The Thirty-Second AAAI Conference
on Artificial Intelligence (AAAI-18)

Effective Heuristics for Committee Scoring Rules

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

Martin Lackner
TU Wien
Vienna, Austria
lackner@dbai.tuwien.ac.at

Dominik Peters
University of Oxford
Oxford, UK
dominik.peters@cs.ox.ac.uk

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

Abstract

Committee scoring rules form an important class of multiwinner voting rules. As computing winning committees under such rules is generally intractable, in this paper we investigate efficient heuristics for this task. We design two novel heuristics for computing approximate results of multiwinner elections under arbitrary committee scoring rules; notably, one of these heuristics uses concepts from cooperative game theory. We then provide an experimental evaluation of our heuristics (and two others, known from the literature): we compare the scores of the committees output by our algorithms to the scores of the optimal committees, and also use the two-dimensional Euclidean domain to compare the visual representations of the outputs of our algorithms.

Introduction

The goal of a multiwinner election is to choose a fixed-size subset of items (a committee of candidates) based on the preferences of a group of agents (the voters). Multiwinner elections are a natural model for various tasks, ranging from shortlisting (Barberà and Coelho 2008), through numerous business applications (Lu and Boutilier 2011; 2015; Skowron, Faliszewski, and Lang 2016; Faliszewski et al. 2016a), to tasks involving proportional representation, such as parliamentary elections (Aziz et al. 2017; Brill, Laslier, and Skowron 2017).

While different applications call for different multiwinner voting rules with specific properties, the class of *committee scoring rules*—multiwinner analogues of single-winner scoring rules (introduced by Elkind et al. 2017b)—appears to be rich enough to contain suitable rules for essentially all the settings considered so far in the literature: Faliszewski et al. (2016a; 2016b) map the internal structure of that class and show how to design specific rules for particular applications.

Briefly put, under a committee scoring rule each voter assigns a score to each committee (based on the positions of the committee members in that voter's preference order), and a *winning committee* is one with the maximum total score, computed as the sum of individual voters' scores. Unfortunately, for almost all committee scoring rules it is NP-hard to find a winning committee (Procaccia, Rosenschein, and Zohar 2008; Lu and Boutilier 2011; Skowron, Faliszewski, and Lang 2016; Aziz et al. 2015; Faliszewski et al. 2016b; 2017a) and, thus, one has to resort to exact algorithms with super-polynomial running times (e.g., parameterized algorithms, Betzler, Slinko, and Uhlmann 2013) or to polynomial-time algorithms and heuristics that produce approximate results. While approximately optimal committees may not be acceptable in political scenarios, they may still be useful in business-related settings (see the discussion provided by Faliszewski et al. 2016c).

The goal of this paper is to experimentally evaluate several heuristic algorithms for computing committee scoring rules. As we identify certain flaws with two main heuristics that appear in the literature (the classic greedy algorithm and simulated annealing), we develop two novel heuristics. One of these is based on viewing a given election as a cooperative game and using game-theoretic solution concepts.

Motivation. There are two main heuristics presented in the literature for computing approximate winning committees under various committee scoring rules: the classic greedy algorithm (Lu and Boutilier 2011; Faliszewski et al. 2016a) and simulated annealing (Faliszewski et al. 2017a).[1] The greedy algorithm starts with an empty committee and then adds candidates one-by-one, greedily maximizing the committee score at each step, until k candidates are selected (where k is the desired committee size). Simulated annealing starts by selecting a random initial committee and then proceeds by randomly replacing committee members, keeping a new committee either if it has a higher score than the current one, or with a probability decaying with the algorithm's progress.

Unfortunately, both of these heuristics are somewhat problematic. The main issue with simulated annealing is that it is inherently randomized. While it performs very well in practice (as confirmed by our work), it would be difficult to use it in settings where candidates are competing agents with a preference for being selected. For example, consider a funding agency that has to choose k research proposals (the candidates) to be funded, based on the opinions of a group of experts (the voters). If the agency used a randomized algo-

[1] These references regard committee scoring rules. The greedy algorithm was already analyzed by Nemhauser et al. (1978) and simulated annealing was proposed by Kirkpatrick et al. (1983).

rithm, then researchers whose proposals were rejected could challenge the fairness of the process (e.g., by questioning if the algorithm's random choices were truly random). We note, however, that simulated annealing is perfectly acceptable for other tasks, such as, e.g., selecting a number of items for an Internet store to put on its homepage.

Thus, the agency would prefer a deterministic algorithm, such as the greedy one. However, the quality of its output is typically notably worse than that of simulated annealing (again, as confirmed by our work) and it has a built-in, structural bias against certain candidates (which is very visible in our experiments). To see why this is the case, note that in a given election the greedy heuristic always chooses the same candidate in the first iteration, irrespective of the desired committee size. For example, when computing an approximate Chamberlin–Courant (1983) winning committee, the greedy heuristic always starts by selecting a candidate c with the highest Borda score (see the Preliminaries for formal definitions), even though in later iterations it may choose candidates that (jointly) dominate c.

Our Contribution. There is a need for deterministic algorithms that perform as well as simulated annealing while avoiding the biases of the greedy algorithm. We address this need as follows:

1. We propose two new heuristic algorithms for (a large subclass of) committee scoring rules. The first one is based on the idea of representing multiwinner elections as cooperative games and using appropriate game-theoretic solution concepts to guide the process of selecting members of the winning committee. We hope that this new conceptual perspective can be usefully exploited in other contexts.

   The second heuristic can be seen as the inverse of the greedy algorithm: It starts with the set of all candidates and removes the least useful candidate at each step, until k candidates remain. A similar idea was used in the context of computing proportional rankings (Skowron et al. 2017).

