In this paper we initiate the study of financial asset design with fairness as an explicit goal. We consider variations on the classical problem of optimal portfolio design. In our setting, an individual consumer is specified by her risk tolerance, which corresponds to the variance in returns she is willing to accept in exchange for higher expected returns. We must design a (small) collection of portfolios and assign each consumer to a portfolio at lower or approximately equal risk than her tolerance. Fairness is imposed by demanding that the portfolios designed do not discriminate (in terms of expected returns) against less wealthy clients (or other specified protected groups).

Our main results are algorithms for optimal and near-optimal portfolio design for both social welfare and fairness objectives, both with and without assumptions on the underlying group structure. We describe an efficient algorithm based on an internal two-player zero-sum game that learns near-optimal fair portfolios \textit{ex ante} and show experimentally that it can be used to obtain a small set of fair portfolios \textit{ex post} as well. For the special but natural case in which group structure coincides with risk tolerances (which models the reality that wealthy consumers generally tolerate greater risk), we give an efficient and optimal fair algorithm. We also provide generalization guarantees for the underlying risk distribution that has no dependence on the number of portfolios and illustrate the theory with simulation results.

CCS Concepts: • Theory of computation → Dynamic programming; Algorithmic game theory; • Applied computing → Economics.

ACM Reference Format:
Emily Diana, Travis Dick, Hadi Elzayn, Michael Kearns, Aaron Roth, Zachary Schutzman, Saeed Sharifi-Malvajerdi, and Juba Ziani. 2021. Algorithms and Learning for Fair Portfolio Design. In Proceedings of the 22nd ACM Conference on Economics and Computation (EC ’21), July 18–23, 2021, Budapest, Hungary. ACM, New York, NY, USA, 19 pages. https://doi.org/10.1145/3465456.3467646

1 INTRODUCTION
In this work, we initiate the study of financial portfolio design with fairness as an explicit goal. We consider algorithmic and learning problems in the design of portfolios that adequately serve different subgroups of a population.

We imagine a large collection of individual retail investors or consumers, each of whom has her own tolerance for investment risk in the form of a limit on the standard deviation of returns.
is well-known in quantitative finance that for any set of financial assets, the optimal expected return on investment is increasing with the riskiness of the portfolio. We assume that a large retail investment firm (such as Vanguard or Fidelity) wishes to design portfolios (which will be offered as financial products) to serve these consumers under the common practice of assigning consumers only to portfolios with lower risks than their tolerances. The firm would like to design and offer only a small number of such portfolios — much smaller than the number of consumers — since the execution and maintenance of portfolios is costly and ongoing. The overarching goal is to design the portfolios to minimize consumer regret — the loss of returns due to being assigned to lower-risk portfolios compared to a consumer’s "bespoke" optimal portfolio — both with and without fairness considerations. We highlight that fairness concerns are becoming increasingly more prominent among both practitioners and academics in the financial sector — see discussion in, for instance, [14], [15], [24], and [23].

We consider a notion of group fairness adapted from the literature on fair division. Consumers belong to underlying groups and the designer’s goal is to minimize the maximum regret across groups — i.e. to optimize for the least well-off group. We refer to this notion as minimax fairness, recently studied by [6, 18]. Our models are agnostic to the groups’ definitions. They may be based on standard demographic features such as race, gender, or age; they may be defined with respect to the designer’s particular application, such as classes of employees at a firm; or groups may correspond to risk tolerances themselves, wherein the firm wishes to adequately serve both risk-averse and risk-seeking consumers.

Compared to the approach of constraining regret to be (approximately) equal across groups (which is not even always feasible in our setting), minimax optimal solutions have the property that they Pareto-dominate regret-equalizing solutions: every group has regret that can only be smaller than it would be if regret were constrained to be equalized across groups [6, 18].

1.1 Our Results and Techniques

We begin by specifying the model and notion of regret (Section 2) and then give an efficient dynamic programming solution to the problem of selecting $p$ portfolios to serve a given population absent any considerations of fairness (Section 3). We then turn to the problem of guaranteeing minimax fairness across groups (Section 4). We consider fairness from two perspectives: the randomized ex ante definition where the designer needs only commit to a distribution over $p$-sets of products and the ex post definition where the designer must select a single optimal minimax fair $p$-set of portfolios for the consumers, and we show separation results (Section 4.1) differentiating these two solution concepts. In the ex ante setting (Section 4.2), we leverage techniques from learning in games and cast the problem as a zero-sum game and design an algorithm which simulates no-regret learning in this game to find a minimax fair suite of portfolios in expectation. In the ex post case (Section 4.3) we extend our previous dynamic program to efficiently find an optimal solution when the number of groups is taken to be a constant. We additionally consider the special case where groups’ risk tolerances belong to disjoint intervals and give an ex post algorithm which is efficient even for a large number of groups (Section 4.4). Our final theoretical results consider generalization (Section 5), where we show that when consumer risk tolerances are drawn from some distribution, there are bounds on how well the optimal solution for that sample performs with respect to the distribution overall. We conclude by evaluating our algorithms empirically (Section 6) by designing portfolios using information from real stock market performance data and synthetic individuals partitioned into groups. For the sake of brevity, additional details and complete proofs can be found in the extended version online.\(^1\)

\(^1\)https://arxiv.org/abs/2006.07281
1.2 Related Work

Our work generally falls within the literature on fairness in machine learning, which is too broad to survey in detail here; see [5] for a recent overview. We are perhaps closer in spirit to research in fair division or allocation problems [1, 3, 22], in which a limited resource must be distributed across a collection of players so as to maximize the utility of the least well-off; here, the resource in question is the small number of portfolios to design. However, we are not aware of any technical connections between our work and this line of research. While our work is inspired and motivated by the context of and applications to the portfolio design problem in finance, there are similarities between the problem we study and one-dimensional clustering and one-dimensional facility location problems [2, 4, 11, 20]. In these problems, there are a collection of individuals along the real line, and the algorithm designer must choose a collection of points on the line to serve as ‘centers’ so as to minimize some cost function associated with the individuals accessing the centers. The algorithm we give in Section 3 for the problem absent fairness is a standard dynamic programming formulation for these settings. There is some recent literature on fair facility location problems [13, 16], which attempt to choose a small number of “centers” to minimize a cost function together with constraints informed by the need to serve a large and diverse population. Our work also has similarities to work on designing algorithms for “fair allocation” problems that arise in contexts such as loan administration or predictive policing [7–9].

There seems to have been relatively little explicit consideration of fairness issues in quantitative finance generally and optimal portfolio design specifically. An exception is [12], in which the interest is in fairly amortizing transaction costs of a single portfolio across investors rather than designing multiple portfolios to meet a fairness criterion.

2 MODEL AND PRELIMINARIES

We aim to create products (portfolios) consisting of weighted collections of assets with differing means and standard deviations (risks). Each consumer is associated with a real number \( \tau \in \mathbb{R}_{\geq 0} \), which is the risk tolerance of the consumer — an upper bound on the standard deviation of any acceptable portfolio. We assume that consumers’ risk tolerances are bounded.

