

Reducing Leximin Fairness to Utilitarian Optimization

Eden Hartman, Yonatan Aumann, Avinatan Hassidim and Erel Segal-Halevi

Abstract

Two prominent objectives in social choice are *utilitarian* - maximizing the sum of agents' utilities, and *leximin* - maximizing the smallest agent's utility, then the second-smallest, etc. Utilitarianism is typically computationally easier to attain but is generally viewed as less fair. This paper presents a general reduction scheme that, given a utilitarian solver, produces a distribution over states (deterministic outcomes) that is leximin in expectation. Importantly, the scheme is robust in the sense that, given an *approximate* utilitarian solver, it produces a lottery that is approximately-leximin (in expectation) - with the same approximation factor. We apply our scheme to several social choice problems: stochastic allocations of indivisible goods, giveaway lotteries, and fair lotteries for participatory budgeting.

Extended Version: <https://arxiv.org/abs/2409.10395>

1 Introduction

In social choice, the goal is to find the best choice for society, but 'best' can be defined in many ways. Two frequent, and often contrasting definitions are the *utilitarian best*, which focuses on maximizing the total welfare (i.e., the sum of utilities); and the *egalitarian best*, which focuses on maximizing the least utility. The *leximin best* generalizes the egalitarian one. It first aims to maximize the least utility; then, among all options that maximize the least utility, it chooses the one that maximizes the second-smallest utility, among these — the third-smallest utility, and so forth. Leximin is often the solution of choice in social choice applications, and frequently used (e.g., Freeman et al. [13], Bei et al. [6], Cheng et al. [7], Flanigan et al. [12]).

Calculating the Optimal Choice. Calculating a choice that maximizes utilitarian welfare is often easier than finding one that maximizes egalitarian welfare, while finding one that is leximin optimal is typically even more complex. For example, when allocating indivisible goods among agents with additive utilities, finding a choice (in this case, an allocation) that maximizes the utilitarian welfare can be done by greedily assigning each item to the agent who values it most. Finding an allocation that maximizes the egalitarian welfare, however, is NP-hard [5], even in this relatively simple case.

In this paper, we show that knowing how to efficiently maximize the utilitarian welfare is sufficient in order to find a fair leximin solution.

Contributions. The core contribution of this paper is a general protocol that, when provided with a procedure for optimizing the utilitarian welfare (for a given problem), outputs a solution that optimizes the expected leximin welfare (for the same problem). By *expected* leximin we mean a distribution over deterministic solutions, for which the expectations of the players' utilities is leximin optimal. Crucially, our protocol *extends to approximations*, in the following sense: given an *approximate* solver for the utilitarian welfare, the protocol outputs a solution that approximates the expected leximin optimal, and the same approximation factor is preserved. In all, with our protocol at hand, optimizing expected leximin welfare is no more difficult than optimizing utilitarian welfare.

We demonstrate the reduction significance by applying it to three social choice problems as follows.

First, we consider the classic problem of *allocations of indivisible goods*, where one seeks to fairly distribute a set of indivisible goods among a set of heterogeneous agents. Maximizing the utilitarian welfare in this case is well-studied. Using our reduction, the previously mentioned greedy algorithm