2. We evaluate all our heuristics experimentally on synthetic data, generated using the impartial culture model and the two-dimensional Euclidean model. As our test case, we use the t-Borda family of committee scoring rules (Faliszewski et al. 2017a), which includes the Chamberlin–Courant rule and the k-Borda rule. We find that the greedy algorithm performs relatively poorly, but simulated annealing shows very good performance in all our settings. Our two new heuristics avoid most of the issues of these algorithms while producing high-quality committees.

As a further contribution, we confirm (for the case of the t-Borda family of rules) that committees computed using simulated annealing have visual representations that are very similar to those for optimal committees. Faliszewski et al. (2017a) already used simulated annealing for computing such representations, and our results are in agreement with their observations (they only provided intuitive justifications).

Preliminaries

An *election* is a pair $E = (C, V)$, where $C = \{c_1, \dots, c_m\}$ is a set of *candidates* and $V = (v_1, \dots, v_n)$ is a collection of *voters*. Each voter v is associated with a *preference order* $\succ_v$, i.e., with a ranking of the candidates in C (from best to worst). A *multiwinner voting rule* $\mathcal{R}$ is a function that, given an election $E = (C, V)$ and a positive integer k, $k \leq |C|$, returns a family of size-k *committees* (i.e., size-k subsets of candidates) that tie as winners.[2] Before we discuss committee scoring rules, we first introduce single-winner scoring rules. For a positive integer t, we denote the set $\{1, \dots, t\}$ by $[t]$.

Single-Winner Scoring Rules. Consider an election with m candidates. For a candidate c and a voter v, we write $\mathrm{pos}_v(c)$ to denote the *position* of c in v's preference order (the top candidate has position 1, the next one has position 2, and so on). A *single-winner scoring function* γ_m for m candidates, $\gamma_m \colon [m] \to \mathbb{R}$, is a non-increasing function that associates each possible position in a vote with a score. For example, $\beta_m(i) = m - i$ is the Borda scoring function and $\alpha_t(i)$, the function that is equal to 1 if $i \leq t$ and to 0 otherwise, is the t-Approval scoring function. Given a family $\gamma = (\gamma_m)_{m \in \mathbb{N}}$ of scoring functions (with one function for each number of candidates), we define the *γ-score* of candidate c in election $E = (C, V)$ as $\gamma\text{-score}_E(c) = \sum_{v \in V} \gamma_{|C|}(\mathrm{pos}_v(c))$. A *single-winner scoring rule* $\mathcal{R}_\gamma$ is a function that, given an election, outputs the set of all candidates that have the highest γ-score (as the set of tied γ-winners of the election).

Committee Scoring Rules. Elkind et al. (2017b) extend the idea of single-winner scoring rules to the multiwinner setting. For a voter v in an election (C, V) with $|C| = m$ and a committee S, $|S| = k$, the *position* of S is the sequence resulting from sorting the set $\{\mathrm{pos}_v(c) \mid c \in S\}$ in increasing order. We write $[m]_k$ to denote the set of all length-k increasing sequences of numbers from $[m]$. For two committee positions $I = (i_1, \dots, i_k)$ and $J = (j_1, \dots, j_k)$, $I, J \in [m]_k$, we say that I *dominates* J if $i_1 \leq j_1, \dots, i_k \leq j_k$.

A *committee scoring function* $f_{m,k}$ for m candidates and committee size k, $f_{m,k} \colon [m]_k \to \mathbb{R}$, is a function that associates each committee position with a score in such a way that if some committee position I dominates some committee position J, then it holds that $f_{m,k}(I) \geq f_{m,k}(J)$.

Definition 1 (Elkind et al. (2017b))**.** *Let $f = (f_{m,k})_{k \leq m}$ be a family of committee scoring functions. The* committee scoring rule *$\mathcal{R}_f$ is the multiwinner rule that, given an election $E = (C, V)$ and a committee size k, outputs the committees S that maximize $f\text{-score}_E(S) = \sum_{v \in V} f_{|C|,k}\big(\mathrm{pos}_v(S)\big)$.*

***t*-Borda Family of Rules.** We use the t-Borda family of committee scoring rules as the test cases for our algorithms. Rules in this family are interesting as they form a "path" between the classic k-Borda and Chamberlin–Courant (CC) rules (they were studied in this context by Faliszewski et al. 2017a). The k-Borda rule outputs committees of candidates with the highest individual Borda scores (i.e., it selects individually-excellent candidates), whereas CC chooses diverse committees, where each voter can find an appealing representative.[3] Formally, these rules are defined via the fol-

[2] In practice, it is necessary to have a tie-breaking scheme. Throughout the paper we disregard the tie-breaking issue.

[3] See the chapter of Faliszewski et al. (2017b) for an in-depth discussion of various types of goals for multiwinner voting rules.

lowing scoring functions:

$$f_{m,k}^{k\text{-Borda}}(i_1,\dots,i_k) = \beta_m(i_1) + \cdots + \beta_m(i_k),$$
$$f_{m,k}^{\mathrm{CC}}(i_1,\dots,i_k) = \beta_m(i_1).$$

For the case of CC, a voter's most preferred committee member (i.e., the one associated with i_1 in the committee position) is referred to as the voter's *representative*.

An *ordered weighted average* (OWA) *operator* for committees of size k is a sequence $\Lambda^k = (\lambda_1^k, \dots, \lambda_k^k) \in \mathbb{R}^k$. Given a family $\Lambda = (\Lambda^k)_{k\in\mathbb{N}}$ of OWA operators for all committee sizes, the Λ-*Borda rule* is the committee scoring rule with scoring functions $f_{m,k}^{\Lambda\text{-Borda}}(i_1,\dots,i_k) = \sum_{t=1}^{k} \lambda_t^k \beta_m(i_t)$. For each positive integer t, the t-*best OWA operator* consists of t ones followed by zeros: for instance, the 2-best operators are of the form $(1, 1, 0, \dots)$. The t-*Borda rule* is simply the Λ-Borda rule defined through the family of t-best operators. In this terminology, CC is the 1-Borda rule and, if the committee size is a fixed constant k, then by taking $t = k$, we obtain the k-Borda rule as a member of the t-Borda family.