2.1 Bespoke Problem

We adopt the standard Markowitz framework [17] for portfolio design.\(^2\) Given a set of \( m \) assets with mean \( \mu \in \mathbb{R}^m \) and covariance matrix \( \Sigma \in \mathbb{R}^{m \times m} \), and a consumer risk tolerance \( \tau \), the bespoke portfolio achieves the maximum expected return that the consumer can realize by assigning weights \( a \) over the assets (where the weight of an asset represents the fraction of the portfolio allocated to said asset) subject to the constraint that the overall risk — quantified as the standard deviation of the mixture \( a \) over assets – does not exceed her tolerance \( \tau \). Finding a consumer’s bespoke portfolio can be written as an optimization problem. We formalize the bespoke problem in Equation (1) below and call the solution \( r(\tau) \).

\[
r(\tau) = \max_{a \in \mathbb{R}^m} \left\{ a^T \mu \mid a^T \Sigma a \leq \tau^2, 1^T a = 1 \right\}
\]

Here we note that optimal portfolios are summarized by the function \( r(\tau) \), which is non-decreasing. Since consumer tolerances are bounded, we let \( B \) denote the maximum value of \( r(\tau) \) across all consumers.

\(^2\)A detailed empirical illustration of this framework is provided in Section 6.
2.2 A Regret Notion

In order to frame our problem in terms of standard mathematical formulations, we need a notion of regret. Two considerations inform our particular choice of regret definition. First, we would like the solution to our problem to correspond to a good outcome for consumers, and this requires that the regret of a choice of products measures how well the consumers could have done under some other choice (as opposed to thinking about how well the portfolio designer could have done, which may or may not align with consumer interests). The solution to the bespoke problem is, by definition, the best the consumer could have done given their risk tolerance and thus is the appropriate benchmark. Second, the realized outcome of any given asset or portfolio may include significant randomness, and so it is more useful to consider the expected value of instruments. Hence, the notions of regret we will consider will use the difference in expected return between a consumer’s best choice under our selected portfolio and her bespoke portfolio.

Suppose there are \( n \) consumers to whom we offer a set of products \( c = (c_1, c_2, \ldots, c_p) \in \mathbb{R}_+^p \). For a consumer with tolerance \( \tau \), the utility from picking product \( c_i \in c \) is given by:

\[
    u_{\tau}(c_i) = \begin{cases} 
    r(c_i) & \text{if } c_i \leq \tau \\
    -\infty & \text{otherwise}
    \end{cases}
\]

That is, a consumer gets utility \( r(c_i) \) equal to the return of product \( c_i \) when it satisfies her tolerance constraint, and arbitrarily negative utility for any product above her tolerance. This second part models the fact that consumers are unwilling to accept products above their maximum tolerance level.

The consumers are strategic and aim to maximize their utility. In turn, when offered a set of product \( c = (c_1, c_2, \ldots, c_p) \), a consumer with tolerance \( \tau \) picks the product \( c(\tau) \) that satisfies

\[
    c(\tau) \triangleq \arg\max_{c_i \in c} u_{\tau}(c_i). 
\]

Equivalently, \( c(\tau) \) is a solution to the following optimization problem:

\[
    \max_{c_i \in c} r(c_i) \\
    \text{s.t. } c_i \leq \tau.
\]

By monotonicity of the return function \( r(\cdot) \), we have that

\[
    c(\tau) = \max \{ c_i : c_i \leq \tau, c_i \in c \} 
\]

Our goal is to design \( p \ll n \) products that minimize a notion of regret for a given set of consumers. A product has a risk (standard deviation) which we will denote by \( S \). We assume throughout that, in addition to the selected \( p \) products, there is always a risk-free product (say cash) available that has zero return; we will denote this risk-free product by \( c_0 \equiv 0 \) throughout the paper \( (r(c_0) = 0) \). For a given consumer with risk threshold \( \tau \), the regret of the consumer with respect to a set of products \( c = (c_1, c_2, \ldots, c_p) \in \mathbb{R}_+^p \) is the difference between the return of her bespoke product and the return of her preferred product \( c(\tau) \). To formalize this, the regret of products \( c \) for a consumer with risk threshold \( \tau \) is defined as

\[
    R_{\tau}(c) \triangleq r(\tau) - r(c(\tau)) = r(\tau) - \max_{c_j \leq \tau} r(c_j).
\]

Note since \( c_0 = 0 \) always exists, the \( \max_{c_j \leq \tau} r(c_j) \) term is well defined. Now for a given set of consumers \( S = \{ \tau_i \}_{i=1}^n \), the regret of products \( c \) on \( S \) is simply defined as the average regret of \( c \) on \( S \):

\[
    R_S(c) \triangleq \frac{1}{n} \sum_{i=1}^n R_{\tau_i}(c)
\]
When $S$ includes the entire population of consumers, we call $R_S(c)$ the population regret. The following notion for the weighted regret of $c$ on $S$, given a vector $w = (w_1, \ldots, w_n)$ of weights for each consumer, will be useful in Sections 3 and 4:

$$R_S(c, w) \triangleq \sum_{i=1}^n w_i R_{\tau_i}(c)$$

(7)

Absent any fairness concern, our goal is to design efficient algorithms to minimize $R_S(c)$ for a given set of consumers $S$ and target number of products $p$: $\min_c R_S(c)$. This will be the subject of Section 3. We can always find an optimal set of products as a subset of the $n$ consumer risk thresholds $S$, because if any product $c_j$ is not in $S$, we can replace it by $\min \{ \tau_i \mid \tau_i \geq c_j \}$ without decreasing the return for any consumer. We let $C_p(S)$ represent the set of all subsets of size $p$ for a given set of consumers $S$: $C_p(S) = \{ c = (c_1, c_2, \ldots, c_p) \subseteq S \}$.

We can reduce our regret minimization problem to the following problem:

$$R(S, p) \triangleq \min_{c \in C_p(S)} R_S(c)$$

(8)

Similarly, we can reduce the weighted regret minimization problem to the following problem:

$$R(S, w, p) \triangleq \min_{c \in C_p(S)} R_S(c, w).$$

(9)

**Remark 1.** In the extended version, we consider a more general regret framework that allows consumers to select products with risk higher than their tolerance. In this case, it is not necessarily optimal to always place products on the consumer risk thresholds; in turn, the techniques we use are of independent interest and different from those used in the main body of the paper.

**Mutual fund separation theorems** are a family of results in the study of optimal portfolio design of the form ‘the risk-return profile of any consumer’s optimal portfolio built from the underlying assets can instead be built from a small number of carefully chosen products, called the mutual funds’, see e.g. [19]. We emphasize that due to our modelling choices, these results do not apply in our setting, despite the similarity in principle of designing a small suite of products to serve a broad range of customers. In particular, the power of the mutual fund separation theorems comes from savvy investors being able to take short positions in some funds in order to expand a long position in another. As an illustration, the one-fund theorem offers users a single mutual fund and a risk-free asset. For a user whose risk tolerance exceeds that of the risk value of the fund, their optimal allocation is to take a short position in the risk-free asset (i.e. a fixed-interest cash loan) as leverage for a long position in the mutual fund. However, in our model, we explicitly disallow taking this short position, which is both reflective of limitations in real-world scenarios such as investing in a retirement plan like a 401(k) as well as of our modelling decision to consider a problem in which each consumer must be assigned to exactly one product, rather than being able to mix-and-match or take weighted combinations of the offerings.

