Automated Design of Scoring Rules by Learning from Examples

Ariel D. Procaccia, Aviv Zohar, and Jeffrey S. Rosenschein

School of Engineering and Computer Science
The Hebrew University of Jerusalem
Jerusalem, Israel
{arielpro, avivz, jeff}@cs.huji.ac.il

ABSTRACT

Scoring rules are a broad and concisely-representable class of voting rules which includes, for example, Plurality and Borda. Our main result asserts that the class of scoring rules, as functions from preferences into candidates, is efficiently learnable in the PAC model. We discuss the applications of this result to automated design of scoring rules. We also investigate possible extensions of our approach, and (along the way) we establish a lemma of independent interest regarding the number of distinct scoring rules.

Categories and Subject Descriptors

F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity;
I.2.11 [**Artificial Intelligence**]: Distributed Artificial Intelligence—*Multiagent Systems*;
J.4 [**Computer Applications**]: Social and Behavioral Sciences—*Economics*

General Terms

Algorithms, Theory, Economics

Keywords

Voting, PAC learning

1. INTRODUCTION

Voting is a well-studied method of preference aggregation, in terms of its theoretical properties, as well as its computational aspects [5, 13]; various practical, implemented applications that use voting exist [8, 7]. In an election, n voters express their preferences over a set of m candidates or alternatives. To be precise, each voter is assumed to reveal linear preferences—a ranking of the candidates. The outcome of the election is determined according to a *voting rule*.

1.1 Scoring Rules

The predominant—ubiquitous, even—voting rule in real-life elections is the *Plurality* rule. Under Plurality, each voter awards one point to the candidate it ranks first, i.e., its most preferred alternative. The candidate that accumulated the most points, summed over all voters, wins the election. Another example of a voting rule is the *Veto* rule: each voter "vetoes" a single candidate; the candidate that was vetoed by the fewest voters wins the election. Yet a third example is the *Borda* rule: every voter awards $m-1$ points to its top-ranked candidate, $m-2$ points to its second choice, and so forth—the least preferred candidate is not awarded any points. Once again, the candidate with the most points is elected.

The abovementioned three voting rules all belong to an important family of voting rules known as *scoring rules*. A scoring rule can be expressed by a vector of parameters $\vec{\alpha} = \langle \alpha_1, \alpha_2, \ldots, \alpha_m \rangle$, where each α_l is a real number and $\alpha_1 \geq \alpha_2 \geq \cdots \geq \alpha_m$. Each voter awards α_1 points to its most-preferred alternative, α_2 to its second-most-preferred alternative, etc. Predictably, the candidate with the most points wins. Under this unified framework, we can express our three rules as:

- *Plurality*: $\vec{\alpha} = \langle 1, 0, \ldots, 0 \rangle$.
- *Borda*: $\vec{\alpha} = \langle m-1, m-2, \ldots, 0 \rangle$.
- *Veto*: $\vec{\alpha} = \langle 1, \ldots, 1, 0 \rangle$.

Voting rules are often compared on the basis of various criteria that define potentially desirable properties. We outline below several important criteria, some economic, and some computational.

1. *Anonymity*: The voting rule is symmetric with regard to the voters.
2. *Neutrality*: The voting rule is symmetric with regard to the candidates.
3. *Consistency*: If two disjoint sets of voters elect the same candidate, this candidate is also elected by the union of the two sets.
4. *Majority*: A candidate that is most preferred by a majority of voters wins the election.
5. *Robustness* [14]: The worst-case probability of the outcome of the election *not* changing as a result of a random mistake/fault in the preferences of the voters.

Cite as: Automated Design of Scoring Rules by Learning from Examples, Ariel D. Procaccia, Aviv Zohar and Jeffrey S. Rosenschein, *Proc. of 7th Int. Conf. on Autonomous Agents and Multiagent Systems (AAMAS 2008)*, Padgham, Parkes, Müller and Parsons (eds.), May, 12-16., 2008, Estoril, Portugal, pp. XXX-XXX.

6. *Complexity of Manipulation*: Consider a coalition of voters that aims to improve its utility from the election by voting untruthfully. How computationally difficult is it to find the coalition's optimal vote?

7. *Communication Complexity*: The amount of communication that is required in order to determine the winner of the election.

A good indication of the importance of scoring rules is given by the fact that they are exactly the family of voting rules that are anonymous, neutral, and consistent [17]. Many other properties can be achieved by scoring rules, depending on the exact parameters. To put it differently, different choices of the parameters of a scoring rule yield significantly different voting rules in terms of their properties. As an example, Table 1 compares Plurality, Borda, and Veto, on the basis of the last four properties in our list.

1.2 Our Setting and Approach

We consider the following setting: an entity, which we refer to as the *designer*, has in mind a voting rule (which may reflect the ethics of a society). We assume that the designer is able, for each constellation of voters' preferences with which it is presented, to designate a winning candidate (perhaps with considerable computational effort). In particular, one can think of the designer's representation of the voting rule as a black box that matches preference profiles to winning candidates. This setting is relevant, for example, when a designer has in mind different properties it wants its rule to satisfy; in this case, given a preference profile, the designer can specify a winning candidate that is compatible with these properties.

We would like to find a concise and easily understandable representation of the voting rule the designer has in mind. We refer to this process as *automated design of voting rules*: given a specification of properties, or, indeed, of societal ethics, find an elegant voting rule that implements the specification. In this paper, we do so by learning from examples. The designer is presented with different preference profiles, drawn according to a fixed distribution. For each profile, the designer answers with the winning candidate. The number of queries presented to the designer must intuitively be as small as possible: the computations the designer has to carry out in order to handle each query might be complex, and communication might be costly.[1]

Now, we further assume that the "target" voting rule the designer has in mind, i.e., the one given as a black box, is a scoring rule. This assumption is justified by the discussion in the previous subsection, as many reasonable properties the designer might find desirable are achievable by scoring rules. We would like to produce a scoring rule (represented by its parameters $\vec{\alpha}$) that is as "close" as possible to the target rule. This way, the designer could in principle translate the possibly cumbersome, unknown representation of a voting rule into a succinct one that can be easily understood and computed. Note that this setting was previously presented and advocated by Procaccia et al. [15]

By "close," we mean close with respect to the fixed distribution over preference profiles. More precisely, we would like to construct an algorithm that receives pairs of the form (preferences, winner) drawn according to a fixed distribution D over preferences, and outputs a scoring rule, such that the probability according to D that our scoring rule and the target rule agree is as high as possible. We wish, in fact, to learn scoring rules in the framework of the formal PAC (Probably Approximately Correct) learning model; a concise introduction to this model is given in Section 2.

