Space Complexity and Unnecessary Operations

The algorithm described in the previous section is quite basic. Hence, slight modifications can reduce its computation time and the space complexity. Note that the modifications do not reduce the time complexity.

First, we explain the technique for reducing the space complexity. Transforming the statement of Property 1 gives the following equation. For $2 \leq i \leq n$,

$$f(p_{i-1}, y) = \begin{cases} f(p_i, y) & \text{if } 0 \leq y \leq w_i \\ f(p_i, y) - f(p_{i-1}, y - w_i) & \text{if } w_i \leq y \leq q - 1. \end{cases}$$

From this, we can compute all the elements of $V(f(p_i))$ from $V(f(p_{i+1}))$ in the increasing order of y, in $\mathrm{O}(q)$ time. At the beginning of the algorithm, we compute $V(f(p_{n-1}))$. After computing $Bz(p_n)$, we compute $V(f(p_i)$ and $V(b(p_{i+2}))$ in the backward direction, and compute $Bz(p_i)$. After computing $Bz(p_i)$, $V(f(p_{i-1}))$ and $V(b(p_{i+1}))$ are never referred. Thus, we delete them from the memory. As a result of the deletion, when we compute $Bz(p_i)$, only $V(f(p_{i-1}))$ and $V(b(p_{i+1}))$ are on the memory. Hence the space complexity is reduced to $\mathrm{O}(q)$.

Second, we explain the reduction of unnecessary operations. In the above algorithms, we compute all the elements of $V(f(p_i))$ and $V(b(p_i))$. However, in practical computation, a large number of them are unnecessary. Clearly, we need only non-zero elements, hence we represents $V(f(p_i))$ by a linked list of its non-zero elements. For $2 \leq i \leq n$, suppose that $f(p_i, y) > 0$. From the definition of f, we obtain:

if $0 \leq y < w_i$, then $f(p_{i-1}, y) > 0$, and
if $w_i \leq y \leq q-1$, then $f(p_{i-1}, y) > 0$ or $f(p_{i-1}, y - w_i) > 0$.

Hence, $f(p_{i+1}, y) > 0$ if and only if $f(p_i, y) > 0$ or $f(p_i, y - w_i) > 0$. Therefore, we can compute $V(f(p_i))$ by tracing the list representing $V(f(p_{i-1}))$. Moreover, if $y < q - w(\bar{P}_i)$, then

$$h(p_{i+1}, q-y-1) - h(p_{i+1}, \max\{q-y-w_i-1, 0\} - 1) = 0$$

holds, hence $f(p_{i-1}, y)$ does not need to be computed in this case. In the same way, computation of the values of b and h can be saved.

Theorem 2 *For a weighted majority game with n players and a quota q, Banzhaf indices for all players can be computed in $O(nq)$ time and $O(q)$ space.* ∎

4 Computing Shapley-Shubik Index

Suppose that, at first, a coalition S is an empty set, and players participate to the coalition one by one in an order. We suppose that the coalition changes from a loser to a winner when a player p_i participates in S. Then, we can naturally consider that p_i has power. Let Π_n be the set of permutations with length n. We note that $|\Pi_n| = n!$. If the order occurs randomly and uniformly, then the probability that p_i has power is

$$\mid \{(j_1, ..., j_n) \in \Pi_n | q \leq w(\{p_{j_1}, p_{j_2}, ..., p_i\}) < q + w_i\} \mid \ / \ n!.$$

This is a definition of Shapley-Shubik index [6] of player p_i, denoted by $Ss(p_i)$.