OWA-based committee scoring rules were introduced by Skowron et al. (2016), and were studied in a slightly different model by Aziz et al. (2017) and Lackner and Skowron (2017). Skowron et al. have shown that if t is a fixed constant independent of k, t-Borda rules are NP-hard to compute.

Decomposable Committee Scoring Rules. A committee scoring rule is *decomposable* (Faliszewski et al. 2016a) if it can be defined via scoring functions of the form:

$$f_{m,k}(i_1,\dots,i_k) = \gamma_{m,k}^{(1)}(i_1) + \cdots + \gamma_{m,k}^{(k)}(i_k),$$

where $\gamma = (\gamma_{m,k}^{(t)})_{1\le t\le k\le m}$ is a family of single-winner scoring functions. Decomposable rules are a superset of OWA-based rules; we need them for technical reasons.

Algorithms

Below we describe our four heuristics. The first two are known in the literature and the latter two are due to this paper. We assume that we are given an election $E = (C, V)$ with $|C| = m$, and a target committee size $k \le m$.

Simulated Annealing (SA). This is a classic metaheuristic used for many combinatorial problems (see, e.g., the survey of Suman and Kumar 2006). We use the version tailored for committee scoring rules by Faliszewski et al. (2017a).

Let $f_{m,k}$ be the given committee scoring function. The algorithm uses parameters p, q and T, $0 < p, q < 1$, $T \in \mathbb{N}$. First, we sample a random committee S_0 and then we perform T iterations. In iteration i we compute committee S_i' by replacing a random member of S_{i-1} with a random candidate in $C \setminus S_{i-1}$. If $f_{m,k}\text{-score}_E(S_i') > f_{m,k}\text{-score}_E(S_{i-1})$ then we set $S_i = S_i'$. Otherwise, with probability pq^i we set $S_i = S_i'$ and with probability $1 - pq^i$ we set $S_i = S_{i-1}$. We output a committee S_j, $j \in [T]$, with the highest $f_{m,k}$-score.

We use the same parameters as Faliszewski et al. (2017a); we run $T = 2000$ iterations with $p = 0.02$ and $q = 0.999$. Note that p is the initial probability of accepting a committee worse than the current one, and q models the speed at which this probability decays with the iteration number.

Greedy. This algorithm is an instantiation of the classic heuristic analyzed by Nemhauser et al. (1978) for optimizing submodular set functions. In the context of committee scoring rules, it was first used by Lu and Boutilier (2011) to compute approximate results of the Chamberlin–Courant rule. While all our deterministic heuristics could be called greedy, in the context of multiwinner elections only this algorithm is traditionally referred to as *the* greedy algorithm.

Given a family $f = (f_{m,k})_{k\le m}$ of committee scoring functions, the algorithm proceeds as follows. First, we set $S_0 = \emptyset$ and then we execute k iterations. In the i-th iteration we identify a candidate c that maximizes the $f_{m,i}$-score of $S_{i-1} \cup \{c\}$,[4] and set $S_i = S_{i-1} \cup \{c\}$. We output S_k.

An important property of the greedy algorithm is *committee monotonicity* (Elkind et al. 2017b): if for a given election we compute two committees W_1 and W_2 with $|W_1| < |W_2|$, then $W_1 \subset W_2$ (we ignore tie-breaking issues here). While in some settings this is a useful feature (Skowron et al. 2017), often it is a flaw. In particular, for each t-Borda rule, in the first iteration the greedy algorithm chooses a candidate with the highest Borda score. This is particularly problematic for the case of CC, where in extreme cases it may happen that the Borda winner is not a representative of any voter.

On the positive side, for the t-Borda rules, the algorithm guarantees $1 - 1/e$ approximation ratio with respect to the score of an optimal committee (Skowron, Faliszewski, and Lang 2016).

Removal. The removal algorithm is in some sense an inverse of the greedy one. It starts with the set of all candidates and iteratively removes the candidate that makes the smallest contribution to the score, until it obtains a committee of size k; a similar heuristic is used by Skowron et al. (2017). The difficulty in implementing this idea is that it is not clear which rule should be chosen to evaluate committees of size $k+1, \dots, m$. Below we describe the algorithm customized for the case of Λ-Borda rules; it can be extended to general scoring functions but focusing on the OWA-based rules leads to significantly better performance.

Fix a Λ-Borda rule and let Λ^k be the OWA operator for committees of size k. The algorithm is as follows:

1. Compute a sequence $\Lambda^k, \Lambda^{k+1}, \dots, \Lambda^m$ of OWA operators (e.g., one could define Λ^{i+1}, $k \le i < m$, by appending 0 to Λ^i, but below we propose a different approach). We refer to this sequence as the *OWA schedule*.
2. Let $S_m = C$. Then in iteration i, $i = m-1, \dots, k+1, k$, we identify a candidate c such that the Λ^i-Borda score of $S_{i+1} \setminus \{c\}$ is as high as possible, and let $S_i = S_{i+1} \setminus \{c\}$. We output S_k.

Intuitively, the motivation behind the algorithm is that we are less likely to make a serious mistake when removing a candidate from a large committee than when adding a candidate to a small one (as in the greedy algorithm).

It turns out that for the removal algorithm to perform well, we need to choose the OWA schedule wisely. For example,

[4] If there is more than one such candidate, then our implementation chooses one randomly. If a fully deterministic algorithm were needed, then this decision could be done in some other simple way.

the 0-appending schedule suggested above gave acceptable results in our initial experiments, but the following one worked much better (this is particularly surprising for the case of CC). Let $\Lambda^k = (\lambda_1, \ldots, \lambda_k)$ be the OWA operator for committees of size k. For each $m' = k, \ldots, m$ we let $\Lambda^{m'}$ consist of m'/k entries with value λ_1, followed by m'/k entries with value λ_2, and so on, until the final m'/k entries with value λ_k. If k does not divide m' evenly, then we round m'/k in a natural way; we omit the details as they have little impact on the results.