Further justification for our agenda is given by noting that it might be difficult to compute a voting rule on all instances, but it might be sufficient to simply calculate the election's result on typical instances. The distribution D can be chosen, by the designer, to concentrate on such instances.

The dimension of a function class is a combinatorial measure of the richness of the class; this dimension of a function class is closely related to the number of examples needed to learn it. We give tight bounds on the dimension of the class of scoring rules, providing an upper bound of m, and a lower bound of $m-3$, where m is the number of candidates in an election. In addition, we show that, given a set of examples, one can efficiently construct a scoring rule that is consistent with the examples, if one exists. Combined, these results imply that the class of scoring rules is efficiently learnable. In other words, given a combination of properties which is satisfied by some scoring rule, it is possible to construct a "close" scoring rule in polynomial time.

It is also worthwhile to ask whether it is possible to extend this approach. Specifically, we pose the question: if the designer has some general voting rule in mind, is it possible to learn a "close" scoring rule? We show that the there are voting rules which cannot be "approximated" by scoring rules. Along the way, we show that the number of distinct scoring rules is at most exponential in the number of voters and candidates (whereas the number of voting rules is double exponential).

1.3 Related Work

Currently there exists a small body of work on learning in economic settings. Kalai [9] explores the learnability (in the PAC model) of rationalizable choice functions. These are functions which, given a set of alternatives, choose the element that is maximal with respect to some linear order. Similarly, PAC learning has very recently been applied to computing utility functions that are rationalizations of given sequences of prices and demands [1].

Another prominent example is the paper by Lahaie and Parkes [10], which considers preference elicitation in combinatorial auctions. The authors show that preference elicitation algorithms can be constructed on the basis of existing learning algorithms. The learning model used, exact learning, differs from ours (PAC learning).

Conitzer and Sandholm [2] have studied automated mechanism design, in the more restricted setting where agents have numerical valuations for different alternatives. They propose automatically designing a truthful mechanism for every preference aggregation setting. However, they find that, under two solution concepts, even determining whether there exists a deterministic mechanism that guarantees a certain social welfare is an $\mathcal{NP}$-complete problem. The authors also show that the problem is tractable when designing a randomized mechanism. In more recent work [3], Conitzer and Sandholm put forward an efficient algorithm for designing deterministic mechanisms, which works only in very

[1]Note that while some properties, like high Complexity of Manipulation, do not naturally name the winner of the election, other properties may hint at the winner, or designate it directly (e.g., Majority, Condorcet Winner, etc.).

	Majority	Robustness	Manipulation	Communication
Plurality	Yes	$\geq \frac{m-2}{m-1}$ [14]	$\mathcal{P}$ [5]	$\Theta(n \log m)$ [4]
Borda	No	$\leq \frac{1}{m}$ [14]	$\mathcal{NP}$-complete [5]	$\Theta(nm \log m)$ [4]
Veto	No	$\geq \frac{m-2}{m-1}$ [14]	$\mathcal{NP}$-complete [5]	?

Table 1: Different scoring rules greatly differ in the desiderata they satisfy

limited scenarios. In short, our setting, goals, and methods are completely different—in the general voting context, even framing computational complexity questions is problematic, since the goal cannot be specified with reference to expected social welfare.

Most importantly, this paper complements the paper of Procaccia, Zohar, Peleg and Rosenschein which studies the learnability of voting trees [15], with, similarly, the goal of automated design of voting trees. Voting trees are voting rules that iteratively compare pairs of candidates based on the preference of the majority of voters; these rules can be succinctly represented. Voting trees are shown to be learnable in certain settings, but are not efficiently learnable (unlike scoring rules). It is important to note that the families of scoring rules and voting trees are practically disjoint, and that the learning-theoretic results in this paper are far stronger.

1.4 Structure of the Paper

In Section 2 we give an introduction to the PAC model. In Section 3, we describe our setting and rigorously prove that the class of scoring rules is efficiently learnable. In Section 4, we discuss extending our approach, and in Section 5, we give our conclusions.

2. PRELIMINARIES

In this section we give a very short introduction to the PAC model and the generalized dimension of a function class. A more comprehensive (and slightly more formal) overview of the model, and results concerning the dimension, can be found in [12].

In the PAC model, the learner is attempting to learn a function $f : X \to Y$, which belongs to a class $\mathcal{F}$ of functions from X to Y. The learner is given a *training set*—a set of points in X, $x_1, x_2, \ldots, x_t$, which are sampled i.i.d. (independently and identically distributed) according to a distribution D over the sample space X. D is unknown, but is fixed throughout the learning process. In this paper, we assume the "realizable" case, where a target function $f^*(x)$ exists, and the given training examples are in fact labeled by the target function: $\{(x_k, f^*(x_k))\}_{k=1}^t$. The *error* of a function $f \in \mathcal{F}$ is defined as

$$err(f) = \Pr_{x \sim D}[f(x) \neq f^*(x)]. \quad (1)$$

$\epsilon > 0$ is a parameter given to the learner that defines the *accuracy* of the learning process: we would like to achieve $err(h) \leq \epsilon$. Notice that $err(f^*) = 0$. The learner is also given an *accuracy* parameter $\delta > 0$, that provides an upper bound on the probability that $err(h) > \epsilon$:

$$\Pr[err(h) > \epsilon] < \delta. \quad (2)$$

We now formalize the discussion above:

Definition 2.1.

1. A *learning algorithm* L is a function from the set of all training examples to $\mathcal{F}$ with the following property: given $\epsilon, \delta \in (0, 1)$ there exists an integer $s(\epsilon, \delta)$—the *sample complexity*—such that for any distribution D on X, if Z is a sample of size at least s where the samples are drawn i.i.d. according to D, then with probability at least $1 - \delta$ it holds that $err(L(Z)) \leq \epsilon$.
2. L is an *efficient* learning algorithm if it always runs in time polynomial in $1/\epsilon$, $1/\delta$, and the size of the representations of the target function, of elements in X, and of elements in Y.
3. A function class $\mathcal{F}$ is *(efficiently) PAC-learnable* if there is an (efficient) learning algorithm for $\mathcal{F}$.