For computing Shapley-Shubik index, a dynamic programming algorithm has been proposed [4, 2]. This algorithm also computes the power indices by computing the values of a function defined in a similar way to Banzhaf index. Here we denote the function by f. For $1 \leq i \leq n$, $0 \leq y \leq q-1$, and $0 \leq k \leq i$, function f is defined as

$$f(p_i, k, y) = \mid \{S | S \subseteq P_i, w(S) = y, |S| = k\} \mid.$$

For any player p_i, the number of permutations $(j_1, ..., j_n)$ satisfying $\{p_{j_1}, p_{j_2} ..., p_i\} = S$ is $(|S|-1)!(n-|S|)!$, thus,

$$\begin{aligned} Ss(p_i) \times n! &= \mid \{(j_1, ..., j_n) \in \Pi_n | q \leq w(\{p_{j_1}, p_{j_2}, ..., p_i\}) < q + w_i\} \mid \\ &= \sum_{S | p_i \notin S, q - w_i \leq w(S) \leq q-1} |S|!(n - |S| - 1)! \\ &= \sum_{y=q-w_i}^{q-1} \sum_{k=0}^{i-1} \sum_{S | p_i \notin S, w(S)=y, |S|=k} k!(n-k-1)!, \end{aligned}$$

thereby

$$Ss(p_n) \times n! = \sum_{y=q-w_n}^{q-1} \sum_{k=0}^{n-1} f(p_{n-1}, k, y).$$

Therefore, $Ss(p_n)$ can be computed from $V(f(p_{n-1}))$. To compute f, the following property similar to Property 1 is used.

Property 2 *For any i, k, and y satisfying $2 \leq i \leq n, 1 \leq k \leq i$ and $0 \leq y \leq q-1$,*

$$f(p_i,k,y) = \begin{cases} f(p_{i-1},k,y) + f(p_{i-1},k-1,y-w_i) & \text{if} \quad y \geq w_i \\ f(p_{i-1},k,y) & \text{if} \quad y < w_i. \end{cases}$$

∎

From this property, $V(f(p_i))$ can be computed from $V(f(p_{i-1}))$ in $\mathrm{O}(nq)$ time. Since each element of $V(f(p_1))$ can be computed directly, $V(f(p_{n-1}))$ and $Ss(p_n)$ can be computed in $\mathrm{O}(n^2q)$ time. The indices of the other players can be computed in the same way by exchanging the indices. Hence, computing $Ss(p_i)$ for all players takes $\mathrm{O}(n^3q)$ time. This is the framework of the algorithm in [4, 2].

The algorithm also seems to have no unnecessary operations involved in the dynamic programming. However, as in the case of Banzhaf index, computing $Ss(p_i)$ for all players involves unnecessary operations, that is, solving n similar dynamic programming problems.

Although the idea of the improved algorithm for Banzhaf index seems to be applicable to Shapley-Shubik index, it does not work efficiently. If b is defined as the same way as Banzhaf index as follows,

$$b(p_i,k,y) = \mid \{S | S \subseteq \bar{P}_i, w(S) = y, |S| = k\} \mid,$$

$Ss(p_i)$ is given by

$$Ss(p_i) \times n! = \sum_{z=0}^{q-1} \sum_{k=0}^{i-1} \sum_{y=\max\{0,q-w_i-z\}}^{q-1-z} \sum_{l=0}^{n-(i-1)} f(p_{i-1},k,z) \times b(p_{i+1},l,y) \times |k+l|!|n-k-l-1|!.$$

The right side of the equation includes four summations. By the idea of using a function h defined in the previous section, the third summation can be eliminated, however three summations still remain. Hence, we have to spend $\mathrm{O}(n^2q)$ time to compute $Ss(p_i)$ for a player p_i in this way. This does not decrease the time complexity.

Our improved algorithm uses b and h defined as follows

$$\begin{aligned} b(p_i,k,y) &= \sum_{S, S \subseteq \bar{P}_i, w(S)=y} (|S|+k)!(n-|S|-k-1)! \text{ and} \\ h(p_i,k,z) &= \begin{cases} \sum_{y=0}^{z} b(p_i,k,y) & \text{if} \quad 0 \leq y \leq q-1 \\ 0 & \text{if} \quad y = -1 \end{cases}. \end{aligned}$$

Property 3 *For i and k with $1 \leq i < n$ and $0 \leq k \leq i$,*

$$b(p_i,k,y) = \begin{cases} b(p_{i+1},k,y) + b(p_{i+1},k+1,y-w_i) & \text{if} \quad y \geq w_i \\ b(p_{i+1},k,y) & \text{if} \quad y < w_i \end{cases}.$$

