Efficient Approximation Algorithms for the Inverse Semivalue Problem

Ilias Diakonikolas
University of Wisconsin-Madison
Madison, United States
ilias@cs.wisc.edu

Chrystalla Pavlou
TurinTech AI
London, United Kingdom
chrystalla@turintech.ai

John Peebles
Princeton University
Princeton, United States
johnpeeb@gmail.com

Alistair Stewart
Web 3 Foundation
Zug, Switzerland
stewart.al@gmail.com

ABSTRACT

Weighted voting games are typically used to model situations where a number of agents vote against or for a proposal. In such games, a proposal is accepted if a weighted sum of the votes exceeds a specified threshold. As the influence of a player over the outcome is not in general proportional to her assigned weight, various power indices have been proposed to measure each player's influence. The inverse power index problem is the problem of designing a weighted voting game that achieves a set of desirable influences as they are measured by a predefined power index. Recent work has shown that exactly solving the inverse power index problem is computationally intractable when the power index is in the class of semivalues — a broad class that includes the popular Shapley value and Banzhaf index. In this work, we design efficient approximation algorithms for the inverse semivalue problem. We develop a unified methodology that leads to computationally efficient algorithms that solve the inverse semivalue problem to any desired accuracy. We perform an extensive experimental evaluation of our algorithms on both synthetic and real inputs. Our experiments show that our algorithms are scalable and achieve higher accuracy compared to previous methods in the literature.

KEYWORDS

Voting Theory; Social Choice Theory; Cooperative Games; Shapley Value; Banzhaf Index; Power index

ACM Reference Format:
Ilias Diakonikolas, Chrystalla Pavlou, John Peebles, and Alistair Stewart. 2022. Efficient Approximation Algorithms for the Inverse Semivalue Problem. In *Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), Online, May 9–13, 2022*, IFAAMAS, 9 pages.

1 INTRODUCTION

1.1 Background and Motivation

Weighted voting games are multiplayer cooperative games modeling the following scenario: There are n binary voters, each associated with a non-negative weight, and the voting outcome is affirmative if and only if the total weight of the affirmative voters exceeds a specified threshold. Weighted voting games have been extensively studied in cooperative game theory, social choice, and voting theory over the course of several decades. In addition to their fundamental importance, these games arise in a number of practical scenarios, including the voting system of the European Union, stockholder companies, and resource allocation in multi-agent systems [5, 8, 14].

Given a weighted voting game, a natural question is how to measure the power or influence of a player on the result of the game. Interestingly enough, the power of a player is not always proportional to her weight. In order to systematically quantify the influence of a player, a number of *power indices* have been proposed and studied in the literature, including the Shapley value (or Shapley-Shubik index) [33, 34], the Banzhaf index [4], the broader class of semivalues [12, 36], and others. The (forward) problem of computing the players' power indices in a given game has received significant attention and its computational complexity has been characterized in many settings (see, e.g., [2, 10, 31]).

In this work, we study the *inverse power index problem* with a focus on the class of *semivalues*: Given a set of target semivalues, design a weighted voting game (i.e., assign appropriate weights to the players) with this set of semivalues (if such a game exists). As the proceeding discussion will explain, this problem has been extensively studied in social choice theory and learning theory. Recent work [11] established that the *exact version* of the inverse semivalue problem – i.e., designing a weighted voting game whose semivalues are exactly equal to the target ones – is computationally intractable. Notably, the aforementioned hardness result does not rule out the existence of efficient *approximation* algorithms for the inverse problem.

1.2 Our Contributions

The main contribution of this paper is a simple and unified approach that yields computationally efficient approximation algorithms for the inverse semivalue problem. Prior to this work, no efficient approximation algorithms were known for this general class of power indices. We note that efficient approximation algorithms [6, 7] were previously designed for Banzhaf indices and Shapley values, but it was not clear how to extend these previous algorithms to the broader class of semivalues. Here we develop a general methodology that applies for *any* semivalue. Moreover, even for the Banzhaf and Shapley special cases, our algorithms are faster and more accurate

Proc. of the 21st International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2022), P. Faliszewski, V. Mascardi, C. Pelachaud, M.E. Taylor (eds.), May 9–13, 2022, Online.

compared to existing algorithms from the literature. In addition to their performance guarantees, we show that our algorithms are scalable to large number of voters n and perform very well on a range of both synthetic and real datasets.

We now provide a high-level overview of our algorithmic approach. The main difficulty in designing an efficient algorithm for the inverse semivalues problem is its non-convexity. In fact, given the hardness result of [11], it is unlikely that there exists an exact efficient algorithm. Our main insight is that one can obtain efficient *approximation* algorithms by considering an appropriate convex relaxation of the problem. This novel convex relaxation that we formulate is a key conceptual contribution of our work.

In more detail, we proceed as follows: First, instead of working with the target semivalues directly, our algorithm works with an essentially equivalent set of parameters, which we call *correlation coefficients* (Definition 3). Our goal then becomes to design a weighted voting game with a given set of correlation coefficients. This problem is equivalent to the problem we started with, hence still computationally hard. We instead consider a convex relaxation of the latter problem, where the goal is to find a *Linear Bounded Function (LBF)* (Definition 4) whose vector of correlation coefficients is equal to the target vector. It turns out that the latter task can be formulated as a convex optimization problem. While we do not know how to solve this convex optimization problem exactly (even evaluating the objective function or its gradient exactly is $\sharp P$-hard), we can use first-order methods to approximate the optimal solution to any desired accuracy. Our algorithm for this task runs in time polynomial in the number of players, n, and $1/\epsilon$, where ϵ is the accuracy parameter. At the end of this process, we output the weighted voting game with the corresponding weights.

Interestingly, the goal of our convex relaxation is not to find a solution with near-optimal objective value, but rather an *approximate stationary point*, i.e., a point with small gradient. Indeed, the gradient of our objective is $\widehat{f_w} - c$, where c is the target vector and $\widehat{f_w}$ is the vector of correlation coefficients of the current hypothesis (Proposition 8). We show that we can use first-order methods, e.g., (stochastic) gradient descent to find an approximate stationary point in polynomial time. An issue that arises is that we do not have access to exact gradients, but we show that an approximate gradient oracle (or a stochastic gradient) suffices for our purposes. In general, the smoothness parameter of our objective is quite large and can be proportional to n, the number of players. As a result, the number of iterations of gradient descent will scale with n. To overcome this difficulty, we develop a more sophisticated algorithm with faster runtime. To achieve this, we leverage the fact that the bad smoothness parameter exists due to a single direction, while in the orthogonal complement the function is 1-smooth.

1.3 Related Work

Several heuristic algorithms for the inverse Banzhaf index problem have been proposed in the literature. Aziz, Patterson, and Leech [3] proposed an algorithm that given target Banzhaf indices and a desired ℓ_2-distance bound, outputs a weighted voting game that has Banzhaf indices within the desired distance. Their algorithm starts with an initial real weight vector, and iteratively updates the weights by interpolating a best fit curve. Unfortunately, no theoretical guarantees are provided regarding the convergence of this method. In fact, it is not difficult to construct simple examples where their algorithm converges to a solution that is far from the optimal solution (see full version of the paper). Heuristic algorithms for the same problem were also proposed by Laruelle and Widgren [24] and Leech [28]. The algorithm of [24] was evaluated on an EU power distribution dataset in [24] and [35]. More recently, an experimental evaluation was performed by Nijs and Wilmer [9], where it was shown that there are inputs where the algorithm does not perform well. The algorithm from [28] was applied on IMF [26] and EU [25] datasets, again without any performance guarantees.