The sample complexity of a learning algorithm for $\mathcal{F}$ is closely related to a measure of the class's combinatorial richness known as the generalized dimension.

Definition 2.2. Let $\mathcal{F}$ be a class of functions from X to Y. We say $\mathcal{F}$ *shatters* $S \subseteq X$ if there exist two functions $g, h \in \mathcal{F}$ such that

1. For all $x \in S$, $g(x) \neq h(x)$.
2. For all $S_1 \subseteq S$, there exists $f \in \mathcal{F}$ such that for all $x \in S_1$, $f(x) = h(x)$, and for all $x \in S \setminus S_1$, $f(x) = g(x)$.

Definition 2.3. Let $\mathcal{F}$ be a class of functions from a set X to a set Y. The *generalized dimension* of $\mathcal{F}$, denoted by $D_G(\mathcal{F})$, is the greatest integer d such that there exists a set of cardinality d that is shattered by $\mathcal{F}$.

Lemma 2.4. *[12, Lemma 5.1] Let X and Y be two finite sets and let $\mathcal{F}$ be a set of total functions from X to Y. If $d = D_G(F)$, then $2^d \leq |\mathcal{F}|$.*

The generalized dimension of a function provides both upper and lower bounds on the sample complexity of algorithms.

Theorem 2.5. *[12, Theorem 5.1] Let $\mathcal{F}$ be a class of functions from X to Y of generalized dimension d. Let L be an algorithm such that, when given a set of t labeled examples $\{(x_k, f^*(x_k))\}_k$ of some $f^* \in \mathcal{F}$, sampled i.i.d. according to some fixed but unknown distribution over the instance space X, produces an output $f \in \mathcal{F}$ that is consistent with the training set. Then L is an (ϵ, δ)-learning algorithm for $\mathcal{F}$ provided that the sample size obeys:*

$$s \geq \frac{1}{\epsilon}\left((\sigma_1 + \sigma_2 + 3) D_G(\mathcal{F}) \ln 2 + \ln\left(\frac{1}{\delta}\right)\right) \quad (3)$$

where σ_1 and σ_2 are the sizes of the representation of elements in X and Y, respectively.

THEOREM 2.6. *[12, Theorem 5.2] Let $\mathcal{F}$ be a function class of generalized dimension $d \geq 8$. Then any (ϵ, δ)-learning algorithm for $\mathcal{F}$, where $\epsilon \leq 1/8$ and $\delta < 1/4$, must use sample size $s \geq \frac{d}{16\epsilon}$.*

3. LEARNING SCORING RULES

Before diving in, we introduce some notation. Let $N = \{1, 2, \ldots, n\}$ be the set of voters, and let $C = \{c_1, c_2, \ldots, c_m\}$ be the set of candidates. Let $\mathcal{L}$ be the set of linear preferences[2] over C; each voter has preferences $\succ^i \in \mathcal{L}$. We denote the *preference profile*, consisting of the voters' preferences, by $\succ^N = \langle \succ^1, \succ^2, \ldots, \succ^n \rangle$.

Let $\vec{\alpha}$ be a vector of real numbers such that $\alpha_l \geq \alpha_{l+1}$ for all $l = 1, \ldots, m-1$. Let $f_{\vec{\alpha}} : \mathcal{L}^N \to C$ be the scoring rule defined by the vector $\vec{\alpha}$, i.e., each voter awards α_l points to the candidate it ranks in the l'th place, and the rule elects the candidate with the most points.

Since several candidates may have maximal scores in an election, we must adopt some method of tie-breaking. Our method works as follows. Ties are broken in favor of the candidate that was ranked first by more voters; if several candidates have maximal scores and were ranked first by the same number of voters, the tie is broken in favor of the candidate that was ranked second by more voters; and so on.[3]

Let $\mathcal{S}_m^n$ be the class of scoring rules with n voters and m candidates. Our goal is to learn, in the PAC model, some target function $f_{\vec{\alpha}^*} \in \mathcal{S}_m^n$. To this end, the learner receives a training set $\{(\succ_k^N, f_{\vec{\alpha}^*}(\succ_k^N)\}_k$, where each $\succ_k^N$ is drawn from a fixed distribution over $\mathcal{L}^N$; let $c_{j_k} = f_{\vec{\alpha}^*}(\succ_k^N)$. For the profile $\succ_k^N$, we denote by $\pi_{j,l}^k$ the number of voters that ranked candidate c_j in place l. Notice that candidate c_j's score under the preference profile $\succ_k^N$ is $\sum_l \pi_{j,l}^k \alpha_l$.

3.1 PAC-Learnability of $\mathcal{S}_m^n$

Our main goal in this section is to prove the following theorem.

THEOREM 3.1. *For all $n, m \in \mathbb{N}$, the class $\mathcal{S}_m^n$ is efficiently PAC-learnable.*

By Theorem 2.5, in order to prove Theorem 3.1 it is sufficient to validate the following two claims: 1) that there exists an algorithm which, for any training set, runs in time polynomial in n, m, and the size of the training set, and outputs a scoring rule which is consistent with the training set (assuming one exists); and 2) that the generalized dimension of the class $\mathcal{S}_m^n$ is polynomial in n and m.

REMARK 3.2. It is possible to prove Theorem 3.1 by using a transformation between scoring rules and sets of linear threshold functions. Indeed, it is well-known that the VC dimension (the restriction of the generalized dimension to boolean-valued functions) of linear threshold functions over $\mathcal{R}^d$ is $d+1$. In principle, it is possible to transform a scoring rule into a linear threshold function which receives (generally speaking) vectors of rankings of candidates as input. Given a training set of profiles, we could transform it into a training set of rankings and use a learning algorithm.

However, we are interested in producing an accurate scoring rule according to a distribution D on preference profiles, which represents typical profiles. It is possible to consider a many-to-one mapping between distributions over profiles and distributions over the abovementioned vectors of rankings. Unfortunately, when this procedure is used, it is nontrivial to guarantee that the learned voting rule succeeds according to the original distribution D. Moreover, this procedure seems to require an increase in sample complexity compared to the analysis given below. Therefore, we proceed with the more "direct" agenda outlined above and detailed below.

It is rather straightforward to construct an efficient algorithm that outputs consistent scoring rules. Given a training set, we must choose the parameters of our scoring rule in a way that, for any example, the score of the designated winner is at least as large as the scores of other candidates. Moreover, if ties between the winner and a loser would be broken in favor of the loser, then the winner's score must be strictly higher than the loser's. Our algorithm, given as Algorithm 1, simply formulates all the constraints as linear inequalities, and solves the resulting linear program.