Proof : For any $S \subseteq \bar{P}_i$, if $w(S) < w_i$ then $p_i \notin S$. Hence,

$$b(p_i,k,y) = b(p_{i+1},k,y)$$

holds if $y < w_i$. In the case that $y \geq w_i$,

$$\begin{aligned} b(p_i,k,y) &= \sum_{S | S \subseteq \bar{P}_i, w(S)=y} (|S|+k)!(n-|S|-k-1)! \\ &= \sum_{S | S \subseteq \bar{P}_i, w(S)=y, p_i \notin S} (|S|+k)!(n-|S|-k-1)! \end{aligned}$$

$$
\begin{aligned}
&+ \sum_{S|S\subseteq \bar{P}_i, w(S)=y, p_i\in S} (|S|+k)!(n-|S|-k-1)! \\
= &\sum_{S|S\subseteq \bar{P}_{i+1}, w(S)=y} (|S|+k)!(n-|S|-k-1)! \\
&+ \sum_{S'|S'\subseteq \bar{P}_{i+1}, w(S')=y-w_i} (|S'|+k+1)!(n-|S'|-k-2)! \\
= &\ b(p_{i+1},k,y)+b(p_{i+1},k+1,y-w_i).
\end{aligned}
$$

Therefore, the statement holds. ∎

From the lemma, we can compute $V(b(p_i))$ from $V(b(p_{i+1}))$ in $\mathrm{O}(qn)$ time.

Lemma 1 *There holds*

$$
Ss(p_i)\times n! = \sum_{z=0}^{q-1}\sum_{k=0}^{i-1} \left(f(p_{i-1},k,z)\times\left(h(p_i,k,q-1-z)-h(p_i,k,\max\{q-w_i-z,0\}-1)\right)\right).
$$

Proof : Similar to the previous section, we consider a partition of a coalition S, given by

$$
S\cap P_{i-1}, \text{ and } S\cap \bar{P}_{i+1}.
$$

The condition $w(S)=y$ is equal to $w(S\cap P_{i-1})+w(S\cap \bar{P}_{i+1})=y$. Hence,

$$
\begin{aligned}
& Ss(p_i)\times n! \\
= &\sum_{S|p_i\notin S, q-w_i\le w(S)\le q-1} |S|!(n-|S|-1)! \\
= &\sum_{S_1\subseteq P_{i-1}} \left(\sum_{S_2\subseteq \bar{P}_{i+1}, q-w_i\le w(S_1)+w(S_2)\le q-1} (|S_1|+|S_2|)!(n-|S_1|-|S_2|-1)! \right) \\
= &\sum_{z=0}^{q-1}\sum_{k=0}^{i-1} \left(\sum_{S_1\subseteq P_{i-1}, w(S_1)=z, |S_1|=k} \left(\sum_{y=\max\{q-w_i-z,0\}}^{q-1-z} \sum_{S_2\subseteq \bar{P}_{i+1}, w(S_2)=y} (k+|S_2|)!(n-k-|S_2|-1)! \right) \right) \\
= &\sum_{z=0}^{q-1}\sum_{k=0}^{i-1} \left(|\ \{S_1|S_1\subseteq P_{i-1}, w(S_1)=z, |S_1|=k\}\ | \times \sum_{y=\max\{q-w_i-z,0\}}^{q-1-z} b(p_{i+1},k,y) \right) \\
= &\sum_{z=0}^{q-1}\sum_{k=0}^{i-1} \left(f(p_{i-1},k,z)\times \sum_{y=\max\{q-w_i-z,0\}}^{q-1-z} b(p_{i+1},k,y) \right) \\
= &\sum_{z=0}^{q-1}\sum_{k=0}^{i-1} f(p_{i-1},k,z)\times\left(h(p_i,k,q-1-z)-h(p_i,k,\max\{q-w_i-z,0\}-1)\right).
\end{aligned}
$$

∎

From the lemma, we can compute $Ss(p_i)$ in $\mathrm{O}(nq)$ time by using $V(f(p_{i-1})), V(b(p_{i+1}))$ and $V(h(p_{i+1}))$. Using the technique described in Section 3, we can compute $V(f(p_i))$ in the backward direction, hence the required memory space is $\mathrm{O}(qn)$.

