Achieving Fully Proportional Representation by Clustering Voters

Piotr Faliszewski
AGH University
Krakow, Poland
faliszew@agh.edu.pl

Arkadii Slinko
University of Auckland
Auckland, New Zealand
a.slinko@auckland.ac.nz

Kolja Stahl
TU Berlin, Germany
kstahl@mailbox.tu-berlin.de

Nimrod Talmon*
Weizmann Institute of Science
Rehovot, Israel
nimrodtalmon77@gmail.com

ABSTRACT

Both the Chamberlin–Courant and Monroe rules are voting rules solving the problem of so-called fully proportional representation: they select committees whose members represent the voters so that voters' satisfaction with their assigned representatives is maximized. These rules suffer from a common disadvantage, being that it is computationally intractable to compute the winning committee exactly. As both of these rules, explicitly or implicitly, partition voters, they can be seen as clustering the voters so that the voters in each group share the same representative. This suggests studying approximation algorithms for these voting rules by means of cluster analysis, which is the subject of this paper. We develop several algorithms based on clustering the voters and analyze their performance experimentally.

General Terms

Algorithms, Experimentation

Keywords

multiwinner elections, clustering, voting

1. INTRODUCTION

We study the problem of winner determination under the Chamberlin–Courant [4] and Monroe [13] multiwinner rules. These remarkable rules achieve two goals simultaneously: they select a representative committee, thus solving the problem of proportional representation of voters in an indirect democracy, and also assign a representative to each voter, securing the accountability of elected representatives to the electorate. These two rules are among the very few known rules that achieve both these goals[1] but, unfortunately, computing the winners for both Chamberlin–Courant and Monroe rules is NP-hard [2, 9, 15]. A number of algorithms for this problem has been developed. Some of them are exact, computing winners in polynomial time in restricted domains [2, 19, 22], some are fixed-parameter tractable [2] and hence computing winners efficiently when certain parameters are small, and some are fast approximation algorithms that work on the universal domain [2, 9, 18, 14, 16]. Here we focus on the approximation approach and show that by using randomization and some ideas from cluster analysis, we can design algorithms that perform noticeably better than some of the algorithms known from the literature. We evaluate our algorithms experimentally.

The Chamberlin–Courant and Monroe rules operate as follows. We are given a set C of candidates, a set V of voters, and a number k, which is the size of the committee to be elected. Each voter ranks the candidates from the most to the least desirable one, and the goal is to pick a size-k committee of candidates that, in some sense, represents the voters in the best possible way. A convenient way of thinking about the setting is that we must elect a parliament of predetermined size k to act on behalf of the voters.

To measure the quality of the assignment, there is a universal satisfaction function γ, such that for each number i, $\gamma(i)$ quantifies the satisfaction of a voter from being represented by a candidate that this voter ranks as its ith best.[2] The Monroe and Chamberlin–Courant rules match each voter to her[3] representative so that: (a) There are at most k candidates that represent the voters (these are the winners of the election; that is, the parliament members), and (b) the sum of the satisfactions of the voters from their representatives is as high as possible. Under the Monroe rule, we additionally require that every selected candidate represents the same number of voters (plus or minus one, due to divisibility issues). Under Chamberlin-Courant the 'one man, one vote' principle is solved differently: in the elected

*Main work was done while affiliated with TU Berlin.

[1]A common practice of splitting the electorate into electoral districts and using a single-winner voting rule in each district can solve the second problem but not the first. Single transferable vote, STV, can be seen as another example of a rule that achieves proportional representation but not accountability.

[2]The most typical satisfaction function, used in the original definitions of Monroe and Chamberlin–Courant, is the Borda score, which for m alternatives is $\gamma_{\mathrm{Borda}}(i) = m - i$.

[3]For clarity, we refer to the voters as females and to the candidates as males.

Appears in: *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2016), J. Thangarajah, K. Tuyls, C. Jonker, S. Marsella (eds.), May 9–13, 2016, Singapore.*

assembly a representative is given a weight proportional to the number of voters he represents.

The Monroe and Chamberlin–Courant rules have a number of applications beyond selecting representative bodies in indirect democracies (such as parliaments, university senates, etc.), that include various business planning tasks and public decision making problems [6, 9, 10, 17]. For example, one can imagine a company that needs to pick k products that it would offer to its customers (e.g., because it is too expensive to manufacture more than k different types of products). The company could perform market analysis to find out how customers rank possible products, and then use the Chamberlin–Courant rule to compute which k of them would ensure the customers' highest satisfaction (under the assumption that each customer buys only one product, the one that he or she likes best).

Due to the NP-hardness of computing the winners under the Monroe and the Chamberlin–Courant rules, it is usually difficult to use these rules directly. Thus, using approximation algorithms in many circumstances is inevitable. There are two main ways of doing so:

Direct approach. In this case, we use an approximation algorithm for either the Monroe or Chamberlin–Courant rule as a replacement for the rule itself. This is approach is suitable for business-related settings and recommendation systems. In these settings the input data is, typically, already distorted and the additional distortion introduced by using an approximation algorithm is inconsequential. On the other hand, this approach is more controversial in the case of political elections. If one were to use an approximation algorithm as a voting rule, one would have to, for example, argue that it has desirable axiomatic properties (interestingly, Elkind et al. [6] have shown that this is the case for one of the approximation algorithms for the Monroe rule; earlier, Caragiannis et al [3] have shown an approximation algorithm for Dodgson's rule that has better axiomatic properties than the rule itself).

Indirect approach. This approach is suitable in the political elections setting. The election organizers announce that the election is to be held using, say, Monroe's rule with a given satisfaction function. Afterwards, they collect the votes, and make the raw election data available to the society. In a given amount of time, everyone is allowed to submit an election outcome (that is, an assignment of the representatives to the voters). The electoral committee also makes its own calculation. The assignment that leads to the highest total satisfaction is implemented. This approach can be taken in high-stake elections, e.g., in parliamentary elections, where one can expect the interested parties to invest significant effort into obtaining the best—the highest scoring—assignment (naturally, one should expect different parties to try to obtain the highest number of seats for themselves, but since at least two parties would be competing, the result should be fair).