It is plausible that even better OWA schedules are possible and we discuss this issue later. Yet, even with the above schedule our algorithm demonstrates excellent performance.

Banzhaf. Our final heuristic is a variant of the greedy one, but with a game-theoretic criterion for choosing the candidates to be included in the committee.

A *cooperative game* $G = (C, \nu)$ consists of a set of *players* $C = \{c_1, \ldots, c_m\}$ and a *characteristic function* $\nu\colon 2^C \to \mathbb{R}$ such that $\nu(\emptyset) = 0$. Intuitively, for each subset C' of players, $\nu(C')$ is the joint payoff that the players in C' receive for working together. We refer to subsets of players as *coalitions*. We use the notion of the *Banzhaf value* of a player (Banzhaf 1965; Dubey and Shapley 1979), a classic solution concept for cooperative games, but we modify its definition to focus on coalitions of a given size, that are required to contain specific players.

Definition 2. *Given a cooperative game $G = (C, \nu)$, an integer k, $k \leq |C|$, and a coalition W, $|W| \leq k - 1$, we define the $k|W$-*restricted Banzhaf value *of player $c_i \in C$ as*

$$B_G(c_i, k, W) = \sum_{S \subseteq C : W \subseteq S, |S| = k-1} \nu(S \cup \{c_i\}) - \nu(S). \quad (1)$$

The classic Banzhaf value of player c_i is defined as $B_G(c_i) = \frac{1}{2^{m-1}} \sum_{k=1}^{m} B_G(c_i, k, \emptyset)$. Intuitively, it is the expected marginal contribution of a player to a randomly selected coalition and, thus, it can be seen as a measure of a player's importance. The $k|W$-restricted Banzhaf value measures players' importance provided that we need to form a size-k coalition containing the players from set W.

To use Banzhaf values in our algorithms, we express the task of evaluating committee scores as a cooperative game.

Definition 3. *Let $E = (C, V)$ be an election with $C = \{c_1, \ldots, c_m\}$ and let $\mathcal{R}_f$ be a committee scoring rule. We define the game $G(E, \mathcal{R}_f) = (C, \nu)$ so that $\nu(\emptyset) = 0$ and for each nonempty coalition S of candidates (players) we have $\nu(S) = f\text{-score}_E(S)$.*

In words, the players in our game are the candidates and the payoff of a coalition S is simply the score that this coalition—interpreted as a committee—would obtain in the underlying election. Thus, we use the terms *committee* and *coalition* interchangeably.

We are now ready to describe our heuristic (which we refer to as *Banzhaf*). Let $E = (C, V)$ be an election, let k be the committee size, and let $\mathcal{R}_f$ be a committee scoring rule, where $f = (f_{m,k})_{k \leq m}$ is the associated family of committee scoring functions. Let G be the game $G(E, \mathcal{R}_f)$. We proceed exactly as in the greedy algorithm, except that in the i-th iteration we choose a candidate from the set $\operatorname{argmax}_{c \in C \setminus S_{i-1}} B_G(c, k, S_{i-1})$. That is, we pick a candidate with the highest $k|S_{i-1}$-restricted Banzhaf value in G and set $S_i = S_{i-1} \cup \{c\}$. Intuitively, in each step we extend the committee so as to maximize the expected increase in score due to the presence of the added candidate, assuming that the rest of the committee will be chosen uniformly at random. The algorithm was inspired by analogous reasoning in the context of network analysis (Michalak et al. 2015)).

For many types of cooperative games, computing the Banzhaf values is NP-hard; see, e.g., the works of Prasad and Kelly (1990) or Bachrach and Rosenschein (2009). Fortunately, for election-based games that use decomposable committee scoring rules, (restricted) Banzhaf values can be computed in polynomial time. To see this, we first note that we can compute them vote-by-vote, since the Banzhaf value is additive. For an election $E = (C, V)$ and game $G = G(E, \mathcal{R}_f)$, where $\mathcal{R}_f$ is a committee scoring rule, we write $G(v_j)$ to denote the game G with the voter set restricted to v_j only. For each candidate c_i, each integer k, $k \leq |C|$, and each set $W \subseteq C$, $|W| < k$, we have $B_G(c_i, k, W) = \sum_{j=1}^{n} B_{G(v_j)}(c_i, k, W)$. To compute the values $B_{G(v_j)}(c_i, k, W)$, we use the following lemma.

Lemma 1. *Let $f = (f_{m,k})_{k \leq m}$ be a family of decomposable committee scoring rules defined via polynomial-time computable single-winner scoring functions, let $E = (C, V)$ be an election, let k be the committee size, and let $G = (\mathcal{R}_f, E)$ be the game associated with $\mathcal{R}_f$ and E. Then for each voter v in V, each candidate $c \in C$, and each set W such that $W \subseteq C \setminus \{c\}$ and $|W| < k$, the value $B_{G(v)}(c, k, W)$ can be computed in polynomial time.*

Proof. Let $m = |C|$. Let $\gamma_{m,k}^{(1)}, \ldots, \gamma_{m,k}^{(t)}$ be the polynomial-time computable single-winner scoring functions such that $f_{m,k}(i_1, \ldots, i_k) = \gamma_{m,k}^{(1)}(i_1) + \cdots + \gamma_{m,k}^{(k)}(i_k)$. We partition W into two sets, W_A and W_B, such that v ranks all the candidates in W_A before c and all the candidates in W_B after c. Our goal is to compute the sum in (1), which defines $B_{G(v)}(c, k, W)$, by rewriting it as a sum over candidates:

$$B_{G(v)}(c, k, W) = \Delta(c) + \textstyle\sum_{d \in C \setminus \{c\}} \Delta(d), \quad (2)$$

where we will define the Δ-terms below.