Theorem 3 *For a weighted majority game with n players, weights $w_1,...,w_n$ of players and a quota q, Shapley-Shubik indices of all players can be computed in $O(n^2q)$ time and $O(nq)$ space.* ∎

5 Computing Deegan-Packel Indices

A winner is called a *minimal winner* if the removal of any player from the winner is a loser. If a player belongs to a minimal winner S, then the player can be considered to have power. The power can be considered to be proportional to $|S|$, hence we let the power be $1/|S|$. The definition of Deegan-Packel index $Dp(p_i)$ [3] of player p_i is the expected value of the power of p_i under the assumption that every minimal winner occurs in equal probability.

For computing Deegan-Packel index, a dynamic programming algorithm has been proposed [5]. In the same way as the algorithms for computing Banzhaf index and Shapley-Shubik index, the algorithm computes the values of a function by dynamic programming. The function has parameters: a player, a weight, a size, and the minimum weight player of the coalition. Thus, the time complexity of the algorithm is $\mathrm{O}(n^3q)$ for computing an index of a player, and $\mathrm{O}(n^4q)$ time for all players.

In this section, to compute $Dp(p_i)$, we use a new function f having only three parameters. By using the function, we can construct a dynamic programming algorithm computing $Dp(p_i)$ in $\mathrm{O}(n^2q)$ time. In the similar way to the previous sections, computing $Dp(p_i)$ for all players can be done in $\mathrm{O}(n^2q)$ time.

Assume that the indices are assigned to players in the decreasing order of their weights, i.e., $w_i \geq w_j$ for any $1 \leq i < j \leq n$. The order enables the minimal winners to be characterized in a useful way. Let $d(S)$ be the coalition obtained from S by removing the player with maximum index among players in S, and X be the set of all minimal winners. Then,

$$S \in X \Leftrightarrow w(S) \geq q \quad \text{and} \quad w(d(S)) < q.$$

Thus,

$$Dp(p_i) \times |X| = \sum_{S|S\subseteq P_n, p_i\in S, w(S)\geq q, w(d(S))<q} \frac{1}{|S|}.$$

For any $1 \leq i \leq n, 0 \leq y \leq q-1$ and $0 \leq k \leq i$, we define

$$\begin{aligned} f(p_i,k,y) &= |\ \{S|S \subseteq P_i, w(S)=y, |S|=k\}\ |, \quad \text{and} \\ b(p_i,k,y) &= \sum_{S,S\subseteq \bar{P}_i, p_i\in S, w(S)\geq y, w(d(S))<y} \frac{1}{|S|+k}. \end{aligned}$$

The definition of f is the same as that in the previous section.

Property 4 *For any $1 \leq i \leq n-1, 0 \leq k \leq i$ and $w_i \leq y \leq q-1$,*

$$b(p_i,k,y) = b(p_{i+1},k,y-w_i+w_{i+1}) + b(p_{i+1},k+1,y-w_i)$$

Proof :

$$\begin{aligned} b(p_i,k,y) &= \sum_{S|S\subseteq \bar{P}_i, p_i\in S, w(S)\geq y, w(d(S))<y} \frac{1}{|S|+k} \\ &= \sum_{S|S\subseteq \bar{P}_{i+1}, w(S)\geq y-w_i, w(d(S))<y-w_i, p_{i+1}\notin S} \frac{1}{|S|+k+1} \\ &\quad + \sum_{S|S\subseteq \bar{P}_{i+1}, w(S)\geq y-w_i, w(d(S))<y-w_i, p_{i+1}\in S} \frac{1}{|S|+k+1} \end{aligned}$$

$$
\begin{aligned}
&= \sum_{S|S\subseteq \bar{P}_{i+1}, p_{i+1}\in S, w(S)\geq y-w_i+w_{i+1}, w(d(S))<y-w_i+w_{i+1}} \frac{1}{|S|+k} \\
&\quad + \quad b(p_{i+1}, k+1, y-w_i) \\
&= \quad b(p_{i+1}, k, y-w_i+w_{i+1}) + b(p_{i+1}, k+1, y-w_i).
\end{aligned}
$$