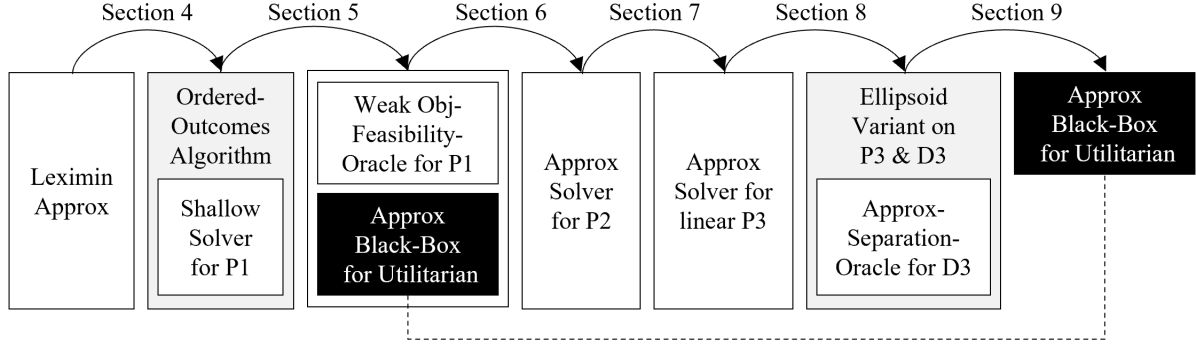


Figure 1: High level description of the reduction algorithm. An arrow from element A to B denotes that the corresponding section reduces problem A to B. White components are implemented in this paper; gray components represent existing algorithms; the black component is the black-box for the utilitarian welfare.

for agents with additive utilities, allows us to achieve a leximin optimal lottery over the allocations in polynomial time. For submodular utilities, approximating leximin to a factor better than $(1 - \frac{1}{e})$ is NP-hard. However, by applying our reduction, existing approximation algorithms for utilitarian welfare can be leveraged to prove that a 0.5-approximation can be obtained deterministically, while the best-possible $(1 - \frac{1}{e})$ -approximation can be obtained with high probability.

Second, we consider the problem of *giveaway lotteries* [2], where there is an event with limited capacity and groups wish to attend, but only-if they can all be admitted together. Maximizing the utilitarian welfare in this setting can be seen as a knapsack problem, for which there is a well-known FPTAS (fully polynomial-time approximation scheme). Using our reduction, we obtain an FPTAS for leximin as well.

Lastly, we consider the problem of *fair lotteries for participatory budgeting*, where one seeks to find a fair lottery over the possible budget allocations. When agents have additive utilities, maximizing the utilitarian welfare can also be interpreted as a knapsack problem (albeit in a slightly different way), which allows us to provide an FPTAS for leximin.

Organization. Section 3 introduces the model and required definitions.

In Sections 4-10, we prove our main result: an algorithm for finding a leximin-approximation using an approximate black-box for utilitarian welfare, while importantly, preserving the same approximation factor. The reduction is done step by step. Section 4 reduces the problem of leximin-approximation to another problem; then, each Section $k \in \{5, 6, 7, 8, 9\}$ reduces the problem introduced in Section $k - 1$ to another problem, where in Section 9 the reduced problem is approximate utilitarian optimization. Section 10 ties the knots to prove the entire reduction, and extends the result to randomized solvers. A schematic description of the reduction structure is provided in Figure 1.

The applications are shown in Section 11. Lastly, Section 12 concludes with some future work directions.

Due to space constraints, all proofs are omitted. Please refer to the [extended version](#) for more details.

1.1 Related Work

Recently, there has been a wealth of research focused on finding a fair leximin lottery for specific problems. Examples include algorithms proposed for Representative Cohort Selection [15], Giveaway Lotteries [2], Allocating Unused Classrooms [19], and Selecting Citizens’ Assemblies [11]. This paper, in contrast, provides a general protocol that can be applied to a wide range of problems.

Alongside these, many papers describe general algorithms for exact leximin optimization [22, 21, 20]. These algorithms usually rely on a solver for single-objective problem, which, in our context, is NP-hard. Recently, Hartman et al. [14] adapted one of these algorithms to work with approximately-optimal solver.

However, designing such a solver remains quite challenging. Our work generalizes these approaches by proving that this algorithm still functions with even a weaker type of solver, which we show can be implemented for many problems.

Another significant area of research focuses on leximin approximations. Since leximin fairness involves multiple objectives simultaneously, it is not straightforward to define what leximin approximation is. Several definitions have been proposed. The definition we employ is related to that of [18, 1]. Other definitions can be found in [14, 15]. An extensive comparison of the different definitions, including examples, is provided in Appendix H.

The work closest to ours is [17], which laid the foundation for this research. Their paper studies the problem of stochastic allocation of indivisible goods (see Section 11.1 for more details), and proposes a reduction from egalitarian welfare to utilitarian welfare for this specific problem. We extend their work in two ways. First, we extend their approach from the allocation problem to any problem where lotteries make sense. Second, we show that a black-box for the utilitarian welfare is even more powerful, as it can also be used for leximin, rather than the egalitarian welfare.

