Structured Proportional Representation

Nimrod Talmon
Weizmann Institute of Science
nimrodtalmon77@gmail.com

ABSTRACT

Multi-winner voting rules aiming at proportional representation, such as those suggested by Chamberlin and Courant [9] and by Monroe [20], partition an electorate into virtual districts, such that a representative is assigned to each district; these districts are formed based on the voters' preferences. In some applications it is beneficial to require certain structural properties to be satisfied by these virtual districts. In this paper we consider situations where the voters are embedded in a network, and we require each virtual district to be connected (with respect to the network). We discuss applications of a corresponding combinatorial problem and study its computational complexity, identifying several variants and special cases which can be solved efficiently.

CCS Concepts

•**Computing methodologies** → **Multi-agent systems;** •**Theory of computation** → *Problems, reductions and completeness;*

Keywords

multiwinner elections; graph algorithms; treewidth

1. INTRODUCTION

We study a generalization of proportional representation multiwinner rules, such as those presented by Chamberlin and Courant [9] and by Monroe [20]; based on the voters' preferences, these rules select a committee of k representatives such that (1) each representative represents a subset of the voters and (2) each voter is represented by exactly one representative. Thus, in effect, the voters are partitioned into pairwise disjoint subsets, which we refer to as *virtual districts*[1], and we assign a representative (i.e., a committee member) to each such district. Ideally, each voter is represented by one of her most-desired alternatives.

In this paper we generalize such proportional representation systems by considering not only the voters' preferences, but take into account also external relations between them. That is, given an election and a network which describes relations between the voters, our goal is to partition the electorate into virtual districts, based on the voters' preferences (as specified by the election), while also requiring each virtual district to satisfy some structural properties, based on the voters' relations (as specified by the network); concretely, in this paper we require each virtual district to be connected (with respect to the external network). Indeed, there are other natural and interesting structural properties which we might require the districts to satisfy; in this paper we study connectivity, as one of the most basic properties of networks; in Section 5 we discuss other structural properties.

Our goal is to get the best of both worlds: we want (1) our chosen committee (consisting of the representatives of all the districts) to best represent our electorate (with respect to the satisfaction of the voters from their assigned representatives) and also to have (2) virtual districts which conform to structural constraints: specifically, to have each district be connected, according to the auxiliary network. A formal definition of the problem considered in this paper, called SPR, is given in Section 2.

We study the computational complexity of SPR. It turns out that efficiently achieving our goals is not possible in general, therefore we concentrate on identifying several well-motivated tractable cases. Specifically, we model the network as a graph, and consider different graph classes, such as trees and graphs with bounded treewidth.

1.1 Motivating Scenarios

Below we describe several scenarios where having connected virtual districts might be beneficial.

Political scenario. Consider a political election to be held, where a size-k committee is to be elected based on the voters' preferences, such that each voter is to be represented by a committee member. In effect, the electorate is partitioned into k virtual districts, based on the voters' representatives (and not based on geography, which is sometimes the case).

We argue that, e.g., prior relationships between voters in each part might influence the effectiveness of the chosen committee. For example, since the representatives might want to discuss various issues with the voters which they represent, voters from each virtual district might periodically meet; thus, friendship relations between them might have an influence on the usefulness of such meetings. Concretely, each voter might be more satisfied if she knows, directly or indirectly, all the voters in her virtual district, thus having each virtual district to be a connected component with respect to the voters' friendship network might be desired.

[1]These are called virtual districts as they resemble electoral districts, but are based on preferences and not on geography.

Appears in: *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2017), S. Das, E. Durfee, K. Larson, M. Winikoff (eds.), May 8–12, 2017, São Paulo, Brazil.*

Commercial scenario. Consider a factory which can manufacture and ship to consumers at most k product types in parallel (i.e., assume k production lines). The factory management might ask for the preferences of their potential customers, and choose k products which, if produced, would satisfy the highest number of people. In effect, the population of potential customers is partitioned into k shipping areas, such that each shipping area contains those potential customers which, among those k products to be produced and shipped by the factory, prefer the same product. For example, a car factory might need to select k colors for its cars, trying to maximize the number of customers which are satisfied with at least one color.

Since the factory has to ship its products to the customers, it might be desired to have each shipping area be connected, with respect to a network, modelling geographic distance between the customers (possibly with some threshold, such that there is an edge between two customers iff the distance between them is upper-bounded by a certain threshold). The reason is that shipping the same product to several customers might be easier if those customers are close to each other; further, each product might be produced in a different geographic place, thus it is cheaper to ship only one product type at a time (say, in each truck).

Multiagent scenario. Consider adding hierarchy to a multiagent system, where each agent has different preferences for the agents it would like to see above it in the hierarchy. In effect, the agents are partitioned into k teams, based on their preferences, and a leader is assigned to each team.

Since leaders might want to broadcast messages to agents in their teams, it might be desired to have good communication properties in each such team. Considering a graph with one vertex for each agent, and where there is an edge between two agents iff they can communicate efficiently (e.g., they are directly connected via some wire), it is apparent that having connected teams would boost their effectiveness, since it would lower the inter-team communication costs.

In each of the scenarios described above, a population of entities (voters, consumers, agents) is to be partitioned into parts (political districts, shipping areas, multiagent teams). When partitioning the population, it is desirable to take into account both the preferences of the entities (the political preferences, the product preferences, the hierarchical preferences), as well as relations between the entities (their friendships, their geographic closeness, their communication links). Specifically, we argue that it is desirable to have each of those virtual districts be connected, with respect to the corresponding network, to increase even further the total satisfaction of the voters, or, if you will, the social welfare.

We further discuss those scenarios, as well as others, in Section 5, where we discuss other structural properties, besides connectivity, to be required from each virtual district.

1.2 Related Work

This paper deals with the computational complexity of a generalization of proportional representation voting rules, such as those suggested by Chamberlin and Courant [9] and Monroe [20] (the difference between these rules is that the latter imposes size constraints on the virtual districts). Winner determination for these rules is NP-hard [23], but they can be solved efficiently for elections with few voters or few alternatives [4] and they can be efficiently approximated [19, 28, 29].

These rules have applications in domains such as recommendation systems [19], resource allocation [29, 27], and facility location [4, 23]. Other generalizations of proportional representation rules exist, most notably the generalization suggested by Skowron et al. [27] (also considered by Aziz et al. [2, 3]), where voters get utilities from several committee members (instead of only from their most-prefered committee member), and the generalization by Elkind and Ismaili [15] which consider a continuum between the utilitarian version and the egalitarian version of these rules.