∎

Lemma 2

$$
Dp(p_i) \times |X| = \sum_{z=0}^{q-1} \sum_{k=0}^{i-1} \left(f(p_{i-1}, k, y) \times b(p_i, k, q-z) \right)
$$

Proof : In a similar way to that described in the previous sections, consider a partition of a coalition S including p_i.

$$
S \cap P_{i-1}, S \cap \bar{P}_i
$$

Then, the condition

$$
w(S) \geq q \quad \text{and} \quad w(d(S)) < q
$$

is equal to

$$
q \leq w(S \cap P_{i-1}) + w(S \cap \bar{P}_i) \quad \text{and} \quad w(S \cap P_{i-1}) + w(d(S \cap \bar{P}_i)) < q,
$$

since $S \cap \bar{P}_i$ is always non-empty. Hence,

$$
\begin{aligned}
& Dp(p_i) \times |X| \\
= & \sum_{S|S\subseteq P_n, p_i\in S, w(S)\geq q, w(d(S))<q} \frac{1}{|S|} \\
= & \sum_{(S_1,S_2)|S_1\subseteq P_{i-1}, S_2\subseteq \bar{P}_i, p_i\in S_2, w(S_1)+w(S_2)\geq q, w(S_1)+w(d(S_2))<q} \frac{1}{|S_1|+|S_2|} \\
= & \sum_{S_1\subseteq P_{i-1}} \left(\sum_{S_2\subseteq \bar{P}_i, p_i\in S_2, q\leq w(S_1)+w(S_2), w(S_1)+w(d(S_2))<q} \frac{1}{|S_1|+|S_2|} \right). \\
= & \sum_{z=0}^{q-1} \sum_{k=0}^{i-1} \left(\sum_{S_1\subseteq P_{i-1}, w(S_1)=z, |S_1|=k} \sum_{S_2\subseteq \bar{P}_i, p_i\in S_2, q\leq z+w(S_2), z+w(d(S_2))<q} \frac{1}{|S_1|+|S_2|} \right). \\
= & \sum_{z=0}^{q-1} \sum_{k=0}^{i-1} \left(|\{S_1|S_1\subseteq P_{i-1}, w(S_1)=z, |S_1|=k\}| \times \sum_{S_2\subseteq \bar{P}_i, p_i\in S_2, q\leq z+w(S_2), z+w(d(S_2))<q} \frac{1}{|S_1|+|S_2|} \right).
\end{aligned}
$$

Substituting f and b into the last equation gives

$$
\sum_{z=0}^{q-1} \sum_{k=0}^{i-1} \left(f(p_{i-1}, k, z) \times b(p_i, k, q-z) \right).
$$

∎

If $i = n$ or $y < w_i$, then $b(p_i, k, y)$ can be computed in constant time. $V(b(p_i))$ can be computed from $V(b(p_{i+1}))$ in $\mathrm{O}(nq)$ time. Hence, we have the following theorem.

Theorem 4 *For a weighted majority game with n players, weights $w_1, ..., w_n$ of players and a quota q, Deegan-Packel indices of all players can be computed in $O(n^2 q)$ time and $O(nq)$ space.* ∎

6 Conclusion

In this paper, we proposed algorithms for computing three popular power indices of weighted games, which are Banzhaf index, Shapley-Shubik index, and Deegan-Packel index. The key idea of our algorithm is to solve the existing dynamic programming for solving the problem in the forward direction and the backward direction, and compute the indices of all players by using the computational results. For n players and the quota q, the time complexities of our algorithms for computing the indices of all players are $\mathrm{O}(nq)$ for Banzhaf index, and $\mathrm{O}(n^2q)$ for Shapley-Shubik index, respectively, while existing algorithms take $\mathrm{O}(n^2q)$ and $\mathrm{O}(n^3q)$ time, respectively. For Deegan-Packel index, we propose a new dynamic programming for computing the index of a player in $\mathrm{O}(n^q)$ time, while the existing algorithm takes $\mathrm{O}(n^3q)$ time. The computation time of indices for all players is reduced from $\mathrm{O}(n^4q)$ to $\mathrm{O}(n^2q)$.

References

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

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

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

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

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

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