A Pseudo-Polynomial Algorithm for Computing Power Indices in Graph-Restricted Weighted Voting Games

Oskar Skibski[1], **Tomasz P. Michalak**[2,3], **Yuko Sakurai**[1], **Makoto Yokoo**[1]
[1]Department of Informatics, Kyushu University, Japan
[2]Department of Computer Science, University of Oxford, UK
[3]Institute of Informatics, University of Warsaw, Poland

Abstract

Weighted voting games allow for studying the distribution of power between agents in situations of collective decision making. While the conventional version of these games assumes that any agent is always ready to cooperate with all others, recently, more involved models have been proposed, where cooperation is subject to restrictions. Following Myerson [1977], such restrictions are typically represented by a graph that expresses available communication links among agents.

In this paper, we study the time complexity of computing two well-known power indices – the Shapley-Shubik index and the Banzhaf index – in the graph-restricted weighted voting games. We show that both are $\#P$-complete and propose a dedicated dynamic-programming algorithm that runs in pseudo-polynomial time for graphs with the bounded treewidth.

1 Introduction

Weighted voting games allow for studying the distribution of power between individual agents in situations of collective decision making. Various aspects of these games have been extensively studied in the multi-agent and AI literature [Chalkiadakis *et al.*, 2011]. The research has especially focused on power indices and their computational properties [Elkind *et al.*, 2009]. In this respect, two most prominent power indices, *i.e.*, the Banzhaf power index and the Shapley-Shubik index, were shown to be $\#P$-complete [Prasad and Kelly, 1990; Deng and Papadimitriou, 1994; Chalkiadakis *et al.*, 2011].

Most works in the literature consider the conventional version of coalitional games, and weighted voting games in particular, where it is idealistically assumed that any agent is always ready to cooperate with all others. Recently, however, attention has been drawn to more involved models, with restrictions on agents' ability to cooperate [Napel *et al.*, 2012; Fernández *et al.*, 2002]. Such restrictions naturally arise in multi-agent systems due to communication costs, distance, distrust, or other factors [Meir *et al.*, 2011].

Restrictions in cooperation can be conveniently expressed in the form of a graph. A canonical model of graph-restricted coalitional games is due to Myerson [1977]. In this formalism, the graph represents communication links between the agents and cooperation between any two agents is possible if and only if there exists a direct or indirect (*i.e.*, through intermediaries) link between them. This rule naturally extends to coalitions – a coalition can cooperate if and only if all the agents involved induce a connected subgraph of a communication graph. The extensions of the Banzhaf and Shapley-Shubik indices to graph-restricted weighted voting games were proposed by Owen [1986] and Myerson [1977], respectively. Building upon Deng and Papadimitriou [1994], we show that, under graph restrictions, computing both indices is also $\#P$-complete.

Our starting point to develop an algorithm for power indices in graph-restricted weighted voting games is the algorithm for standard weighted voting games by Matsui and Matsui [2000]. In particular, the authors showed that the Banzhaf and Shapley-Shubik indices can be computed in pseudo-polynomial time using the dynamic programming approach, where agents are considered one after the other, and their weights are added to the aggregated data on (weights of) previously generated coalitions.

Unfortunately, with graph restrictions the situation becomes much more involved because we have to account for the fact that an agent must be connected to the coalition he cooperates with. In other words, we cannot add the weight of an agent to aggregated data of previously generated coalitions without checking to which of these coalitions the agent is connected to. This is not an easy task. To perform it, we could use Skibski *et al.*'s [2014] algorithm for general graph-restricted games that enumerates all connected coalition in the graph. However, as the number of connected coalitions can be exponential, this algorithm can take an exponential number of steps.

Interestingly, however, for graphs with the bounded treewidth, we propose in this paper an algorithm that computes both the Banzhaf and Shapley-Shubik indices in pseudo-polynomial time. To this end, using tree decomposition, we develop a technique that allows to gather needed information about all connected coalitions in polynomial time, even if the number of connected coalitions is exponential. Overall, the time complexity of our algorithm is $O(2^{d^2+d}dn^5q^2)$, where n is the number of agents, d is the treewidth, and q is the voting quota.

2 Preliminaries

Let N be the set of agents. In the classic form, a *coalitional game* is a tuple (N, v), where v is the *characteristic function* that assigns a payoff to every coalition $S \subseteq N$ with $v(\emptyset) = 0$.

Weighted voting games: An important class of coalitional games are *weighted voting games*. Such a game is defined by a sequence of non-negative integers $Q = [q; w_1, w_2, \ldots, w_n]$, where w_i is the number of votes (or weight) of agent i, and q is a threshold (*quota*) needed for a coalition to win. All winning coalitions are assigned the value of 1 while all losing coalitions the value of 0. Formally:

$$v_Q(S) = \begin{cases} 1 & \text{if } \sum_{i \in S} w_i \geq q \\ 0 & \text{otherwise.} \end{cases} \quad (1)$$

We will denote the set of all winning coalitions in Q by $\mathcal{W}_Q$.

As common in the literature, we assume that the quota is higher than a half of all votes, *i.e.*, $\frac{1}{2}\sum_{i \in N} w_i < q \leq \sum_{i \in N} w_i$. This condition implies that the weighted voting game is *proper*: if S is a winning coalition, then $N \setminus S$ is a losing one, and does not violate supperadditivity ($v_Q(S \cup T) \geq v_Q(S) + v_Q(T)$ for every disjoint $S, T \subseteq N$). Typically, we will refer to the coalitional game simply by its characteristic function (*e.g.*, v) or by Q to a weighted voting game.