2 Preliminaries

We denote the set $\{1, \dots, n\}$ by $[n]$ for $n \in \mathbb{N}$.

Mathematical Programming. Throughout the paper, we frequently use mathematical programming to define optimization problems. A program is characterized by the following three elements. (1) Constraints: used to define the set of feasible solutions, which forms a subset of $\mathbb{R}^m$ for some $m \in \mathbb{N}$. (2) Type: the program can be either a maximization or minimization program. (3) Objective function: assigns an objective value to each feasible solution. The goal is to find a feasible solution with an optimal objective value (either maximal or minimal, depending on the problem type).

For a given program, P , the *feasible region*, denoted by $F(P)$, represents the set of vectors satisfying all the constraints of P . We say that a vector $\mathbf{v} \in F(P)$ is a *feasible solution for P* (also: a solution for P or feasible for P), and denote its objective value according to the objective function of P by $obj(P, \mathbf{v})$.

3 Model and Definitions

The setting postulates a set of n agents $N = \{1, \dots, n\}$, and a set of deterministic options S – this set represents the possible deterministic allocations – in the fair division setting, or the possible budget allocations – in a budgeting application. For simplicity from now on, we refer to S as *states* and number them $S = \{s_1, \dots, s_{|S|}\}$. We seek solutions that are *distributions* over states. Formally, an *S -distribution* is a probability distribution over the set S , and X is the set of all such distributions:

$$X = \left\{ \mathbf{x} = (x_1, \dots, x_{|S|}) \in \mathbb{R}_{\geq 0}^{|S|} \mid \sum_{j=1}^{|S|} x_j = 1 \right\}.$$

Importantly, we allow the number of states, $|S|$, to be exponential in n . This implies that even describing a solution requires exponential time. However, our algorithms will return *sparse solutions* – that is, the number of states with positive probability will be polynomial in n . These solutions can be efficiently described as a list of states with positive probability, along with their corresponding probabilities.

Following Kawase and Sumita [17], we assume that solutions are represented in *sparse form* – a list of indices of entries with positive values, together with their corresponding values.

Definition 3.1 (A Poly-sparse Vector). *A vector, $\mathbf{v} \in \mathbb{R}_{\geq 0}^m$ for some $m \in \mathbb{N}$, is a poly-sparse if no more than polynomial number (in n) of its entries are non-zero.*

Degenerate-State. We assume that there exists a single degenerate-state, $s_d \in S$, that gives all the agents utility 0. This will allow us to also consider sub-probabilities over the other states, which give positive utility to some agents. The use of degenerate-state makes sense in our setting, as we assume all utilities are non-negative.

Utilities. The utility of agent $i \in N$ from a given state is described by the function $u_i: S \rightarrow \mathbb{R}_{\geq 0}$, which is provided as a value oracle.¹ The utility of agent i from a given S -distribution, $\mathbf{x} \in X$, is the expectations $E_i(\mathbf{x}) = \sum_{j=1}^{|S|} x_j \cdot u_i(s_j)$. The vector of expected utilities of all agents from a solution $\mathbf{x}$ is denoted by $\mathbf{E}(\mathbf{x}) = (E_1(\mathbf{x}), \dots, E_n(\mathbf{x}))$; and referred to as the expected vector of $\mathbf{x}$.

3.1 Leximin Fairness

We aim for a leximin-optimal S -distribution: one that maximizes the smallest expected-utility, then, subject to that, maximizes the second-smallest expected-utility, and so on. It is formally defined below.