Whatever the approach, it is important to have efficient approximation algorithms for our rules. Luckily, in both cases it is perfectly natural for the algorithms to be randomized. Randomization is noncontroversial in the direct approach due to the fact that, typically, the input data is distorted anyway (e.g, is based on the market analysis, which itself is noisy), and in the indirect approach it does not matter how one obtains the committee as long as its total satisfaction is good.

Our contribution. Currently, the best approximation algorithms for the Monroe and Chamberlin–Courant rules (in terms of the guaranteed approximation ratio) are due to Skowron et al. [18]. Another, very practical, approximation algorithm (for the Chamberlin–Courant rule only) is due to Lu and Boutilier [9]. In this paper we provide heuristic means of improving the outcomes of these algorithms and we show, through appropriate experiments, that our techniques indeed lead to substantial improvements.

Our main idea is to treat the problem of computing the winning committee as a clustering problem. Consider a situation where we are given a specific committee of size k. For both the Monroe and the Chamberlin–Courant rules, it is possible to compute the optimal assignment of the voters to the candidates in this committee efficiently [2]. This assignment partitions the voters into natural clusters (each cluster contains the voters represented by the same candidate).

On the other hand, given a partition, the optimal way to select a committee of representatives would be to assign to voters of each part the Borda winner of the subelection conducted in that part. We can now put both considerations together in the following algorithm. Starting from a certain initial partition and initial assignment, we can reassign representatives optimally, then calculate the Borda winners in each part of the new partition and form the committee and the assignment for the next iteration. Each step increases the total satisfaction of the voters and we repeat this process until we reach a fixed-point, where the representative of each cluster is its local winner. It is easy to see that this procedure is an incarnation of the standard iterative clustering algorithm [5, 8] (in a certain way, our idea also resembles cluster sampling; see, e.g., the textbook of Thompson [20]).

It is quite intuitive that the quality of the committee output by the above procedure depends strongly on the quality of the initial committee, and there are several ways in which to choose it. One can, for example, select the candidates at random, or use the output of an approximation algorithm, e.g., that of Lu and Boutilier [9] or Skowron et al. [18]. Neither of these approaches appears to be perfect, though, since both quickly lead to finding local optima that the algorithm cannot escape (in the former case, this is due to starting from a random committee that, possibly, cannot be improved well in our iterative fashion; in the latter case, because the starting committee is *too good*). Yet, we find that combining both these strategies gives very good results.

We evaluate our clustering algorithm on a number of elections that we obtain using the Polya-Eggenberger urn model [1]. It turns out that the results of our best algorithm, on the average, are noticeably better than those obtained using approximation algorithms known from the literature.

2. PRELIMINARIES

An election E is a pair (C, V), where $C = \{c_1, \ldots, c_m\}$ is a set of candidates and $V = (v_1, \ldots, v_n)$ is the collection of voters. Each voter v_i has a total order $\succ_{v_i}$ over C, referred to as her preference order. We write $\mathrm{rank}_v(c)$ to denote the position of candidate c in v's preference order (e.g., if v ranks c on the top position, then $\mathrm{rank}_v(c) = 1$).

A multiwinner voting rule $\mathcal{R}$ is a function that, given an election $E = (C, V)$ and an integer k, $k \leq |C|$, returns a set $\mathcal{R}(E, k)$ of size-k winning committees; each committee in the set ties as a winner of the election. Below we describe the Monroe [13] and Chamberlin–Couarant rules [4].

Assignment functions. Consider an election $E = (C, V)$ with $V = (v_1, \dots, v_n)$, and let k be the size of the committee to be elected. A k-CC-assignment function for this election is a function $\Phi\colon V \to C$ such that $|\Phi(V)| = |\{\Phi(v_i) \mid v_i \in V\}| \leq k$. Intuitively, we say that the candidate $\Phi(v_i)$ is the representative of voter v_i, and we say that candidate c represents voters from the set $\Phi^{-1}(c) = \{v_i \in V \mid \Phi(v_i) = c\}$. A k-Monroe-assignment function is a k-CC-assignment function that additionally satisfies the following condition: For each candidate c that represents a nonempty set of voters, it holds that $\lfloor \frac{n}{k} \rfloor \leq |\Phi^{-1}(c)| \leq \lceil \frac{n}{k} \rceil$.

Satisfaction function. Consider an election $E = (C, V)$ with the candidate set $C = \{c_1, \dots, c_m\}$. We say that the function $\gamma\colon \{1, \dots, m\} \to \mathbb{N}$ is a satisfaction function if it is nonincreasing. Intuitively, $\gamma(i)$ is the measure of satisfaction that a voter gets from being represented by the candidate that she ranks as her i'th best. The classical versions of the Chamberlin-Courant and Monroe rules use the Borda satisfaction function $\gamma_{\mathrm{Borda}}(i) = m - i$.

Monroe and Chamberlin–Courant rules. Consider an election $E = (C, V)$, the size of the committee k, and a satisfaction function γ. Let Φ be a k-CC-assignment function for E. We define the total satisfaction of the voters with Φ to be:

$$\Gamma(\Phi) = \textstyle\sum_{v \in V} \gamma(\mathrm{rank}_v(\Phi(v))).$$

The Chamberlin–Courant rule for γ, denoted γ-CC, outputs all size-k committees W for which there exists a k-CC-assignment function Φ such that: (a) $\Gamma(\Phi)$ has maximum value, and (b) $\Phi(V) \subseteq W$.[4] The Monroe rule for γ, denoted γ-Monroe, is defined in the same way, but with the exception that we consider k-Monroe-assignment functions only.