Graph-restricted games: An implicit assumption embedded in the above definitions is that all agents can cooperate with each other. This is, however, not always the case as there may exists various physical or procedural restrictions on which coalitions are feasible and which are not. A popular model of such a situation was introduced by Myerson [1977]. Formally, Myerson's *graph-restricted game* is a triple (N, v, G), where $G = \langle N, E \rangle$ is an undirected graph and $E \in N \times N$[1] is a set of unordered pairs which denote links between agents. G describes the underlining communication structure between the agents with the interpretation that an agent can cooperate only with nodes connected to him directly (called *neighbours* and denoted $\mathcal{N}(i)$), or through a path. At the same token, coalition S can cooperate if it is connected in G, *i.e.*, any two agents of S are connected in G by a path contained in S. The set of all connected coalitions in G is denoted $\mathcal{C}_G$. If S is not connected, the agents in this coalition cooperate in the connected components of S:

$$S/G = \{S' \subseteq S \mid S' \textit{ is a maximal connected subgraph of } S\}.$$

Formally, $S' \subseteq S$ is a maximal connected subset of S in G, if it is connected and there exists no superset $S'' \supsetneq S'$ connected in G. Given the above, the *graph-restricted game*, denoted v/G, is defined as follows:

$$v/G(S) = \sum_{T \in S/G} v(T). \quad (2)$$

Note that the definition of restricted game is – though not explicitly – based on the assumption that game is superadditive. Otherwise, we would end up with a paradox: the sum of values of coalitions could be bigger than the value they can get together, *i.e.*, graph-restriction would yield better performance.

Graph-restricted weighted voting games: In this paper, we are interested in *graph-restricted weighted voting games* which combine features of weighted voting games with graph-restricted games [Napel *et al.*, 2012]. Formally, a graph-restricted weighted voting game is a pair (Q, G), where Q is a weighted voting game of agents N, and $G = \langle N, E \rangle$ is an underlining communication structure. In these games:

- the value of coalition S is the sum of the values of its connected components S/G (formula (2)); and
- the value of a connected component $T \in S/G$ equals 1 only if the sum of weights of its members exceeds the threshold: $\sum_{i \in T} w_i \geq q$ (formula (1)).

Based on the assumption that threshold exceeds half of the sum of weights of all agents, we have that in every coalition S there exists at most one connected component with non-zero value. Consequently:

$$v_Q/G(S) = \begin{cases} 1 & \text{if } \sum_{i \in T} w_i \geq q \textit{ for some } T \in S/G, \\ 0 & \text{otherwise.} \end{cases} \quad (3)$$

Consider a winning coalition from game Q which is connected in G. We will denote the set of all such coalitions, called *winning connected coalitions*, by $\mathcal{WC}$. Formally:

$$\mathcal{WC} = \mathcal{W}_Q \cap \mathcal{C}_G = \{S \subseteq N \mid \sum_{i \in S} w_i \geq q, \textit{ and } S \textit{ is connected}\}.$$

A set of winning connected coalitions with agent i is denoted $\mathcal{WC}_i$. Now, rephrasing the definition of characteristic function v_Q/G, the coalition is winning in graph-restricted weighted voting games, if it contains a winning connected coalitions, *i.e.*, $v_Q/G(S) = 1$, iff $S/G \cap \mathcal{WC} \neq \emptyset$.

Power indices: Let us now consider the issue of measuring power of agents in weighted voting games. Two well-known power indices are the Shapley-Shubik index [Shapley and Shubik, 1954] and the Banzhaf index [Penrose, 1946]. To formalize them, let us denote by $\Pi(N)$ the set of all permutations of N and by S_i^π the coalition made of agents which are in a permutation $\pi \in \Pi(N)$ before i. Now, the Shapley-Shubik index and the Banzhaf index can be defined as follows, for all $i \in N$:

$$SSI_i(v) = \frac{1}{|N|!} \sum_{\pi \in \Pi(N)} \left(v(S_i^\pi \cup \{i\}) - v(S_i^\pi)\right), \quad (4)$$

$$BI_i(v) = \frac{1}{2^{|N|-1}} \sum_{S \subseteq N \setminus \{i\}} \left(v(S \cup \{i\}) - v(S)\right), \quad (5)$$

respectively (we use a normalized version of Banzhaf index proposed by Dubey and Shapley [1979]). Intuitively, both indices remunerate agents by considering the number of times they are *pivotal*, *i.e.*, how often a given agent changes a losing coalition to a winning one. While the Banzhaf index is a simple average over all coalitions, the Shapley-Shubik index considers all possible orders in which agents could join the game one after the other. Both indices can be applied to an

[1] $A \times B$ denotes the set of unordered pairs (a, b), $a \in A$, $b \in B$.

arbitrary coalitional game (N, v) (not only to weighted voting games). In this case the Shapley-Shubik index is called the Shapley value.

Our aim in this paper is to compute the Shapley-Shubik and the Banzhaf indices in graph-restricted weighted voting games, *i.e.*, to compute $SSI_i(v_Q/G)$ and $BI_i(v_Q/G)$ for all $i \in N$. We note that the Shapley-Shubik index applied to the restricted game v_Q/G has an alternative name, *i.e.*, the Myerson value [Napel *et al.*, 2012], while the Banzhaf index applied to the restricted game v_Q/G is also called the restricted Banzhaf index [Owen, 1986]. However, for clarity, we will refer to them the Shapley-Shubik index and the Banzhaf index, respectively.