For each candidate $d \in C \setminus \{c\}$ and each $t \in [k]$, we let $\mathcal{C}(d, t)$ denote the set of coalitions S such that: (a) $W \subseteq S$; (b) $|S| = k - 1$; (c) $d \in S$; and (d) d is voter v's t-th most desirable member of S. We define $r(d)$ to be 0 if v ranks d ahead of c, and 1 otherwise. Further, we define $\Delta(d)$ to be

$$\textstyle\sum_{t=1}^{k} \sum_{S \in \mathcal{C}(d,t)} \left(\gamma_{m,k}^{(t)}(\mathrm{pos}_v(d)) - \gamma_{m,k-1}^{(t+r(d))}(\mathrm{pos}_v(d))\right)$$
$$= \textstyle\sum_{t=1}^{k} |\mathcal{C}(d, t)| \cdot \left(\gamma_{m,k}^{(t)}(\mathrm{pos}_v(d)) - \gamma_{m,k-1}^{(t+r(d))}(\mathrm{pos}_v(d))\right).$$

We define $\Delta(c)$ in a similar (but not identical) way. For each $t \in [k]$, we let $\mathcal{C}(c, t)$ be the set of coalitions S such that $W \subset S$, $|S| = k - 1$ and c is voter v's t-th most preferred candidate in $S \cup \{c\}$. Then set

$$\Delta(c) = \textstyle\sum_{t=1}^{k} \sum_{S \in \mathcal{C}(c,t)} \gamma_{m,k}^{(t)}(\mathrm{pos}_v(c))$$
$$= \textstyle\sum_{t=1}^{k} |\mathcal{C}(c, t)| \cdot \gamma_{m,k}^{(t)}(\mathrm{pos}_v(c)).$$

It is easy to see that (2) holds with these definitions.

To complete the proof, it suffices to note that, for each candidate $c \in C$ and each $t \in [k]$, the value $|\mathcal{C}(c,t)|$ can be computed in polynomial time. E.g., for $t > |W_A|$ we have:

$$|\mathcal{C}(c,t)| = \binom{\mathrm{pos}_v(c) - 1 - |W_A|}{t - 1 - |W_A|} \cdot \binom{m - \mathrm{pos}_v(c) - |W_B|}{t - k - |W_B|}$$

The idea behind the formula above is as follows. For c to be ranked in the t-th position among the candidates in $S \cup \{c\}$, S has to contain exactly $t-1$ candidates that v ranks ahead of c. S has to contain all members of W, so it contains the $|W_A|$ members of W ranked ahead of t, and it suffices to add the missing $t - 1 - |W_A|$ candidates in an arbitrary way (altogether there are $\mathrm{pos}_v(c) - 1 - |W_A|$ candidates that do not belong to W and that v ranks ahead of c). The number of ways in which we can choose members of S that v ranks after c is calculated in a similar manner. □

Lemma 1 immediately implies the following theorem.

Theorem 2. *For each decomposable committee scoring rule $\mathcal{R}_f$ defined via polynomial-time computable single-winner scoring functions, there is a polynomial-time algorithm that computes the (restricted) Banzhaf values for each candidate in a given election.*

We leave the question of whether there are natural committee scoring rules for which computing restricted Banzhaf values is NP-hard as interesting future work.

Experimental Evaluation

We performed two sets of experiments to evaluate our algorithms. In the first one, we tested how the performance of our heuristics for the t-Borda family of rules changes with t. In the second one, we focused on the CC rule and checked how the size of the committee influences the results.

Experimental Setup. We consider synthetically generated elections with $m = 100$ candidates and $n = 100$ voters. We generate voters' preferences as follows:

1. In the *impartial culture (IC)* model, for each voter we draw her preference order uniformly at random from the set of all possible linear orders.
2. In the *2D uniform square (2D)* model, each candidate and each voter is associated with a point drawn uniformly at random from the square $[-3,3] \times [-3,3]$ (these points are called the *ideal points* of the respective candidates and voters). Each voter forms her ranking by sorting the candidates in order of increasing Euclidean distances of their ideal points from her own ideal point. This distribution is an example of the two-dimensional Euclidean model and is used, e.g., in the experiments of Elkind et al. (2017a).

Let v be a voter in some election with m candidates, and let S be a committee of size k, such that $\mathrm{pos}_v(S) = (i_1, \ldots, i_k)$. For each $t \in [k]$, the t-Borda score that v assigns to S is $\beta_m(i_1) + \cdots + \beta_m(i_t) = mt - i_1 - i_2 - \cdots - i_t$. We define the *reverse t-Borda score* that v assigns to S to be $i_1 + \cdots + i_t$. The reverse t-Borda score of S in the full election is the sum of the reverse t-Borda that it gets from the voters.

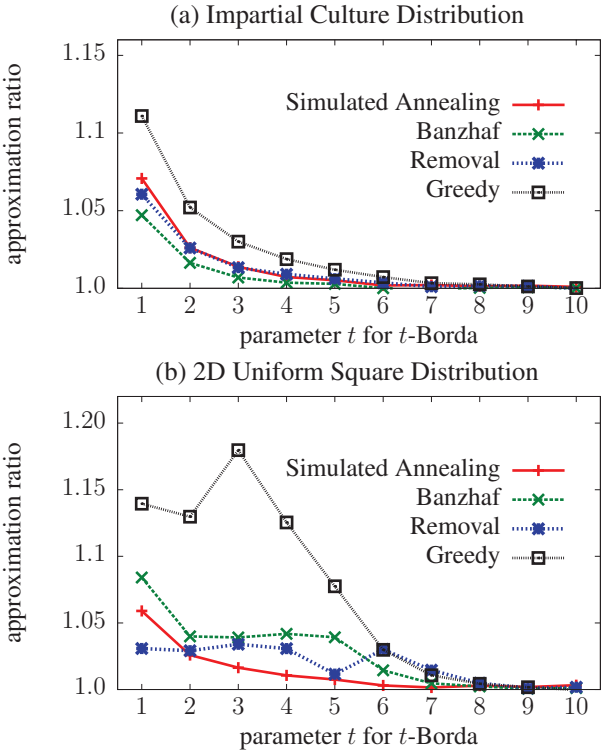

Figure 1: Ratios of average reverse t-Borda scores computed by our heuristics to those computed using the exact ILP-based algorithm for our two preference generation models ($m = 100$, $n = 100$, $k = 10$).