The Leximin Order. For $\mathbf{v} \in \mathbb{R}^n$, let $\mathbf{v}^\uparrow$ be the corresponding vector sorted in non-decreasing order, and $v_i^\uparrow$ the i -th smallest element (counting repetitions). For example, if $\mathbf{v} = (1, 4, 7, 1)$ then $\mathbf{v}^\uparrow = (1, 1, 4, 7)$, and $v_3^\uparrow = 4$. For $\mathbf{v}, \mathbf{u} \in \mathbb{R}^n$, we say that $\mathbf{v}$ is (weakly) *leximin-preferred* over $\mathbf{u}$, denoted $\mathbf{v} \succeq \mathbf{u}$, if one of the following holds. Either $\mathbf{v}^\uparrow = \mathbf{u}^\uparrow$ (in which case they are *leximin-equivalent*, denoted $\mathbf{v} \equiv \mathbf{u}$). Or there exists an integer $1 \leq k \leq n$ such that $v_i^\uparrow = u_i^\uparrow$ for $i < k$, and $v_k^\uparrow > u_k^\uparrow$ (in which case $\mathbf{v}$ is *strictly* leximin-preferred over $\mathbf{u}$, denoted $\mathbf{v} \succ \mathbf{u}$). Note that the $\mathbf{E}(\mathbf{x})$'s are n -tuples, so the leximin order applies to them.

Observation 3.1. Let $\mathbf{v}, \mathbf{u} \in \mathbb{R}^n$. Exactly one of the following holds: either $\mathbf{v} \succeq \mathbf{u}$ or $\mathbf{u} \succ \mathbf{v}$.

Leximin Optimal. $\mathbf{x}^* \in X$ is a *leximin-optimal* S -distribution if $\mathbf{E}(\mathbf{x}^*) \succeq \mathbf{E}(\mathbf{x})$ for all $\mathbf{x} \in X$.

Approximation. Throughout the paper, $\alpha \in (0, 1]$ is a multiplicative approximation ratio.

Definition 3.2 (Leximin-Approximation). We say that an S -distribution, $\mathbf{x}^A \in X$, is an α -leximin-approximation if $\mathbf{E}(\mathbf{x}^A) \succeq \alpha \cdot \mathbf{E}(\mathbf{x})$ for all $\mathbf{x} \in X$.

Observation 3.2. An S -distribution is a 1-leximin-approximation if-and-only-if it is leximin-optimal.

3.2 Utilitarian Optimization

Utilitarian Optimal. We say that $\mathbf{x}^{uo} \in X$ is a *utilitarian-optimal* S -distribution if it maximizes the sum of expected utilities. That is, $\sum_{i=1}^n E_i(\mathbf{x}^{uo}) \geq \sum_{i=1}^n E_i(\mathbf{x})$ for any $\mathbf{x} \in X$.

Stochasticity is Unnecessary. In fact, for utilitarian welfare, there always exists a deterministic solution – a single state – that maximizes the sum of expected utilities.

Lemma 3.3. Let $s^{uo} \in \arg \max_{s \in S} \sum_{i=1}^n u_i(s)$. Then $\sum_{i=1}^n u_i(s^{uo}) \geq \sum_{i=1}^n E_i(\mathbf{x})$ for any $\mathbf{x} \in X$.

Algorithms for utilitarian welfare are typically designed for this deterministic setting (where the goal is to find a utilitarian-optimal *state*). Our reduction requires a deterministic solver.

The proposed reduction requires a utilitarian welfare solver that is robust to scaling each utility function by a different constant. Formally:

¹This means that the algorithm does not have a direct access to the utility function. Rather, given $s \in S$, the value $u_i(s)$ can be obtained in $\mathcal{O}(1)$ for any $i = 1, \dots, n$.

(I) α -Approximate Black-Box for Utilitarian Welfare

Input: n non-negative constants $c_1, \dots, c_n$.

Output: A state, $s^{uo} \in S$, for which $\sum_{i=1}^n c_i \cdot u_i(s^{uo}) \geq \alpha \sum_{i=1}^n c_i \cdot u_i(s)$ for any $s \in S$.

When $\alpha = 1$, we say that we have an *exact* black-box.

Many existing solvers for the utilitarian welfare are inherently robust to scaling, as they handle a class of utilities that are closed under this operation. For instance, in the division of goods, various classes of utilities, such as additive and submodular, are closed under constant scaling.