3 Closed-Form Formulas

Let us now derive more compact closed-form formulas for the indices. They show that the key challenge in computing both indices is to count the number of winning connected coalitions of a given size and with a given number of neighbours.

Recall that the basic idea behind both the Shapley-Shubik index and the Banzaf index is to remunerate agents by considering how often they are pivotal in different coalitions. In conventional weighted voting games, all such situations can be found by considering all coalition that almost reach the threshold – for a given agent i, all sets $S \not\ni i$ such that $q > \sum_{j \in S} \omega_j \geq q - \omega_i$ [Matsui and Matsui, 2000].

Graph-restricted voting games are more involved since we have to account for the communication structure between the agents. In particular, agent i will not turn loosing coalitions to winning ones if it is not connected to them. On the other hand, i may turn out to be pivotal in a coalition with an arbitrary large sum of weights if, without i, this coalition is disconnected. More precisely, agent i is pivotal in S if:

(a) $S \setminus \{i\}$ does not contain a winning connected coalition, and

(b) S contains a winning connected coalition.

Thus, agent i must produce a winning connected coalition by connecting some (non-winning) components of $S \setminus \{i\}$, joining a single component, or forming a new singleton component (if $\omega_i > q$). Thus, the set of all *pivotal winning connected coalition* of agent i is defined as follows:

$$\mathcal{PWC}_i = \{S \in \mathcal{WC}_i \mid ((S \setminus \{i\})/G) \cap \mathcal{WC} = \emptyset\}. \quad (6)$$

In the two following theorems we provide the general formula for the Shapley-Shubik index and the Banzhaf index in graph-restricted weighted voting games. We define $\mathcal{N}(S)$ as a set of neighbours of S: $\mathcal{N}(S) = \left(\bigcup_{i \in S} \mathcal{N}(i)\right) \setminus S$.

Theorem 1. *Let (Q, G) be a graph-restricted weighted voting game with the set of winning connected coalitions $\mathcal{WC}$. The Shapley-Shubik index satisfies the following formulas:*

$$SSI_i(v_Q/G) = \sum_{S \in \mathcal{PWC}_i} \gamma_1^S = \sum_{S \in \mathcal{WC}_i} \gamma_1^S - \sum_{\substack{S \in \mathcal{WC} \\ i \in \mathcal{N}(S)}} \gamma_2^S,$$

where $\gamma_1^S = \frac{(|S|-1)!(|\mathcal{N}(S)|)!}{(|\mathcal{N}(S)|+|S|)!}$ and $\gamma_2^S = \frac{(|S|)!(|\mathcal{N}(S)|-1)!}{(|\mathcal{N}(S)|+|S|)!}$.

Proof. Let us consider formula (4) for the Shapley-Shubik index. Since, by definition, (Q, G) is proper, there are no negative marginal contributions. The only case when agent $i \in N$ may be pivotal and have the positive marginal contribution of 1 is when T is a winning coalition, but $T \setminus \{i\}$ is losing. Let us denote the set of such coalitions with $\mathcal{WT}_i$. Formally: $\mathcal{WT}_i = \{T \in \mathcal{W} \mid i \in T, T \setminus \{i\} \notin \mathcal{W}\}$.

Now, we can rewrite formula (4) as follows:

$$SSI_i(v_Q/G) = \frac{1}{|N|!} \sum_{\pi \in \Pi(N): S_i^\pi \cup \{i\} \in \mathcal{WT}_i} 1. \quad (7)$$

Recall that coalitions in $\mathcal{WT}_i$ do not have to be necessarily connected, but, from the assumption on q, they have to contain exactly one winning connected coalition: $S = \mathcal{WT}_i \cap \mathcal{WC}_i$. Since $S \setminus \{i\}$ cannot belong to $\mathcal{WC}_i$ (this would imply that $T \setminus \{i\} \in \mathcal{W}$), we have that $S \in \mathcal{PWC}_i$, and:

$$SSI_i(v_Q/G) = \frac{1}{|N|!} \sum_{S \in \mathcal{PWC}_i} \sum_{\pi \in \Pi(N): S \in (S_i^\pi \cup \{i\})/G} 1. \quad (8)$$

Let us now compute the number of permutations such that S is a connected component of S_i^π. To this end, all other agents from S have to come before agent i, all neighbors of S after i, and all other agents do not play a role. Simple calculations show that there are $\frac{(|S|-1)!\mathcal{N}(S)!}{(|S|+|\mathcal{N}(S)|)!}$ such permutations which concludes the proof of the first part of the formula.

Now, consider the set difference $\mathcal{WC}_i \setminus \mathcal{PWC}_i$. This set consists of winning connected coalitions C that, without agent i, contain winning connected component $S = C \setminus \{i\} \cap \mathcal{WC}$. Note that i must be a neighbour of S. Analysing the permutations, we see from the set of permutations that contain winning connected coalitions in $S_i^\pi \cup \{i\}$ we have to exclude those that contain winning connected coalitions adjacent to i in S_i^π:

$$SSI_i(v_Q/G) = \frac{1}{|N|!} \sum_{S \in \mathcal{WC}_i} \sum_{\pi \in \Pi(N): S \in (S_i^\pi \cup \{i\})/G} 1 - \frac{1}{|N|!} \sum_{S \in \mathcal{WC}, i \in \mathcal{N}(S)} \sum_{\pi \in \Pi(N): S \in S_i^\pi/G} 1.$$

Now, it is enough to calculate permutations as above. □

Theorem 2. *Let (Q, G) be a graph-restricted weighted voting game with the set of winning connected coalitions $\mathcal{WC}$. The Banzhaf index satisfies the following formulas:*

$$BI_i(v_Q/G) = \sum_{S \in \mathcal{PWC}_i} \gamma^S = \sum_{S \in \mathcal{WC}_i} \gamma^S - \sum_{\substack{S \in \mathcal{WC} \\ i \in \mathcal{N}(S)}} \gamma^S,$$

where $\gamma^S = 1/2^{|S|+|\mathcal{N}(S)|}$.