Fatima, Wooldridge, and Jennings [17] gave an iterative approximation algorithm for the inverse Shapley value problem that given target Shapley values, a quota, and a desired average percentage difference, outputs a weighted voting game with the given quota that has Shapley values within the desired distance. While it is shown that each iteration runs in quadratic time and that the algorithm eventually converges, no theoretical guarantees are given regarding the convergence rate.

We also note that an exact algorithm for both the inverse Banzhaf index and inverse Shapley value problems was given by Kurz [21]. Unfortunately, the algorithm has runtime exponential in n, as it relies on integer linear programming. As expected, the algorithm is slow in practice even for small weighted voting games: For example, the algorithm takes 55 minutes even for simple instances with 15 players. Another exact and exponential-time algorithm is given by Keijzer, Klos, and Zhang [8]. This algorithm is based on an enumeration of all weighted voting games with n players.

The inverse power index problem is of significant interest in other fields, including computational complexity and learning theory, where the inverse Banzhaf index problem is known as the "Chow parameters problem". The Chow parameters of a linear threshold function are equivalent to the non-normalized Banzhaf indices [13] of the corresponding weighted voting game, and therefore the inverse Banzhaf index problem is tantamount to the Chow parameters problem. Recent works [6, 7, 30] have obtained polynomial time approximation algorithms with provable performance guarantees for the inverse problem with respect to the Banzhaf indices and the Shapley values.

Finally, a number of complexity results have been established concerning weighted voting games, see, e.g., [2, 10, 14–16]. Most relevant to this paper is the work of [11] which established that the exact version of the inverse semivalue problem is computationally intractable. A related line of work [1, 23] examines which non-negative vectors can be well approximated by various power index vectors of simple games.

2 PRELIMINARIES

Notation. We write $wt(x)$ to denote the *weight* of a Boolean vector $x \in \{-1, 1\}^n$, i.e., the number of 1's in x. We use $\text{sign} : \mathbb{R} \to \mathbb{R}$ for the function that takes value 1 if $z \geq 0$ and value -1 if $z < 0$. We will denote by $P_1 : \mathbb{R} \to [-1, 1]$ the function defined as $P_1(a) = a$ if $|a| \leq 1$ and $P_1(a) = \text{sign}(a)$, otherwise. We denote by $\mathrm{I} : \mathbb{R} \to \mathbb{R}$ the function that takes value 1 if $-1 \leq a \leq 1$ and value 0 otherwise. We will denote by q the function $q : \mathbb{R} \to \mathbb{R}^+$ defined

as $q(a) = \begin{cases} |a| & a < -1 \text{ or } a > 1 \\ \frac{1+a^2}{2} & -1 \le a \le 1 \end{cases}$. We use $\mathbf{I}$ to denote the identity matrix and $\preceq$ to denote the standard Löwner ordering on matrices. We say a function is β-smooth if its gradient is β-Lipschitz.

Our object of study are linear threshold functions (LTFs).

Definition 1 (Linear Threshold Function)**.** *A linear threshold function (LTF) is any function $h_{w,\theta} : \{-1,1\}^n \to \{-1,1\}$ such that $h_{w,\theta}(x) = \text{sign}(w \cdot x - \theta)$ for some weight vector $w \in \mathbb{R}^n$ and threshold $\theta \in \mathbb{R}$.*

Note that weighted voting games are equivalent to LTFs with non-negative weights.

Semivalues. We start with the definition of semivalues [32] in terms of weighting coefficients, as characterized in [12].

Definition 2 (Semivalues)**.** *For a positive integer n, a probability vector $p^n = (p^n_0, \ldots, p^n_{n-1}) \in \mathbb{R}^n$ is a vector such that $\sum_{t=0}^{n-1} \binom{n-1}{t} p^n_t = 1$ and $p^n_t \ge 0$, for $t \in \{0, \ldots, n-1\}$. The i-th semivalue, $i \in [n]$, corresponding to the probability vector p^n of a function $f : \{-1,1\}^n \to \{-1,1\}$ is*

$$\widetilde{f}^{p^n}(i) = \sum_{x \in \{-1,1\}^n : x_i = -1} p^n_{wt(x)} f(x) x_i + \sum_{x \in \{-1,1\}^n : x_i = 1} p^n_{wt(x)-1} f(x) x_i \,.$$

Setting $p^n_{-1} = p^n_n = 0$, the vector of semivalues can be reformulated as follows:

$$\widetilde{f}^{p^n}(i) = \frac{1}{2} \sum_{x \in \{-1,1\}^n} f(x) x_i \left(p^n_{wt(x)} + p^n_{wt(x)-1} \right) + \frac{1}{2} \sum_{x \in \{-1,1\}^n} f(x) \left(p^n_{wt(x)-1} - p^n_{wt(x)} \right) \,. \quad (1)$$

Definition 3 (Correlation Coefficient)**.** *For a Boolean function $f : \{-1,1\}^n \to \{-1,1\}$, we will write*

$$\widehat{f}^{p^n}(i) := \sum_{x \in \{-1,1\}^n} \mu_{p^n}(x) f(x) x_i$$

for the normalized first term of (1), where

$$\mu_{p^n}(x) := \frac{p^n_{wt(x)} + p^n_{wt(x)-1}}{\sum_{x \in \{-1,1\}^n} p^n_{wt(x)} + p^n_{wt(x)-1}} \,.$$

We note that the vector of correlation coefficients of a function f is essentially equivalent to its vector of semivalues. It is convenient for our algorithms to work with the vector of correlation coefficients instead. Correlation coefficients for the special case of Shapley values were also used in [7].

Special Cases. The Shapley values and Banzhaf indices are the semivalues defined by $p^n_t = \frac{(n-t-1)!t!}{n!}$ [34] and $p^n_t = \frac{1}{2^{n-1}}$ [13], respectively. We also study another important family of semivalues, the binomial semivalues, defined by $p^n_t = p^t(1-p)^{n-t-1}$ [19], where $p \in [0,1]$.

Inverse Semivalue Problem. Let n denote the number of players and consider the semivalues defined by a known probability vector p^n. Given a vector $c = (c_1, \ldots, c_n)$ of target semivalues with the promise that some LTF has these semivalues, the aim is to find such an LTF.

Name: SV_{p^n}-Inverse Problem

Input: Vector $(c_1, \ldots, c_n) \in \mathbb{Q}^n$ and threshold $\theta \in \mathbb{Q}$ such that there exists a $v \in \mathbb{Q}^n$ with $\widetilde{h}^{p^n}_{v,\theta}(i) = c_i$, for $i \in [n]$.