Algorithm 1 Given a training set, the algorithm returns a scoring rule which is consistent with the given examples, if one exists.

for $k \leftarrow 1 \ldots t$ **do**
  $C_k \leftarrow \emptyset$
  for all $j \neq j_k$ **do** ▷ c_{j_k} is the winner in example k
    $\vec{\pi}^{\Delta} \leftarrow \vec{\pi}_{j_k}^k - \vec{\pi}_j^k$
    $l_0 \leftarrow \min\{l : \pi_l^{\Delta} \neq 0\}$
    if $\pi_{l_0}^{\Delta} < 0$ **then** ▷ Ties are broken in favor of c_j
      $C_k \leftarrow C_k \cup \{c_j\}$
    end if
  end for
end for
return a feasible solution $\vec{\alpha}$ to the following linear program:

$$\begin{array}{l} \forall k,\ \forall c_j \in C_k, \sum_l \pi_{j_k,l}^k \alpha_l \geq \sum_l \pi_{j,l}^k \alpha_l + 1 \\ \forall k,\ \forall c_j \notin C_k, \sum_l \pi_{j_k,l}^k \alpha_l \geq \sum_l \pi_{j,l}^k \alpha_l \\ \forall l = 1, \ldots, m-1 \quad \alpha_l \geq \alpha_{l+1} \\ \forall l,\ \alpha_l \geq 0 \end{array}$$

A linear program can be solved in time that is polynomial in the number of variables and inequalities [16]; it follows that Algorithm 1's running time is polynomial in n, m, and the size of the training set.

REMARK 3.3. Notice that any vector $\vec{\alpha}$ with a polynomial representation can be scaled to an equivalent vector of integers which is also polynomially representable. In this case, the scores are always integral. Thus, instead of using a strict inequality in the LP's first set of constraints, we can use a weak inequality with an additive factor of 1.

REMARK 3.4. Although the transformation between learning scoring rules and learning linear threshold functions mentioned in Remark 3.2 has some drawbacks as a learning

[2] A binary relation which is antisymmetric, transitive, and total.

[3] In case several candidates have maximal scores and identical rankings everywhere, break ties arbitrarily—say, in favor of the candidate with the smallest index.

method, results on the computational complexity of learning linear threshold functions can be leveraged to obtain computational efficiency. Indeed, well-known algorithms such as Winnow [11] suit this purpose.

REMARK 3.5. Algorithm 1 can also be used to check, with high probability, if the voting rule the designer has in mind is indeed a scoring rule, as described (in a different context) by Kalai [9] (we omit the details here). This further justifies the setting in which the voting rule the designer has in mind is known to be a scoring rule.

So, it remains to demonstrate that the generalized dimension of $\mathcal{S}_m^n$ is polynomial in n and m. The following lemma shows this.

LEMMA 3.6. *The generalized dimension of the class S_m^n is at most m:*

$$D_G(S_m^n) \leq m.$$

PROOF. According to Definition 2.3, we need to show that any set of cardinality $m+1$ cannot be shattered by $\mathcal{F}$. Let $S = \{\succ_k^N\}_{k=1}^{m+1}$ be such a set, and let h, g be the two social choice functions that disagree on all preference profiles in S. We shall construct a subset $S_1 \subseteq S$ such that there is no scoring rule $f_{\vec{\alpha}}$ that agrees with h on S_1 and agrees with g on $S \setminus S_1$.

Let us look at the first preference profile from our set, $\succ_1^N$. We shall assume without loss of generality that $h(\succ_1^N) = c_1$, while $g(\succ_1^N) = c_2$, and that in $\succ_1^N$ ties are broken in favor of c_1. Let $\vec{\alpha}$ be some parameter vector. If we are to have $h(\succ_1^N) = f_{\vec{\alpha}}(\succ_1^N)$, it must hold that

$$\sum_{l=1}^{m} \pi_{1,l}^1 \cdot \alpha_l \geq \sum_{l=1}^{m} \pi_{2,l}^1 \cdot \alpha_l, \tag{4}$$

whereas if we wanted $f_{\vec{\alpha}}$ to agree with g we would want the opposite:

$$\sum_{l=1}^{m} \pi_{1,l}^1 \cdot \alpha_l < \sum_{l=1}^{m} \pi_{2,l}^1 \cdot \alpha_l \tag{5}$$

More generally, we define, with respect to the profile $\succ_k^N$, the vector $\vec{\pi}_\Delta^k$ as the vector whose l'th coordinate is the difference between the number of times the winner under h and the winner under g were ranked in the l'th place:[4]

$$\vec{\pi}_\Delta^k = \vec{\pi}_{h(\succ_k)}^k - \vec{\pi}_{g(\succ_k)}^k. \tag{6}$$

Now we can concisely write necessary conditions for $f_{\vec{\alpha}}$ agreeing with h or g, respectively, by writing:[5]

$$\vec{\pi}_\Delta^k \cdot \vec{\alpha} \geq 0 \tag{7}$$

$$\vec{\pi}_\Delta^k \cdot \vec{\alpha} \leq 0 \tag{8}$$

Notice that each vector $\vec{\pi}_\Delta^k$ has exactly m coordinates. Since we have $m+1$ such vectors (corresponding to the $m+1$ profiles in S), there must be a subset of vectors that is linearly dependent. We can therefore express one of the vectors as a linear combination of the others. Without loss of generality, we assume that the first profile's vector can be written as a combination of the others with parameters β_k, not all 0:

$$\vec{\pi}_\Delta^1 = \sum_{k=2}^{m+1} \beta_k \cdot \vec{\pi}_\Delta^k \tag{9}$$

Now, we shall construct our subset S_1 of preference profiles, on which $f_{\vec{\alpha}}$ agrees with h, as follows:

$$S_1 = \{k \in \{2, \ldots, m+1\} : \ \beta_k \geq 0\} \tag{10}$$

Suppose, by way of contradiction, that $f_{\vec{\alpha}}$ agrees with h on $\succ_k^N$ for $k \in S_1$, and with g on the rest. We shall examine the value of $\vec{\pi}_\Delta^1 \cdot \vec{\alpha}$:

$$\begin{aligned} \vec{\pi}_\Delta^1 \cdot \vec{\alpha} &= \sum_{k=2}^{m+1} \beta_k \cdot \vec{\pi}_\Delta^k \cdot \vec{\alpha} \\ &= \sum_{k \in S_1} \beta_k \cdot \vec{\pi}_\Delta^k \cdot \vec{\alpha} + \sum_{k \notin S_1 \cup \{1\}} \beta_k \cdot \vec{\pi}_\Delta^k \cdot \vec{\alpha} \\ &\geq 0 \end{aligned} \tag{11}$$

The last inequality is due to the construction of S_1—whenever β_k is negative, the sign of $\vec{\pi}_\Delta^k \cdot \vec{\alpha}$ is non-positive ($f_{\vec{\alpha}}$ agrees with g), and whenever β_k is positive, the sign of $\vec{\pi}_\Delta^k \cdot \vec{\alpha}$ is non-negative (agreement with h).