Proof. The proof is analogous to the one of Theorem 1. For the Banzhaf index, the key formula (8) iterates not over permutations, but coalitions of N that contain S. Thus, calculations are easier – the probability that S forms a component in a random coalition equals $1/2^{|S|+|\mathcal{N}(S)|}$, as coalition must contain S, but not a neighbour of S. We omit details due to space constraints. □

Algorithm 1: General algorithm for power indices in graph-restricted weighted voting games

Input: Weighted voting game $Q = [q; w_1, \ldots, w_n]$, graph $G = \langle N, E \rangle$
Output: Shapley-Shubik index $SSI_i(v^Q/G)$, and Banzhaf index $BI_i(v^Q/G)$ of game (Q, G)

calculate table A_i;
for $s \leftarrow 0$ **to** $|N|$ **do**
  for $k \leftarrow 0$ **to** $|N| - s$ **do**
    for $w \leftarrow q$ **to** $\sum_{j \in N} \omega_j$ **do**
      $SSI_i(v^Q/G) \leftarrow SSI_i(v^Q/G) +$
        $\mathcal{WC}_i[s,k,w] \frac{(s-1)(k)!}{(s+k)!} - \mathcal{NWC}_i[s,k,w] \frac{(s)(k-1)!}{(s+k)!}$;
      $BI_i(v^Q/G) \leftarrow BI_i(v^Q/G) +$
        $\mathcal{WC}_i[s,k,w] \frac{1}{2^{s+k}} - \mathcal{NWC}_i[s,k,w] \frac{1}{2^{s+k}}$;

Before we move on to our main algorithm, we introduce the general scheme for calculating both power indices. Looking at the formulas from Theorems 1 and 2, we see that it involves enumerating two sets: $\mathcal{WC}_i$ and $\mathcal{NWC}_i = \{S \in \mathcal{WC} \mid i \in \mathcal{N}(S)\}$. Set $\mathcal{NWC}_i$ is the set of all winning connected coalitions that contain at least one neighbour of agent i, but not i itself. Now, each element of both sets contributes to the power index a value which depends on the size of the set and number of neighbours. Thus, for a given set of connected coalitions $A \subseteq \mathcal{C}$, we will associate a table $A[]$ such that $A[s,k,w]$ is the number of connected coalition of size s with k neighbours and sum of weights w:

$$A[s,k,w] = |\{S \in \mathcal{C} \mid |S| = s, |\mathcal{N}(S)| = k, \sum_{j \in S} \omega_i = w\}|.$$

Algorithm 1 presents the general scheme for calculating the indices using tables $\mathcal{WC}_i[]$ and $\mathcal{NWC}_i[]$. Our main algorithm presented in the next section will focus on calculating both tables. We end this section with the complexity result.

Proposition 1. *Calculating the Shapley-Shubik index and Banzhaf index in graph-restricted weighted voting games is $\#P$-complete.*[2]

Proof. Deng and Papadimitriou [1994] proved that calculating the Shapley-Shubik index in the standard weighted voting games is $\#P$-complete (similarly, the Banzhaf index). This problem can be reduced to our problem, as it can be modelled using a complete graph as a restriction. Finally, computing $n!SSI_i(v_Q/G)$ and $2^{N-1}BI_i(v_Q/G)$ can be considered as the number of accepting paths of nondeterministic Turing machine, so both problems are in class $\#P$, and – because of the reduction – they are $\#P$-complete. □

[2] $\#P$ is the set of counting problems associated with the decision problems in the set NP. A problem is $\#P$-complete if and only if it is in $\#P$ and every problem in $\#P$ can be reduced to it by a polynomial counting reduction [Valiant, 1979].

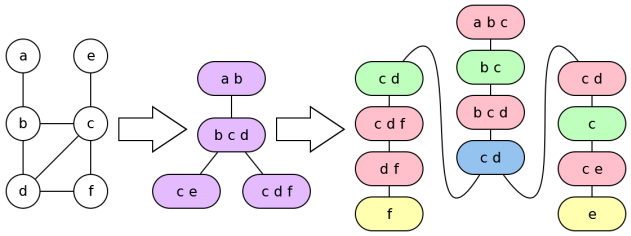

Figure 1: An example of a tree decomposition with width 2. In the middle, a standard tree decomposition. On the right, a nice tree decomposition: yellow nodes have type LEAF, red – INTRODUCE, green – FORGET, and blue – JOIN.

4 Traversing Induced Connected Subgraphs

The key challenge in implementing the formulas from the previous section is to count the number of winning connected coalitions of a given size and with a given number of neighbours. In this section, we propose a pseudo-polynomial algorithm that does this task in graphs with bounded treewidth.