### 2.3 Group Fairness: ex post and ex ante

Like much of the literature, we focus on group fairness, but our setting is somewhat unique. In the setting without fairness, the individuals’ risk tolerances are the only features that determine the regret-minimizing suite of portfolios. When we then consider how this solution might adversely impact certain populations, we additionally consider the group membership of the individuals. Our models are agnostic to where these group labels come from. A designer may well wish to enforce fairness with respect to traditional demographic groups, such as race, gender, or age. The groups may represent something specific to the problem itself, such as a corporation providing retirement accounts that properly serve the different types of employees at that firm. There are
natural instances of this problem where risk tolerances correlate or even coincide with group membership, such as in designing investment options for retirement where the designer wishes to be fair with respect to consumers’ ages, since individuals tend to become less tolerant of risk as they grow older. Finally, it may even be the case that risk tolerances themselves define the group structure, where the designer is simply concerned that ‘low risk’, ‘medium risk’ and ‘high risk’ consumers are all adequately served by the selected portfolios.

We additionally remark that this very group structure can capture a notion of individual fairness by taking each user to be the single member of their own “group”. In the special case where every group contains a single consumer, we can use a slight modification of the dynamic program from Section 3 find a \( p \)-set of products minimizing the minimax individual regret for \( n \) consumers in time \( O(n^2 p) \).

Our approach also engenders nuance in the timing of fairness considerations. That is, as we use randomized algorithms, one may consider the regret that will be achieved in expectation before the randomness of the algorithm is realized, or first realize the randomness and then measure fairness with respect to the portfolios ultimately selected. Both notions are useful: ex ante fairness is easier to satisfy computationally, and may be a reasonable definition in settings where the portfolios are modified frequently, consumers enter and exit the system, or the risk tolerances of consumers are not known in advance. On the other hand, the stronger notion of ex post fairness captures settings where redesign opportunities may be few and far between or the consumers’ tolerances are known in advance.

Now suppose consumers are partitioned into \( g \) groups: \( S = \{G_k\}_{k=1}^g \). Each \( G_k \) consists of the consumers of group \( k \) represented by their risk thresholds. We will often abuse notation and write \( i \in G_k \) to denote that consumer \( i \) has threshold \( \tau_i \in G_k \). Given this group structure, minimizing the regret of the whole population absent any constraint might lead to some groups incurring much higher regret than others. With fairness concerns in mind, we turn to the design of efficient algorithms to minimize the maximum regret over groups (we call this maximum “group regret”):

\[
\mathcal{R}_{\text{fair}}(S, p) \triangleq \min_{c \in \mathcal{C}_p(S)} \left\{ \max_{1 \leq k \leq g} R_{G_k}(c) \right\}
\]  

(10)

The set of products \( c \) that solves the above minimax problem will be said to satisfy ex post minimax fairness (for brevity, we call this “fairness” throughout). One can relax Program (10) by allowing the designer to randomize over sets of \( p \) products and output a distribution over \( \mathcal{C}_p(S) \) (as opposed to one deterministic set of products) that minimizes the maximum expected regret of groups:

\[
\hat{\mathcal{R}}_{\text{fair}}(S, p) \triangleq \min_{C \in \Delta(\mathcal{C}_p(S))} \left\{ \max_{1 \leq k \leq g} \mathbb{E}_{c \sim C} \left[ R_{G_k}(c) \right] \right\}
\]  

(11)

where \( \Delta(A) \) represents the set of probability distributions over the set \( A \), for any \( A \). The distribution \( C \) that solves the above minimax problem will be said to satisfy ex ante minimax fairness — meaning fairness is satisfied in expectation before realizing any set of products drawn from the distribution \( C \) — but there is no fairness guarantee on the realized draw from \( C \). Such a notion of fairness is useful in settings in which the designer has to make repeated decisions over time and has the flexibility to offer different sets of products in different time steps. In Section 4, we provide algorithms that solve both problems cast in Programs (10) and (11). We note that while there is a simple integer linear program (ILP) that solves Equations (10) and (11), such an ILP is often intractable to solve. We use it in our experiments on small instances to evaluate the quality of our efficient algorithms.
3 REGRET MINIMIZATION ABSENT FAIRNESS

In this section, we provide an efficient dynamic programming algorithm for finding the set of p products that minimizes the (weighted) regret for a collection of consumers (absent fairness constraints). This dynamic program will be used as a subroutine in our algorithms for finding optimal products for minimax fairness. A firm uninterested in fairness could use such a program (choosing weights increasing in customer profitability) to minimize regret for profitable customers and consequently, maximize its own profits; the fact that this approach is only an ingredient, not sufficient alone, in finding a fair portfolio highlights the fact that the fairness-constrained problem is considerably more technically challenging.

Let $S = \{\tau_i\}_{i=1}^n$ be a collection of consumer risk thresholds and $w = (w_i)_{i=1}^n$ be their weights, such that $\tau_1 \leq \cdots \leq \tau_n$ (without loss of generality). The key idea is as follows: suppose that consumer index $z$ defines the riskiest product in an optimal set of $p$ products. Then all consumers $z, \ldots, n$ will be assigned to that product and the consumers $1, \ldots, z-1$ will not be. Therefore, if we knew the highest risk product in an optimal solution, we would be left with a smaller sub-problem in which the goal is to optimally choose $p-1$ products for the first $z-1$ consumers. Our dynamic programming algorithm finds the optimal $p'$ products for the first $n'$ consumers for all values of $n' \leq n$ and $p' \leq p$.

More formally for any $n' \leq n$, let $S[n'] = \{\tau_i\}_{i=1}^{n'}$ and $w[n']$ denote the $n'$ lowest risk consumers and their weights. For any $n' \leq n$ and $p' \leq p$, let $T(n', p') = R(S[n'], w[n'], p')$ be the optimal weighted regret achievable in the sub-problem using $p'$ products for the first $n'$ weighted consumers. We make use of the following recurrence relations:

**Lemma 1.** The function $T$ defined above satisfies the following properties:

1. For any $1 \leq n' \leq n$, we have $T(n', 0) = \sum_{i=1}^{n'} w_i r(\tau_i)$.
2. For any $1 \leq n' \leq n$ and $0 \leq p' \leq p$, we have

   $$T(n', p') = \min_{z \in \{p', \ldots, n\}} \left( T(z-1, p' - 1) + \sum_{i=z}^{n'} w_i (r(\tau_i) - r(\tau_z)) \right).$$

The running time of our dynamic programming algorithm, which uses the above recurrence relations to solve all sub-problems, is summarized below.

**Theorem 1.** There exists an algorithm that, given a collection of consumers $S = \{\tau_i\}_{i=1}^n$ with weights $w = (w_i)_{i=1}^n$ and a target number of products $p$, outputs a collection of products $c \in C_p(S)$ with minimal weighted regret: $R_S(c, w) = R(S, w, p)$. This algorithm runs in time $O(n^3 p)$.

**Proof.** The algorithm computes a table containing the values $T(n', p')$ for all values of $n' \leq n$ and $p' \leq p$ using the above recurrence relations. The first column, when $p' = 0$, is computed using property 1 from Lemma 1, while the remaining columns are filled using property 2. By keeping track of the value of the index $z$ achieving the minimum in each application of property 2, we can also reconstruct the optimal products for each sub-problem.