It is well-known that computing even a single winning committee under either Monroe or Chamberlin–Courant rules is NP-hard [9, 15] and hard in the parameterized sense [2]. Nonetheless, given a set of k candidates, it is possible to efficiently compute an optimal k-CC-assignment function and an optimal k-Monroe-assignment function that uses these candidates as representatives. For the case of γ-CC, this is obvious (by simply picking, for each voter, the candidate from the given set of k candidates which she prefer the best) and for Monroe there is a network-flow-based algorithm which is due to Betzler et al. [2]; later, we will refer to this algorithm as the BSU-algorithm.

We use variants of these algorithms that also work for partial committees, that is, for sets of *up to* k candidates. In the case of γ-CC rules, this is trivial (for each $k' \leq k$, every k'-CC-assignment function is also a k-CC-assignment function). For the case of γ-Monroe rules, the situation is a bit more complicated because each candidate can represent at most $\lceil \frac{n}{k} \rceil$ voters. So, when we consider partial committees for γ-Monroe, we also consider partial k-Monroe-assignment functions. A partial k-Monroe-assignment function is a regular k-Monroe-assignment functions that can assign the *null* candidate $\perp$ to some voters. There is no constraint on the number of voters to whom the null candidate is assigned, and we assume that for each voter v we have $\gamma(\mathrm{rank}_v(\perp)) = 0$.

Approximation algorithms. Let γ be a satisfaction function and let $\mathcal{R}$ be either the Monroe or Chamberlin–Courant rule for γ. We consider an election $E = (C, V)$ and committee size k. For $\beta \in [0, 1]$, we say that a k-$\mathcal{R}$-assignment function Φ is a β-approximation of $\mathcal{R}(E, k)$, if it holds that $\Gamma(\Phi) \geq \beta \cdot \mathrm{OPT}$, where OPT is the highest possible total voter satisfaction under $\mathcal{R}$ in the election E. We also speak of committees as β-approximations: We say that a committee W is β-approximate for a given election, if the optimal assignment function computed for this committee is β-approximate. A β-approximation algorithm for $\mathcal{R}$ is an algorithm that always outputs a β-approximate committee.

There are several approximation algorithms for Monroe and Chamberlin–Courant rules [9, 16, 14, 18]. We will discuss the two most relevant ones to our work in Section 3.2, where we describe the initialization step of our heuristics.

3. ALGORITHMS

In this section we describe our algorithms. We first describe two of their most important components, the clustering heuristic and the initialization heuristics (which, in fact, can be used on its own, without the clustering step), and then, based on these, we suggest six algorithms.

3.1 Clustering Heuristic

Let $\mathcal{R}$ be either Monroe or CC. We are given an election $E = (C, V)$, a committee size k, and a satisfaction function γ. The goal is to compute a β-approximate solution for $\mathcal{R}$, for as high a value of β as possible.

Our algorithms are inspired by the heuristic algorithms that are commonly employed for k-means clustering [8] and proceed according to the following steps:

1. Obtain an initial committee W_1 (e.g., by random choice, but we will describe more involved scenarios later). Then, using the BSU-algorithm, compute the initial assignment function Φ_1 that uses the candidates from W_1.

2. Keep generating assignment functions Φ_i, for $i = 2, 3, \dots$, using the following steps, until you generate a function Φ_i that is equal to Φ_{i-1}:

   (a) For each candidate $c \in W_{i-1}$, compute the set of voters $V_c = \Phi_{i-1}^{-1}(c)$.

   (b) For each V_c computed in the previous step, find a candidate c' such that $\sum_{v \in V_c} \gamma(\mathrm{rank}_v(c'))$ is maximal.

   (c) Let W_i be the set of candidates c' computed in the previous step. If $|W_i| \leq k$, add to it $k - |W_i|$ randomly chosen candidates.

   (d) Using the BSU-algorithm, compute the assignment function Φ_i for committee W_i.

3. When the procedure stops for some value $i = t$, output the committee W_t and the assignment function Φ_t.

To get some feeling for how the procedure works, let us consider an example.

[4]This last condition, if a bit weird at first sight, takes into account the possibility that while the committee is required to contain k candidates, there can be fewer than k candidates that represent any voters.

Example 1. *Consider an election with the set of candidates $\{a, b, c\}$ and the following voters: $v_1 \colon a \succ b \succ c$, $v_2 \colon c \succ b \succ a$, $v_3 \colon c \succ b \succ a$. We use a satisfaction function γ such that $\gamma(1) = 4$, $\gamma(2) = 3$, and $\gamma(3) = 0$. We set the committee size to be $k = 2$. The optimal solution is $\{a, c\}$ and irrespective of which committee our clustering heuristic starts with, after a single iteration it outputs the optimal solution. For example, if it starts with $W_1 = \{b, c\}$, then we have Φ_1 such that $\Phi^{-1}(b) = \{v_1\}$ and $\Phi^{-1}(c) = \{v_2, v_3\}$. The algorithm computes $W_2 = \{a, c\}$ and terminates.*

Let us now discuss some details of the procedure. First, we note that it always terminates. This is so, because in every iteration within Step 2, if we obtain $\Phi_i \neq \Phi_{i-1}$, then it must be the case that $\Gamma(\Phi_i) > \Gamma(\Phi_{i-1})$.[5] In fact, if the satisfaction function is polynomially bounded, then the clustering procedure terminates in polynomial time (in essence, because there are only polynomially many different scores a committee may have, and the score of a committee improves in every step).

Proposition 1. *If the satisfaction function used by $\mathcal{R}$ is polynomially bounded, then the clustering procedure runs in polynomial time.*

Naturally, the clustering heuristic never outputs a solution that is worse than the initial assignment. Also, as we will also see in the experimental evaluation of our heuristics, the choice of the initial committee W_1 is very important and needs to balance two factors. On the one hand, the committee W_1 should be of a relatively high quality, so that the algorithm starts near a very good solution. On the other hand, it should be sufficiently random, to avoid getting stuck in local maxima.