Therefore, by equation (5), we have that $f(\succ_1^N) \neq c_2 = g(\succ_1^N)$. However, it holds that $1 \notin S_1$, and we assumed that $f_{\vec{\alpha}}$ agrees with g outside S_1—this is a contradiction. □

Theorem 3.1 is thus proven. The upper bound on the generalized dimension of S_m^n is quite tight: in the next subsection we show a lower bound of $m-3$.

3.2 Lower Bound for the Generalized Dimension of $\mathcal{S}_m^n$

Theorem 2.6 implies that a lower bound on the generalized dimension of a function class is directly connected to the complexity of learning it. In particular, a tight bound on the dimension gives us an almost exact idea of the number of examples required to learn a scoring rule. Therefore, we wish to bound $D_G(\mathcal{S}_m^n)$ from below as well.

THEOREM 3.7. *For all $n \geq 4$, $m \geq 4$, $D_G(\mathcal{S}_m^n) \geq m-3$.*

PROOF. We shall produce an example set of size $m-3$ which is shattered by $\mathcal{S}_m^n$. Define a preference profile $\succ_l^N$, for $l = 3, \ldots, m-1$, as follows. For all l, the voters $1, \ldots, n-1$ rank candidate c_j in place j, i.e., they vote $c_1 \succ_l^i c_2 \succ_l^i \cdots \succ_l^i c_m$. The preferences $\succ_l^n$ (the preferences of voter n in profile $\succ_l^N$) are defined as follows: candidate 2 is ranked in place l, candidate 1 is ranked in place $l+1$; the other candidates are ranked arbitrarily by voter n. For example, if $m = 5$, $n = 6$, the preference profile $\succ_3^N$ is:

$\succ_3^1$	$\succ_3^2$	$\succ_3^3$	$\succ_3^4$	$\succ_3^5$	$\succ_3^6$
c_1	c_1	c_1	c_1	c_1	c_3
c_2	c_2	c_2	c_2	c_2	c_4
c_3	c_3	c_3	c_3	c_3	c_2
c_4	c_4	c_4	c_4	c_4	c_1
c_5	c_5	c_5	c_5	c_5	c_5

[4]There is some abuse of notation here; if $h(\succ_k^N) = c_l$ then by $\vec{\pi}_{h(\succ_k)}^k$ we mean $\vec{\pi}_l^k$.

[5]In all profiles except $\succ_1^N$, we are indifferent to the direction in which ties are broken.

LEMMA 3.8. *For any scoring rule $f_{\vec{\alpha}}$ with $\alpha_1 = \alpha_2 \geq 2\alpha_3$ it holds that:*

$$f_{\vec{\alpha}}(\succ_l^N) = \begin{cases} c_1 & \alpha_l = \alpha_{l+1} \\ c_2 & \alpha_l > \alpha_{l+1} \end{cases}$$

PROOF. We shall first verify that c_2 has maximal score. c_2's score is at least $(n-1)\alpha_2 = (n-1)\alpha_1$. Let $j \geq 3$; c_j's score is at most $(n-1)\alpha_3 + \alpha_1$. Thus, the difference is at least $(n-1)(\alpha_1 - \alpha_3) - \alpha_1$. Since $\alpha_1 = \alpha_2 \geq 2\alpha_3$, this is at least $(n-1)(\alpha_1/2) - \alpha_1 > 0$, where the last inequality holds for $n \geq 4$.

Now, under preference profile $\succ_l^N$, c_1's score is $(n-1)\alpha_1 + \alpha_{l+1}$ and c_2's score is $(n-1)\alpha_1 + \alpha_l$. If $\alpha_l = \alpha_{l+1}$, the two candidates have identical scores, but c_1 was ranked first by more voters (in fact, by $n-1$ voters), and thus the winner is c_1. If $\alpha_l > \alpha_{l+1}$, then c_2's score is strictly higher—hence in this case c_2 is the winner. □

Armed with Lemma 3.8, we will now prove that the set $\{\succ_l^N\}_{l=3}^{m-1}$ is shattered by $\mathcal{S}_m^n$. Let $\vec{\alpha}^1$ be such that $\alpha_1^1 = \alpha_2^1 \geq 2\alpha_3^1 = \alpha_4^1 = \cdots = \alpha_m^1$, and $\vec{\alpha}^2$ be such that $\alpha_1^1 = \alpha_2^1 \geq 2\alpha_3^1 > \alpha_4^1 > \cdots > \alpha_m^1$. By the lemma, for all $l = 3, \ldots, m-1$, $f_{\vec{\alpha}^1}(\succ_l^N) = c_1$, and $f_{\vec{\alpha}^2}(\succ_l^N) = c_2$.

Let $T \subseteq \{3, 4, \ldots, m-1\}$. We must show that there exists $\vec{\alpha}$ such that $f_{\vec{\alpha}}(\succ_l^N) = c_1$ for all $l \in T$, and $f_{\vec{\alpha}}(\succ_l^N) = c_2$ for all $l \notin T$. Indeed, configure the parameters such that $\alpha_1 = \alpha_2 > 2\alpha_3$, and $\alpha_l = \alpha_{l+1}$ iff $l \in T$. The result follows directly from Lemma 3.8.

4. ON LEARNING SCORING RULES "CLOSE" TO TARGET RULES

Heretofore, we have concentrated on trying to learn scoring rules. In particular, we have assumed that there are scoring rules that are consistent with given training sets. We have motivated our attention to this specific family of rules by noting that they are exactly the family of anonymous, neutral, and consistent rules, and demonstrating that it is possible to obtain a variety of properties by adjusting the parameters that define scoring rules.