Goal: Output $w \in \mathbb{Q}^n$ such that $\widetilde{h}^{p^n}_{w,\theta}(i) = c_i$, for $i \in [n]$.

For ease of notation, in the rest of the paper, we denote by $\widehat{f}$ the $\widehat{f}^{p^n}$ parameters, by $\widetilde{f}$ the $\widetilde{f}^{p^n}$ parameters, and by μ the distribution μ_{p^n}.

3 WARMUP: INVERSE BANZHAF PROBLEM

We start with the inverse Banzhaf problem – the special case of the SV_{p^n}-Inverse problem where p^n is the uniform distribution on $\{\pm 1\}^n$.

Optimization Formulation. A difficulty which arises when solving the inverse Banzhaf problem is that LTFs are discontinuous, which makes them difficult to optimize over. As such, we follow prior work on this problem, which first relaxes the problem [6] as follows: Instead of trying to find an LTF with given Banzhaf indices, one instead searches for a linear *bounded* function with these Banzhaf indices. Linear bounded functions are a class of functions that generalize LTFs.

Definition 4 (Linear Bounded Function)**.** *A linear bounded function (LBF) is any function $f_{w,\theta} : \{-1,1\}^n \to [-1,1]$ such that $f_{w,\theta}(x) = P_1(w \cdot x - \theta)$ for some weight vector $w \in \mathbb{R}^n$ and threshold $\theta \in \mathbb{R}$.*

To see that LBFs generalize LTFs, note that any LTF with weight vector w can be written as an LBF by taking an LBF with a weight vector that is a sufficiently large scalar multiple of w. We now give the statement of the inverse Banzhaf problem for LBFs.

Inverse Banzhaf Problem for LBFs
Input: Vector $c = (c_1, \ldots, c_n) \in \mathbb{Q}^n$ and $\theta \in \mathbb{Q}$ such that there exists a $v \in \mathbb{Q}^n$ with $\widetilde{f}_{v,\theta}(i) = c_i$ for $i \in [n]$.
Output: A weight vector $w = (w_1, \ldots, w_n) \in \mathbb{Q}^n$ for an LBF f such that $\widetilde{f}_{w,\theta}(i) = c_i$ for $i \in [n]$.

We show that this problem can be reformulated as the problem of minimizing the following convex function. In fact, we show the stronger result that the ℓ_2-norm of the gradient of the function on the weights w below is equal to the ℓ_2 distance of the Banzhaf indices $\widetilde{f}_{w,\theta}$ of the LBF f with weights w from the desired parameters c. For simplicity, we consider the case where the threshold θ is set to 0 and we defer the general case to the full version of the paper.

Proposition 5. Given a vector c of target Banzhaf indices, consider the convex and 1-smooth function $g : \mathbb{R}^n \to \mathbb{R}$ defined as $g(w) = \mathbb{E}_{x \sim \{-1,1\}^n}[q(w \cdot x)] - w \cdot c$. Then for the LBF $f_{w,0} = P_1(w \cdot x)$, we have $\|\nabla g(w)\|_2 = \|\widetilde{f}_{w,0} - c\|_2$.

Proof. For $l \in \{1, \ldots, n\}$, we have:

$$\frac{\partial g}{\partial w_l} = \mathbb{E}_{x \sim \{-1,1\}^n}[P_1(w \cdot x) x_l] - c_l = \widetilde{f}_{w,0}(l) - c_l,$$

and so $\nabla g(w) = \widetilde{f}_{w,0} - c$.

We now prove smoothness. For $l, j \in \{1, .., n\}$, we have:

$$\frac{\partial^2 g}{\partial w_l \partial w_j} = \mathbb{E}_{x \sim \{-1,1\}^n}[I(w \cdot x) x_l x_j].$$

Thus, the Hessian matrix of g is the following:

$$\mathbf{H}_g(w) = \frac{1}{2^n} \sum_{x \in \{-1,1\}^n} I(w \cdot x)\ xx^\top.$$

Since for any $x \in \{-1,1\}^n$, it holds that $0 \le I(w \cdot x) \le 1$, we get that

$$\mathbf{H}_g(w) \preceq \frac{1}{2^n} \sum_{x \in \{-1,1\}^n} xx^\top = \mathbf{I}.$$

This implies that g is 1-smooth. □

Optimization via (Stochastic) Gradient Descent. Assuming an exact first-order oracle for g, we may solve the inverse Banzhaf problem for LBFs by minimizing g using the standard gradient descent method. Specifically, as g is 1-smooth, for any $0 < \epsilon < 1$, we can find a $w \in \mathbb{R}^n$ such that $\|\nabla g(w)\|_2 \le \epsilon$, i.e., an LBF $f_{w,0}$ such that $\|\widetilde{f}_{w,0} - c\|_2 \le \epsilon$, in $O(1/\epsilon^2)$ iterations by the standard analysis of gradient descent for smooth functions.

However, since computing the Banzhaf indices of an LTF is #P-hard [31], we cannot have an efficient first-order oracle. Instead, we can approximately solve the inverse Banzhaf problem in $O(1/\epsilon^2)$ iterations given access to an *approximate* first-order oracle via a folklore analysis (see full version).

Fact 6. For any $\epsilon > 0$, given access to ϵ-additive approximate gradients, an LBF $f_{w,0}$ with weight vector $w \in \mathbb{R}^n$ such that $\|\widetilde{f}_{w,0} - c\|_2 \le O(\epsilon)$ can be computed with $O(1/\epsilon^2)$ iterations of gradient descent. One can implement the gradient oracle in time $O(n^2 \log(n)/\epsilon^2)$.

As shown in the full version, standard gradient descent succeeds even given access to an ϵ-additive approximate (as opposed to exact) gradient oracle, which itself can be implemented using standard sampling techniques. Specifically, each gradient oracle call uses $O(n \log(n)/\epsilon^2)$ samples and runs in time $O(n^2 \log(n)/\epsilon^2)$. We note that instead of gradient descent with approximate gradients, one can use stochastic gradient descent (SGD). In this case, each stochastic gradient oracle call uses only one sample and $O(n/\epsilon^4)$ iterations suffice to get an ϵ-approximate solution.

The following theorem summarizes the main result of this section for the inverse Banzhaf problem for LBFs.

Theorem 7. *Given a vector of desired Banhzaf indices c, we can efficiently compute a weight vector w such that the LBF $f_{w,0}$ has $\widetilde{f}_{w,0}$ parameters that approximate c within ℓ_2 error ϵ. The algorithm runs in time $\tilde{O}\left(n^2/\epsilon^4\right)$, where the $\tilde{O}()$ hides logarithmic factors in its argument.*

4 INVERSE SEMIVALUES PROBLEM

In this section, we describe our algorithms for the general inverse semivalues problem.

Optimization Formulation. We solve the SV_{p^n}-Inverse problem by solving an essentially equivalent problem, which we term the "inverse correlation coefficients problem" or "inverse $\widehat{f}$ problem", where $\widehat{f}$ are the correlation coefficients (Definition 3). For the special case of Banzhaf indices, the correlation coefficients are equal to the Banzhaf indices.