3.2 Initialization Heuristics

Our algorithms for computing the initial assignment are based on the greedy approximation algorithms of Lu and Boutilier [9] and Skowron et al. [18] (specifically, we merge the ideas of Skowron et al.'s Algorithms C and GM, which generalizes the algorithm of Lu and Boutilier). We first present the general skeleton of our initialization algorithm and then describe how to instantiate it to obtain the particular algorithms we are interested in (the algorithms of Lu and Boutilier and Skowron et al. are special cases of these).

We are given an election $E = (C, V)$, a committee size k, and a satisfaction function γ. We also have a parameter d that specifies how many partial committees we keep after each iteration (the idea of keeping several committees is due to Skowron et al. [18]). Our algorithm proceeds as follows:

1. We form the initial set D_0 of partial committees that contains only one partial committee, the empty set.
2. We execute k iterations, in each one we extend each partial committee in D by one candidate. Each iteration is either a *greedy iteration* or a *randomized iteration*, depending on a given criterion (to be specified separately for each of the algorithms which we use). Let i be the number of the iteration to be executed:

   Greedy iteration. For each partial committee W from D_{i-1}, we form all partial committees that result from extending W by a single candidate (not yet in W). We insert all the partial committees that we obtain into set D'_i. For each partial committee in D'_i we compute an optimal assignment Φ (using the BSU-algorithm) and compute its voter satisfaction $\Gamma(\Phi)$. We form the set D_i by picking up to d partial committees from D'_i that have the highest voter satisfactions.

   Randomized iteration. We start by setting D_i to be empty. Then we repeat the following procedure d times: We choose one of the d committees from D_{i-1} uniformly at random and extend it by one of the candidates not in the committee, who is also chosen uniformly at random. We put the resulting partial committee into D_i.
3. Finally, D_k contains up to d committees, each with k candidates. We output the one with the highest voter satisfaction (as computed by the BSU-algorithm).

Based on this general skeleton, we form the following algorithms for the initialization step (we set the parameter d to either be 1—for the simplest settings—or to be 5, since the results of Skowron et al. [18, Fig. 13] suggest that this is a sufficiently large value to obtain the benefit of keeping multiple committees, and, at the same time, is small enough to not affect the running time too much):

G1. This algorithm uses greedy iterations only and $d = 1$. This is exactly the algorithm of Lu and Boutlier [9] for the Chamberlin–Courant family of rules and the GM algorithm of Skowron et al. [18] for the case of the Monroe family of rules. The algorithm guarantees that the committee it outputs is $(1 - \frac{1}{e})$-approximate. (For the case of γ_{Borda}-Monroe, the approximation ratio is guaranteed to be $1 - \frac{k-1}{2(m-1)} - \frac{H_k}{k}$, where m is the number of candidates and H_k is the k'th harmonic number [18]).

G5. This is the same algorithm as G1, except that it uses parameter $d = 5$. Naturally, it provides the same approximation guarantees as G1.

D1. This algorithm combines greedy iterations and randomized ones in the following way. We are given two additional parameters, p and e, such that $0 < p < 1$ and $e \in \mathbb{R}_+$, and at the i'th iteration we make a randomized step with probability $p^{i \cdot e}$ and a greedy step with probability $1 - p^{i \cdot e}$. (The name of the algorithms stands for *decaying probability of a random iteration.*) We set the parameter $d = 1$.

D5. The same as D1, but with $d = 5$.

R(1,5). This algorithm combines the two above algorithms. We run D1 five times to obtain five committees. Then we run D5 five times to obtain another five committees. Out of these ten committees, we output the one with the highest voter satisfaction.

While algorithm R(1,5) may appear somewhat ad hoc, the underlying idea is quite clear. We run algorithms D1 and D5 to explore the space of committees that should be *relatively good*, due to the greedy steps, but are not *too good*, due to the randomized steps. We use both D1 and D5 to obtain some more varied committees. Naturally, in practical situations, one should likely tweak the algorithm in various ways. Perhaps use only D5 and not D1, perhaps use more or fewer iterations, etc. Our point here is not necessarily to show the most effective initialization algorithm, but rather to show

[5]Technically, for this to hold we need to assume that when computing the candidates c', we break ties in favor of the representative c.

the power of combining greedy and randomized steps (indeed, if one were to use our algorithms to compute the result of a political election, as in the indirect approach from the introduction, one would likely try as many various combinations of the parameters of the algorithms as time would permit).

3.3 Putting the Algorithms Together

Altogether, in our experimental section we consider six algorithms. The first three are, simply, the initialization algorithms G1, G5, and R(1,5) that we have described above. The next three, algorithms G1C, G5C, and R(1,5)C, first use the respective initialization algorithm to find a set of committees (we let the algorithms output all the committees they construct, without picking the best one), then we apply the clustering heuristic to each of them, and finally output the committee with the highest voter satisfaction. The algorithms with the clustering step are bound to be better than those without it. It is interesting—and we evaluate it in the next section—how often and to what extent the clustering step improves the solution.

For algorithms R(1,5) and R(1,5)C, we use the following parameters. We set $p = 0.75$ and $e = 1$ (e.g., in the first iteration, D1 has 0.75 probability of executing a randomized step, in the second iteration ≈ 0.56, in the third iteration ≈ 0.42, and so on).

4. EXPERIMENTAL ANALYSIS

We now present our experimental results. We have three main goals. First, we would like to evaluate how well our algorithms perform (that is, how close are the solutions they produce to optimal outcomes). Second, we would like to evaluate how effective is the randomization in algorithms R(1,5) and R(1,5)C, as compared to the other, fully deterministic, algorithms. Third, we would like to evaluate how effective the clustering step is.

4.1 Experimental Setup

We have run our algorithms in a number of experiments. For each experiment we have generated 500 instances of elections (we used the Polya-Eggenberger urn model; see below), each with $m = 100$ candidates and $n = 100$ voters. For each experiment we used three satisfaction functions: γ_{Borda}, γ_{convex}, and $\gamma_{\mathrm{concave}}$.