In this paper we generalize these rules by taking into account a network, connecting the voters which participate in the election. Various scenarios of voting in presence of networks were studied. For example, Boldi et al. [6] studied voting systems which let voters specify delegates to represent them; several researchers [11, 13, 24, 25, 26] studied how social networks can influence the election winners and the ability of the population to recover ground-truths; other researchers [7, 8, 10] studied how social networks can affect manipulative actions performed by external agents.

Last, we mention the recent paper by Igarashi et al. [17] which is closely related to the current paper: On one hand, they consider an election being held on a social network, and also require some kind of connectivity. On the other hand, their problem is more constrained, and they study it from a cooperative game theoretic perspective.

2. PRELIMINARIES

For $n \in \mathbb{N}$, we denote the set $\{1, \ldots, n\}$ by $[n]$.

2.1 Elections and Voting Rules

An election $E = (A, V)$ consists of a set of alternatives $A = \{a_1, \ldots, a_m\}$ and a collection of voters $V = (v_1, \ldots, v_n)$. In this paper we concentrate on Approval elections, where a voter v is associated with a subset $A_v \subseteq A$ of the alternatives which she approves (for simplicity, the voters are females while the alternatives are males). Such a voter is said to be an A_v-voter and we denote such a situation by writing "$v : A_v$". For example, with $A = \{a_1, a_2, a_3\}$, by writing $v : \{a_2, a_3\}$ we mean that v approves a_2 and a_3, but disapproves a_1; we further say that v is an $\{a_2, a_3\}$-voter.

A multiwinner voting rule $\mathcal{R}$ is a function that, given an election $E = (A, V)$ and an integer $k \in [m]$, outputs a set $\mathcal{R}(E, k)$ of k-element subsets of A. Each subset $S \subseteq A$ of k alternatives is called a *committee* and each member of $\mathcal{R}(E, k)$ is called a *winning committee*. A single-winner rule is a multiwinner rule for $k = 1$. Under (single-winner) Approval, each alternative gets one point from each voter which approves him, and the alternatives with the highest number of points tie as co-winners.

In this paper we study a generalization of (Approval) *Chamberlin–Courant* (in Section 5 we discuss how our algorithmic results transfer to *Monroe*). Under the Chamberlin–Courant multiwinner voting rule, for a committee S, we consider all *assignment functions* $\Phi : V \to S$; that is, an assignment function Φ assigns a committee member $a \in S$ (a representative) to each voter $v \in V$. The *score* of a committee S and an assignment Φ for this committee equals the number of voters which are assigned to a committee member which they approve. The *score* of a committee S is taken to be the maximum over all its possible assignments Φ, and the highest-scoring committee wins (if several committees have the same highest score, then all of them tie as co-winners).

2.2 Parameterized Complexity

A parameterized problem, where some part of the input is declared as the *parameter*, is said to be *fixed-parameter tractable* (equivalently, is in FPT) if there is an algorithm that solves it in time $f(k) \cdot |I|^{o(1)}$, where $|I|$ is the size of the encoding of the given instance, k is the value of the parameter, and f is some computable function. There is a hardness hierarchy of parameterized problems, where it is widely-believed that a W-hard problem is not FPT. An algorithm running in time $O(n^k)$ shows containment in XP. For more details on parameterized complexity, we refer to textbooks on this subject [14, 16, 22].

2.3 The SPR problem

The Structured Proportional Representation problem (SPR in short), which is the subject of the current paper, can be described as follows. We are given an election, consisting of a set A of m alternatives and a collection V of n voters. Each voter corresponds to a vertex in a given vertex-labeled graph $G = (V, E)$, such that the label on each vertex corresponds to the vote of the voter (indeed, there is a slight clash of notation regarding V, but since vertices correspond to voters, we keep it that way). In this paper we concentrate on approval elections, where each voter specifies a subset of alternatives: a voter approves the alternatives in this subset, while she disapproves all the remaining alternatives. Analogously, each vertex v in G (corresponding to a voter v in the election) is labeled by the subset of alternatives $A_v \subseteq A$ approved by the voter v.

Given such a labeled graph, the task is to partition it into k connected components, which we usually refer to as *virtual districts*, and to assign a representative $a \in A$ to each of these virtual districts. We define the *satisfaction* of a virtual district with respect to its representative to be the number of voters in the district which approve the district's representative (since vertices correspond to voters, we use voters and vertices interchangeably). The goal is to maximize the overall satisfaction, that is, the sum (over the virtual districts) of the satisfaction. A more formal definition follows. (Indeed, while the problem is described as a maximization problem, the corresponding decision problem can be defined easily, by supplying another integer input, standing for the desired total satisfaction, and requiring the total satisfaction to be lower-bounded by it).

Definition 1. *For a voter v approving $A_v \subseteq A$ and a representative $a \in A$, we define* $\text{score}(v, a) = 1$ *iff* $a \in A_v$ *(a* satisfied *voter) and* 0 *otherwise (an* unsatisfied *voter).*

> (Approval) SPR
> **Input:** A set of alternatives A, a graph $G = (V, E)$ where each vertex $v \in V$ is labeled by a subset $A_v \subseteq A$ of alternatives, and a desired number of districts k.
> **Task:** Partition the vertices of G into k connected components, $V_1, \ldots, V_k$, and choose k corresponding alternatives, $rep(V_1), \ldots, rep(V_k)$, such as to maximize $\sum_{i \in [k]} \sum_{v \in V_i} \text{score}(v, rep(V_i))$.

Indeed, the SPR problem corresponds to (Approval) Chamberlin–Courant with the additional requirement that each district needs to be connected, with respect to an auxiliary graph (which connects the voters participating in the election).

Two variants of SPR. In the definition above, we allow repetitions in the list $rep(V_1), \ldots, rep(V_k)$ of representatives, such that, even for $i \neq j$, it might be the case that $rep(V_i) = rep(V_j)$. In effect, an alternative might serve as the representative of several districts; importantly, the number of different connected components is always k. This variant is called the *non-unique* variant and models, for example, situations where the alternatives correspond to parties, and so a certain party can win in several districts (since, e.g., it can nominate different politicians to each district).

We also consider the *unique* variant, where repetitions in the list of representatives are not allowed; this variant models, for example, situations where the alternatives correspond to politicians, and so it is not desirable to have several districts represented by a single politician.

Recalling the scenarios described at the beginning of Section 1, we mention that our commercial scenario seems to be better modeled using the non-unique variant (since we might ship the same product to different regions but we cannot ship more than k product types), while our multiagent scenario seems to be better modeled using the unique variant (since we prefer having each leader leading only one team, for which it can take full care of).