For the problem of matching correlation coefficients of LBFs for distributions other than the uniform distribution, one might attempt the same approach as in the previous section: Formulate the same convex optimization problem and approximately minimize it. Unfortunately, for arbitrary distributions, the resulting convex optimization is not 1-smooth in general. Indeed, we will see that the smoothness parameter could potentially be as bad as $O(n)$. We nonetheless obtain the same runtime as before (up to logarithmic factors). We achieve this by first showing that the problem only has undesirable smoothness in a single direction. Then, we design an algorithm whose runtime only depends logarithmically on the smoothness in this bad direction.

We now define the inverse $\widehat{f}$ problem for LBFs and its convex formulation.

Inverse $\widehat{f}$ Parameters Problem for LBFs
Input: Vector $c = (c_1, \ldots, c_n) \in \mathbb{Q}^n$ and threshold $\theta \in \mathbb{Q}$ such that there exists a $v \in \mathbb{Q}^n$ with $\widehat{f}_{v,\theta}(i) = c_i$, for $i \in [n]$.
Output: A weight vector $w = (w_1, \ldots, w_n) \in \mathbb{Q}^n$ for an LBF f such that $\widehat{f}_{w,\theta}(i) = c_i$ for $i \in [n]$.

Similarly with the inverse Banzhaf problem, the inverse $\widehat{f}$ problem can be formulated as the problem of minimizing the convex function g', where g' is defined in the same way as g except that the expectation term is with respect to the distribution μ (instead of the uniform distribution).

Proposition 8. Given a vector c of target $\widehat{f}$ parameters, consider the convex and n-smooth function $g' : \mathbb{R}^n \to \mathbb{R}$ defined by $g'(w) = \mathbb{E}_{x \sim \mu}[q(w \cdot x)] - w \cdot c$. Then for the LBF $\widehat{f}_{w,0}$ with weights w, $\|\nabla g'(w)\|_2 = \|\widehat{f}_{w,0} - c\|_2$.

Proof. The analysis of the gradient is identical to that of the Banzhaf indices case. For the Hessian, we have that

$$\mathbf{H}_{g'}(w) = \sum_{x \in \{-1,1\}^n} \mu(x) I(w \cdot x)\ xx^\top.$$

Observing that $\mathrm{trace}(\mathbf{H}_{g'}(w)) = n$, we get that $\lambda_{max}(\mathbf{H}_{g'}(w)) \le n$, and therefore $\mathbf{H}_{g'}(w) \preceq n\mathbf{I}$. So g' is n-smooth.

□

We can similarly use gradient descent with approximate gradient oracle to solve the above convex optimization problem. Unfortunately, this leads to runtime $\Omega(n^3/\epsilon^4)$ (see full version). In the rest of this section, we design a more sophisticated and faster algorithm for this problem.

4.1 Gradient Descent with One Bad Direction

In this subsection, we develop an algorithm for the inverse $\widehat{f}$ parameters problem with running time better than standard gradient descent. The algorithm takes advantage of the fact that the Hessian has large eigenvalues in at most one direction. We first formally state this fact and then we give the algorithm.

Proposition 9. For any distribution μ on vectors that is symmetric with respect to permutations, there exists $a \in \mathbb{R}$ such that $\mathbb{E}_{x \sim \mu}[xx^\top] \preceq M$, where $M = [M_{ij}]$ with $M_{ij} = a$, $i \ne j$, and $M_{ii} = 1$.

See full version for the proof.

Thus, the Hessian of the convex function to be minimized is bounded above by the Hessian in the above proposition. The all-ones vector is an eigenvector of this upper bound. Its corresponding

eigenvalue on this vector is $1 + (n-1)a$. All other eigenvalues of the matrix are $1 - a$. Thus, there is only one "bad" direction with a large eigenvalue.

In the following, we will use z to denote the bad direction. We emphasize that our methods work for any convex function with any (known) fixed bad direction. We prove the following.

Theorem 10. *Consider a function g on n variables which is β_1-smooth in every direction orthogonal to some fixed known bad direction z in which it is β_2-smooth, where $\beta_2 \geq \beta_1$. Suppose we have an ϵ-additive approximate gradient oracle and ϵ^2/β_1-additive approximate function value oracle. Suppose also that we are given a point w^0 and the minimizer of g is w^*. Then we can find a point where the ℓ_2 norm of the gradient is $O(\epsilon)$-small using*

$$O\left(\frac{\beta_1 \cdot [g(w^0) - g(w^*)]}{\epsilon^2} \cdot \log\left(\frac{\beta_2 \cdot [g(w^0) - g(w^*)]}{\epsilon^2}\right)\right)$$

queries to the two oracles plus $O(n)$ additional work per query.

Note that this is only a logarithmic factor away from the runtime one could get if the function were β_1-smooth in every direction.

Applying this theorem to the inverse $\widehat{f}$ parameters problem gives the following corollary:

Theorem 11. *Given a vector of desired $\widehat{f}$ parameters c, we can efficiently compute a weight vector such that the LBF with this weight vector has $\widehat{f}$ parameters that approximate c within ℓ_2-error ϵ. The algorithm runs in time $\tilde{O}(n^2)/\mathrm{poly}(\epsilon)$.*

Due to space constraints, we provide only an overview in the main body and defer a full treatment of the algorithm to the full version of the paper. The algorithm is motivated by the following intuition. Suppose one has a convex function g which has β_1 smoothness in every orthogonal to the z direction, in which it has β_2 smoothness. Suppose we are at some point x, then do a search along the line $x + \alpha z$ for all values of α to minimize g along this line. Call the resulting point y. The gradient of g at y is orthogonal to the z direction. Thus, if we take a gradient step for g at y, this step can also be interpreted as a gradient step for the restriction of g to the hyperplane orthogonal to the z direction which contains y. This restricted function has better smoothness β_1, so gradient descent makes as much progress as if the function were actually β_1-smooth in all directions.

The preceding intuition suggests an algorithm in which we perform binary search to approximately minimize g in the z-direction, take a gradient step orthogonal to the z direction, and repeat. Specifically, in each iteration we do the following, given a starting point x.

(1) Approximately minimize g in the z-direction starting at x. For technical reasons, we must ensure that both the gradient and function value are approximately minimized, so this step has two parts that ensure each of these requirements.
(2) Take the gradient at this new point, orthogonalize it with respect to the bad direction, and take a step in this orthogonalized direction.