To bound the running time, observe that the sums appearing in both properties can be computed in $O(1)$ time, after pre-computing all partial sums of the form $\sum_{i=1}^{n'} w_i r(\tau_i)$ and $\sum_{i=1}^{n'} w_i$ for $n' \leq n$. Computing these partial sums takes $O(n)$ time. With this, property 1 can be evaluated in $O(1)$ time, and property 2 can be evaluated in $O(n)$ time (by looping over the values of $z$). In total, we can fill out all $O(np)$ table entries in $O(n^2 p)$ time. Reconstructing the optimal set of products takes $O(p)$ time. \qed
4 REGRET MINIMIZATION WITH GROUP FAIRNESS

In this section, we consider the problem of choosing a set of $p$ products in a way that satisfies group fairness. As noted before, groups can correspond to traditional demographic classes like race or gender, context-specific partitions such as type of employee, or can simply correspond to partitions of risk tolerances. Importantly, in our notion of minimax fairness, our objective is to optimize the welfare of the worst-off group under our choice. Thus we are implicitly enforcing fairness to protect all groups from inequity, while still achieving good overall performance. This may not be appropriate in certain settings - for instance, if the designer wished to prioritize specific groups to remedy historical inequity, they may prefer to solve a weighted regret-minimization using the dynamic program described in Section 3 and weight the groups they wish to protect.

Formally, we study the problem of choosing $p$ products when the consumers can be partitioned into $g$ groups and we want to optimize minimax fairness across groups, for both the ex post minimax fairness Program (10) and the ex ante minimax fairness Program (11).

We start the discussion of minimax fairness by showing a separation between the ex post objective in Program (10) and the ex ante objective in Program (11). More precisely, we show in subsection 4.1 that the objective value of Program (10) can be $\Omega(g)$ times higher than the objective value of Program (10).

In the remainder of the section, we provide algorithms to solve Programs (10) and (11). In subsection 4.2, we provide an algorithm that solves Program (11) to any desired additive approximation factor via no-regret dynamics. In subsection 4.3, we provide a dynamic program approach that finds an approximately optimal solution to Program (10) when the number of groups $g$ is small. Finally, in subsection 4.4, we provide a dynamic program that solves Program (10) exactly in a special case of our problem in which the groups are given by disjoint intervals of consumer risk thresholds.

4.1 Separation Between Randomized and Deterministic Solutions

The following theorem shows a separation between the minmax (expected) regret achievable by deterministic versus randomized strategies (as per Programs (10) and (11)); in particular, the regret $R_{\text{fair}}$ of the best deterministic strategy can be $\Omega(g)$ times worse than the regret $\hat{R}_{\text{fair}}$ of the best randomized strategy:

**Theorem 2.** For any $g$ and $p$, there exists an instance $S$ consisting of $g$ groups such that

$$\frac{\hat{R}_{\text{fair}}(S, p)}{R_{\text{fair}}(S, p)} \leq \frac{1}{p + 1} \left\lfloor \frac{p + 1}{g} \right\rfloor$$

In the following theorem, we show that for any instance of our problem, by allowing a multiplicative factor $g$ blow-up in the target number of products $p$, the optimal deterministic minmax value will be at least as good as the randomized minmax value with $p$ products.

**Theorem 3.** We have that for any instance $S$ consisting of $g$ groups, and any $p$,

$$R_{\text{fair}}(S, gp) \leq \hat{R}_{\text{fair}}(S, p)$$
Proof. Fix any instance $S = \{G_k\}_{k=1}^g$ and any $p$. Let $c_k^* \triangleq \arg\min_{c \in C_p(S)} R_{G_k}(c)$ which is the best set of $p$ products for group $G_k$. We have that

$$R_{\text{fair}}(S, gp) = \min_{c \in C_p(S)} \max_{1 \leq k \leq g} R_{G_k}(c) \leq \max_{1 \leq k \leq g} R_{G_k}(\bigcup_k c_k^*) \leq \max_{1 \leq k \leq g} R_{G_k}(c_k^*) \leq \max \mathbb{E}_{c \sim \mathcal{C}} [R_{G_k}(c)]$$

where the last inequality follows from the definition of $c_k^*$ and it holds for any distribution $C \in \Lambda(C_p(S))$. \hfill \qed

### 4.2 An Algorithm to Optimize for \textit{ex ante} Fairness

In this section, we provide an algorithm to solve the \textit{ex ante} Program (11). Remember that the optimization problem is given by

$$\widehat{R}_{\text{fair}}(S, p) \triangleq \min_{C \in \Lambda(C_p(S))} \left\{ \max_{1 \leq k \leq g} \mathbb{E}_{c \sim \mathcal{C}} [R_{G_k}(c)] \right\}$$

Algorithm (1) relies on the dynamics introduced by Freund and Schapire [10] to solve Program (11). The algorithm interprets this minimax optimization problem as a zero-sum game between the designer, who wants to pick products to minimize regret, and an adversary, whose goal is to pick the highest regret group. This game is played repeatedly, and agents update their strategies at every time step based on the history of play. In our setting, the adversary uses the multiplicative weights algorithm to assign weights to groups (as per Freund and Schapire [10]) and the designer best-responds using the dynamic program from Section 3 to solve Equation (12) to choose an optimal set of products, noting that

$$\mathbb{E}_{k \sim D(t)} \left[ R_{G_k}(c) \right] = \sum_{k \in [g]} D_k(t) \sum_{i \in G_k} \frac{R_{G_k}(c)}{|G_k|} = \sum_{i=1}^n R_{G_k}(c) \sum_{k \in [g]} \frac{D_k(t)\mathbb{I}\{i \in G_k\}}{|G_k|} = R_S(c, w(t))$$

where $w_i(t) \triangleq \sum_{k \in [g]} \frac{D_k(t)\mathbb{I}\{i \in G_k\}}{|G_k|}$ denotes the weight assigned to agent $i$, at time step $t$.

Theorem 4 shows that the time-average of the strategy of the designer in Algorithm 1 is an approximate solution to minimax problem (11).

**Theorem 4.** Suppose that for all $i \in [n]$, $r(\tau_i) \leq B$. Then for all $T > 0$, Algorithm 1 runs in time $O(Tn^2p)$ and the output distribution $C_T$ satisfies

$$\max_{k \in [g]} \mathbb{E}_{c \sim C_T} [R_{G_k}(c)] \leq \widehat{R}_{\text{fair}}(S, p) + B \left( \sqrt{\frac{2 \ln g}{T} + \frac{\ln g}{T}} \right).$$

**Proof.** Note that the action space of the designer $C_p(S)$ and the action set of the adversary $\{G_k\}_{k=1}^g$ are both finite, so our zero-sum game can be written in normal form. Further, $u_k(t) \in [0, 1]$, noting that the return of each agent is in $[0, B]$ — so must be the average return of a group. Therefore,
We note that this can be done in time \( t \). The approximation statement is obtained by noting that for any distribution \( G \), the no-regret player sets \( D(t + 1) = \frac{D_k(t)\beta u_k(t)}{\sum_{h=1}^q D_h(t)\beta w_h(t)} \) for all \( k \in [g] \). The no-regret player sets \( D(t + 1) \) via multiplicative weight update with \( \beta = \frac{1}{1 + \sqrt{2\ln g}} \in (0, 1) \), as follows:

\[
D_k(t + 1) = \frac{D_k(t)\beta u_k(t)}{\sum_{h=1}^q D_h(t)\beta w_h(t)} \quad \forall k \in [g].
\]