2.4 Overview of Our Results

We are interested in understanding the computational complexity of (Approval) SPR, especially when restricted to certain graph classes, both for the non-unique variant and for the unique variant. Our main results can be summarized as follows.

- Both the unique variant of SPR and the non-unique variant of SPR are NP-hard, even when each voter can approve only one alternative, the number of districts is two, the number of alternatives is three, and the graph is planar (planar graphs can model, for example, geographic relations).
- With respect to trees, there is a sharp contrast between the unique variant of SPR and the non-unique variant of SPR: the unique variant is NP-hard even on paths while the non-unique variant is polynomial-time solvable on trees.
- The non-unique variant of SPR can be efficiently solved on graphs with bounded treewidth; for constant number of districts, the unique variant of SPR can also be efficiently solved on such graphs.

2.5 Trees and Treewidth

Since our positive algorithmic results are for trees and graphs of bounded treewidth, it might be useful to briefly discuss those graph classes.

Trees are important mainly since they can naturally model hierarchical structures: as an archetypal example, consider peers in an hierarchical organization. We also mention that one might first compute a spanning tree for an arbitrary graph, and then execute our efficient tree algorithms on this spanning tree. Treewidth (see, e.g., [14]) is a well-accepted measure of similarity of graphs to trees, where trees have treewidth 1, and graphs which are similar to trees (in some specific sense) have small treewidth. Treewidth plays an important role in the design of many exact and approximation algorithms for a variety of NP-hard problems and is practically useful in the context of social networks [1].

DEFINITION 2 (TREEWIDTH). *Let $G = (V(G), E(G))$ be a graph. Let $T = (V(T), E(T))$ be a tree and $B : V(T) \to 2^{V(G)}$. The pair (T, B) is a* valid tree decomposition *of G if it holds that: (1) $\bigcup_{x \in V(T)} B(x) = V(G)$ for each $x \in V(T)$; (2) For each $\{u, v\} \in E(G)$, there exists some $x \in V(T)$ such that $u, v \in B(x)$; and (3) For each $v \in V(G)$, the set of vertices x of T for which $v \in B(x)$ induces a connected subtree of T. The* width *of a valid tree decomposition (T, B) is $\max_{x \in V(T)} |B(x)| - 1$. The* treewidth *of a graph G, denoted by $\omega(G)$, is the minimum width over all its valid tree decompositions.*

3. INITIAL RESULTS

We next prove a hardness result, demonstrating that SPR is intractable, even when each voter can approve only one alternative, the number of districts is two, there are only three alternatives, and the graph is planar. This stands in contrast to winner determination for Chamberlin–Courant, which is polynomial-time solvable for constant number of districts.

THEOREM 1. *SPR is NP-hard, even when each voter can approve only one alternative, the number of districts is two, there are only three alternatives, and the graph is planar.*

PROOF. We reduce from the following problem.

> 2-DISJOINT CONNECTED SUBGRAPHS (2-DCS)
> **Input:** A graph $G = (V, E)$ and disjoint nonempty sets of vertices, Z_1 and Z_2.
> **Question:** Are there vertex-disjoint connected subgraphs of G, $G_1 = (V_{G_1}, E_{G_1})$ and $G_2 = (V_{G_2}, E_{G_2})$, such that $Z_1 \subseteq V_{G_1}$ and $Z_2 \subseteq V_{G_2}$?

The 2-DCS problem is known to be NP-hard even on planar graphs [30]. Given an instance of 2-DCS we create an instance of SPR, as follows. We use the same input graph G, and create three alternatives: z_1, z_2, and d; the alternatives z_1 and z_2 would correspond to the nonempty sets Z_1 and Z_2, while d would act as a dummy alternative. We let each vertex $v \in Z_1$ to approve the alternative z_1; we let each vertex $v \in Z_2$ to approve the alternative z_2; and we let all other vertices $v \in V \setminus (Z_1 \cup Z_2)$ to approve d. For each vertex $v \in Z_1$, we create $n^2 - 1$ new dummy vertices, and connect them to v; we let these vertices to also approve the alternative z_1. Similarly, for each vertex $v \in Z_2$, we create $n^2 - 1$ new dummy vertices, and connect them to v; we let these vertices to also approve the alternative z_2. We set the number of districts k to two. Since, for NP-hardness, we shall use the decision version of SPR, we set the desired total satisfaction to be $n^2 \cdot (|Z_1| + |Z_2|)$. This finishes the polynomial-time reduction.

Next we prove the correctness of the reduction. Given a solution for the instance of 2-DCS, we let the voters corresponding to the vertices of G_1 (that is, V_{G_1}), together with all other dummy vertices approving z_1, to form one virtual district, V_1. Similarly, we let the voters corresponding to the vertices of G_2 (that is, V_{G_2}), together with all other dummy vertices approving z_2, to form another virtual district, V_2. Notice that the dummy vertices approving z_1 are connected to vertices in G_1, while the dummy vertices approving z_2 are connected to vertices in G_2. Then, since G_1 and G_2 are both connected, it follows that the virtual districts V_1 and V_2 are connected as well.

In SPR, we are required to assign all vertices to districts, so we shall assign the remaining vertices in $V \setminus V_{G_1} \cup V_{G_2}$: we assign them arbitrary to either V_1 or V_2 while making sure that both V_1 and V_2 remain connected (this can be done, e.g., by iteratively assigning these vertices, always making sure that connectivity is preserved; indeed, we might need to iterate several times). We set z_1 to be the representative of V_1 and z_2 to be the representative of Z_2. The resulting solution to SPR has indeed total satisfaction of $n^2 \cdot (|Z_1| + |Z_2|)$, since each voter in $Z_1 \cup Z_2$, and each corresponding dummy vertex, is satisfied by her representative.

For the other direction, let us consider a solution for the instance of SPR, which achieves a total satisfaction of at least $n^2 \cdot (|Z_1| + |Z_2|)$. First, consider a vertex v in Z_1 and one of her dummy vertices u. We claim that we can assume, without loss of generality, that each v and u are in the same district. If this is not the case, then it means that u is alone in her district, since u is connected only to v. Then, the total satisfaction cannot be $n^2 \cdot (|Z_1| + |Z_2|)$: this can be seen by considering all possible ways of representing the district containing v by either z_1, z_2, or d. Thus, we conclude that we can assume that, for each vertex v in $Z_1 \cup Z_2$, the dummy vertices of v are in the same district as v.