Function γ_{Borda} is the Borda satisfaction function, i.e., $\gamma_{\mathrm{Borda}}(i) = m - i$, for $i \in [m]$. Functions γ_{convex} and $\gamma_{\mathrm{concave}}$ are certain convex and concave functions (respectively) and are depicted in Figure 1. Formally, they are defined as:

$$\gamma_{\mathrm{concave}}(i) = m\left(\frac{-m+i+1}{m-1}\right)^7 + m \text{ , } \gamma_{\mathrm{convex}}(i) = m\left(\frac{m-i-1}{m-1}\right)^7 .$$

In most cases we have considered relatively small committees ($k = 3$ and $k = 5$), but we have also considered several much larger committees ($k = 27$ and $k = 47$). For each experiment, we run our six algorithms, and additionally we computed the optimal solution using an ILP solver (see the works of Lu and Boutilier [9] and Skowron et al. [18] for ILP formulations of the winner determination problems).[6]

[6]This is why we chose elections with 100 candidates and 100 voters: They are small enough to compute optimal solutions using ILP, but large enough to have some nontrivial structure.

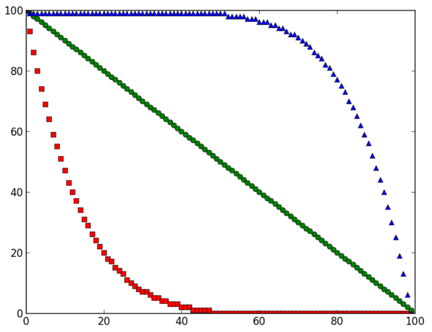

Figure 1: Our satisfaction functions: γ_{convex} (red rectangles; bottom), γ_{Borda} (green circles; center), and $\gamma_{\mathrm{concave}}$ (blue triangles; top).

We generated the elections using the Polya–Eggenberger urn model (see the works of Skowron et al. [18], Berg and Lepelley [1], McCabe-Dansted and Slinko [12], Walsh [21], and Erdelyi et al. [7] for some other examples where this model was used). This model operates as follows: For a candidate set C, we create an urn which initially contains one copy of each of the $|C|!$ possible preference orders over C. To generate a vote, we draw a preference order randomly from the urn and include it in the election. Then we return this order back to the urn, together with $\alpha \cdot |C|!$ additional copies, where α is a parameter of contagion (for $\alpha = 0$ we would get the impartial culture model, where votes are drawn uniformly at random).

The sheer number of experiments that we have run is too large for presenting all the results which we have obtained. Thus, in the discussion below, we present only some representative results (we show results for $k = 3$, $\alpha \in \{0.05, 0.25\}$, for both Monroe and CC, for all our satisfaction functions; we also show three other interesting experiments).

4.2 Discussion of the Results

We present the results of our experiments in Figures 2–16. For each of the plots we indicate the voting rule (Chamberlin–Courant or Monroe), the satisfaction function ($\gamma_{\mathrm{concave}}$, γ_{Borda}, γ_{convex}), the size of the committee (3, 5, 27, 47), and the value of the parameter α for the urn model (0.05 or 0.25). The plots show the distribution of the voter satisfaction achieved by each of our algorithms (averaged over 500 elections generated in each given experiment), scaled so that the optimal satisfaction is 1. The red line (in the middle of each rectangle) indicates the median quality of the obtained solutions. The blue rectangle covers solutions from the 25th to 75th percentile (i.e., it shows the range of the satisfaction values of solutions worse than the best quarter of the obtained solutions, but better than the worst quarter of the solutions). The top and bottom black lines indicate 1.5 times the interquartile range, and smaller blue crosses below and above them indicate outliers.

In each of our plots, each column stands for one of our six algorithms. The number next to G1C (G5C, R(1,5)C) is the number of instances for which G1C (G5C, R(1,5)C) performed better than G1 (G5, R(1,5)). This number shows how frequently the clustering step provides an improvement.

How well do our algorithms perform? All our algorithms achieve very good approximation ratios, though there are noticeable differences in their quality. Typically, G1 performs worst of all (but still, the median result tends to be at least 0.95 of the voter satisfaction in the optimal solution) and R(1,5)C tends to perform best.

How effective are the randomization steps? It is quite evident that introducing the randomization steps has very positive impact on the quality of the generated solutions. If we compare the results for R(1,5) and R(1,5)C to the results for G5 and G5C, respectively, then the randomized algorithms never perform noticeably worse, and often achieve slightly better results (but see the next point).

How effective is the clustering step? It turns out that in most scenarios applying the clustering step improves the solution by, at least, some amount, and in some scenarios it improves it greatly. This is most evident when we compare the results of algorithms R(1,5) and R(1,5)C for the Monroe family of rules. In this case, applying the clustering step often improves (quite significantly, judging by the improvement of the median satisfaction of the voters) the solution in many more than half of the instances. On the other hand, if we compare the results of G1 and G1C, or of G5 and G5C, then the clustering step does help, but on fewer instances and to a lesser extent. This confirms that generating an appropriate initial committee is very important.

Contrasting the figures corresponding to $\alpha = 0.05$ to the figures corresponding to $\alpha = 0.25$, we see that with the increase of the coefficient of contagion (i.e., the homogeneity of the population) the clustering step is becoming noticeably more useful. The reason for this is not exactly clear but, intuitively, with the increase of contagion the clusters in the data become more pronounced, hence the clustering step becomes more useful.

5. SUMMARY

We have considered the problem of computing approximate solutions for the Monroe and Chamberlin–Courant rules via algorithms combining the existing greedy approximation algorithms, randomized steps, and clustering. We have shown that putting these ideas together leads to noticeably more effective algorithms than previously known from the literature. In particular, our results have shown that interleaving greedy steps with randomized steps leads to committees that are very good starting points for the clustering step. Our algorithms can quite easily be modified to satisfy soft constraints such as "try to find as good a committee as possible with as many members of a given party as possible."

For future work, it would be interesting to perform similar experiments as ours on real election data (e.g., from PrefLib [11]). Second, it would be interesting to find a principled way of setting the parameters of our procedures (we relied on ad hoc settings). Third, we are interested in theoretical explanations of why interleaving greedy and random step gives better outcomes than greedy steps alone.