\textbf{Algorithm 1:} 2-Player Dynamics for the Ex Ante Minimax Problem

\begin{algorithm}
\textbf{Initialization:} The no-regret player picks the uniform distribution \( D(1) = \left( \frac{1}{g}, \ldots, \frac{1}{g} \right) \in \Delta([g]) \), for \( t = 1, \ldots, T \) do

The best-response player chooses \( c(t) = (c_1(t), \ldots, c_p(t)) \in C_p(S) \) so as to solve

\[
c(t) = \arg\min_{c \in C_p(S)} E_{k-D(t)}[R_{G_k}(c)];
\]

the no-regret player observes \( u_k(t) = R_{G_k}(c(t))/B \) for all \( k \in [g] \). The no-regret player sets \( D(t + 1) \) via multiplicative weight update with \( \beta = \frac{1}{1 + \sqrt{2\ln g}} \in (0, 1) \), as follows:

\[
D_k(t + 1) = \frac{D_k(t)\beta u_k(t)}{\sum_{h=1}^q D_h(t)\beta w_h(t)} \quad \forall k \in [g].
\]

\end{algorithm}

| Output: \( C_T \): the uniform distribution over \( \{c(t)\}^T_{t=1} \).

\[
\text{our minimax game fits the framework of Freund and Schapire [10], and we have that}
\]

\[
\max_{k \in [g]} E_{c \sim \Delta_C} \left[ R_{G_k}(c) \right] \\
\leq \min_{C \in \Delta(C_p(S))} \max_{G \in \Delta([g])} E_{k-G, c \sim \Delta} \left[ R_{G_k}(c) \right] \\
+ B \left( \sqrt{\frac{2\ln g}{T}} + \frac{\ln g}{T} \right). 
\]

The approximation statement is obtained by noting that for any distribution \( G \) over groups,

\[
\max_{G \in \Delta([g])} E_{k-G, c \sim \Delta} \left[ R_{G_k}(c) \right] = \max_{1 \leq k \leq g} E_{c \sim \Delta} \left[ R_{G_k}(c) \right].
\]

With respect to running time, note that at every time step \( t \in [T] \), the algorithm first solves

\[
c(t) = \arg\min_{c \in C_p(S)} E_{k-D(t)}[R_{G_k}(c)].
\]

We note that this can be done in time \( O(n^2p) \) as per Section 3, remembering that

\[
E_{k-D(t)}[R_{G_k}(c)] = R_{S}(c, w(t))
\]

i.e., Equation (12) is a weighted regret minimization problem. Then, the algorithm computes \( u_k(t) \) for each group \( k \), which can be done in time \( O(n) \) (by making a single pass through each customer \( i \) and updating the corresponding \( u_k \) for \( i \in G_k \)). Finally, computing the \( g \) weights takes \( O(g) \leq O(n) \) time. Therefore, each step takes time \( O(n^2p) \), and there are \( T \) such steps, which concludes the proof. \( \square \)

Importantly, Algorithm 1 outputs a distribution over \( p \)-sets of products; Theorem 4 shows that this distribution \( C_T \) approximates the ex ante minimax regret \( \overline{R}_{\text{fair}} \). \( C_T \) can be used to construct a deterministic set of product with good regret guarantees: while each \( p \)-set in the support of \( C_T \) may have high group regret, the union of all such \( p \)-sets must perform at least as well as \( C_T \) and therefore meet benchmarks \( \overline{R}_{\text{fair}} \) and \( R_{\text{fair}} \). However, this union may lead to an undesirable blow-up in the number of deployed products. The experiments in Section 6 show how to avoid such a blow-up in practice.
4.3 An *ex post* Minimax Fair Strategy for Few Groups

In this section, we present a dynamic programming algorithm to find \( p \) products that approximately optimize the *ex post* maximum regret across groups, as per Program (10). In contrast to the no-regret algorithm in the previous section, the runtime of the algorithm here will have an exponential dependence on \( g \), the number of groups.

Recall, the optimization problem is given by

\[
\mathcal{R}_{\text{fair}}(S, p) = \min_{c \in C_p(S)} \left\{ \max_{1 \leq k \leq g} R_{G_k}(c) \right\}
\]  

(10)

The algorithm aims to build a set containing all \( g \)-tuples of average regrets \((R_{G_1}, \ldots, R_{G_g})\) that are simultaneously achievable for groups \( G_1, \ldots, G_g \). However, doing so may be computationally infeasible, as there can be as many regret tuples as there are ways of choosing \( p \) products among \( n \) consumer thresholds, i.e. \(^n_p\) of them. Instead, we discretize the set of possible regret values for each group and build a set that only contains rounded regret tuples via recursion over the number of products \( p \). The resulting dynamic program runs efficiently when the number of groups \( g \) is a small constant and can guarantee an arbitrarily good additive approximation to the minimax regret.

We provide the main guarantee of our dynamic program below and defer the full dynamic program and all technical details to the extended version.

**Theorem 5.** Fix any \( \epsilon > 0 \). There exists a dynamic programming algorithm that, given a collection of consumers \( S = \{\tau_i\}_{i=1}^n \), \( g \) groups \( \{G_k\}_{k=1}^g \), and a target number of products \( p \), finds a product vector \( c \in C_p(S) \) with

\[
\max_{k \in [g]} R_{G_k}(c) \leq \mathcal{R}_{\text{fair}}(S, p) + \epsilon
\]

in time

\[
O\left(n^2p\left(\left\lceil \frac{Bp}{\epsilon} \right\rceil + 1\right)^g\right).
\]

This running time is efficient when \( g \) is small, with a much better dependency in parameters \( p \) and \( n \) than the brute force approach that searches over all \(^n_p\) ways of picking \( p \) products.

4.4 The Special Case of Interval Groups

We now consider the special case where the groups \( G_1, \ldots, G_k \) are defined by disjoint intervals of risk thresholds. This structure can occur in cases where risk tolerances correlate with group membership. For example, groups defined based on age or employee type might correlate strongly with risk thresholds, or we might define the protected groups based on risk thresholds directly.

For this special case we provide an efficient algorithm for finding a collection of \( p \) products that minimize *ex post* minimax group regret.

Our main algorithm in this section is for a decision version of the fair product selection problem, in which we are given interval groups \( G_1, \ldots, G_g \), a number of products \( p \), a target regret \( \kappa \), and the goal is to output a collection of \( p \) products such that every group has average regret at most \( \kappa \) if possible or else output INFEASIBLE. We can convert any algorithm for the decision problem into one that approximately solves the minimax regret problem by performing binary search over the target regret \( \kappa \) to find the minimum feasible value. Finding an \( \epsilon \)-suboptimal set of products requires \( O(\log \frac{B}{\epsilon}) \) runs of the decision algorithm, where \( B \) is a bound on the minimax regret.

Our decision algorithm processes the groups in order of increasing risk thresholds, choosing products \( c^{(k)} \subset G_k \) when processing group \( G_k \). Given the maximum risk product \( x = \max(\cup_{h=1}^{k-1} c^{(h)}) \) chosen for groups \( G_1, \ldots, G_{k-1} \), we say that a product set \( c \subset G_k \) is *valid* for group \( G_k \) with default product \( x \) if the regret of group \( G_k \) is at most \( \kappa \) when offered products \( c \cup \{x\} \). Our algorithm chooses
Algorithm 2: Fair Product Decision Algorithm

Input: Interval groups $G_1, \ldots, G_g$, max products $p$, target regret $\kappa$

1. Let $c \leftarrow \emptyset$
2. For $k = 1, \ldots, g$,
   a. If $V(G_k, p - |c|, \max(c), \kappa) = \emptyset$ output INFEASIBLE
   b. Otherwise, let $c^{(k)}$ be an efficient set of products in $V(G_k, p - |c|, \max(c), \kappa)$.
   c. Let $c \leftarrow c \cup c^{(k)}$.
3. Output $c$.