In this section, we push the envelope by asking the following question. Given examples that are consistent with some general voting rule, is it possible to learn a scoring rule that is "close" to the target rule?

Mathematically, we are actually asking whether there exist target voting rules f^* such that $\min_{f_{\vec{\alpha}} \in \mathcal{S}_m^n} err(f_{\vec{\alpha}})$ is large. This of course depends on the underlying distribution D. In the rest of this section, the implicit assumption is that D is the simplest nontrivial distribution over profiles, namely the uniform distribution. Nevertheless, the uniform distribution usually does not reflect real preferences of voters; this is an assumption we are making for the sake of analysis. In light of this discussion, the definition of distance between voting rules is going to be the fraction of preference profiles on which the two rules disagree.

DEFINITION 4.1. A voting rule $f : \mathcal{L}^N \to C$ is a *c-approximation* of a voting rule g iff f and g agree on a c-fraction of the possible preference profiles:

$$|\{\succ^N \in \mathcal{L}^N : \ f(\succ^N) = g(\succ^N)\}| \geq c \cdot (m!)^n.$$

In other words, the question is: given a training set $\{(\succ_k^N, f(\succ_j^N)\}_k$, where $f : \mathcal{L}^N \to C$ is some voting rule, how hard is it to learn a scoring rule that c-approximates f, for c that is close to 1?

It turns out that the answer is: it is impossible. Indeed, there are voting rules that disagree with any scoring rule on almost all of the preference profiles; if the target rule f is such a rule, it is impossible to find, and of course impossible to learn, a scoring rule that is "close" to f.

For instance, consider the following voting rule that we call *flipped veto*: each voter awards one point to the candidate it ranks *last*; the winner is the candidate with the most points. This rule is of course not reasonable as a preference aggregation method, but still—it is a valid voting rule.

PROPOSITION 4.2. *Let $f_{\vec{\alpha}}$ be a scoring rule. Then $f_{\vec{\alpha}}$ is at most a $1/m$-approximation of flipped veto.*

PROOF. Let $\succ^N$ be a preference profile such that $f_{\vec{\alpha}}(\succ^N) = \text{flipped veto}(\succ^N) = c^*$, for some $c^* \in C$. Define a set $A_{\succ^N} \subseteq \mathcal{L}^N$ as follows: each profile in the set is obtained by switching the place of a candidate $c \in C$, $c \neq c^*$, with the place of c^*, in the ordering of each voter that did not rank c^* last.[6] For a preference profile $\succ_1^N \in A_{\succ^N}$ that was obtained by switching c with c^*, clearly the winner under flipped veto is still c^*, as this rule takes into account only candidates ranked last. In addition, under $f_{\vec{\alpha}}$, the score of c in $\succ_1^N$ is at least as large as the score of c^* in $\succ^N$ (voters that have not switched the two candidates are ones that rank c^* last, and the score of the other candidates remains unchanged—hence $f_{\vec{\alpha}}(\succ_1^N) = c$. It follows that for any preference profile in $A_{\succ^N}$, $f_{\vec{\alpha}}$ and flipped veto do not agree.

We claim that for any two preference profiles $\succ_1^N$ and $\succ_2^N$ on which $f_{\vec{\alpha}}$ and flipped veto agree, it holds that $A_{\succ_1^N} \cap A_{\succ_2^N} = \emptyset$. Indeed, assume that there exists $\succ^N \in A_{\succ_1^N} \cap A_{\succ_2^N}$. It cannot be the case that the same candidate was switched with c^* in order to obtain $\succ^N$ from both $\succ_1^N$ and $\succ_2^N$—that would imply $\succ_1^N$ and $\succ_2^N$ are identical. Therefore, assume w.l.o.g. that c_1 was switched with c^* in $\succ_1^N$ (only in the rankings of voters that did not rank c^* last), and c_2 was switched with c^* in $\succ_2^N$. But this means that both c_1 and c_2 are winners in $\succ^N$ (by the fact that c^* was a winner in both $\succ_1^N$ and $\succ_2^N$)—a contradiction.

In addition, in any two preference profiles $\succ_1^N$ and $\succ_2^N$ such that

$$f_{\vec{\alpha}}(\succ_1^N) = \text{flipped veto}(\succ_1^N) = c^*,$$

and

$$f_{\vec{\alpha}}(\succ_2^N) = \text{flipped veto}(\succ_2^N) = c^{**},$$

it holds that $A_{\succ_1^N} \cap A_{\succ_2^N} = \emptyset$, as flipped veto elects c^* in all profiles in $A_{\succ_1^N}$, but elects c^{**} in all profiles in $A_{\succ_2^N}$.

It follows that for every preference profile on which $f_{\vec{\alpha}}$ and flipped veto agree, there are at least $m-1$ distinct profiles on which the two voting rules disagree; this proves the proposition. □

Even more interestingly, it is possible to show that almost every voting rule cannot be approximated by scoring rules by a factor better than $1/2$.

THEOREM 4.3. *Let $\epsilon, \delta > 0$. For large enough values of n and m, at least a $(1-\delta)$-fraction of the voting rules F :*

[6] It cannot be the case that all voters ranked c^* last, by our tie-breaking assumption.

$\mathcal{L}^n \to \{c_1, \ldots, c_m\}$ *satisfy the following property: no scoring rule in* $\mathcal{S}^n_m$ *is a* $(1/2+\epsilon)$*-approximation of* F*.*

REMARK 4.4. Proposition 4.2 can seemingly be circumvented by removing the requirement that in a scoring rule defined by a vector $\vec{\alpha}$, $\alpha_l \geq \alpha_{l+1}$ for all l. Indeed, flipped veto is essentially a scoring rule with $\alpha_m = 1$ and $\alpha_l = 0$ for all $l \neq m$. However, the constant voting rule which always elects the same candidate seems to have the same inapproximability ratio, even when this property of scoring rules is not taken into account. Moreover, Theorem 4.3 also holds when scoring rules are not assumed to satisfy this property.

In order to prove the theorem, we require the following lemma, which may be of independent interest.

LEMMA 4.5. *There exists a polynomial* $p(n,m)$ *such that for all* $n, m \in \mathbb{N}$, $|\mathcal{S}^n_m| \leq 2^{p(n,m)}$.