At times, however, rescaling can result in diverging from the problem definition. For example, if the problem definition assumes that utilities are normalized to sum up to 1, then the definition is not robust to rescaling. In such cases, technically, a polynomial solver need not be able to solve the scaled version. Hence, we explicitly include the assumption that the solver is robust to rescaling.

4 Main Loop

The main algorithm used in the reduction is described in Algorithm 1. It is adapted from the Ordered Outcomes algorithm of [20] for finding an exact leximin-optimal solution. It uses a given solver for the maximization program below.

The Program P1. The program is parameterized by an integer $t \in N$ and $t-1$ constants $(z_1, \dots, z_{t-1})$; its only variable is $\mathbf{x}$ (a vector of size $|S|$ representing an S -distribution):

$$\begin{aligned} \max \quad & \sum_{i=1}^t E_i^\uparrow(\mathbf{x}) - \sum_{i=1}^{t-1} z_i & s.t. \quad & (P1.1) \quad \sum_{j=1}^{|S|} x_j = 1 \\ & & & (P1.2) \quad x_j \geq 0 \quad j = 1, \dots, |S| & (P1) \\ & & & (P1.3) \quad \sum_{i=1}^{\ell} E_i^\uparrow(\mathbf{x}) \geq \sum_{i=1}^{\ell} z_i \quad \forall \ell \in [t-1] \end{aligned}$$

Constraints (P1.1–2) simply ensure that $\mathbf{x} \in X$. Constraint (P1.3) says that for any $\ell < t$, the sum of the smallest ℓ expected-utilities is at least the sum of the ℓ constants $z_1, \dots, z_\ell$. The objective of a solution² $\mathbf{x} \in F(\mathbf{P1})$ is the difference between the sum of its smallest t expected-utilities and the sum of the $t-1$ constants $z_1, \dots, z_{t-1}$.

Algorithm 1. The algorithm has n iterations. In each iteration t , it uses the given solver for P1 with the following parameters: the iteration counter t , and $(z_1, \dots, z_{t-1})$ that were computed in previous iterations. The solver returns a solution for this P1, denoted by $\mathbf{x}^t$. If $t < n$ then its objective value, denoted by z_t , is used in the following iterations as an additional parameter. Finally, the solution $\mathbf{x}^n$ generated at the last application of the solver is returned by the main loop.

Notice that the program evolves in each iteration. For any $t \in [n-1]$, the program at iteration $t+1$ differs from the program at iteration t in two ways: first, the objective function changes; and second, an additional constraint is introduced as part of Constraint (P1.3), $\sum_{i=1}^t E_i^\uparrow(\mathbf{x}) \geq \sum_{i=1}^t z_i$ (for $\ell = t$). This constraint is equivalent to: $\sum_{i=1}^t E_i^\uparrow(\mathbf{x}) - \sum_{i=1}^{t-1} z_i \geq z_t$, which essentially ensures that any solution for following programs achieves an objective value at-least z_t according to the objective function of the program at iteration t . In other words, this constraint guarantees that as we continually improving the situation. This implies, in particular, that $\mathbf{x}^t$ remains feasible for the $(t+1)$ -th program.

²See Preliminaries for more details.

Algorithm 1 Main Loop

Input: A solver for **P1**.

- 1: **for** $t = 1$ to n **do**
 - 2: Let $\mathbf{x}^t$ be the solution returned by applying the given solver with t and $(z_1, \dots, z_{t-1})$.
 - 3: Let $z_t := \text{obj}(\mathbf{P1}, \mathbf{x}^t)$.
 - 4: **end for**
 - 5: **return** $\mathbf{x}^n$.
-

Observation 4.1. Let $t \in [n - 1]$. The solution obtained in the t -th iteration of Algorithm 1, $\mathbf{x}^t$, is also feasible for the $(t + 1)$ -th program.