Acknowledgments

We are grateful to the reviewers for their comments. Piotr Faliszewski was supported by NCN grant DEC-2012/06/M/ST1/00358. Arkadii Slinko was supported by the Royal Society of NZ Marsden Fund UOA-254. Kolja Stahl was supported by DFG project PAWS (NI 369/10). Nimrod Talmon was supported by the DFG Research Training Group MDS (GRK 1408).

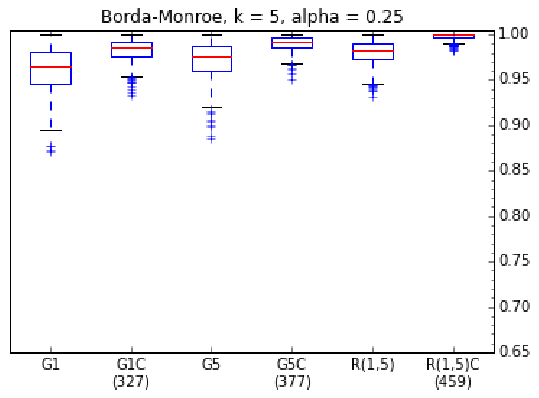

Figure 2: Performance of our algorithms for the Monroe voting rule, committee size = 5, Borda scores, and the urn model with $\alpha = 0.25$.

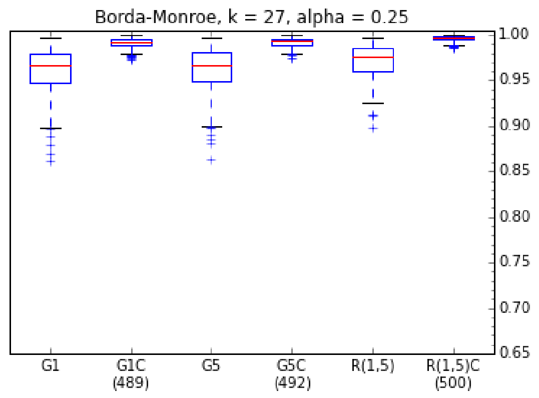

Figure 3: Performance of our algorithms for the Monroe voting rule, committee size = 27, Borda scores, and the urn model with $\alpha = 0.25$.

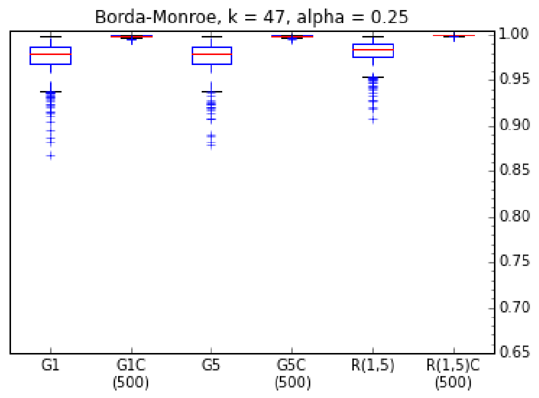

Figure 4: Performance of our algorithms for the Monroe voting rule, committee size = 47, Borda scores, and the urn model with $\alpha = 0.25$.

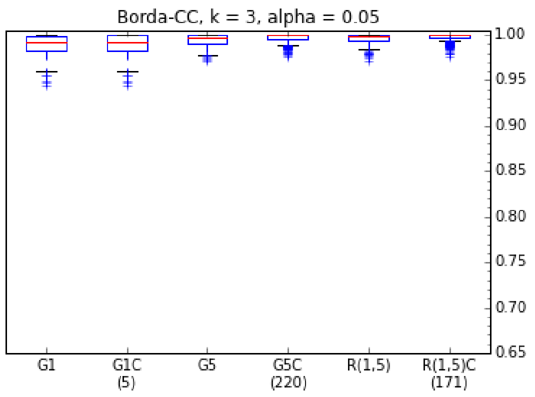

Figure 5: Performance of our algorithms for the Chamberlin–Courant voting rule, committee size = 3, Borda scores, and the urn model with $\alpha = 0.05$.

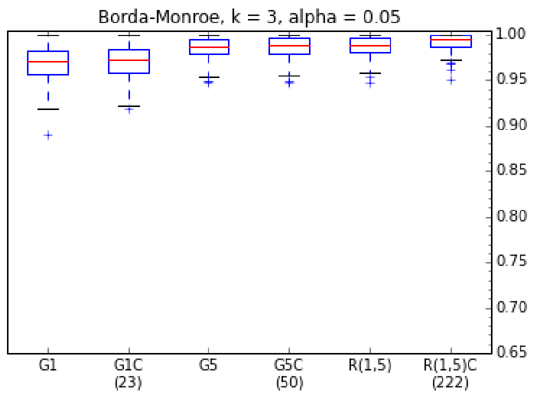

Figure 8: Performance of our algorithms for the Monroe voting rule, committee size = 3, Borda scores, and the urn model with $\alpha = 0.05$.

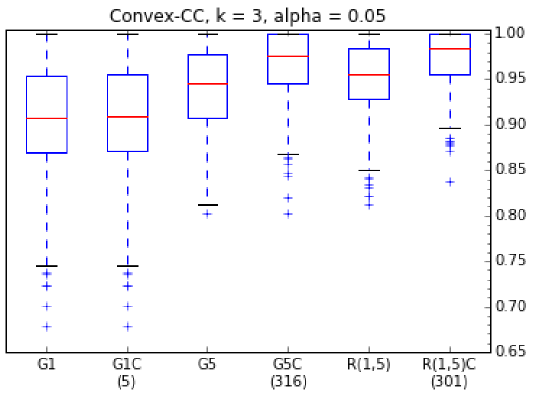

Figure 6: Performance of our algorithms for the Chamberlin–Courant voting rule, committee size = 3, Convex scores, and the urn model with $\alpha = 0.05$.