In more detail, the One Bad Direction algorithm builds a solution w for the regularized function g' through the following iterative process:

(1) **Step 1: Binary Search in z-Direction.** Given a point w^t, search in the z direction from w^t via binary search to find a point $w^{t+1} = w^t + \alpha z$ such that (1) $|\langle \nabla g'(w^{t+1}), z\rangle| \leq 6\epsilon_1$ and (2) $g'(w^{t+1}) \leq g'(w^{t+1^*}) + 8\epsilon_2$, where w^{t+1^*} minimizes g' in the z-direction, i.e., $w^{t+1^*} = w^t + \alpha^* z$ for some $\alpha^* \in \mathbb{R}$ and $g'(w^*_{t+1}) \leq g'(w_t + \alpha z)$ for any $\alpha \in \mathbb{R}$. The binary search algorithm to find such a point using only ϵ_1 additively approximate gradients and ϵ_2 additive approximate access to the function value first finds an interval S of points along the z search direction that contains the optimal point w^{t+1^*} and for which the derivative in t he z direction at each point in S is at most $6\epsilon_1$. Then, we search within this set for a point which satisfies the function value constraint. That is, it consists of the two following steps:
   (a) **Step 1a: Finding S.** To find S we start with some outer interval S' that is a superset of S. We find the left and right endpoints of S by running two separate instances of binary search on the derivative of g' in the z direction. For the right endpoint, it suffices to search and output any point in S' where the approximate derivative in the z direction is in $[2\epsilon_1, 5\epsilon_1]$. For the left endpoint, it suffices to look for an approximate derivative in $[-5\epsilon_1, -2\epsilon_1]$.
   (b) **Step 1b: Satisfying function value nearly minimal constraint.** We use a variant of binary search on function value to find w^{t+1} such that $g'(w^{t+1}) \leq g'(w^{t+1^*}) + 8\epsilon_2$. In each step, we query the value of g' at four equally spaced points starting from the left endpoint and going to the right endpoint of the interval. Let a, b, c, d be the points ordered by $a < b < c < d$ according to their order in the z direction, with further left points first. Let g'' denote the approximate evaluation oracle for g'. If $g''(a), g''(b), g''(c), g''(d)$ are all within $\pm 3\epsilon_2$ of each other, output the minimum of b and c. Otherwise, we consider the quantity $g''(b) - g''(c)$. If $g''(b) - g''(c) > 2\epsilon_2$, we replace the endpoint a with b. Similarly, if $g''(c) - g''(b) > 2\epsilon_2$, we replace the endpoint d with c. Otherwise, we have $|g''(b) - g''(c)| \leq 2\epsilon_2$. In this case, we replace a with b and d with c, recursing on the subinterval b, c.
(2) **Step 2: Gradient Descent in the Orthogonal Direction.** Given w^{t+1} found in Step 1, we take the step $w^{t+2} = w^{t+1} - \eta r$, where $r = \tilde{\nabla} g'(w^{t+1}) - \langle \tilde{\nabla} g'(w^{t+1}), z\rangle z$ is the orthogonal component to the z direction of an approximately computed gradient $\tilde{\nabla} g'(w^{t+1})$ and $0 < \eta < 1$.

We prove in the full paper version that Theorem 11 gives an algorithm for the inverse semivalue problem for LBFs, under some mild assumptions on the probability distributions defining the semivalues.

Theorem 12. *For any semivalues defined by p^n such that $p_{t-1} - p_t = k(2t-n)(p_{t-1} + p_t)$ for a constant k, for all $t \in \{1, \ldots, n-1\}$, for any $\epsilon > 0$, $\delta > 0$, given a vector of desired semivalues c, we can compute a weight vector $w \in \mathbb{R}^n$ such that the LBF $f_{w,0}$ has $\|\widetilde{f_{w,0}} - c\|_2 \leq \epsilon$ with probability $1 - \delta$, in time $\mathrm{poly}(n, 1/\epsilon)$.*

As discussed earlier, to overcome the fact that LTFs are difficult to optimize over, we relaxed the inverse semivalue problem by searching for an LBF. Summarizing our main results for LBFs, we

have the following: Theorem 7 shows that finding a weight vector of an LBF $f_{w,0}$ that approximately matches some target Banzhaf indices is solvable in $\tilde{O}(n^2/\epsilon^4)$ time. We then propose in Theorem 10 an algorithm for the inverse $\widehat{f}$ parameters problems and we show that finding an LBF $f_{w,0}$ that approximately achieves some target coordinate coefficients is solvable in $\tilde{O}(n^2)/\text{poly}(\epsilon)$. Finally, using the proposed algorithm for the inverse $\widehat{f}$ problem, we give an algorithm for the inverse semivalue problem for LBFs, under some natural assumptions on the probability distributions defining the semivalues, in Theorem 12.

Recalling that the goal of the inverse semivalue problem is to output an LTF (weighted voting game), in the end of our processes we output the corresponding LTF. That is, if the output of our algorithms for the inverse semivalue problem is the LBF $f_{w,0} = P_1(w \cdot x)$, we output the LTF $h_{w,0} = \text{sign}(w \cdot x)$.

Although it remains an open question to bound the distance of the indices of an LBF function and the corresponding LTF function for general semivalues, using previous structural results [6, 7] about the Banzhaf index and the Shapley values, we are able to bound the ℓ_2-distance between the target indices and the indices of the output LTF function for these important semivalue classes. We expect analogous structural results to hold for all semivalues.

Concretely, for the case of the Banzhaf index, [6] established that if an LTF f and a bounded function $g : \{-1,1\}^n \to [-1,1]$ have small Banzhaf distance, i.e., $\|\widetilde{f} - \widetilde{g}\|_2 \leq O(\epsilon^{O(\log^2(1/\epsilon))})$, then the ℓ_1-distance of f and g is at most $O(\epsilon)$, i.e., $\mathbb{E}_{x\sim\{-1,1\}^n}[|f(x) - g(x)|] \leq O(\epsilon)$ (see full version). Hence, if we solve the inverse Banzhaf index for LBFs with accuracy $\epsilon^{O(\log^2(1/\epsilon))}$, we obtain an LBF g that is ϵ-close in ℓ_1-distance to the LTF function f that matches the target Banzhaf indices. Combining the above theorem with Theorem 7, we get the following corollary for the inverse Banzhaf index problem.

Theorem 13. *For any $\epsilon > 0$ and $\delta > 0$, given a vector of target Banzhaf indices c, a weight vector $w \in \mathbb{R}^n$ such that the LTF $h_{w,0}$ has $\mathbb{E}_{x\in\{-1,1\}^n}[|h_{w,0}(x) - h_{v,0}(x)|] \leq O(\epsilon)$ with probability $1-\delta$, can be computed in $\tilde{O}(n^2)(1/\epsilon)^{O(\log^2(1/\epsilon))}$ time, where $h_{v,0}$ is the target LTF whose Banzhaf indices are equal to c.*

Qualitatively similar structural results were obtained for the case of Shapley values [7]. Similarly with the Banzhaf index case, running our algorithms for the inverse Shapley values problem for LBFs with appropriate error, guarantees that the corresponding LTF will be in small distance from the target Shapley values, and gives an algorithm for the inverse Shapley value problem for LTFs. Due to space constraints, we give the statement and proof of this corollary in the full version of the paper.

Our experiments for the Shapley value and the Banzhaf index indicate that such large accuracy for the inverse problems for LBFs is not required in practice to achieve small distance between the LBF and the corresponding LTF: In all test cases, we have found that the distance achieved by the output LBF and the target is comparable with the distance of the corresponding LTF and the target. In the case of the Banzhaf index, we observe that the two distances are essentially the same (see full version for more details).