Solving P1 Exactly. By Ogryczak and Śliwiński [20] (Theorem 1), the returned $\mathbf{x}^n$ is leximin-optimal when the solver for **P1** is *exact* - that is, it returns a solution $\mathbf{x}^t \in F(\mathbf{P1})$ with optimal objective value — $\text{obj}(\mathbf{P1}, \mathbf{x}^t) \geq \text{obj}(\mathbf{P1}, \mathbf{x})$ for any solution $\mathbf{x} \in F(\mathbf{P1})$. However, in some cases, no efficient exact solver for **P1** is known. An example is stochastic allocation of indivisible goods among agents with submodular utilities, described in Section 11.1 — in this case, it is NP-hard even for $t = 1$, as its optimal solution maximizes the egalitarian welfare (see [17] for the hardness proof).

Solving P1 Approximately. Our initial attempt to deal with this issue was to follow the approach of Hartman et al. [14]. They consider an *approximately-optimal* solver for **P1**, which returns a solution $\mathbf{x}^t \in F(\mathbf{P1})$ with approximately-optimal objective value — $\text{obj}(\mathbf{P1}, \mathbf{x}^t) \geq \alpha \cdot \text{obj}(\mathbf{P1}, \mathbf{x})$ for any solution $\mathbf{x} \in F(\mathbf{P1})$. When $t = 1$, the algorithm of Kawase and Sumita [17] is an approximate-solver for **P1**. However, for $t > 1$, their technique no longer works due to fundamental differences in the structure of the resulting programs. Designing such a solver is very challenging; all our efforts to design such a solver for several NP-hard problems were unsuccessful.

Solving P1 Shallowly. Our first contribution is to show that Algorithm 1 can work with an even weaker kind of solver for **P1**, that we call a *shallow solver*. The term "shallow" is used since the solver returns a solution whose objective value is optimal only with respect to a *subset* of the feasible solutions, as described below. Recall that s_d is the degenerate-state that gives utility 0 to all agents, and x_d is its probability according to $\mathbf{x}$. We consider the set of solutions that use at-most α fraction of the distribution for the other states, which give positive utility to some agents:

$$X_{\leq \alpha} = \{\mathbf{x} \in X \mid \sum_{j \neq d} x_j \leq \alpha\}.$$

A *shallow-solver* is defined as follows:

(II) α -Shallow-Solver for P1

Input: An integer $t \in N$ and rationals $z_1, \dots, z_{t-1}$.

Output: A solution $\mathbf{x}^t \in F(\mathbf{P1})$ such that $\text{obj}(\mathbf{P1}, \mathbf{x}^t) \geq \text{obj}(\mathbf{P1}, \mathbf{x})$ for any $\mathbf{x} \in F(\mathbf{P1}) \cap X_{\leq \alpha}$.

In words: the solver returns a solution $\mathbf{x}^t \in F(\mathbf{P1})$ whose objective value is guaranteed to be optimal comparing only to solutions that are also in $X_{\leq \alpha}$. This is in contrast to an exact solver, where the objective value of the returned solution is optimal comparing to all solutions.³ Clearly, when $\alpha = 1$, we get an exact solver, as $X_{\leq 1} = X$.

Notice that $\mathbf{x}^t$ does not required to be in $X_{\leq \alpha}$, so its objective value might be *strictly-higher* than the optimal objective value of the set $F(\mathbf{P1}) \cap X_{\leq \alpha}$.

Lemma 4.2. Given an α -shallow-solver for **P1**, Algorithm 1 returns an α -leximin-approximation.

³See Table 1 in Appendix B for comparison of the solvers.

5 A Shallow-Solver for P1

To design a shallow solver for P1, we use a *weak objective-feasibility-oracle*, defined as follows:

(III) α -Weak Objective-Feasibility-Oracle for P1

Input: An integer $t \in N$ and rationals $z_1, \dots, z_{t-1}$, and another rational z_t .

Output: One of the following claims regarding z_t :

- **Feasible:** $\exists \mathbf{x} \in F(\mathbf{P1})$ s.t. $\text{obj}(\mathbf{P1}, \mathbf{x}) \geq z_t$. In this case, the oracle returns such $\mathbf{x}$.
- **Infeasible-Under- $X_{\leq \alpha}$:** $\nexists \mathbf{x} \in F(\mathbf{P1}) \cap X_{\leq \alpha}$ s.t. $\text{obj}(\mathbf{P1}, \mathbf{x}) \geq z_t$.