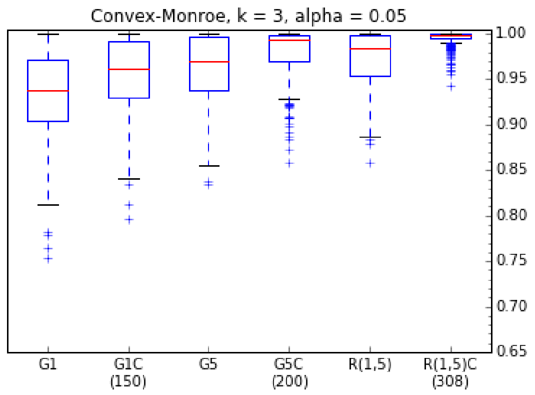

Figure 9: Performance of our algorithms for the Monroe voting rule, committee size = 3, Convex scores, and the urn model with $\alpha = 0.05$.

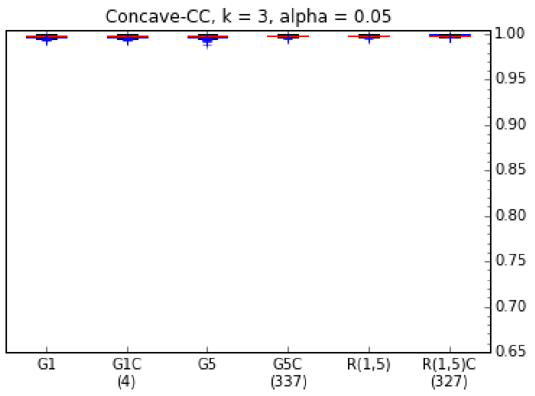

Figure 7: Performance of our algorithms for the Chamberlin–Courant voting rule, committee size = 3, Concave scores, and the urn model with $\alpha = 0.05$.

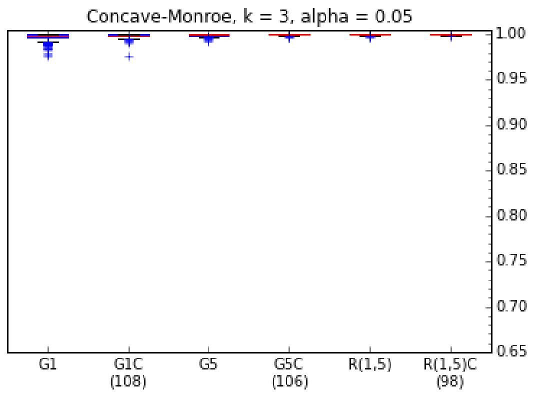

Figure 10: Performance of our algorithms for the Monroe voting rule, committee size = 3, Concave scores, and the urn model with $\alpha = 0.05$.

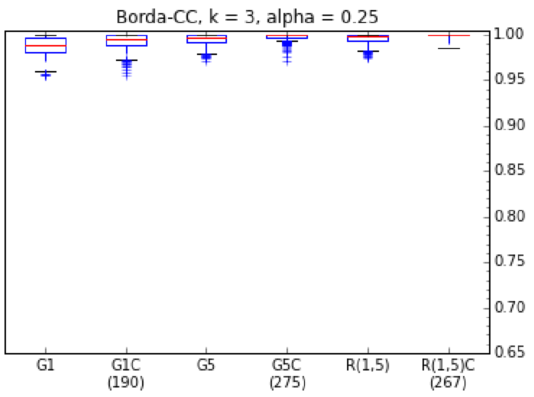

Figure 11: Performance of our algorithms for the Chamberlin–Courant voting rule, committee size = 3, Borda scores, and the urn model with $\alpha = 0.25$.

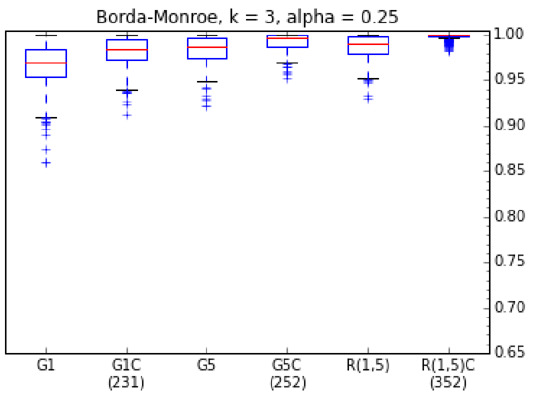

Figure 14: Performance of our algorithms for the Monroe voting rule, committee size = 3, Borda scores, and the urn model with $\alpha = 0.25$.

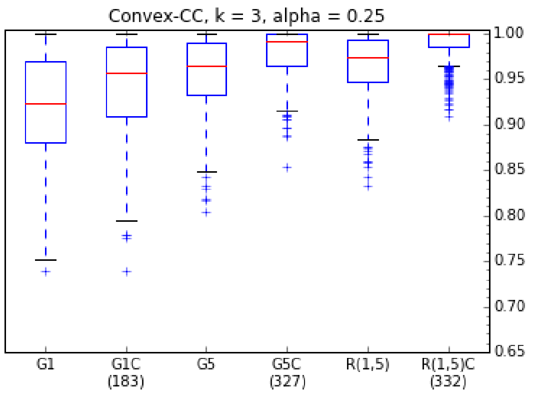

Figure 12: Performance of our algorithms for the Chamberlin–Courant voting rule, committee size = 3, Convex scores, and the urn model with $\alpha = 0.25$.

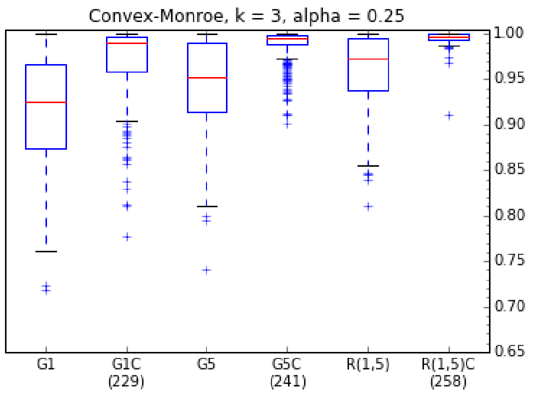

Figure 15: Performance of our algorithms for the Monroe voting rule, committee size = 3, Convex scores, and the urn model with $\alpha = 0.25$.