We start with some basic definitions. Given a graph $G = \langle N, E \rangle$, the *tree decomposition* is a tree $\langle X, T \rangle$ such that, with each node $x \in X$, a subset of N is associated, denoted B_x, and such that the following three conditions are satisfied:

(1) for every node $i \in N$, $i \in B_x$ for some x;
(2) for every edge $(i,j) \in E$, $\{i,j\} \subseteq B_x$ for some x;
(3) for every node $i \in N$, nodes x such that $i \in B_x$ form a subtree of $\langle X, T \rangle$.

The *width of the tree decomposition* is the size of the biggest set of nodes minus one. Now, the *treewidth* of the graph is the minimum width among all possible tree decompositions. Even though finding the tree decomposition with the minimal treewidth is NP-hard, a heuristic method is able to find one with reasonably small treewidth [Bodlaender, 1996].

A rooted tree decomposition is called *nice* if every node x is of one of the four types:

- LEAF: no children and $|B_x| = \{v\}$ for some v;
- INTRODUCE: one child y with $B_x = B_y \cup \{v\}$ for some v;
- FORGET: one child y with $B_x = B_y \setminus \{v\}$ for some v;
- JOIN: two children, y_1, y_2, such that $B_x = B_{y_1} = B_{y_2}$.

See Figure 1 for an example. Any tree decomposition of width d and n nodes can be turned into a nice tree decomposition of width d and $O(dn)$ nodes in time $O(d^2 n)$ [Kloks, 1994]. Thus, without the difference for the time complexity of our algorithm, we consider only nice tree decompositions.

In our problem, we aim to find all winning connected coalitions (those containing agent i or adjacent to him). In the graph notation, connected coalitions correspond to induced connected subgraphs – graph $\langle S, M \rangle$ is an induced connected subgraph of $\langle N, E \rangle$ if $S \subseteq N$ and $M = (S \times S) \cap E$ consist of all edges between nodes from S that appear in E. We will not discuss not-induced subgraphs; hence, we will often skip this word. The goal of our algorithm is to find all (induced) connected subgraphs with given properties.

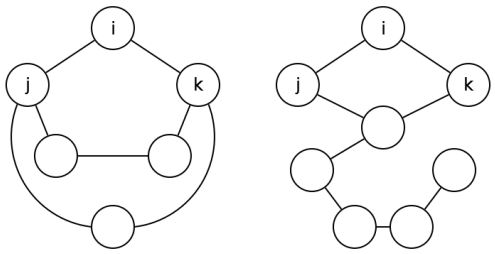

Figure 2: Example of two graphs with similar properties.

Assume that $\langle X, T\rangle$ is a nice tree decomposition such that node i is in the set B_r of the root $r \in X$. In our algorithm, we will traverse the tree decomposition bottom-up and, for every node, construct a data structure c_x based on the data structures of the children. However, accounting for all connected subgraphs based only on children is far from trivial. Consider the following example that corresponds to INTRODUCE node with i added to set $\{j, k\}$.

Example 1. *Let $G = \langle N, E\rangle$ be a graph and let $X = \{i, j, k\} \subseteq N$. We denote by $c_x(S)$ the number of induced connected subgraphs that contain exactly nodes $S \subseteq X$. Now, assume node i has only two neighbours j and k, not connected directly to each other. Based on the information about connected subgraphs of j and k (without i), we would like to calculate the number of connected subgraphs with i. We can easily see that $c_x(\{i\}) = 1$, $c_x(\{i, j\}) = c_x(\{j\})$ and $c_x(\{i, k\}) = c_x(\{k\})$. Consider $c_x(\{i, j, k\})$. If j and k are connected only by i (i.e., $c_x(\{j, k\}) = 0$), then the number of induced connected subgraphs that contain all three nodes can be easily calculated: $c_x(\{i, j, k\}) = c_x(\{j\}) \cdot c_x(\{k\})$. However, if j and k are connected with a path that does not go through i, then this number cannot be determined based on $c_x(\{j\})$, $c_x(\{k\})$ $c_x(\{j, k\})$. To see that, consider two graphs from Figure 2. While the $c_x(\{j\}) = c_x(\{k\}) = 6$ and $c_x(\{j, k\}) = 5$ for both graphs, the number $c_x(\{i, j, k\})$ equals 7 for the left graph and 6 for the right one.*

To tackle this problem, we provide the following observation. The proposition below states that, to account for all induced connected subgraphs with an additional node, it is enough to consider all induced connected subgraphs of a graph with all possible edges between nodes connected to the new node. While, in general, it is an exponential task, given a bounded treewidth, it will require the exponential number of steps, but only in respect to the treewidth.

Proposition 2. *Let $G = \langle N, E\rangle$ be a graph, and $i \in N$ be a node. Then, there exists a bijection between a set of induced connected subgraphs with $i \cup \mathcal{N}(i)$ in graph G, and induced connected subgraphs with $\mathcal{N}(i)$ in graph G' obtained by removing node i and connecting nodes from X in clique.*

Let us describe our algorithm, the pseudocode of which is presented in Algorithm 2. For clarity of presentation, we focus first on the algorithm that calculates set $\mathcal{WC}_i$. Later, we discuss a simple modification that allows for computing set $\mathcal{NWC}_i$ simultaneously. In Algorithm 2, we find first the tree decomposition $\langle X, T\rangle$ and adjust it in such a way that the root