In this section we focus on reverse t-Borda scores instead of the standard ones because reverse scores are far more robust. For example, consider an election E with m candidates and n voters, and a committee W. If we formed an election E' by adding m new candidates to E, all ranked below the original ones, then the t-Borda score of W in E' would exceed its score in E by nmt points. On the other hand, the reverse t-Borda scores of W would be the same in E and E'. Further, reverse scores have a very natural interpretation. For example, the average reverse 1-Borda score (i.e., the average reverse CC score) that voters assign to a committee is the average position in which the voters rank their representatives. Thus we feel that reverse t-Borda scores are better suited for experimental comparisons than the standard scores.

To compute the true outputs of our voting rules, we formulated them as integer linear programs (ILPs) (such formulations can be found in the work of Skowron et al. (2016)) and solved these ILPs using the CPLEX ILP solver.

Experiments Regarding t-Borda. In this series of experiments we fixed the committee size to be $k = 10$ and we evaluated how our heuristics perform with respect to the t-Borda rules, with t ranging from 1 to 10. For each heuristic (as well as for the exact ILP-based algorithm), each $t \in [10]$, and each of the preference generation models, we generated 5000 elections and computed results using the t-Borda scoring function. Then, for each t and for each algorithm, we

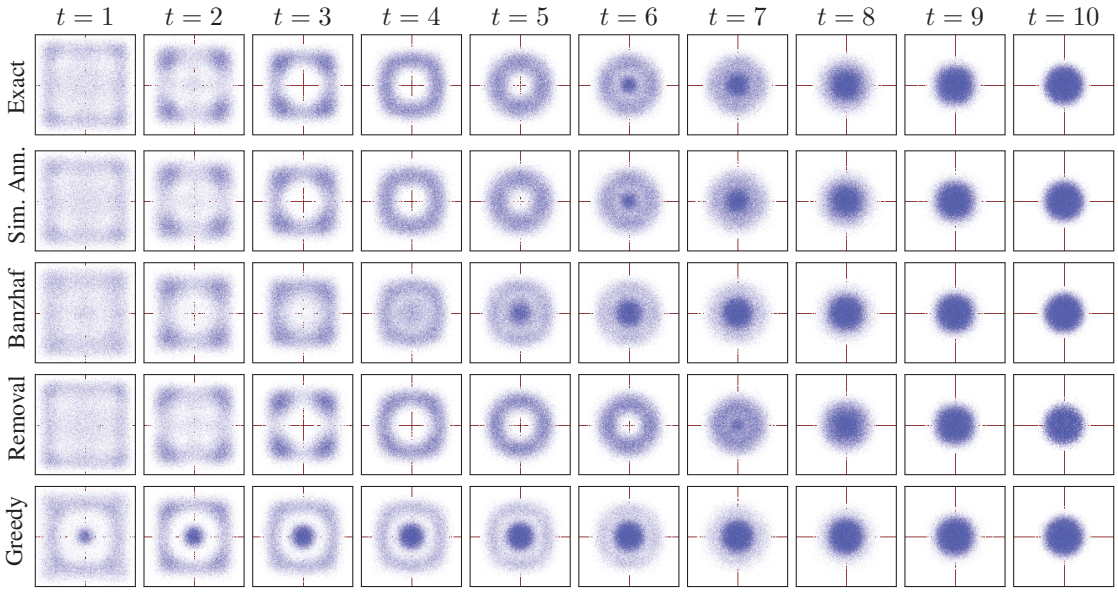

Figure 2: Histograms showing frequency of including candidates from given areas of the $[-3, 3] \times [-3, 3]$ square in the winning committees, depending on the algorithm and the value of t (2D uniform square distribution, $m = 100$, $n = 100$, $k = 10$).

computed the average reverse t-Borda score of the resulting committees. In Figure 1 we present the ratios of these scores to those of the optimal committees. For example, ratio ≈ 1.18 for the greedy algorithm with 3-Borda scoring, for the 2D uniform square distribution, means that on average the greedy algorithm produced committees whose reverse 3-Borda scores were 18% higher than the optimal scores.

Following Elkind et al. (2017a), for the 2D uniform square distribution we also generated histograms showing how often candidates from given areas of the $[-3, 3] \times [-3, 3]$ square are included in the winning committee, depending on the value of t and the algorithm used. Specifically, we partitioned the square into 120×120 equal-sized cells, and for the committees computed by each of the algorithms we calculated how many of their members fall into each cell. We present the results of these computations as histograms on Figure 2 (the darker a given cell is, the more candidates fell into it; we used the formula of Elkind et al. (2017a) for translating numerical values into colors).

Figures 1 and 2 indicate that simulated annealing (SA) performs very well; in particular, its histograms are practically indistinguishable from those for the actual rule. In contrast, the greedy algorithm is the weakest performer. Indeed, it consistently achieves the worst approximation ratios and its histograms visibly show the bias that comes from selecting the Borda winner in the first iteration (see the dark blue spots in the centers of the histogram for $t \in \{1, \ldots, 5\}$; importantly, these spots are not present in the histograms computed using the exact algorithm). From Figure 1 we can also conclude that both the removal algorithm and the Banzhaf algorithm perform very well: their approximation error never exceeds 10% and declines rapidly with t. In fact, for the IC model the Banzhaf algorithm achieves the best approximation ratios, and the removal algorithm performs as well as SA. Intuitively, the good performance of the Banzhaf algorithm in this setting is to be expected: when selecting a committee member, it assumes that the remaining members would be chosen uniformly at random, which matches the spirit of the IC model. For the 2D uniform square model, the removal algorithm outperforms the Banzhaf algorithm by some margin (the advantage is not huge, but noticeable).