Comparing the runtime of our proposed algorithms for LTFs and the algorithms in [6, 7], we note that they all have runtime guarantees with polynomial dependence on n for any fixed ϵ. We note that the prior algorithms [6, 7] have not been implemented and evaluated experimentally. For the case of the Banzhaf indices, we expect that our SGD-based algorithm is significantly faster than that of [6]. For the case of the Shapley values, our One Bad Direction algorithm has a significantly better runtime dependence on n (namely $\tilde{O}(n^2)$) compared to [7] (where the poly(n) dependence is unspecified and is certainly at least $\Omega(n^3)$) and, importantly, extends to all semivalues.

5 EXPERIMENTS

We performed an empirical evaluation of practical variants of the above algorithms for the Banzhaf indices, the Shapley values, and the binomial semivalues on synthetic and real inputs. All experiments were done on a laptop computer with a 2.7 GHz Intel Core i7 CPU and 16 GB of RAM. Our experiments show that our algorithms are scalable and achieve high accuracy on both synthetic and real inputs.

5.1 Implementation Details

Algorithms. We implemented the mini-batch stochastic gradient descent method (SGD) and a stochastic variant (SBD) of the one bad direction algorithm. In our implementation, we replaced the binary search in the z-direction with a constant-size step in the z-direction, taking into account only the direction of the gradient. After each such step, we take a gradient descent step orthogonalized with respect to the z-direction. We implemented our SGD and SBD algorithms for Shapley values, binomial semivalues, and Banzhaf indices. In our implementation, we used mini-batch size equal to 50. Pseudocode of the SBD algorithm is given in the full version.

Selecting stepsizes for SGD and SBD. We determined the formulas to use for selecting step size by plotting a data set of hand optimized step sizes for synthetically generated problems and picking parameters that appeared to fit these plots well. Specifically, we ran each of the five algorithms (SGD for Banzhaf, Shapley, binomial semivalues, and SBD for Shapley and binomial) on an initial set of synthetic inputs with n taking various values between 30 and 230. From this, we obtained initial stepsizes for SGD and SBD of $\eta_{\text{init}} \sim \frac{0.19\cdot\text{norm}}{n\cdot\text{norm}_2}$, where norm $= z \cdot$ target is the size of the projection of the input to the z direction and $\text{norm}_2 = \|\text{target} - (z \cdot \text{target}) \cdot z\|_2$ is the size of the orthogonal projection of the input to z direction. We also observed that the stepsizes for the binomial semivalues fit the value $\eta_{\text{init}} \sim \frac{0.6\cdot\text{norm}}{n\cdot\text{norm}_2}$. Regarding the Banzhaf index, we found $\eta = 10/11$ to be a good fit. For SGD and SBD for the Shapley values, we decrease the initial stepsizes by the logarithm of the iteration count. For the SGD for the Banzhaf indices, we decrease the stepsize by a factor of 0.9999 in each iteration.

Estimating the Error. We measured the convergence of our algorithms with respect to the ℓ_2-distance between the target and the semivalues achieved by the output weight vectors. Namely, for a weight vector w, a threshold θ, and a target vector c, we measured the following distances:

- Banzhaf distance: $\|\widetilde{f}^B_{w,\theta} - c\|_2$
- Shapley distance: $\|\widetilde{f}^S_{w,\theta} - c\|_2$
- Correlation Coefficients distance: $\|\widehat{f}_{w,\theta} - c\|_2$

We estimated the aforementioned distances with additive error ϵ with high probability $1-\delta$ using $m = \lceil \frac{2 \cdot n \cdot \log(\frac{2n}{\delta})}{\epsilon^2} \rceil$ samples. In our experiments, we used $\epsilon = 0.01$ for $0 \leq n \leq 230$, $\epsilon = 0.018$ for $n > 230$ and $\delta = 0.1$.

5.2 Synthetic Inputs

Experiments with synthetic target inputs corresponding to various weighted voting games allow us to examine and verify the convergence of our algorithms. Our experiments validate that high accuracy is achieved by both SGD and SBD on all inputs.

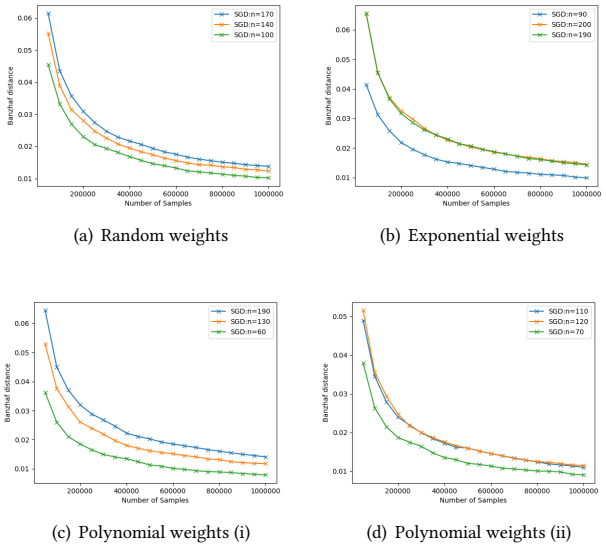
Generating Target Inputs. We used synthetic Shapley values and Banzhaf indices targets with the number of variables n ranging from 30 to 400. To generate a target Shapley values (Banzhaf indices) vector of a specific size n, we selected a weight vector of size n from a family of weights and we computed the Shapley values (Banzhaf index) of the LTF function defined by the sampled weights and $\theta = 0$. We examined the following families of weights:

- Random weights: We select n integer weights uniformly at random from the interval $[1, 10000n]$.
- Exponential weights: For $1 \leq i \leq n$, $w_i = 1.1^i$.
- Polynomial weights (i): For $1 \leq i \leq n$, $w_i = i \cdot n$.
- Polynomial weights (ii): For $1 \leq i \leq n$, $w_i = 1 + 0.5i + 0.2i^2 + 0.005i^3$.

Inputs from three other weight families were also examined. The definitions of these additional weight families and the corresponding experimental results are given in the full version of the paper (due to space limitations).

Banzhaf Index. We evaluated the performance of SGD for the Banzhaf index on 20 synthetic target inputs from each weight family with n ranging from 30 to 230. For each input, we let SGD run for 20000 iterations, and we estimated the Banzhaf distance of the average hypothesis from the target after every 1000 iterations. For each input, we ran SGD 10 times and we plotted the average estimated distance of the 10 repeated runs in Figure 1. Overall, we get that after 20000 iterations very high accuracy is achieved: the average Banzhaf distance over all inputs of the same size over 10 repeated runs ranges from 0.005 for $n = 30$ to 0.012 for $n = 230$.