PROOF. It is true that there are an infinite number of ways to choose the vector $\vec{\alpha}$ that defines a scoring rule. Nevertheless, what we are really interested in is the number of *distinct* voting rules. For instance, if $\vec{\alpha}^1 = 2\vec{\alpha}^2$, then $f_{\vec{\alpha}^1} \equiv f_{\vec{\alpha}^2}$, i.e., the two vectors define the same voting rule.

It is clear that two scoring rules $f_{\vec{\alpha}^1}$ and $f_{\vec{\alpha}^2}$ are distinct only if the following condition holds: there exist two candidates $c_{j_1}, c_{j_2} \in C$, and a preference profile $\succ^N$, such that $f_{\vec{\alpha}^1}(\succ^N) = c_{j_1}$ and $f_{\vec{\alpha}^2}(\succ^N) = c_{j_2}$. This holds only if there exist two candidates c_{j_1} and c_{j_2} and a preference profile $\succ^N$ such that under α^1, c_{j_1}'s score is strictly greater than c_{j_2}'s, and under α_2, either c_{j_2}'s score is greater or the two candidates are tied, and the tie is broken in favor of c_{j_2}.

Now, assume $\succ^N$ induces rankings $\vec{\pi}_{j_1}$ and $\vec{\pi}_{j_2}$. The conditions above can be written as

$$\sum_l \pi_{j_1,l}\alpha^1_l > \sum_l \pi_{j_2,l}\alpha^1_l, \tag{12}$$

$$\sum_l \pi_{j_1,l}\alpha^2_l \leq \sum_l \pi_{j_2,l}\alpha^2_l, \tag{13}$$

where the inequality is an equality only if ties are broken in favor of c_{j_2}, i.e., if $l_0 = \min\{l : \pi_{j_1,l} \neq \pi_{j_2,l}\}$, then $\pi_{j_1,l} < \pi_{j_2,l}$.[7]

Let $\vec{\pi}_\Delta = \vec{\pi}_{j_1} - \vec{\pi}_{j_2}$. As in the proof of Lemma 3.6, equations (12) and (13) can be concisely rewritten as

$$\vec{\pi}_\Delta \cdot \vec{\alpha}^1 > 0 \geq \vec{\pi}_\Delta \cdot \vec{\alpha}^2, \tag{14}$$

where the inequality is an equality only if the first nonzero position in $\vec{\pi}_\Delta$ is negative.

In order to continue, we opt to reinterpret the above discussion geometrically. Each point in $\mathbb{R}^m$ corresponds to a possible choice of parameters $\vec{\alpha}$. Now, each possible choice of $\vec{\pi}_\Delta$ is the normal to a hyperplane. These hyperplanes partition the space into cells: the vectors in the interior of each cell agree on the signs of dot products with all vectors $\vec{\pi}_\Delta$. More formally, if $\vec{\alpha}_1$ and $\vec{\alpha}_2$ are two points in the interior of a cell, then for any vector $\vec{\pi}_\Delta$, $\vec{\pi}_\Delta \cdot \vec{\alpha}^1 > 0 \Leftrightarrow \vec{\pi}_\Delta \cdot \vec{\alpha}^2 > 0$. By equation (14), this implies that any two scoring rules $f_{\vec{\alpha}^1}$ and $f_{\vec{\alpha}^2}$, where $\vec{\alpha}^1$ and $\vec{\alpha}^2$ are in the interior of the same cell, are identical.

[7]W.l.o.g. we disregard the case where $\vec{\pi}_{j_1} = \vec{\pi}_{j_2}$; the reader can verify that taking this case into account multiplies the final result by an exponential factor at most.

What about points residing in the intersection of several cells? These vectors always agree with the vectors in one of the cells, as ties are broken according to rankings induced by the preference profile, i.e., according to the parameters that define our hyperplanes. Therefore, the points in the intersection can be conceptually annexed to one of the cells.

So, we have reached the conclusion that the number of distinct scoring rules is at most the number of cells. Hence, it is enough to bound the number of cells; we claim this number is exponential in n and m. Indeed, each $\vec{\pi}_\Delta$ is an m-vector, in which every coordinate is an integer in the set $\{-n, -n+1, \ldots, n-1, n\}$. It follows that there are at most $(2n+1)^m$ possible hyperplanes. It is known [6] that given k hyperplanes in d-dimensional space, the number of cells is at most $O(k^d)$. In our case, $k \leq (2n+1)^m$ and $d = m$, so we have obtained a bound of:

$$((2n+1)^m)^m \leq (3n)^{m^2} = \left(2^{\log 3n}\right)^{m^2} = 2^{m^2 \log 3n}. \tag{15}$$

□

REMARK 4.6. This lemma implies, according to Lemma 2.4, that there exists a polynomial $p(n,m)$ such that for all $n, m \in \mathbb{N}$, $D_G(\mathcal{S}^n_m) \leq p(n,m)$. However, we have already obtained a tighter upper bound of m.

PROOF OF THEOREM 4.3. We will surround each scoring rule $f_{\vec{\alpha}} \in \mathcal{S}^n_m$ with a "ball" $B(\vec{\alpha})$, which contains all the voting rules for which $f_{\vec{\alpha}}$ is a $(1/2+\epsilon)$-approximation. We will then show that the union of all these balls covers at most a δ-fraction of the set of the space of voting rules. This implies that for at least a $(1-\delta)$-fraction of the voting rules, no scoring rule is a $(1/2+\epsilon)$-approximation.

For a given $\vec{\alpha}$, what is the size of $B(\vec{\alpha})$? As there are $(m!)^n$ possible preference profiles, the ball contains rules that do not agree with $f_{\vec{\alpha}}$ on at most $(1/2-\epsilon)(m!)^n$ preference profiles. For a profile on which there is disagreement, there are m options to set the image under the disagreeing rule.[8] Therefore,

$$|B(\vec{\alpha})| \leq \binom{(m!)^n}{(1/2-\epsilon)(m!)^n} m^{(1/2-\epsilon)(m!)^n}. \tag{16}$$