Next we claim that each of the vertices in $Z_1 \cup Z_2$ has to be satisfied. Otherwise, there is at least one vertex in $Z_1 \cup Z_2$ which is not satisfied, but then, her dummy vertices would be unsatisfied as well, since they approve the same alternative and they are in the same district. Since there are $n^2 \cdot (|Z_1| + |Z_2|) + |V \setminus (Z_1 \cup Z_2)| \leq n^2 \cdot (|Z_1| + |Z_2|) + n$ vertices in the graph, it follows that, in this case, the total satisfaction will be at most $n^2 \cdot (|Z_1| + |Z_2| - 1) + n < n^2 \cdot (|Z_1| + |Z_2|)$. Thus, we conclude that all vertices in $Z_1 \cup Z_2$ are satisfied.

Since the vertices in Z_1 approve z_1, while the vertices in Z_2 approve z_2, it follows that the representatives are z_1 and z_2, and then there is one connected district V_1, represented by z_1 and containing all vertices in Z_1, and another connected district V_2, represented by z_2 and containing all vertices in Z_2. Thus, we can partition the graph into two connected components, thus having a solution for 2-DCS. □

REMARK 1. *Notice that in the election constructed by the reduction presented above, each voter approves only one alternative. Thus, the above reduction shows that SPR is NP-hard even for the Plurality version of SPR (under Plurality, each voter gives one point to her most-prefered alternative, and the alternative with the highest number of points wins; that is, for each v, it holds that $|A_v| = 1$). We mention that the reduction can be modified (specifically, by adding further dummy alternatives) to show NP-hardness for t-Approval (where $|A_v| = t$) and t-Veto (where $|A \setminus A_v| = t$).*

REMARK 2. *The proof presented above also shows that the task of partitioning the graph into virtual districts is hard, even when the committee members are given. (Technically, this holds since we can enumerate all possible committees of size two.)*

To gain some further intuition concerning SPR, we next consider very special graphs, namely complete graphs. It is clear that on complete graphs, SPR is equivalent to winner determination for Chamberlin–Courant (since, roughly speaking, the graph structure does not matter). Based on the work of Procaccia et al. [23] and Betlzer et al. [4], we conclude the following.

COROLLARY 1. *On complete graphs, SPR is (1) NP-hard, (2) XP and W-hard wrt. the number of districts k, and (3) FPT wrt. to either the number n of voters or the number m of alternatives.*

Indeed, Theorem 1 proves that SPR is NP-hard even for three alternatives, thus presumably not fixed-parameter tractable for the number m of alternatives. For completeness, we mention that for the number n of voters, however, SPR is fixed-parameter tractable (for any graph): this follows by considering each of the $O(n^n)$ possible partitions of the voters into districts.

4. UNIQUE REPRESENTATIVES AND NON-UNIQUE REPRESENTATIVES

The results presented in Section 3 hold both for the unique variant and the non-unique variant of SPR; as described in Section 2, these two variants are motivated by different scenarios. In this section, we do distinguish between these two variants, and show that, for some graph classes, the choice of variant has critical consequences on the computational complexity of SPR. A *path graph* of length n is composed of vertices $\{v_1, \ldots, v_n\}$ and edges $\{\{v_i, v_{i+1}\} : i \in [n-1]\}$). The following theorem shows that the unique variant of SPR is intractable even on path graphs.

THEOREM 2. *The unique variant of SPR is NP-hard, even on path graphs, at least for 2-Approval.*

PROOF. A coloring of a graph is said to be *convex* if each color class forms a connected component in it. We provide a reduction from the following problem.

> CONVEX RECOLORING
> **Input:** A graph $G = (V, E)$ with a vertex coloring $C : V \to \mathcal{C}$ (where $\mathcal{C}$ is the set of colors) and a budget b.
> **Question:** Can the coloring of G become convex by recoloring at most b vertices?

That is, in the CONVEX RECOLORING problem, we are given a vertex-colored graph, and the task is to recolor a minimum number of vertices in order to make the coloring convex. CONVEX RECOLORING is known to be NP-hard even on path graphs [21]. Given an instance of CONVEX RECOLORING on path graphs, we create an instance for the unique variant of SPR, as follows.

We use the same graph G. We create an alternative for each color, another n dummy alternatives, and let the ith voter to approve the alternative corresponding to her (vertex's) color and the ith dummy alternative. We set the number k of districts to be equal to the number of colors (that is, $k = |\mathcal{C}|$). Since, for NP-hardness, we shall use the decision version of SPR, we set the total satisfaction to be $n - b$. This finishes the description of the reduction. which is computable in polynomial time. Notice that, since the original graph is a path graph, we have that the generated instance for SPR is a path graph as well.

Next we prove the correctness of the reduction. First, given a solution to the instance for SPR, we recolor each of the unsatisfied voters with the color of their representative. That way, we color at most b vertices (since there are at most b unsatisfied voters), and each color class forms a connected component, as needed for CONVEX RECOLORING.

For the other direction, consider a solution for the instance of CONVEX RECOLORING, such that each color class in the recolored graph forms a connected component. We declare each of these connected component to be a district in our solution for SPR. Further, we select the representative of each such district to be the representative corresponding to the color class of the corresponding connected component. Then, all the vertices, besides those that were recolored, are now satisfied. Thus, we have a total satisfaction of $n - b$. The above works if the solution for the instance of CONVEX RECOLORING used all k colors; if this is not the case, i.e., if there are $k' < k$ connected components in the solution for the instance of CONVEX RECOLORS, then we can split single vertices from some components, as we describe next. First, if we have some connected components where the total number of vertices in them is at most $k - k'$, then we split these vertices to have one vertex in each new district, and we let the unique dummy alternative approved by this vertex to represent the district. Then, if we did not reach k districts, then we take a component with more vertices than needed, and remove some of its vertices to make each of them a singleton district (as before); importantly, this can be done while preserving connectivity (for example, by iteratively removing a vertex while preserving connectivity; this would take polynomial time even when implemented naively). Notably, the total satisfaction of the electorate would be still $n - b$, as needed. □

The situation for the non-unique variant is different, since it can be efficiently solved even on trees.

THEOREM 3. *On trees, the non-unique variant of SPR is polynomial-time solvable.*

PROOF. We describe an algorithm based on applying dynamic programming twice, in a nested way. The first dynamic program is used for traversing the tree, while the second dynamic program is used for iterating over children of inner nodes. First, we arbitrarily root the tree. Then, starting from the leaves, we proceed in a bottom-up fashion, and consider different subtrees, until we reach the root.