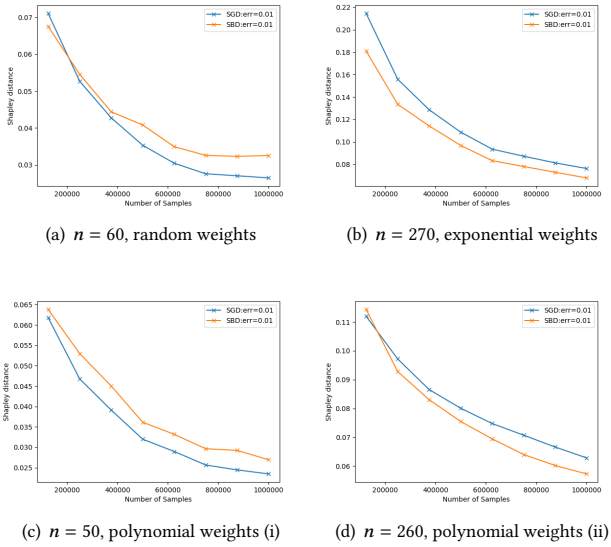
Shapley Values. We evaluated the error convergence of SGD and SBD and compared their performance on 32 synthetic target inputs from each weight family with n ranging from 30 to 390. The results are given in Figure 2. For each input, we let the algorithms run for 20000 iterations and we estimated the distance between the Shapley values of the average hypothesis and the Shapley target after every 2500 iterations. Our results indicate that both algorithms achieve small Shapley distance after 20000 iterations: the Shapley distance ranges from ~ 0.02 for $n = 30$ to ~ 0.07 for $n = 390$. Comparing the performance of SGD and SBD, we observe that although SGD outperforms SBD on some of the inputs, SBD outperforms SGD on a larger fraction of the inputs and achieves smaller average distance when considering only inputs with large n ($n > 150$). More specifically, we have the following results: we get that out of total 224 synthetic targets, SGD outperforms SBD on 123 inputs. If we consider only the cases where one of the two algorithms outperforms the other by at least 10%, we have that SBD outperforms SGD on 39 inputs and SGD outperforms SBD on 29 inputs. Focusing on inputs with large n, we have that SBD outperforms SGD on 91 out of 154 inputs with size at least 130, with SBD outperforming on 37 of them by at least 10% and SGD outperforming on 4.



(a) Random weights (b) Exponential weights

(c) Polynomial weights (i) (d) Polynomial weights (ii)

Figure 1: Average performance of SGD for the Banzhaf index over 10 repeated runs on three inputs with different input sizes n from each of the weight families for 20000 iterations.



(a) $n = 60$, random weights (b) $n = 270$, exponential weights

(c) $n = 50$, polynomial weights (i) (d) $n = 260$, polynomial weights (ii)

Figure 2: Average performance with respect to the Shapley distance of SGD and SBD over 5 repeated runs on one input from each of the weight families with size n for 20000 iterations.

Binomial Semivalues. We note that in addition to the Banzhaf indices and the Shapley values, we evaluated the performance of our

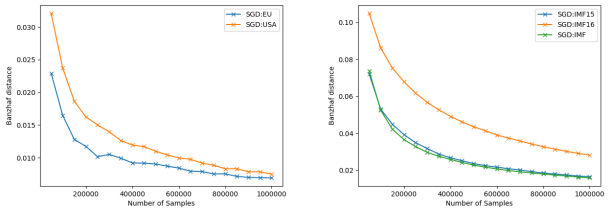
algorithms on binomial semivalues. We get the following results: both SGD and SBD attain small distance from the target after 20000 iterations, but SBD outperforms SGD on inputs with large n. Due to space limitations, the results are given in the full version of the paper.

5.3 Real Inputs

We evaluated our algorithms on a range of real inputs that have received considerable attention in the literature:

- EU: semivalues of the supermajority weighted voting game ($\theta \sim 73\%$ of the sum of weights) with size $n = 27$ and weights equal to the weights that were used in the voting procedures of the European Union Council under the Treaty of Nice [18].
- USA: semivalues of the majority weighted voting game with size $n = 51$ and weights equal to the Electoral College votes allocated to each US state according to the 1990 Census data, used in the 2000 presidential elections [27].
- IMF, IMF16, IMF15: semivalues of the majority weighted voting games with sizes $n = 189$, $n = 188$, $n = 188$ and weights equal to the voting shares of the IMF members in 2019 [20], 2016, and 2015[22, 29].

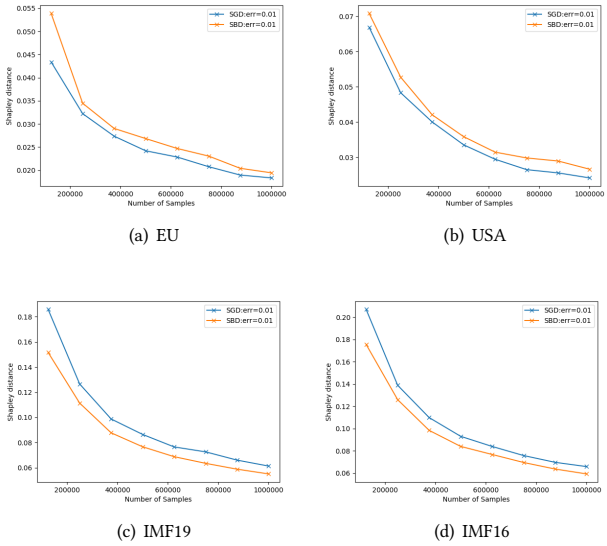
The results of our experiments for the Banzhaf index and the Shapley values on the real inputs verify that small error is achieved by our algorithms after 20000 iterations. Specifically, we have that the average estimated Banzhaf distance over 10 repeated runs is 0.0069 for EU, 0.0075 for USA, and 0.0159 for IMF, 0.0282 for IMF16 and 0.0166 for IMF15. Figure 3 shows the results of SGD on the Banzhaf index of the five real inputs.



(a) EU and USA (b) IMF, IMF15 and IMF16

Figure 3: Average performance of SGD for the Banzhaf index over 10 repeated runs on the Banzhaf index of the weighted majority games of EU, USA and IMF for 20000 iterations.

Comparing the performance of SGD and SBD on the Shapley values of the real inputs, we observe a similar trend with their performance on synthetic inputs: SGD outperforms SBD on inputs with small n, i.e., the EU and USA, and SBD outperforms SGD on inputs with large n, i.e., on the three IMF inputs. Figure 4 shows that after 20000 iterations, SGD outperforms SBD by $\sim 6.0\%$ and $\sim 10.2\%$ on EU and USA, respectively, whereas SBD outperforms SGD by $\sim 11.3\%$, $\sim 9.8\%$ and $\sim 11.1\%$ on IMF, IMF15 and IMF16, respectively (the results for IMF15 are given in the full version).

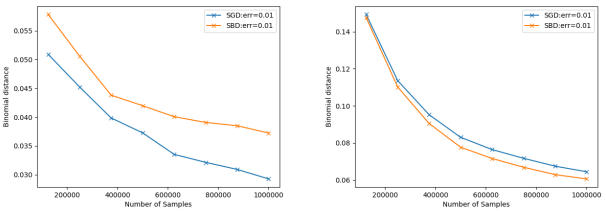


(a) EU (b) USA

(c) IMF19 (d) IMF16

Figure 4: Average performance with respect to Shapley distance of SGD and SBD over 5 repeated runs on EU, USA and IMF.