Algorithm 2: DP-algorithm calculating table $\mathcal{WC}_i$

Input: Weighted voting game $Q = [q; w_1, \ldots, w_n]$, graph $G = \langle N, E\rangle$ with treewidth d, agent i
Output: Table $\mathcal{WC}_i[]$

find a nice tree decomposition $\langle X, T\rangle$ of G with width d and root r such that $i \in B_r$;
add Forget nodes, so that $B_r = \{i\}$ for root r;
$S \leftarrow$ *empty stack*; `// main stack with nodes from X`
$Q \leftarrow$ *empty queue*; `// auxiliary queue to build S`
$Q.add($*root r*$)$;
while Q *is not empty* **do** `// building S top-down`
  $x \leftarrow Q.pop()$;
  $Q.add($*children of* $x)$;
  $S.add(x)$;
`// we are traversing tree ⟨X,T⟩ bottom-up`
while S *is not empty* **do**
  $x \leftarrow S.pop()$; `// all children of x are processed`
  switch type of x **do**
    case LEAF*: (with $B_x = \{v\}$)*
      $c_x(\emptyset, \emptyset) = \emptyset$;
      $c_x(\{v\}, \emptyset) = \{(1, 0, \omega_v)\}$; `// tuple (s,k,w)`
    case INTRODUCE*: (with child y and $v = B_x \setminus B_y$)*
      foreach $S \subseteq B_y$, $M \subseteq 2^{B_y \times B_y}$, $N_v \subseteq B_y$ **do**
        $M_v \leftarrow (v \times N_v)$;
        if $S \cap N_v \neq \emptyset$ **then**
          foreach $(s, k, w) \in c_y(S, M)$ **do**
            $c_x(S, M \cup M_v).add((s, k+1, w))$;
          `// based on Proposition 2`
          foreach $(s, k, w) \in c_y(S, M \cup (N_v \times N_v))$ **do**
            $c_x(S \cup \{v\}, M \cup M_v).add((s+1, k, w+\omega_v))$;
        else
          $c_x(S, M \cup M_v) = c_y(S, M)$;
          if $S \neq \emptyset$ **then** $c_x(S \cup \{v\}, M \cup M_v) = \emptyset$;
          else $c_x(S \cup \{v\}, M \cup M_v) = \{(1, |M_v|, \omega_v)\}$;
    case FORGET*: (with child y and $v = B_y \setminus B_x$)*
      $M_v \leftarrow \{(v, j) \in E \mid j \in B_x\}$; `// real edges`
      foreach $S \subseteq B_x$, $M \subseteq 2^{B_x \times B_x}$ **do**
        $c_x(S, M) = c_y(S, M \cup M_v) \cup c_y(S \cup \{v\}, M \cup M_v)$;
    case JOIN*: (with children y_1, y_2)*
      foreach $S \subseteq B_x$, $M \subseteq 2^{B_x \times B_x}$ **do**
        if $S \neq \emptyset$ **then**
          foreach $(s_1, k_1, w_1) \in c_{y_1}(S, M)$ **do**
            foreach $(s_2, k_2, w_2) \in c_{y_2}(S, M)$ **do**
              $s \leftarrow s_1 + s_2 - |S|$;
              $k \leftarrow k_1 + k_2 - |\{(S \times B_x) \cap M\}|$;
              $\omega \leftarrow \omega_1 + \omega_2 - \sum_{i \in S} \omega_i$;
              $c_x(S, M).add(s, k, \omega)$;
        else $c_x(S, M) = c_{y_1}(S, M) \cup c_{y_2}(S, M)$;
return $c_x(\{i\}, \emptyset)$; `// x = root of ⟨X,T⟩ here`

node r consists of only agent i: $B_r = \{i\}$ (lines 1–2). Then, we prepare stack S with all nodes in a bottom-up order, *i.e.*, in such a way that children are processed before their parent (lines 3–8). Technically, we do it by going through the tree in a top-down order, and by adding children at the top of the parent in the stack. Then, our main loop starts.

In each step of the main loop, we calculate data structure c_x for node $x \in X$ based on the data gathered in children nodes. Let us denote by B_x^* the set of nodes that appear in a subtree rooted at x. For a given subset of (potential) edges[3] $M \subseteq (B_x \times B_x)$, consider a graph G restricted to nodes B_x^* and with the edges between B_x replaced by M. Now, for each subset of nodes $S \subseteq B_x$, in variable $c_x(S, M)$, we store (parameters of) all connected coalitions of this graph that intersect with B_x on S. Let us define the following set:

$$D_x(S, M) = \{C \in \mathcal{WC}(\langle N, E \setminus (B_x \times B_x) \cup M \rangle) \mid C \cap B_x = S\}.$$

Then, $c_x(S, M)$ is a multiset

$$c_x(S, M) = \{(|C|, |\mathcal{N}(C)|, \sum_{j \in C} \omega_j) \mid C \in D_x(S, M)\}.$$

Before we proceed, note that only the FORGET case involves real edges of graph G. This is because, until we remove a node, we have to consider all possible edges between it and other nodes in order to apply the technique from Proposition 2. When we remove the node, because of the tree decomposition structure, we know it will not occur again.

Let us now briefly describe all four cases of nodes. The LEAF case (lines 12–14) is trivial. In the INTRODUCE case (lines 15–26), in order to extend structure to another node for every pair (S, M), we consider also any $N_v \subseteq B_y$ so that we account for all possible edges that v can have. Now, if – in given scenario with $M \cup (v \times N_v)$ edges – v is connected to S, then every subgraph with S gains a new neighbour (lines 19–20). To count subgraphs that contain $S \cup \{i\}$, we use the technique from Proposition 2. On the other hand, if v is not connected to S, then coalitions with S are just copied, and $S \cup \{v\}$ exists only if S is empty.