Let T denote the whole tree, and let $\text{root}(T)$ denote its arbitrarily-chosen root. Similarly, let us denote the root of a subtree T' of T by $\text{root}(T')$. In particular, if T' is composed of only one vertex, then this vertex is denoted by $\text{root}(T')$ (that is, T' is a subtree, while $\text{root}(T')$ is a node: its root).

For a subtree T', an integer $k' \in [k]$, and an alternative $a' \in A$, let the value $\text{Rec}(T', k', a')$ be the score of an optimal way of partitioning the vertices in the subtree T' into k' parts, such that the upper-most part (that is, the part where $\text{root}(T')$ is in) is represented by the alternative a'. For ease of presentation, let us define $\text{Rec}(T', k') := \max_{a' \in A}(\text{Rec}(T', k', a'))$. The algorithm will eventually return the value of $\text{Rec}(T, k)$. Let us describe how to compute different values of $\text{Rec}(T', k', a')$ by considering several cases, differentiated by the number of children that $\text{root}(T')$ has.

Case 1: $\text{root}(T')$ is a leaf. In this case, $\text{Rec}(T', 1, a')$ is 1 if the voter corresponding to $\text{root}(T')$ is an a'-voter, and is 0 otherwise. That is:

$$\text{Rec}(T', 1, a') = \text{score}(\text{root}(T'), a').$$

Case 2: $\text{root}(T')$ has one child. In this case, we take the maximum between two possibilities, differentiated by

whether root(T') and her child are in the same district or in different districts. That is:

$$\mathrm{Rec}(T', k', a') = \max\big(\mathrm{Rec}(T' \setminus \mathrm{root}(T'), k' - 1) + \mathrm{score}(\mathrm{root}(T'), a'), \mathrm{Rec}(T' \setminus \mathrm{root}(T'), k', a') + \mathrm{score}(\mathrm{root}(T'), a')\big).$$

Case 3: root(T') has at least two children. This case is more involved, and we solve it with another dynamic program. Using our second dynamic program, we can efficiently decide how to distribute the k' districts assigned to T' between the children of root(T'). Next we describe our second dynamic program.

For ease of presentation, let us arbitrarily order the children of root(T'). For a subtree T' and an integer c', let us denote the subtree containing all vertices contained in the first c' children of root(T'), including root(T') itself, by $T'_{c'}$. For a subtree T', an integer c', an integer $k' \in [k]$, and an alternative $a' \in A$, let the value $\mathrm{Rec}'(T', c', k', a')$ be the score of an optimal way of partitioning the vertices of $T'_{c'}$ into k' parts, such that the upper-most part (that is, the part where root(T') is in) is assigned to the alternative a'. Indeed, if root(T') has c children, then $\mathrm{Rec}(T', k', a') = \mathrm{Rec}'(T', c, k', a')$. Let us describe how to compute the values of $\mathrm{Rec}'(T', c', k', a')$. If $c' = 1$, then we proceed similarly to the case where root(T') has one child (indeed, when $c' = 1$, we consider only one child of T'). Otherwise, if $c' > 1$, then we proceed as follows. Since we have k' districts to partition T' into, we shall decide how to distribute these k' districts between the different children of root(T'). The main observation we use now is that, when considering the c'th child of root(T') (whose subtree is denoted by c'-child(T')), it is enough to consider only the number of districts which are already assigned to former children of root(T'). Formally:

$$\mathrm{Rec}'(T', c', k', a') = \max_{k'' \in [k']}\big(\quad (1)$$
$$\mathrm{Rec}'(T', c' - 1, k' - k'', a') \quad (2)$$
$$+ \max(\mathrm{Rec}(c'\text{-child}(T'), k''), \quad (3)$$
$$\mathrm{Rec}(c'\text{-child}(T'), k'' + 1, a'))\big) \quad (4)$$

In the above equation, k'' stands for the number of districts assigned to c'-child(T'). For each value of k'', we compute the optimal partition of the former children of root(T') (in (2)), and consider two possibilities, differentiated by whether root(C') and root(c'-child(T')) are in different districts (in (3)) or in the same district (in (4)).

This finishes the description of our algorithm. Next we prove its correctness and provide an analysis of its running time. We begin with a correctness proof.

Intuitively, our dynamic programs check all possibilities, thus, in particular, consider all solutions. More formally, consider a solution. In induction, we assume that the algorithm is correct for trees of size at most n, and we consider trees with n vertices.

For the base, if the tree has only one vertex, then, since we consider each alternative a' when we compute $\mathrm{Rec}(T', k', a')$, it follows that for the right alternative, we will return the correct total satisfaction, which is 1.

If the root has only one child, then there are only two possibilities for the situation in the solution: either the root is in the same district as her child, or the root is in her own singleton district. We check both possibilities in Case 2.

Last, if the root has at least two children, then we employ our second dynamic program. In this case, it holds that there must be a division of the districts into the children of the root, and our second dynamic program considers each of these possible divisions. To conclude, we see that our algorithm checks all possible partitions, and takes the one with the highest total satisfaction; thus correctness follows.

For the running time, let us first compute the size of the table of the first dynamic program. Since we have a cell for each node root(T') in the tree, for each number k' of districts, and for each alternative a', it follows that the size of the table of the first dynamic program is $n \cdot k \cdot m$.

Next, we compute the time we spend in the computation of each node, for the first dynamic program. The amount of work for a leaf is $O(1)$, and for a node with only one child is $O(m)$, since we consider all possible representatives for its child. To compute the time we spend in the computation of a node with at least two children, let us first compute the size of the table of the second dynamic program. Since we have a cell for each node root(T') in the tree, for each number c' of children, for each number k' of districts, and for each alternative a', it follows that the size of the table of the second dynamic program is $n \cdot n \cdot k \cdot m$. For each node in the second dynamic program, the time we spend is $k \cdot m$, since we consider each $k'' \in [k']$ and we consider all representatives for its c'-child. Thus, naturally, the amount of work for a node in the first dynamic program is the highest for nodes with at least two children, and in this case it is $n \cdot n \cdot k \cdot m \cdot k \cdot m$.

To conclude, we have that the total running time is $nkmnnkmkm$, which equals to $(nmk)^3$. □

Intuitively, the reason for the computational complexity difference between the non-unique variant of SPR (which is shown, in Theorem 3, to be polynomial-time solvable on trees) and the unique variant of SPR (which is shown, in Theorem 2, to be NP-hard even on paths), is the following: for the non-unique variant, we do not need to remember the exact committee members that represent certain subtrees, as it is sufficient to only count the number of committee members already assigned. For the unique variant, however, this is not the case, since committee members that are already assigned to districts cannot represent any further voters.