Lemma 2. For any interval groups $G_1, \ldots, G_k$, number of products $p$, and target regret $\kappa$, Algorithm 2 will output a set of at most $p$ products for which every group has regret at most $\kappa$ if one exists, otherwise it outputs INFEASIBLE.

It remains to provide an algorithm that finds an efficient set of products in the set $V(S, p', x, \kappa)$ if one exists. The following Lemma shows that we can use a slight modification of the dynamic programming algorithm from Theorem 1 to find such a set of products in $O(|S|^2 p')$ time if one exists, or output INFEASIBLE.

Lemma 3. There exists an algorithm for finding an efficient set of products in $V(S, p', x, \kappa)$ if one exists and outputs INFEASIBLE otherwise. The running time of the algorithm is $O(|S|^2 p')$.

Proof. A straightforward modification of the dynamic program described in Section 3 allows us to solve the problem of minimizing the regret of population $S$ when using $p'$ products, and the initial option is given by a product with risk limit $x \leq \tau$ for all $\tau \in S$ (instead of $c_0$). The dynamic program runs in time $O(|S|^2 p')$, and tracks the optimal set of $p''$ products to serve the $z - 1$ lowest risk consumers in $S$ while assuming the remaining consumers are offered product $\tau_z$, for all $z \leq |S|$ and $p'' < p'$. Assuming $V(S, p', x, \kappa)$ is non-empty, one of these $z$'s corresponds to the highest product in an efficient solution, and one of the values of $p''$ corresponds to the number of products used in said efficient solution. Therefore, the corresponding optimal choice of products is efficient: since it is optimal, the regret remains below $\kappa$. It then suffices to search over all values of $p'$ and $z$ after running the dynamic program, which takes an additional time at most $O(|S|^p')$. □
Combined, the above results prove the following result:

**Theorem 6.** There exists an algorithm that, given a collection of consumers divided into interval groups \( S = \{G_k\}_{k=1}^g \) and a number of products \( p \), outputs a set \( c \) of \( p \) products satisfying 
\[
\max_{k \in [g]} \mathbb{R}_{G_k}(c) \leq \mathbb{R}(S, p) + \epsilon
\]
and runs in time \( O(\log(\frac{B}{\epsilon})p \sum_{k=1}^g |G_k|^2) \), where \( B \) is a bound on the maximum regret of any group.

**Proof.** Run binary search on the target regret \( \kappa \) using Algorithm 2 together with the dynamic program. Each run takes \( O(p \sum_{k=1}^g |G_k|^2) \) time, and we need to do \( O(\log \frac{B}{\epsilon}) \) runs. \( \square \)

5  **GENERALIZATION GUARANTEES**

5.1 **Generalization for Regret Minimization Absent Fairness**

Suppose now that there is a distribution \( S \) over consumer risk thresholds. Our goal is to find a collection of \( p \) products that minimizes the expected (with respect to \( S \)) regret of a consumer when we only have access to \( n \) risk limits sampled from \( S \). For any distribution \( S \) over consumer risk limits and any \( p \), we define \( \mathbb{R}_S(c) = \mathbb{E}_{\tau \sim S}[\mathbb{R}(\tau(c))] \), which is the *distributional* counterpart of \( \mathbb{R}_S(c) \).

In Theorem 7, we provide a generalization guarantee that shows it is enough to optimize \( \mathbb{R}_S(c) \) when \( S \) is a sample of size \( n \geq 2B^2\epsilon^{-2} \log(4/\delta) \), all drawn independently from the distribution \( S \).

**Theorem 7 (Generalization Absent Fairness).** Let \( r : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R} \) be any non-decreasing function with bounded range \( B \) and \( S \) be any distribution over \( \mathbb{R}_{\geq 0} \). For any \( \epsilon > 0 \) and \( \delta > 0 \), and for any target number of products \( p \), if \( S = \{\tau_i\}_{i=1}^n \) is drawn independently from \( S \), provided that 
\[
n \geq \frac{2B^2 \log(4/\delta)}{\epsilon^2}
\]
then with probability at least \( 1 - \delta \), we have
\[
\sup_{c \in \mathbb{R}_{\geq 0}^p} |\mathbb{R}_S(c) - \mathbb{R}_S(c)| \leq \epsilon.
\]

The sample complexity bound in Theorem 7 has a standard square dependency in \( 1/\epsilon \) but is independent of the number of products \( p \). This result may come as a surprise, especially given that standard measures of the sample complexity of our function class do depend on \( p \). Indeed, in the extended version, we show that Pollard’s pseudo-dimension [21] for this class of problems is lower bounded by \( p \). The derived sample complexity bounds are loose, requiring at least \( n \geq O(\tilde{B}^2 p / \epsilon^2) \) sampled consumers.

5.2 **Generalization for Fairness Across Several Groups**

In the presence of \( g \) groups, we can think of \( S \) as being a mixture over \( g \) distributions, say \( \{D_k\}_{k=1}^g \), where \( D_k \) is the distribution of group \( G_k \), for every \( k \). We let the weight of this mixture on component \( D_k \) be \( \pi_k \). Let \( \pi_{\min} = \min_{1 \leq k \leq g} \pi_k \). In this framing, a sample \( \tau \sim D \) can be seen as first drawing \( k \sim \pi \) and then \( \tau \sim D_k \). Note that we have sampling access to the distribution \( D \) and cannot directly sample from the components of the mixture: \( \{D_k\}_{k=1}^g \).

**Theorem 8 (Generalization with Fairness).** Let \( r : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R} \) be any non-decreasing function with bounded range \( B \) and \( D \) be any distribution over \( \mathbb{R}_{\geq 0} \). For any \( \epsilon > 0 \) and \( \delta > 0 \), and for any target number of products \( p \), if \( S = \{\tau_i\}_{i=1}^n \) consisting of \( g \) groups \( \{G_k\}_{k=1}^g \) is drawn i.i.d. from \( D \) provided that 
\[
n \geq \frac{2}{\pi_{\min}} \left( \frac{4B^2 \log(8g/\delta)}{\epsilon^2} + \log(2g/\delta) \right)
\]
then with probability at least $1 - \delta$, we have

$$\sup_{C \in \Delta(\mathbb{R}^p_{\geq 0}), k \in [g]} \left| \mathbb{E}_{c \sim C} [R_G_k(c)] - \mathbb{E}_{c \sim C} [R_D_k(c)] \right| \leq \varepsilon.$$

6 EXPERIMENTS

In this section, we present experiments that complement our theoretical results.