The FORGET case (lines 27–30), as discussed above, is the only one that uses real edges of graph G. When forgetting node v, for each subset of nodes S, we join connected subgraphs that, in the child node, were assigned to S and to $S \cup \{v\}$.

In the JOIN case (lines 31–40), node x has two children. Consider two subgraphs – one from each child – that contain exactly S (S is connected according to some set of edges M). The important observation is that the intersection of both these subgraphs is also set S. In other words, subgraphs cannot intersect outside of S. This is because, from the tree decomposition definition, if some element would appear in both children's subtrees, then it must appear in their parent node. The same applies to joined neighbours – they are limited to set B_x. Thus, every pair yields a new connected subgraph with sum of values decreased by the intersection – S. However, if intersection S is an empty set, then subgraphs cannot be merged, and sets of subgraphs are simply added. Finally, variable $c_x(\{i\}, \emptyset)$ consists parameters of all connected subgraphs with i, and $c_x(\emptyset, \emptyset)$ – without i (line 41). Note, however, that we cannot easily distinguish those that contain neighbours of agent i, *i.e.*, set $\mathcal{NWC}_i$. Let us describe below a simple modification that allows for finding this set simultaneously.

To this end, we add the fourth element a to the tuple of parameters stored in $c_x(S, M)$. Now, $a = 1$ if connected subgraph contains a neighbour of i, and 0 otherwise. Updating this information is easy – whenever we add a new tuple (lines 14, 26), we set it to 0. When we join two coalitions (lines 36–39), we take a maximum of both values. Now, when forgetting a node v, if v is a neighbour of i, then, in all sets that contained v ($c_x(S \cup \{v\}, M \cup M_v)$ in line 30), we update the value of a to 1. At the end, $\mathcal{NWC}_i$ consists of all tuples from $c_x(\emptyset, \emptyset)$ with $a = 1$ (line 41). This modification does not change the complexity of the algorithm.

Time and Space Complexity: Let $n = |N|$. Crucial for the time and space complexity is the data structure used for multiset $c_x(S, M)$. Here, we assume that $c_x(S, M)$ is a three-dimension table $[0..n+1][0..n+1][0..2q]$, where q is a threshold and the dimensions corresponds to the size of the coalition, number of neighbours and sum of weights. Thus, for every possible tuple (s, k, w) we store the number of occurrences. That way, we are achieving pseudo-polynomial time and space complexity. Note that for special cases – for example, when the number of connected subgraphs is polynomial in n – different data structure may be better and lead to *real* polynomial time.

Let us consider the time complexity of our algorithm. The initialization (lines 1–9) has complexity $O(d^2 n)$ (for $n = |N|$) and produces a tree decomposition with at most $O(dn)$ nodes. Thus, the main loop has $O(dn)$ iterations. In the LEAF case, we perform $O(1)$ steps. In the JOIN case, for every subset of B_x ($O(2^d)$) and subset of possible edges between B_x ($O(2^{d^2})$), we have to multiply two c_x structures ($O(n^4 q^2)$), which overall gives time $O(2^{d^2+d} n^4 q^2)$. Analogous analysis leads to $O(2^{d^2+2d} n^2 q)$ and $O(2^{d^2+d} n^2 q)$ for the INTRODUCE and FORGET cases, respectively. Thus, each iteration have complexity $O(2^{d^2+d} n^4 q^2)$. Overall, the time complexity of our algorithm is $O(2^{d^2+d} d n^5 q^2)$ which yields pseudo-polynomial time if d is bounded.

As for the space complexity, note that in every node of tree decomposition we store 2^{d^2+d} arrays of size $O(n^2 q)$. As there are at most $O(dn)$ nodes, the space complexity is $O(2^{d^2+d} n^2 q)$.

5 Related work

One of the main motivations behind the graph-restricted weighted voting games is modelling the political alliances – a topic that raised considerable attention in the literature. The first computational analysis is due to Fernández *et al.* [2002] who proposed polynomial time algorithms for Banzhaf and Shapley-Shubik indices for trees based on generating functions. Palestini [2005] proved, that under some strong condition on the set of winning coalitions, Shapley-Shubik index can be calculated in graph-restricted weighted voting

[3] We note that we could only consider sets of edges M that include the real set of edges between $B_x \times B_x$. For the clarity of presentation we consider all possible sets.

games in the polynomial time. Recently, Napel *et al.* [2012] performed the (non-computational) monotonicity analysis of power indices graph-restricted weighted voting games.

In a related work, See *et al.* [2014] introduced compatibility weighted voting games, where all winning coalitions have to be cliques. Aziz *et al.* [2009] considered computational complexity of the power indices in the spanning connectivity game. Here, however, agents are represented by edges, not nodes.

Game-theoretic solution concepts, such as the Banzhaf index and the Shapley-Shubik index have been used as an advanced centrality measures [Gomez *et al.*, 2003; Michalak *et al.*, 2013; 2015]. Thus, our paper directly contributes to this growing body of literature (see Tarkowski et al., [2015] for an overview).

6 Conclusions

In this paper, we studied the problem of computing the Shapley-Shubik index and the Banzhaf index in graph-restricted weighted voting games. For graph with a bounded treewidth, we introduced a general technique for traversing all induced connected subgraphs in polynomial time. We used it to design a pseudo-polynomial algorithm for both indices. In our future work, we would like to study computational properties of other power indices under graph restrictions.