Binomial Semivalues. We obtained qualitatively similar results as in the Shapley values case: SBD outperforms SGD on inputs with large n, i.e., on the three IMF inputs; SGD outperforms SBD on inputs with small n, i.e., on the EU and USA inputs. Figure 5 shows that SGD outperforms SBD by $\sim 27.3\%$ on the USA input, whereas SBD outperforms SGD on IMF by $\sim 2.7\%$.



(a) USA (b) IMF16

Figure 5: Average performance with respect to binomial semivalues distance of SGD and SBD for the binomial semivalues ($p = 0.3$) over two repeated runs on USA and IMF.

Runtime. We measured the runtime of SBD and SGD for Shapley values and binomial semivalues, and the runtime of SGD for the Banzhaf index, when we let the algorithms run for 20000 iterations (the measured time does not include the time needed to estimate the error of the average hypothesis). Our experiments confirm that our algorithms are scalable. We have that over two repeated runs the runtime of SGD for the Banzhaf index ranges from $\sim 1.69s$ for $n = 30$ to $\sim 7.75s$ for $n = 390$; for Shapley values ranges from $\sim 6.48s$ to $\sim 85.80s$; and for binomial semivalues ranges from $\sim 5.56s$ to $\sim 75.52s$. The runtimes of SBD are comparable.

REFERENCES

[1] Noga Alon and Paul H Edelman. 2010. The inverse Banzhaf problem. *Social Choice and Welfare* 34, 3 (2010), 371–377.

[2] Haris Aziz. 2008. Complexity of comparison of influence of players in simple games. In *Proceedings of the Second International Workshop on Computational Social Choice*. 61–72.

[3] Haris Aziz, Mike Paterson, and Dennis Leech. 2007. Efficient algorithm for designing weighted voting games. In *IEEE Intl. Multitopic Conf.* 1–6.

[4] John F Banzhaf III. 1964. Weighted voting doesn't work: A mathematical analysis. *Rutgers L. Rev.* 19 (1964), 317.

[5] Georgios Chalkiadakis, Michael Wooldridge, and Herve Moulin. 2016. *Weighted Voting Games*. Cambridge University Press, 377–396.

[6] Anindya De, Ilias Diakonikolas, Vitaly Feldman, and Rocco A Servedio. 2014. Nearly optimal solutions for the chow parameters problem and low-weight approximation of halfspaces. *Journal of the ACM (JACM)* 61, 2 (2014), 11.

[7] Anindya De, Ilias Diakonikolas, and Rocco A Servedio. 2017. The inverse Shapley value problem. *Games and Economic Behavior* 105 (2017), 122–147.

[8] Bart De Keijzer, Tomas B Klos, and Yingqian Zhang. 2014. Finding optimal solutions for voting game design problems. *Journal of Artificial Intelligence Research* 50 (2014), 105–140.

[9] Frits de Nijs and Daan Wilmer. 2012. Evaluation and Improvement of Laruelle-Widgren Inverse Banzhaf Approximation. *arXiv preprint arXiv:1206.1145* (2012).

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

[11] Ilias Diakonikolas and Chrystalla Pavlou. 2019. On the complexity of the inverse semivalue problem for weighted voting games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 1869–1876.

[12] Pradeep Dubey, Abraham Neyman, and Robert James Weber. 1981. Value theory without efficiency. *Mathematics of Operations Research* 6, 1 (1981), 122–128.

[13] Pradeep Dubey and Lloyd S Shapley. 1979. Mathematical properties of the Banzhaf power index. *Mathematics of Operations Research* 4, 2 (1979), 99–131.

[14] Edith Elkind, Leslie Ann Goldberg, Paul W Goldberg, and Michael Wooldridge. 2008. On the Dimensionality of Voting Games.. In *Proceedings of the 23rd AAAI Conference on Artificial Intelligence*. 69–74.

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

[16] Piotr Faliszewski and Lane A Hemaspaandra. 2008. The complexity of power-index comparison. In *International Conference on Algorithmic Applications in Management*. Springer, 177–187.

[17] Shaheen Fatima, Michael Wooldridge, and Nicholas R Jennings. 2008. An anytime approximation method for the inverse Shapley value problem. In *Proceedings of the 7th International Joint Conference on Autonomous Agents and Multiagent Systems-Volume 2*. International Foundation for Autonomous Agents and Multiagent Systems, 935–942.

[18] Josep Freixas. 2004. The dimension for the European Union Council under the Nice rules. *European Journal of Operational Research* 156, 2 (2004), 415–419.

[19] Josep Freixas and M Albina Puente. 2002. Reliability importance measures of the components in a system based on semivalues and probabilistic values. *Annals of Operations Research* 109, 1-4 (2002), 331–342.

[20] IMF. 2020. *International Monetary Fund*. https://www.imf.org/external/np/sec/memdir/members.aspx

[21] Sascha Kurz. 2012. On the inverse power index problem. *Optimization* 61, 8 (2012), 989–1011.

[22] Sascha Kurz. 2016. Computing the Power Distribution in the IMF. *Available at SSRN 2742118* (2016).

[23] Sascha Kurz. 2016. The inverse problem for power distributions in committees. *Social Choice and Welfare* 47, 1 (2016), 65–88.

[24] Annick Laruelle and Mika Widgrén. 1998. Is the allocation of voting power among EU states fair? *Public Choice* 94, 3-4 (1998), 317–339.

[25] Dennis Leech. 2002. Designing the voting system for the EU Council of Ministers. *Public Choice* 113, 3-4 (2002), 437–464.

[26] Dennis Leech. 2002. Voting power in the governance of the International Monetary Fund. *Annals of Operations Research* 109, 1-4 (2002), 375–397.

[27] Dennis Leech. 2003. Computing power indices for large voting games. *Management Science* 49, 6 (2003), 831–837.

[28] Dennis Leech. 2003. Power indices as an aid to institutional design: the generalised apportionment problem. *Yearbook on New Political Economy* (2003).

[29] D Leech and R Leech. 2013. *A new analysis of a priori voting power in the IMF: Recent quota reforms give little cause for celebration*. 389–410. https://doi.org/10.1007/978-3-642-35929-3_21

[30] Ryan O'Donnell and Rocco A Servedio. 2011. The chow parameters problem. *SIAM J. Comput.* 40, 1 (2011), 165–199.

[31] Kislaya Prasad and Jerry S Kelly. 1990. NP-completeness of some problems concerning voting games. *International Journal of Game Theory* 19, 1 (1990), 1–9.

[32] Alvin E Roth (Ed.). 1988. *The Shapley value: essays in honor of Lloyd S. Shapley*. Cambridge University Press.

[33] Lloyd S Shapley. 1953. A value for n-person games. *Contributions to the Theory of Games* 2, 28 (1953), 307–317.

[34] Lloyd S Shapley and Martin Shubik. 1954. A method for evaluating the distribution of power in a committee system. *American political science review* 48, 3 (1954), 787–792.

[35] Matthias Sutter. 2000. Fair allocation and re-weighting of votes and voting power in the EU before and after the next enlargement. *Journal of Theoretical Politics* 12, 4 (2000), 433–449.

[36] Robert James Weber. 1979. *Subjectivity in the valuation of games*. Technical Report. Yale Univ New Haven Conn Cowles Foundation For Research In Economics.