It is interesting to compare the histograms generated by the algorithms. SA generates essentially the same histograms as the exact algorithm, and the greedy algorithm has a clear bias (as argued above). The Banzhaf and removal algorithms produce histograms very similar to those of the exact algorithm for $t \in \{1, 2, 3\} \cup \{8, 9, 10\}$. For the remaining values, removal tends to focus too much on the "outer ring", whereas the Banzhaf algorithm focuses too much on the "inner disc". It is remarkable that these methods make different kinds of errors. Thus, deciding which algorithm to prefer should depend on the application. For instance, for the funding agency example mentioned in the introduction, when choosing $k = 10$ proposals using 5-Borda, the Banzhaf algorithm would likely be superior to the removal one; the proposals from the "inner disc" are those that are individually ranked highly (they have the highest individual Borda scores), but the algorithm also selects some more diverse proposals from the "outer ring". Arguably, in this case the Banzhaf algorithm might even be more appealing than the original 5-Borda voting rule.

Experiments Regarding Chamberlin–Courant. Our second set of experiments focuses on the CC rule for varying committee sizes. Besides the four heuristics, we evaluated the Ranging algorithm,[5] designed specifically for the CC rule

[5]Ranging is a variant of Algorithm P of Skowron et al. (2015) due to Elkind et al. (2017a), and forms the basis of a PTAS for the CC rule. Given an election, the algorithm considers each $\ell \in [m]$

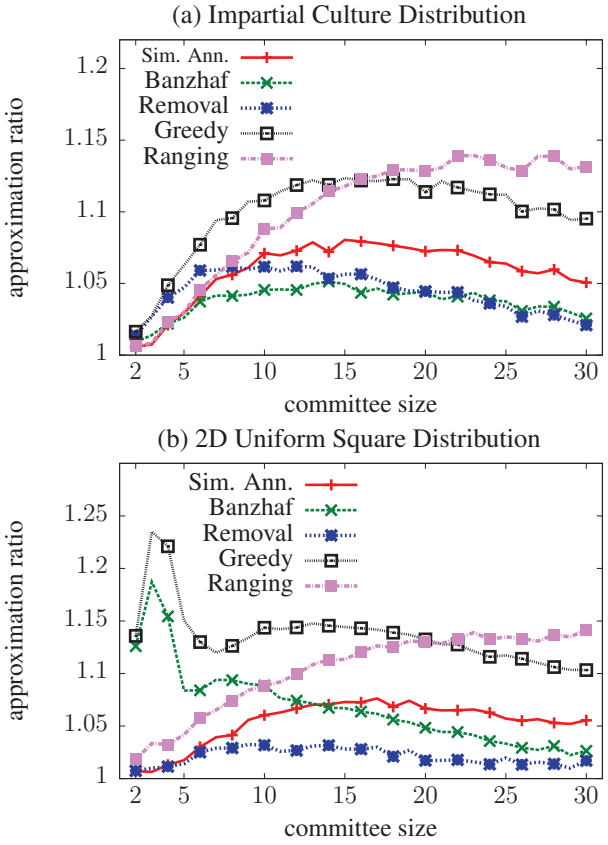

Figure 3: Ratios of average reverse CC scores (average positions of the voters' representatives) computed by our heuristics to those computed using the exact algorithm for our two preference generation models ($m = 100$, $n = 100$).

and known to provide the strongest theoretically-established approximation guarantee for this rule.

For each of the algorithms, each committee size $k \in \{2, \ldots, 30\}$, and each model of preferences, we generated 1000 elections and computed the winning committees. In Figure 3 we present the ratios of the achieved average reverse CC scores to those computed using the exact algorithm (recall that for the case of CC, the reverse scores are simply the average positions of the voters' representatives).

The results in Figure 3 are quite intriguing. We see that the greedy algorithm is the worst performer for $k \leq 20$, but then the ranging algorithm becomes the worst, even though it has the strongest theoretical guarantees. On the other hand, the removal algorithm offers excellent performance for all committee sizes and both preference models. The Banzhaf algorithm also performs very well, especially for larger committees. For the IC model, where we expect the Banzhaf algorithm to do well, its performance is indeed better than that of the removal algorithm for most cases, but even there the removal algorithm is sometimes marginally better.

and greedily finds a size-k committee such that as many voters as possible rank some committee member in the top ℓ positions. Among these committees, it selects one with the highest CC score.

Performance of SA. Notably, for the case of CC, the Banzhaf and removal algorithms outperform SA for a range of committee sizes. This is due to our choice of parameters p, q and T; SA would perform better if we executed more iterations or re-ran the algorithm a few times for each election. However, even with the current parameters SA produces histograms that are very close to those for the actual rules (for the t-Borda experiments),[6] which indicates that this variant of SA is a reasonable benchmark. It is thus remarkable that polynomial-time computable deterministic rules, such as removal or the Banzhaf algorithm, can beat this benchmark.

OWA Schedules for Removal. One of our most intriguing results is the performance of the removal algorithm, which strongly relies on the specific OWA schedule that we used. We also performed initial experiments with the 0-appending schedule (which, for example, would always use OWAs of the form $(1, 0, \ldots, 0)$ for the CC rule), and the results were notably worse. For example, for the 2D uniform square distribution, CC rule, and committee size $k = 10$, the average position of a representative computed using our current removal algorithm is, on average, only 3% further from the top than the optimal one, whereas for the 0-appending schedule the average error is 10%. It may be possible to design even better OWA schedules for the removal algorithm, and we view this as as an interesting direction for future studies.