It turns out that the efficient algorithm presented in the proof of Theorem 3 for the non-unique variant of SPR on trees can be generalized to graphs with bounded treewidth (see Section 2.5 for a brief introduction on treewidth).

The overall idea of the efficient algorithm for graphs with bounded treewidth is the following: since graphs with bounded treewidth have small separators, it is enough to consider all possible assignments of alternatives to the vertices in each separator, corresponding to bags in the computed tree decomposition; then, it is sufficient to keep track of the number of districts already assigned to vertices corresponding to bags inside the subtree rooted in the separator.

THEOREM 4. *On graphs with bounded treewidth ω, the non-unique variant of SPR is polynomial-time solvable.*

PROOF. We use the concept of a nice tree decomposition [5]: in short, a tree decomposition can be efficiently converted into a *nice* tree decomposition, with $O(n\omega)$ nodes, the same treewidth, and with the special property that there are only the following four *types* of nodes:

Leaf node: Only one vertex is associated with this node.

Introduce Node: The node has one child, and if B is the set of vertices associated with its child, then there exist some $v \notin B$ for which it holds that the set of vertices associated with the node is $B \cup \{v\}$.

Forget Node: The node has one child, and if B is the set of vertices associated with the node, then there exist some $v \notin B$ for which it holds that the set of vertices associated with its child is $B \cup \{v\}$.

Join Node: The node has two children, and if B is the set of vertices associated with the node, then each of the children is also associated with the same set B.

Given a nice tree decomposition, we next describe a dynamic program for solving SPR on it. To this end, for a node T' associated with z vertices, let us arbitrarily order those z vertices and denote them by $[v_1, \ldots, v_z]$. For a node T' (in the tree decomposition), a number k' of districts, and z alternatives, $[a_1, \ldots, a_z]$, possibly with duplicates, let us define $\text{Rec}(T', k', [a_1, \ldots, a_z])$ to be the optimal total satisfaction for the vertices in the subtree rooted at T', when partitioning them into k' districts, such that, for each $i \in [z]$, v_i is represented by the alternative a_i. Notice that the alternatives $[a_1, \ldots, a_z]$ are also ordered, to match the vertices $[v_1, \ldots, v_z]$. For ease of presentation, let us define $\text{Rec}(T', k') := \max_{[a_1,\ldots,a_z] \in A^z}(\text{Rec}(T', k', [a_1, \ldots, a_z]))$ (recall that A is the set of alternatives). The algorithm would return $\text{Rec}(T, k)$ for the root T of the tree.

To describe the dynamic program, it is enough to describe how to compute $\text{Rec}(T', k', [a_1, \ldots, a_z])$ for each of the four different types of nodes in the nice tree decomposition.

T' is a leaf node: Let T' be a leaf node, associated with the vertex v. First, if $k' = 0$ or $z \neq 1$, then there is no solution; thus, in this case, we define $\text{Rec}(T', k', [a_1, \ldots, a_z]) := -\infty$. Otherwise, if $k' = 1$ and $z = 1$, then we shall check whether v is satisfied by a_1; thus, in this case, we define $\text{Rec}(T', 1, [a_1])$ to be 1 if $a_1 \in A_v$ and 0 otherwise (recall that A_v is the set of alternatives approved by v).

T' is an introduce Node: Let T' be an introduce node, associated with the vertices $[b_1, \ldots, b_{x+1}]$ (ordered in this arbitrary order), with one child, denoted by child, which is associated with the vertices $[b_1, \ldots, b_x]$ (ordered in this arbitrary order).

First, we shall check whether it is possible to assign the alternatives $[a_1, \ldots, a_{x+1}]$ to the vertices $[b_1, \ldots, b_{x+1}]$, using at most k' districts. To this end, we shall compute the number of connected districts induced by assigning each alternative a_i to his corresponding vertex b_i. If such an assignment results in more than k' connected districts, then we define $\text{Rec}(T', k', [a_1, \ldots, a_{x+1}]) := -\infty$.

Otherwise, we compute the satisfaction of the vertices $[b_1, \ldots, b_{x+1}]$ from this assignment of the alternatives $[a_1, \ldots, a_{x+1}]$; let us denote this value, which is equal to $|\{i \in [x+1] : a_i \in A_{b_i}\}|$, by $\text{score}(T')$.

We have two possibilities to consider now. First, if b_{x+1} (the vertex which is introduced in T') is in her own district (that is, there is no other vertex b_i, for $i \in [x]$, which is adjacent to b_{x+1} and for which $a_{x+1} = a_i$), then, in this case, we define $\text{Rec}(T', k', [a_1, \ldots, a_{x+1}]) := \text{score}(b_{x+1}, a_{x+1}) + \text{Rec}(\text{child}, k' - 1, [a_1, \ldots, a_x])$, since one of the districts is "used" by b_{x+1} (recall that $\text{score}(b_{x+1}, a_{x+1})$ is 1 iff b_{x+1} approves a_{x+1}); otherwise, if b_{x+1} is not defining his own district (that is, there is another vertex b_i, for $i \in [x]$, which is adjacent to b_{x+1}, and for which $a_{x+1} = a_i$), then, in this case, we define $\text{Rec}(T', k', [a_1, \ldots, a_{x+1}])$ to be equal to $\text{score}(b_{x+1}, a_{x+1}) + \text{Rec}(\text{child}, k', [a_1, \ldots, a_x])$.

T' is a forget Node: Let T' be a forget node, associated with the vertices $[b_1, \ldots, b_x]$ (ordered in this arbitrary order), with one child, denoted by child, which is associated with the vertices $[b_1, \ldots, b_{x+1}]$ (ordered in this arbitrary order). We check all options of assigning an alternative to the vertex b_{x+1} (the vertex that was forgotten), and take the maximum over these options; that is, in this case, we define $\text{Rec}(T', k', [a_1, \ldots, a_x] := \max_{a \in A}(\text{Rec}(\text{child}, k', [a_1, \ldots, a_x, a])$.

T' is a join Node: Let T' be a join node, associated with the vertices $[b_1, \ldots, b_x]$ (ordered in this arbitrary order), with two children, denoted by child_1 and child_2, both associated with the same vertices (in the same arbitrary order). We check all options of distributing the k' districts between the two children of T'; that is, we define

$$\begin{aligned}\text{Rec}(T', k', [a_1, \ldots, a_x] := \max_{k'' \in [k']} (\\ \text{Rec}(\text{child}_1, k'', [a_1, \ldots, a_x]) + \\ \text{Rec}(\text{child}_2, k' - k'', [a_1, \ldots, a_x]) \ - \text{score}(T')).\end{aligned}$$

Recall that $\text{score}(T')$ is the total satisfaction that the vertices $[b_1, \ldots, b_x]$ get by assigning $[a_1, \ldots, a_x]$ to them; we decrement the value by $\text{score}(T')$ since it is incremented already twice, once in each child of T'.