6.1 Data

The underlying data for our experiments consists of time series of daily closing returns for 50 publicly traded U.S. equities over a 15-year period beginning in 2005 and ending in 2020; the equities chosen were those with the highest liquidity during this period. From these time series, we extracted the average daily returns and covariance matrix, which we then annualized by the standard practice of multiplying returns and covariances by 252, the number of trading days in a calendar year. The mean annualized return across the 50 stocks is 0.13, and all but one are positive due to the long time period spanned by the data. The correlation between returns and risk (standard deviation of returns) is 0.29 and significant at $P = 0.04$. The annualized returns and covariances are then the basis for the computation of optimal portfolios given a specified risk limit $\tau$ as per Section 2.

In Figure 1(a), we show a scatter plot of risk vs. returns for these 50 stocks. For sufficiently small risk values, the optimal portfolio has almost all of its weight in cash, since all of the equities have higher risk and insufficient independence. At intermediate values of risk, the optimal portfolio concentrates its weight on just the seven stocks highlighted in red in Figure 1(a) and listed in the table in Figure 1(b). This figure also plots the optimal risk-return frontier, which generally lies to the northwest (lower risk and higher return) of the stocks themselves, due to the optimization’s exploitation of independence. The black dot highlights the optimal return for risk tolerance 0.1, for which we show the optimal portfolio weights in Figure 1(b). Note that once the risk tolerance reaches that of the single stock with highest return (the red point lying on the optimal frontier, representing Netflix), the frontier becomes flat, since at that point the optimal portfolio is one fully invested in this single stock.

6.2 Algorithms and Benchmarks

We consider both population and group regret and a number of algorithms: the integer linear program (ILP) for optimizing group regret; an implementation of the no-regret dynamics (NR) for group regret described in Algorithm 1; two strategies for “sparsifying” NR (described below); the dynamic program (DP) of Section 3 for optimizing population regret; and a greedy heuristic, which iteratively chooses the product that reduces population regret the most.\footnote{For ease of understanding, here we consider a restriction of the Markowitz portfolio model of [17] and Equation (1) in which short sales are not allowed, i.e. the weight assigned to each asset must be non-negative.} We will evaluate each of these algorithms on both types of regret and provide further details in the extended version.

Note that the NR algorithm (Algorithm 1) outputs a distribution over sets of $p$ products; a natural way of extracting a fixed set of products is to take the union of the support, but in principle this could lead to far more than $p$ products. We propose two sparsification techniques that extract $p$ products from the support, both of which try to remove redundant products (i.e., products with very similar returns). Our first strategy, which we refer to as NR-Sparse-G, is a greedy algorithm\footnote{In the extended version, we show that the average population return $f_S(c) = \frac{1}{n} \sum_{j} \max_{c_i \leq \tau} r(c_j)$ is submodular, and thus the greedy algorithm, which has the advantage of $O(np)$ running time compared to the $O(n^2p)$ of the DP, also enjoys the standard approximate submodular performance guarantees.}
that repeatedly finds the pair of products with closest returns and removes the higher risk one until only \( p \) products remain. The second algorithm is based on a version of one-dimensional asymmetric \( k \)-center clustering, which we refer to as NR-Sparse-C. NR-Sparse-C optimally selects \( p \) products from the NR support such that the maximum difference in return from any product in the NR support to the closest selected product with lower risk is minimized. Details for NR-Sparse-C are given in the extended version.

![Graph](image)

| Company          | Allocation |
|------------------|------------|
| Apple            | 10.6%      |
| Amazon           | 4.3%       |
| Gilead Sciences  | 2.3%       |
| Monster Beverage | 8.3%       |
| Netflix          | 8.7%       |
| NVIDIA           | 0.3%       |
| Ross Stores      | 6%         |
| Cash reserves    | 59.5%      |

(a) Risk vs. Returns: Scatterplot and Optimal Frontier. (b) Optimal Portfolio for Risk = 0.1.

Fig. 1. Asset Risks and Returns, Optimal Frontier and Portfolio Weights

### 6.3 Experimental Design and Results

Our first experiment compares the population and minimax group regrets of the algorithms we have discussed on two different population distributions as we vary the number of products, \( p \). Both population distributions have \( n = 100 \) consumers divided into \( g = 5 \) groups and the consumers belonging to group \( i \in \{1, \ldots, 5\} \) have risk thresholds drawn from a Gaussian distribution with mean \( \mu_i = i/10 \) and standard deviation \( \sigma_i = 0.01 \). Any negative risk thresholds are set to 0. In the first population distribution, which we call the unbalanced distribution, each consumer belongs to group 2 with probability \( 5/9 \), and each of the remaining groups with probability \( 1/9 \). In the second population distribution, called the balanced distribution, consumers belong to each group with equal probability. We compare the NR algorithm run for \( T = 200 \) iterations, both NR sparsification algorithms, the dynamic program from Section 3, and the greedy algorithm. Results are averaged over 100 populations sampled from the population distribution. Further experimental details are given in the extended version.

**Unbalanced Distribution.** Figure 2 shows the performance of each algorithm averaged over 100 samples from the unbalanced population distribution for \( p \in \{5, \ldots, 9\} \) products. The NR algorithm achieves the lowest population and minimax group regret. However, we are plotting the regret of the union of the products in the support of the output distribution, which contains many more than \( p \) products: for \( p \in \{5, 6, 7, 8, 9\} \), NR used an average of \( \{10.31, 10.3, 10.75, 11.62, 12.53\} \) products on the unbalanced distribution, respectively. The ILP algorithm achieves the optimal minimax group regret with exactly \( p \) products, but is significantly more computationally expensive than the other methods; for \( p = 5 \), the average solve time for the ILP was 17.64 seconds, while the
dynamic program ran in just 0.0008 seconds on average. For larger sets of consumers, solving the ILP becomes intractable. The two NR sparsification algorithms have the next best minimax group regret, and for $p \in \{5, 6, 7\}$, using NR-Sparse-C with $p + 1$ products results in minimax group regret lower than the ILP solution with $p$ products. Finally, the dynamic program and greedy algorithm for minimizing population regret have the worst minimax group regret across all values of $p$. As expected, the dynamic program achieves the optimal average population regret, followed closely by the greedy algorithm.

![Fig. 2. Algorithm Performance on the Unbalanced Population Distribution](image)

Balanced Distribution. Figure 3 plots the same performance measures for the balanced distribution. As in the unbalanced distribution, the NR algorithm achieves the lowest population and minimax group regret, but this is again due to it using more than $p$ products. Unlike the unbalanced distribution, there is less variation in the performance of the algorithms, and our NR sparsification strategies no longer outperform the dynamic program and greedy algorithms on the minimax group regret. One possible explanation is that on the balanced distribution, solutions that minimize the overall population regret also do a reasonably good job at minimizing the minimax group regret, making this an easy case for the DP algorithm. Consider the case where $p = 5 = g$. Intuitively, since the groups form mostly non-overlapping risk intervals, the optimal minimax group regret solution will tend to select one product from each group. On the other hand, since the groups are all equally sized, minimizing the average population regret also requires that we choose products to serve each group (by symmetry, there is no reason for the algorithm to focus on any one group more than any other). When these intuitions hold, an optimal set of products for minimizing the average population regret may be nearly optimal for minimizing the minimax group regret as well. This is consistent with our empirical results, where we see that the gaps in both population and minimax group regret between the ILP and DP solutions are small.