How large is this expression? Let $B'(\vec{\alpha})$ be the set of all voting rules that disagree with $f_{\vec{\alpha}}$ on *exactly* $(1/2+\epsilon)(m!)^n$ preference profiles. It holds that

$$\begin{aligned} |B'(\vec{\alpha})| &= \binom{(m!)^n}{(1/2+\epsilon)(m!)^n}(m-1)^{(1/2+\epsilon)(m!)^n} \\ &= \binom{(m!)^n}{(1/2-\epsilon)(m!)^n}((m-1)^{1+2\epsilon})^{1/2(m!)^n} \\ &\geq \binom{(m!)^n}{(1/2-\epsilon)(m!)^n} m^{1/2(m!)^n}, \end{aligned} \tag{17}$$

where the last inequality holds for a large enough m. But since the total number of voting rules, $m^{(m!)^n}$, is greater than the number of rules in $B'(\vec{\alpha})$, we have:

$$\begin{aligned} \frac{m^{(m!)^n}}{B(\vec{\alpha})} \geq \frac{B'(\vec{\alpha})}{B(\vec{\alpha})} &\geq \frac{\binom{(m!)^n}{(1/2-\epsilon)(m!)^n} m^{1/2(m!)^n}}{\binom{(m!)^n}{(1/2-\epsilon)(m!)^n} m^{(1/2-\epsilon)(m!)^n}} \\ &= m^{\epsilon(m!)^n}. \end{aligned} \tag{18}$$

[8]This way, we also take into account voting rules that agree with $f_{\vec{\alpha}}$ on more than $(1/2+\epsilon)(m!)^n$ profiles.

Therefore

$$B(\vec{\alpha}) \leq \frac{m^{(m!)^n}}{m^{\epsilon(m!)^n}} = m^{(1-\epsilon)(m!)^n}. \quad (19)$$

If the union of balls is to cover at least a δ-fraction of the set of voting rules, we must have $|\mathcal{S}_m^n| \cdot m^{(1-\epsilon)(m!)^n} \geq \delta \cdot m^{(m!)^n}$; equivalently, it must hold that $|\mathcal{S}_m^n| \geq \delta \cdot m^{\epsilon(m!)^n}$. However, Lemma 4.5 implies that $|\mathcal{S}_m^n|$ is exponential in n and m, so for large enough values of n and m, the above condition does not hold. □

5. CONCLUSIONS

We have shown that the class of scoring rules is efficiently learnable in the PAC model. We have argued that, given a black box specification of the choice criteria of the society, learning from examples allows one to efficiently (albeit approximately) design such a rule. The black box reflects some ideal voting rule the designer has in mind, which satisfies, for instance, different desirable properties. The designer thus essentially translates a cumbersome representation of a voting rule (hidden within the black box) to a concisely represented voting rule which is easy to understand and apply.

We mentioned that scoring rules can capture a wide variety of properties. However, in Section 4 we explored extending our approach, and showed that many voting rules cannot be approximated using scoring rules. However, this negative result relied implicitly on assuming a uniform distribution over profiles. Moreover, it might be the case that some of the important families of voting rules can be approximated by scoring rules. Therefore, we do not rule out at this point the application of our approach to designing general voting rules by directly learning scoring rules which approximate them.

Comparing our learning-theoretic results with those obtained in the context of learning voting trees [15], we conclude that scoring rules are far superior to voting trees with respect to the approach of automated design of voting rules. Indeed, in general, learning voting trees entails a training set of exponential size. Furthermore, finding a voting tree that is consistent with the given training set is $\mathcal{NP}$-hard.

6. ACKNOWLEDGMENT

This work was partially supported by Israel Science Foundation grant #898/05. Ariel Procaccia is supported by the Adams Fellowship Program of the Israel Academy of Sciences and Humanities.

7. REFERENCES

[1] E. Beigman and R. Vohra. Learning from revealed preference. In *Proceedings of the Seventh ACM Conference on Electronic Commerce*, pages 36–42, 2006.

[2] V. Conitzer and T. Sandholm. Complexity of mechanism design. In *Proceedings of the Eighteenth Annual Conference on Uncertainty in Artificial Intelligence*, pages 103–110, 2002.

[3] V. Conitzer and T. Sandholm. An algorithm for automatically designing deterministic mechanisms without payments. In *Proceedings of the Third International Joint Conference on Autonomous Agents and Multiagent Systems*, pages 128–135, 2004.

[4] V. Conitzer and T. Sandholm. Communication complexity of common voting rules. In *Proceedings of the Sixth ACM Conference on Electronic Commerce*, pages 78–87, 2005.

[5] V. Conitzer, T. Sandholm, and J. Lang. When are elections with few candidates hard to manipulate? *Journal of the ACM*, 54:1–33, 2007.

[6] H. Edelsbrunner. *Algorithms in Combinatorial Geometry*, volume 10 of *EATCS Monographs on Theoretical Computer Science*. Springer, 1987.

[7] S. Ghosh, M. Mundhe, K. Hernandez, and S. Sen. Voting for movies: the anatomy of a recommender system. In *Proceedings of the Third Annual Conference on Autonomous Agents*, pages 434–435, 1999.

[8] T. Haynes, S. Sen, N. Arora, and R. Nadella. An automated meeting scheduling system that utilizes user preferences. In *Proceedings of the First International Conference on Autonomous Agents*, pages 308–315, 1997.

[9] G. Kalai. Learnability and rationality of choice. *Journal of Economic Theory*, 113(1):104–117, 2003.

[10] S. Lahaie and D. C. Parkes. Applying learning algorithms to preference elicitation. In *Proceedings of the Fifth ACM Conference on Electronic Commerce*, pages 180–188, 2004.

[11] N. Littlestone. Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. *Machine Learning*, 2:285–318, 1988.

[12] B. K. Natarajan. *Machine Learning: A Theoretical Approach*. Morgan Kaufmann, 1991.

[13] A. D. Procaccia and J. S. Rosenschein. Junta distributions and the average-case complexity of manipulating elections. *Journal of Artificial Intelligence Research*, 28:157–181, February 2007.

[14] A. D. Procaccia, J. S. Rosenschein, and G. A. Kaminka. On the robustness of preference aggregation in noisy environments. In *The Sixth International Joint Conference on Autonomous Agents and Multiagent Systems (AAMAS 2007)*, pages 416–422, Honolulu, Hawaii, May 2007.

[15] A. D. Procaccia, A. Zohar, J. Peleg, and J. S. Rosenschein. Learning voting trees. In *Proceedings of the Twenty-Second Conference on Artificial Intelligence (AAAI 2007)*, pages 110–115, 2007.

[16] R. J. Vanderbei. *Linear Programming: Foundations and Extensions*. Springer, 2nd edition, 2001.

[17] H. P. Young. Social choice scoring functions. *SIAM Journal of Applied Mathematics*, 28(4), 1975.