This concludes the description of the algorithm. For correctness, it can be observed, and formally proven in induction, that we preserve correctness for each of the node types. We omit these straight-forward arguments.

For the time complexity, we shall first compute the size of the table. We have a cell for each node T' in the tree decomposition, for each number k' of districts, and for each vector $[a_1, \ldots, a_z]$, where $z \leq \omega + 1$, where each a_i is taking values from A. Thus, the size of the table is $n \cdot k \cdot m^{\omega+1}$. The amount of time spent for each cell is different for each node type: for leaves, we spend $O(1)$ time; for introduce nodes, we spend $O((\omega+1)^2)$ time, since we compute the number of districts induced by the assignment; for forget nodes, we spend $O(m)$ time, since we consider all possible representatives for the forgotten node; for join nodes, we spend $O(k)$ time, since we consider all possible distributions of the k' districts between the two children. All in all, we have time complexity of $n \cdot k \cdot m^{\omega+1} \cdot \max((\omega+1)^2, m, k)$, which is generally governed by $O(m^\omega \cdot n)$. □

A similar result as stated in Theorem 4 for the non-unique variant of SPR is, presumably, not possible for the unique variant of SPR (recall that Theorem 2 shows that the unique variant of SPR is NP-hard even on paths). However, if we consider only constant number of districts, then it turns out that the unique variant of SPR is polynomial-time solvable on graphs with bounded treewidth.

The algorithm is, in a sense, a modification of the algorithm described in the proof of Theorem 4. Specifically, for the unique variant of SPR it is not sufficient to keep track only of the *number* of districts already assigned to a subtree in the tree decomposition, since we do not allow an alternative to represent several (disconnected) districts. Thus, instead of keeping track of that number, we keep track of the exact alternatives which are used as representatives for the vertices inside the subtree rooted in each separator.

THEOREM 5. *On graphs with bounded treewidth ω, the unique variant of SPR is polynomial-time solvable for constant number of districts.*

PROOF. We use an algorithm which has a lot of similarities with the algorithm described in the proof of Theorem 4. Here, however, instead of computing the value $\mathrm{Rec}(T', k', [a_1, \ldots, a_z])$, corresponding to the maximum total satisfaction when assigning the alternatives $[a_1, \ldots, a_z]$ to the vertices of T', while using k' alternatives as the representatives for the vertices corresponding to the subtree rooted at T', we compute the following. For a node T' (in the tree decomposition), a set of alternatives $S' \subseteq A$, and a list of alternatives $[a_1, \ldots, a_z]$, we define $\mathrm{Rec}(T', S', [a_1, \ldots, a_z])$ to be the maximum total satisfaction when assigning the alternatives $[a_1, \ldots, a_z]$ to the vertices of T', while using the alternatives in S' as representatives for the vertices corresponding to the subtree rooted at T'.

Indeed, this corresponds to keeping track of the exact representatives of the subtree, and not only of the number of such representatives, which is sufficient for the non-unique variant of SPR, but is not sufficient for the unique variant of SPR.

Due to space constraints, we defer the full description of the algorithm, together with its proof of correctness and analysis of its running time, to the full version (the proof is available upon request). □

Let us end the technical part of the paper by mentioning that Theorems 3, 4, and 5 might follow by expressing SPR in Monadic Second Order Logic and applying corresponding meta-theorems [12, 18]. Anyhow, our proofs are combinatorial and thus give more insight to the problem structure and are generally more efficient.

5. OUTLOOK

We mention some directions for future research.

Requiring other structural properties. One of the main points of this paper is that imposing structure on virtual districts might be beneficial for some applications related to systems of proportional representation. That is, in certain cases where proportional representation is desirable, it might make sense to take into account underlying networks; then, one might try to require certain structural properties from the virtual districts, to hopefully gain more desirable societal outcomes.

As one of the most basic structural properties of graphs, in this paper we chose to require each virtual district to be connected. It is natural to impose other structural restrictions on the virtual districts.

As examples, we might require each virtual district to:

1. have a small diameter or a small sum of distances: this might model geographical closeness, e.g., as described in the commercial scenario in Section 1.1;
2. be highly-connected: this might be quantified, e.g., using vertex connectivity, edge expansion, or graph density: the resulting districts might be more noise-robust;
3. have a light spanning tree, according to some given edge weights: a corresponding partition would have good communication properties, since broadcasting, e.g., would be easier (recall the multiagent scenario, described in Section 1.1.

Size-constrained districts. It is natural to consider SPR on top of the Monroe rule [20] (instead of on top of the Chamberlin–Courant rule), by requiring all virtual districts to be of roughly the same size.

We mention, importantly, that our algorithmic results seem to transfer to this model as well. Specifically, we can enforce specific district sizes in our dynamic programs used in the proof of Theorem 3 and we can make sure, at least for constant number of districts, that the districts created by the algorithm presented in the proofs of Theorem 4 and 5 would be of the required sizes.

Approximation and heuristics. It is not clear whether SPR can be efficiently approximated. Even if we cannot prove approximation guarantees, studying heuristics is of interest. Below we briefly discuss two heuristic ideas.

Using spanning trees. We might get good approximation algorithms by first finding a spanning tree for the given graph, and then operating only on that spanning tree (recall that, at least for the non-unique variant, we can efficiently solve SPR for trees).

Using diffusion processes and local search. Consider the following process. We start from k arbitrary vertices, called *initiators*, such that each initiator defines its own district. Then, in each iteration, we perform a diffusion step, where each initiator selects one vertex to be added to her district while still preserving connectivity (naturally, if possible, the initiator would select a vertex with similar preferences to her own). We end the diffusion process when the partition into districts is finalized. Further, it might be worthwhile to perform local improvement steps, moving voters between districts while preserving connectivity.

Egalitarian variant. In this paper we study a *maxsum* variant of SPR, where our goal is to maximize the sum of satisfaction, or, equivalently, to minimize the *sum* of *dissatisfaction*. In certain situations, however, it might be desirable to minimize the *maximum* dissatisfaction, thus studying a *minmax* variant of SPR. Indeed, this would be the egalitarian variant of SPR (instead of the utilitarian variant, which is the subject of the current paper).