Acknowledgements

Oskar Skibski and Makoto Yokoo were supported by JSPS KAKENHI Grant (24220003) and Yuko Sakurai was supported by JSPS KAKENHI Grant (15K12101). Tomasz Michalak was supported by the European Research Council under Advanced Grant 291528 ("RACE") and the Polish National Science grant DEC-2013/09/D/ST6/03920.

References

[Aziz *et al.*, 2009] Haris Aziz, Oded Lachish, Mike Paterson, and Rahul Savani. Power indices in spanning connectivity games. In *Algorithmic Aspects in Information and Management*, pages 55–67. Springer, 2009.

[Bodlaender, 1996] Hans L Bodlaender. A linear-time algorithm for finding tree-decompositions of small treewidth. *SIAM J. on Computing*, 25(6):1305–1317, 1996.

[Chalkiadakis *et al.*, 2011] Georgios Chalkiadakis, Edith Elkind, and Michael Wooldridge. Computational aspects of cooperative game theory. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 5(6):1–168, 2011.

[Deng and Papadimitriou, 1994] Xiaotie Deng and Christos H Papadimitriou. On the complexity of cooperative solution concepts. *Mathematics of Operations Research*, 19(2):257–266, 1994.

[Dubey and Shapley, 1979] Pradeep Dubey and Lloyd S Shapley. Mathematical properties of the banzhaf power index. *Math. of OR*, 4(2):99–131, 1979.

[Elkind *et al.*, 2009] Edith Elkind, Leslie Ann Goldberg, Paul W Goldberg, and Michael Wooldridge. On the computational complexity of weighted voting games. *Annals of Mathematics and Artificial Intelligence*, 56(2):109–131, 2009.

[Fernández *et al.*, 2002] Julio R Fernández, Encarnacion Algaba, Jesús M Bilbao, A Jiménez, Nieves Jiménez, and Jorge J López. Generating functions for computing the myerson value. *Annals of OR*, 109(1-4):143–158, 2002.

[Gomez *et al.*, 2003] Daniel Gomez, Enrique González-Arangüena, Conrado Manuel, Guillermo Owen, Monica del Pozo, and Juan Tejada. Centrality and power in social networks: a game theoretic approach. *Mathematical Social Sciences*, 46(1):27–54, 2003.

[Kloks, 1994] Ton Kloks. *Treewidth: computations and approximations*, volume 842. Springer Science & Business Media, 1994.

[Matsui and Matsui, 2000] Tomomi Matsui and Yasuko Matsui. A survey of algorithms for calculating power indices of weighted majority games. *Journal-Operations Research Society of Japan*, 43:71–86, 2000.

[Meir *et al.*, 2011] Reshef Meir, Jeffrey S Rosenschein, and Enrico Malizia. Subsidies, stability, and restricted cooperation in coalitional games. In *IJCAI*, pages 301–306, 2011.

[Michalak *et al.*, 2013] T. P. Michalak, T. Rahwan, P. L. Szczepański, O. Skibski, R. Narayanam, M. J. Wooldridge, and N. R. Jennings. Computational analysis of connectivity games with applications to the investigation of terrorist networks. In F. Rossi, editor, *IJCAI'13*, pages 293–301. AAAI Press, 2013.

[Michalak *et al.*, 2015] T.P. Michalak, T. Rahwan, O. Skibski, and M. Wooldridge. Defeating terrorist networks with game theory. *Intelligent Systems, IEEE*, 30(1):53–61, Jan 2015.

[Myerson, 1977] Roger B Myerson. Graphs and cooperation in games. *Math. of OR*, 2(3):225–229, 1977.

[Napel *et al.*, 2012] Stefan Napel, Andreas Nohn, and José Maria Alonso-Meijide. Monotonicity of power in weighted voting games with restricted communication. *Math. Social Sciences*, 64(3):247–257, 2012.

[Owen, 1986] Guillermo Owen. Values of graph-restricted games. *SIAM J. on Algebraic Discrete Methods*, 7(2):210–220, 1986.

[Palestini, 2005] Arsen Palestini. Reformulation of some power indices in weighted voting games. *Homo Oeconomicus*, 2005.

[Penrose, 1946] Lionel S Penrose. The elementary statistics of majority voting. *Journal of the Royal Statistical Society*, pages 53–57, 1946.

[Prasad and Kelly, 1990] Kislaya Prasad and Jerry S Kelly. Np-completeness of some problems concerning voting games. *Int. J. of Game Theory*, 19(1):1–9, 1990.

[See *et al.*, 2014] Abigail See, Yoram Bachrach, and Pushmeet Kohli. The cost of principles: Analyzing power in compatibility weighted voting games. In *AAMAS*, pages 37–44, 2014.

[Shapley and Shubik, 1954] Lloyd S Shapley and Martin Shubik. A method for evaluating the distribution of power in a committee system. *Am. Pol. Sc. Rev.*, 48(03):787–792, 1954.

[Skibski *et al.*, 2014] Oskar Skibski, Tomasz P Michalak, Talal Rahwan, and Michael Wooldridge. Algorithms for the Shapley and Myerson values in graph-restricted games. In *AAMAS*, pages 197–204, 2014.

[Tarkowski *et al.*, 2015] M. Tarkowski, T. Michalak, T. Rahwan, E. Elkind, and M. Wooldridge. Game-theoretic network centrality measures—a critical survey. Technical report, CS-RR-15-02, Computer Science Department, University of Oxford, 2015.

[Valiant, 1979] Leslie G Valiant. The complexity of computing the permanent. *Theoretical Computer Science*, 8(2):189–201, 1979.