Our second experiment explores generalization. We fix $p = 5$ and use the NR-Sparse-C algorithm to choose products. We draw a test set of 5000 consumers from the unbalanced distribution described above. For sample sizes of $\{25, 50, ..., 500\}$ consumers, we obtain product sets using NR-Sparse-C and calculate the incurred regret using these products on both the training and test sets. We repeat this process 1000 times for each sample size and average them. This is plotted in Figure 4; we observe that measured both by population regret as well as by group regret, the test regret converges to the training regret as the sample size increases. The decay rate is roughly $1/\sqrt{n}$, as suggested by
theory, but our theoretical bound is loose due to sub-optimal constants. Training regret increases with sample size because, for a fixed number of products, it is harder to satisfy a larger number of consumers.

Fig. 3. Algorithm Performance on the Balanced Population Distribution

Fig. 4. Generalization for NR-Sparse-C with $p = 5$ on the unbalanced distribution.

CONCLUSION

Our work is the first (to our knowledge) to study portfolio design from the perspective of fairness, and examining other consumer-facing applications of these classical optimization problems is a rich area for future study. Additionally, we develop a particular definition of group fairness which constrains the maximum regret across all groups, but there may be other notions of fairness to consider in this setting which may lead to different algorithmic results. Finally, we note that because

---

5The theoretical bound is roughly an order of magnitude higher than the experimental bound; we do not plot it as it makes the empirical errors difficult to see.
of the similarity of our problem to a variant of a one-dimensional facility location problem, our work may point to interesting extensions in the fair facility location domain, which is also a relatively new and exciting area of study.

As with many papers in the fairness in machine learning literature, this paper is written with positive broader impacts in mind but also is at risk of providing a technical facade of “fairness” that might mask more serious underlying issues. Our aim in this paper is to provide efficient algorithms to allow financial companies to choose products not just to optimize for the overall benefit for their customers on average (which naturally favors majority groups over minority groups) but to instead optimize for the least well-off subgroup. Providing algorithms for this task, as well as simply clearly laying out the goal, helps lessen the institutional frictions that might otherwise prevent companies from even investigating fairness motivated objectives. This is particularly salient in the finance domain where algorithmic processes such as quantitative trading and so-called ‘robo-advisors’ are already widely used in consumer-facing settings. We therefore see fair algorithms such as those in this work as demonstrating that these practices can be improved for the benefit of those users who may be harmed by a standard profit-maximization approach to algorithm design.

On the other hand, we stress that we are guaranteeing only a particular technical notion of fairness that does not aim to address many underlying financial inequities. There is always a risk with such technologies that they will be applied and then used to justify doing nothing more, because some notion of “fairness” has been satisfied. We additionally recognize that the similarities between the finance context we study and certain kinds of one-dimensional facility location problems may lead to future work that lifts our algorithms into that context. We aim to avoid these outcomes with a clear discussion of what our methods can (and cannot) promise and we urge researchers building on this work to make similar considerations.

REFERENCES

[1] Siddharth Barman and Sanath Kumar Krishnamurthy. 2017. Approximation algorithms for maximin fair division. In Proceedings of the 2017 ACM Conference on Economics and Computation. 647–664.

[2] Richard Bellman. 1973. A note on cluster analysis and dynamic programming. Mathematical Biosciences 18, 3-4 (1973), 311–312.

[3] Eric Budish. 2011. The combinatorial assignment problem: Approximate competitive equilibrium from equal incomes. Journal of Political Economy 119, 6 (2011), 1061–1103.

[4] Danny Z. Chen and Haitao Wang. 2011. New Algorithms for 1-D Facility Location and Path Equipartition Problems. In Algorithms and Data Structures, Frank Dehne, John Iacono, and Jörg-Rüdiger Sack (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 207–218.

[5] Alexandra Chouldechova and Aaron Roth. 2020. A snapshot of the frontiers of fairness in machine learning. Commun. ACM 63, 5 (2020), 82–89.

[6] Emily Diana, Wesley Gill, Michael Kearns, Krishnaram Kenthapadi, and Aaron Roth. 2020. Convergent Algorithms for (Relaxed) Minimax Fairness. arXiv preprint arXiv:2011.03108 (2020).

[7] Kate Donahue and Jon Kleinberg. 2020. Fairness and utilization in allocating resources with uncertain demand. In Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency. 658–668.

[8] Hadi Elzayn, Shahin Jabbari, Christopher Jung, Michael Kearns, Seth Neel, Aaron Roth, and Zachary Schutzman. 2019. Fair algorithms for learning in allocation problems. In Proceedings of the Conference on Fairness, Accountability, and Transparency. 170–179.

[9] Danielle Ensign, Sorelle A Friedler, Scott Neville, Carlos Scheidegger, and Suresh Venkatasubramanian. 2017. Runaway feedback loops in predictive policing. arXiv preprint arXiv:1706.09847 (2017).

[10] Yoav Freund and Robert E Schapire. 1996. Game theory, on-line prediction and boosting. In Proceedings of the ninth annual conference on Computational learning theory. 325–332.

[11] Refael Hassin and Arie Tamir. 1991. Improved complexity bounds for location problems on the real line. Operations Research Letters 10, 7 (1991), 395–402.

[12] Dan A Lancu and Nikolaos Trichakis. 2014. Fairness and efficiency in multiportfolio optimization. Operations Research 62, 6 (2014), 1285–1301.
[13] Christopher Jung, Sampath Kannan, and Neil Lutz. 2019. A Center in Your Neighborhood: Fairness in Facility Location. arXiv preprint arXiv:1908.09041 (2019).

[14] A. Klein. [n.d.]. Reducing bias in AI-based financial services. https://www.brookings.edu/research/reducing-bias-in-ai-based-financial-services/ Accessed: October 06, 2020.

[15] Michelle Seng Ah Lee and Luciano Floridi. 2020. Algorithmic fairness in mortgage lending: from absolute conditions to relational trade-offs. Available at SSRN (2020).

[16] Sepideh Mahabadi and Ali Vakilian. 2020. (Individual) Fairness for $k$-Clustering. arXiv preprint arXiv:2002.06742 (2020).

[17] Harry Markowitz. 1952. PORTFOLIO SELECTION. The Journal of Finance 7, 1 (1952), 77–91. https://doi.org/10.1111/j.1540-6261.1952.tb01525.x arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1540-6261.1952.tb01525.x

[18] Natalie Martinez, Martin Bertran, and Guillermo Sapiro. 2020. Minimax Pareto Fairness: A Multi Objective Perspective. In Proceedings of the 37th International Conference on Machine Learning. Vienna, Austria, PMLR 119.

[19] Robert C Merton. 1972. An analytic derivation of the efficient portfolio frontier. Journal of financial and quantitative analysis (1972), 1851–1872.

[20] Frank Nielsen and Richard Nock. 2014. Optimal interval clustering: Application to Bregman clustering and statistical mixture learning. IEEE Signal Processing Letters 21, 10 (2014), 1289–1292.

[21] David Pollard. 1984. Convergence of stochastic processes. New York: Springer-Verlag.

[22] Ariel D Procaccia and Junxing Wang. 2014. Fair enough: Guaranteeing approximate maximin shares. In Proceedings of the fifteenth ACM conference on Economics and computation. 675–692.

[23] Janis Sarra. 2014. Fairness in financial markets, reconceiving market norms. In Forum on Public Policy: A Journal of the Oxford Round Table. Forum on Public Policy.

[24] Suzanne B Shu and Sylvia Morelli Ucla. 2012. I Applying Fairness Theories to Financial Decision Making. (2012).