The egalitarian variant might be computationally easier, as, for example, a greedy algorithm which keeps adding voters to a district until the given upper bound on the dissatisfaction has been reached, might work in some cases (at least for trees, but possibly for other graph classes and variants as well).

Scoring-based SPR. In this paper we study SPR for Approval elections, where each over is associated with a subset of her approved alternatives. It is natural to study SPR for scoring-based voting rules, where each voter is associated with a ranking over the alternatives (thus, correspondingly, each vertex would be labeled with a permutation over the alternatives). Not surprisingly, SPR for scoring-based voting rules remains NP-hard, as can be seen by introducing further dummy alternatives to the construction described in the proof of Theorem 1. Further, there is some hope for efficient approximation algorithms, since scoring rules, such as Borda, offer smoother distinction between alternatives; as a result, problems defined on scoring rules sometimes tend to be easier to approximate than on Approval elections.

Acknowledgments

We thank the anonymous reviewers for valuable comments.

REFERENCES

[1] Aaron B. Adcock, Blair D. Sullivan, and Michael W. Mahoney. Tree-like structure in large social and information networks. In *Proceedings of ICDM'13*, pages 1–10, 2013.

[2] Haris Aziz, Markus Brill, Vincent Conitzer, Edith Elkind, Rupert Freeman, and Toby Walsh. Justified representation in approval-based committee voting. In *Proceedings of AAAI'15*, pages 20–25, 2015.

[3] Haris Aziz, Serge Gaspers, Joachim Gudmundsson, Simon Mackenzie, Nicholas Mattei, and Toby Walsh. Computational aspects of multi-winner approval voting. In *Proceedings of AAMAS'15*, pages 107–115, 2015.

[4] Nadja Betzler, Arkadii Slinko, and Johannes Uhlmann. On the computation of fully proportional representation. *Journal of Artificial Intelligence Research*, pages 475–519, 2013.

[5] Hans L. Bodlaender and Arie Koster. Combinatorial optimization on graphs of bounded treewidth. *The Computer Journal*, 51(3):255–269, 2008.

[6] Paolo Boldi, Francesco Bonchi, Carlos Castillo, and Sebastiano Vigna. Voting in social networks. In *Proceedings of CIKM'09*, pages 777–786, 2009.

[7] Robert Bredereck, Piotr Faliszewski, Rolf Niedermeier, and Nimrod Talmon. Large-scale election campaigns: Combinatorial shift bribery. In *Proceedings of AAMAS'15*, pages 67–75, 2015.

[8] Laurent Bulteau, Jiehua Chen, Piotr Faliszewski, Rolf Niedermeier, and Nimrod Talmon. Combinatorial voter control in elections. *Theoretical Computer Science*, 589:99–120, 2015.

[9] John R. Chamberlin and Paul N. Courant. Representative deliberations and representative decisions: Proportional representation and the Borda rule. *American Political Science Review*, 77(03):718–733, 1983.

[10] Jiehua Chen, Piotr Faliszewski, Rolf Niedermeier, and Nimrod Talmon. Elections with few voters: Candidate control can be easy. In *Proceedings of AAAI'15*, pages 2045–2051, 2015.

[11] Vincent Conitzer. Should social network structure be taken into account in elections? *Mathematical Social Sciences*, 64(1):100–102, 2012.

[12] Bruno Courcelle and Joost Engelfriet. *Graph structure and monadic second-order logic: a language-theoretic approach*, volume 138. Cambridge University Press, 2012.

[13] John A. Doucette, Hadi Hosseini, Alan Tsang, Kate Larson, and Robin Cohen. Voting with social networks: Truth springs from argument amongst friends. *Presented at the 2nd Workshop on Exploring Beyond the Worst Case in Computational Social Choice; Held as part of AAMAS'15*, 2015.

[14] Rodney G. Downey and Michael R. Fellows. *Fundamentals of Parameterized Complexity*. Springer, 2013.

[15] Edith Elkind and Anisse Ismaili. OWA-based extensions of the Chamberlin–Courant rule. In *Proceedings of ADT'15*, pages 486–502, 2015.

[16] Jörg Flum and Martin Grohe. *Parameterized Complexity Theory*. Elsevier, 2006.

[17] Ayumi Igarashi, Dominik Peters, and Edith Elkind. Group activity selection on social networks. In *Proceedings of AAMAS '17*, 2017. To appear.

[18] Michael Lampis. Algorithmic meta-theorems for restrictions of treewidth. *Algorithmica*, 64(1):19–37, 2012.

[19] Tyler Lu and Craig Boutilier. Budgeted social choice: From consensus to personalized decision making. In *Proceedings of IJCAI'11*, pages 280–286, 2011.

[20] Burt L. Monroe. Fully proportional representation. *American Political Science Review*, 89(04):925–940, 1995.

[21] Shlomo Moran and Sagi Snir. Convex recolorings of strings and trees: Definitions, hardness results and algorithms. *Journal of Computer and System Sciences*, 74(5):850–869, 2008.

[22] Rolf Niedermeier. *Invitation to Fixed-Parameter Algorithms*. Oxford University Press, 2006.

[23] Ariel D. Procaccia, Jeffrey S. Rosenschein, and Aviv Zohar. On the complexity of achieving proportional representation. *Social Choice and Welfare*, 30(3):353–362, 2008.

[24] Ariel D. Procaccia, Nisarg Shah, and Eric Sodomka. Ranked voting on social networks. In *Proceedings of IJCAI'15*, pages 2040–2046, 2015.

[25] Amirali Salehi-Abari and Craig Boutilier. Empathetic social choice on social networks. In *Proceedings of AAMAS'14*, pages 693–700, 2014.

[26] Sigal Sina, Noam Hazon, Avinatan Hassidim, and Sarit Kraus. Adapting the social network to affect elections. In *Proceedings of AAMAS'15*, pages 705–713, 2015.

[27] Piotr Skowron, Piotr Faliszewski, and Jerome Lang. Finding a collective set of items: From proportional multirepresentation to group recommendation. In *Proceedings of AAAI'15*, pages 2131–2137, 2015.

[28] Piotr Skowron, Piotr Faliszewski, and Arkadii Slinko. Achieving fully proportional representation is easy in practice. In *Proceedings of AAMAS'13*, pages 399–406, 2013.

[29] Piotr Skowron, Piotr Faliszewski, and Arkadii Slinko. Achieving fully proportional representation: Approximability results. *Artificial Intelligence*, 222:67–103, 2015.

[30] Pim van't Hof, Daniël Paulusma, and Gerhard J. Woeginger. Partitioning graphs into connected parts. *Theoretical computer science*, 410(47-49):4834–4843, 2009.