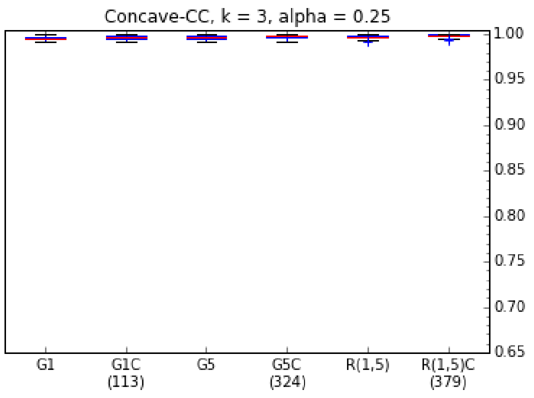

Figure 13: Performance of our algorithms for the Chamberlin–Courant voting rule, committee size = 3, Concave scores, and the urn model with $\alpha = 0.25$.

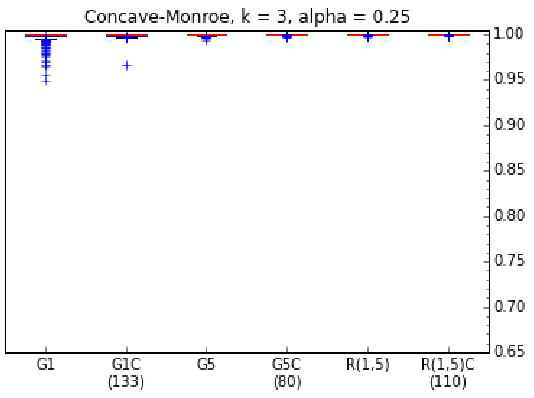

Figure 16: Performance of our algorithms for the Monroe voting rule, committee size = 3, Concave scores, and the urn model with $\alpha = 0.25$.

REFERENCES

[1] S. Berg and D. Lepelley. On Probability Models in Voting Theory. *Statistica Neerlandica*, 48(2):133–146, 1994.

[2] N. Betzler, A. Slinko, and J. Uhlmann. On the computation of fully proportional representation. *Journal of Artificial Intelligence Research*, 47:475–519, 2013.

[3] I. Caragiannis, C. Kaklamanis, N. Karanikolas, and A. Procaccia. Socially desirable approximations for Dodgson's voting rule. In *Proceedings of the 11th ACM Conference on Electronic Commerce*, pages 253–262. ACM Press, June 2010.

[4] B. Chamberlin and P. Courant. Representative deliberations and representative decisions: Proportional representation and the Borda rule. *American Political Science Review*, 77(3):718–733, 1983.

[5] R. O. Duda, P. E. Hart, et al. *Pattern classification and scene analysis*, volume 3. Wiley New York, 1973.

[6] E. Elkind, P. Faliszewski, P. Skowron, and A. Slinko. Properties of multiwinner voting rules. In *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems*, pages 53–60, May 2014.

[7] G. Erdélyi, M. R. Fellows, J. Rothe, and L. Schend. Control complexity in Bucklin and fallback voting: An experimental analysis. *J. Comput. Syst. Sci.*, 81(4):661–670, 2015.

[8] J. A. Hartigan. *Clustering algorithms.* John Wiley & Sons, Inc., 1975.

[9] T. Lu and C. Boutilier. Budgeted social choice: From consensus to personalized decision making. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, pages 280–286, 2011.

[10] T. Lu and C. Boutilier. Value-directed compression of large-scale assignment problems. In *Proceedings of the 29th AAAI Conference on Artificial Intelligecne*, pages 1182–1190, 2015.

[11] N. Mattei and T. Walsh. Preflib: A library for preferences. In *Proceedings of the 3nd International Conference on Algorithmic Decision Theory*, pages 259–270, 2013.

[12] J. C. McCabe-Dansted and A. Slinko. Exploratory analysis of similarities between social choice rules. *Group Decision and Negotiation*, 15(1):77–107, 2006.

[13] B. Monroe. Fully proportional representation. *American Political Science Review*, 89(4):925–940, 1995.

[14] J. Oren and B. Lucier. Online (budgeted) social choice. In *Proceedings of the 28th AAAI Conference on Artificial Intelligecne*, pages 1456–1462, July 2014.

[15] A. Procaccia, J. Rosenschein, and A. Zohar. On the complexity of achieving proportional representation. *Social Choice and Welfare*, 30(3):353–362, 2008.

[16] P. Skowron and P. Faliszewski. Fully proportional representation with approval ballots: Approximating the MaxCover problem with bounded frequencies in FPT time. In *Proceedings of the 29th AAAI Conference on Artificial Intelligecne*, pages 2124–2130, 2015.

[17] P. Skowron, P. Faliszewski, and J. Lang. Finding a collective set of items: From proportional multirepresentation to group recommendation. In *Proceedings of the 29th AAAI Conference on Artificial Intelligecne*, 2015.

[18] P. Skowron, P. Faliszewski, and A. Slinko. Achieving fully proportional representation: Approximability result. *Artificial Intelligence*, 222:67–103, 2015.

[19] P. Skowron, L. Yu, P. Faliszewski, and E. Elkind. The complexity of fully proportional representation for single-crossing electorates. *Theoretical Computer Science*, 569:43–57, 2015.

[20] S. Thompson. *Sampling.* Wiley, 2012.

[21] T. Walsh. Where are the hard manipulation problems? *Journal of Artificial Intelligence Research*, pages 1–29, 2011.

[22] L. Yu, H. Chan, and E. Elkind. Multiwinner elections under preferences that are single-peaked on a tree. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence*, pages 425–431, 2013.