Running Times. So far, we have mostly disregarded the running times of our algorithms. There are two reasons for that. First, all our algorithms are sufficiently fast that they can be executed on practically relevant election instances. Second, when one computes election results, the quality of the results is far more important than the running times of the algorithms (provided they are not prohibitive). Nevertheless, to give the reader some idea of the relative running times of our algorithms, we mention that, to compute the results for 100 candidates, 100 voters, and committee size $k = 10$ (on an office machine with Intel i5 760 CPU and 8GB of RAM), the general Banzhaf algorithm requires about $24s$, removal requires about $10s$, SA requires about $1s$, and greedy requires about $0.5s$. For the case of CC, however, we designed an optimized Banzhaf algorithm, which runs under $1s$. We defer more detailed analysis to the full version of the paper.

Conclusions

We have evaluated four algorithms for approximately computing committee scoring rules: simulated annealing, which is randomized, and greedy, removal, and Banzhaf algorithms, which are deterministic. We have shown that simulated annealing performs very well in essentially all settings and provides visual representations of results that match the exact ones nearly perfectly. The greedy algorithm usually performs worst, but the other two algorithms often match or outperform simulated annealing (and are also deterministic). This is very relevant from a practical point of view, as randomized voting rules are unacceptable in many settings.

Our work leaves several directions for future research. For example, it would be interesting to consider further heuris-

[6] This confirms that the histograms computed by Faliszewski et al. (2017a), using SA with the same parameters, are meaningful.

tics and metaheuristics, hopefully enlarging the subclass of committee scoring rules which can be efficiently solved in practice. Further, viewing our heuristics as voting rules on their own could yield an interesting axiomatic analysis.

Acknowledgments. Dominik Peters was supported by ERC grant 639945 (ACCORD). Martin Lackner was supported by ERC grant 639945 (ACCORD) and by the Austrian Science Foundation FWF, grant P25518 and Y698. Piotr Faliszewski was supported by the National Science Centre, Poland, under project 2016/21/B/ST6/01509. We thank Edith Elkind for useful discussions, and the reviewers for helpful feedback.

References

Aziz, H.; Gaspers, S.; Gudmundsson, J.; Mackenzie, S.; Mattei, N.; and Walsh, T. 2015. Computational aspects of multiwinner approval voting. In *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems*, 107–115.

Aziz, H.; Brill, M.; Conitzer, V.; Elkind, E.; Freeman, R.; and Walsh, T. 2017. Justified representation in approval-based committee voting. *Social Choice and Welfare* 48(2):461–485.

Bachrach, Y., and Rosenschein, J. S. 2009. Power in threshold network flow games. *Autonomous Agents and Multi-Agent Systems* 18(1):106–132.

Banzhaf, J. 1965. Weighted voting doesn't work: A mathematical analysis. *Rutgers Law Review* 19:317–343.

Barberà, S., and Coelho, D. 2008. How to choose a non-controversial list with k names. *Social Choice and Welfare* 31(1):79–96.

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

Brill, M.; Laslier, J.; and Skowron, P. 2017. Multiwinner approval rules as apportionment methods. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, 414–420.

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

Dubey, P., and Shapley, L. 1979. Mathematical properties of the Banzhaf power index. *Mathematics of Operations Research* 4(2):99–131.

Elkind, E.; Faliszewski, P.; Laslier, J.; Skowron, P.; Slinko, A.; and Talmon, N. 2017a. What do multiwinner voting rules do? An experiment over the two-dimensional Euclidean domain. In *Proceedings of the 31st AAAI Conference on Artificial Intelligence*, 494–501. Full version available at `http://home.agh.edu.pl/~faliszew/2d.pdf`.

Elkind, E.; Faliszewski, P.; Skowron, P.; and Slinko, A. 2017b. Properties of multiwinner voting rules. *Social Choice and Welfare* 48(3):599–632.

Faliszewski, P.; Skowron, P.; Slinko, A.; and Talmon, N. 2016a. Committee scoring rules: Axiomatic classification and hierarchy. In *Proceedings of the 25th International Joint Conference on Artificial Intelligence*, 250–256.

Faliszewski, P.; Skowron, P.; Slinko, A.; and Talmon, N. 2016b. Multiwinner analogues of the plurality rule: Axiomatic and algorithmic views. In *Proceedings of the 30th AAAI Conference on Artificial Intelligence*, 482–488.

Faliszewski, P.; Slinko, A.; Stahl, K.; and Talmon, N. 2016c. Achieving fully proportional representation by clustering voters. In *Proceedings of the 15th International Conference on Autonomous Agents and Multiagent Systems*, 296–304.

Faliszewski, P.; Skowron, P.; Slinko, A.; and Talmon, N. 2017a. Multiwinner rules on paths from k-Borda to Chamberlin–Courant. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 192–198.

Faliszewski, P.; Skowron, P.; Slinko, A.; and Talmon, N. 2017b. Multiwinner voting: A new challenge for social choice theory. In Endriss, U., ed., *Trends in Computational Social Choice*. AI Access.

Kirkpatrick, S.; Gelatt, Jr., C.; and Vecchi, M. 1983. Optimization by simulated annealing. *Science* 220(4598):671–680.

Lackner, M., and Skowron, P. 2017. Consistent approval-based multi-winner rules. Technical Report arXiv:1704.02453 [cs.GT], arXiv.org.

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

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

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

Nemhauser, G.; Wolsey, L.; and Fisher, M. 1978. An analysis of approximations for maximizing submodular set functions. *Mathematical Programming* 14(1):265–294.

Prasad, K., and Kelly, J. 1990. NP-completeness of some problems concerning voting games. *International Journal of Game Theory* 19(1):1–9.

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

Skowron, P.; Lackner, M.; Brill, M.; Peters, D.; and Elkind, E. 2017. Proportional rankings. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, 409–415.

Skowron, P.; Faliszewski, P.; and Lang, J. 2016. Finding a collective set of items: From proportional multirepresentation to group recommendation. *Artificial Intelligence* 241:191–216.

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

Suman, B., and Kumar, P. 2006. A survey of simulated annealing as a tool for single and multiobjective optimization. *Journal of the Operational Research Society* 57(10):1143–1160.