Note that these claims are not mutually exclusive, as z_t can satisfy both conditions simultaneously. In this case, the oracle may return any one of these claims.

Lemma 5.1. *Given an α -weak objective-feasibility-oracle for P1 (III), an α -approximate black-box for the utilitarian welfare (I), and an arbitrary vector in $F(\mathbf{P1})$, an efficient α -shallow-solver for P1 (II) can be designed.*

6 Weak Objective-Feasibility-Oracle for P1

To design a weak objective-feasibility-oracle for P1, we modify P1 as follows. First, we convert the optimization program P1, to a feasibility program (without an objective), by adding a constraint saying that the objective function of P1 is at least the given constant z_t : $\sum_{i=1}^t E_i^\uparrow(\mathbf{x}) \geq \sum_{i=1}^t z_i$. We then make two changes to this feasibility program: (1) remove Constraint (P1.1) that ensures that the sum of values is 1, and (2) add the objective function: $\min \sum_{j=1}^{|S|} x_j$. We call the resulting program P2:

$$\begin{aligned} \min \quad & \sum_{j=1}^{|S|} x_j & \text{s.t.} \quad & \text{(P2.1)} \quad x_j \geq 0 & j = 1, \dots, |S| \\ & & & & \\ & & \text{(P2.2)} \quad & \sum_{i=1}^{\ell} E_i^\uparrow(\mathbf{x}) \geq \sum_{i=1}^{\ell} z_i & \forall \ell \in [t] \end{aligned} \tag{P2}$$

Note that (P2.2) contains the constraints (P1.3), as well as the new constraint added when converting to a feasibility program. As before, the only variable is $\mathbf{x}$. However, in this program, a vector $\mathbf{x}$ can be feasible without being in X , as its elements are not required to sum to 1.

We shall now prove that a weak objective-feasibility-oracle for P1 can be designed given a solver for P2, which returns a poly-sparse⁴ approximately-optimal solution. As P2 is a *minimization* program and as $\alpha \in (0, 1]$, it is defined as follows:

(IV) $\frac{1}{\alpha}$ -Approximately-Optimal-Sparse-Solver for P2

Input: An integer $t \in N$ and rationals $z_1, \dots, z_t$.

Output: A poly-sparse $\mathbf{x}^A \in F(\mathbf{P2})$ such that $\text{obj}(\mathbf{P2}, \mathbf{x}^A) \leq \frac{1}{\alpha} \text{obj}(\mathbf{P2}, \mathbf{x})$ for any $\mathbf{x} \in F(\mathbf{P2})$.

Lemma 6.1. *Given an $\frac{1}{\alpha}$ -approximately-optimal-sparse-solver for P2 (IV), an α -weak objective-feasibility-oracle for P1 (III) can be obtained.*

⁴A poly-sparse vector (def. 3.1) is one whose number of non-zero values can be bounded by a polynomial in n .

7 Approximately-Optimal Solver for P2

The use of $E^\uparrow()$ operator makes both P1 and P2 non-linear. However, Ogryczak and Śliwiński [20] showed that P1 can be “linearized” by replacing the constraints using $E^\uparrow()$ with a polynomial number of linear constraints. We take a similar approach for P2 to construct the following linear program P3:

$$\begin{aligned}
 \min \quad & \sum_{j=1}^{|S|} x_j \quad s.t. \quad (P3.1) \quad x_j \geq 0 \quad j = 1, \dots, |S| \\
 (P3.2) \quad & \ell y_\ell - \sum_{i=1}^n m_{\ell,i} \geq \sum_{i=1}^{\ell} z_i \quad \forall \ell \in [t] \\
 (P3.3) \quad & m_{\ell,i} \geq y_\ell - \sum_{j=1}^{|S|} x_j \cdot u_i(s_j) \quad \forall \ell \in [t], \forall i \in [n] \\
 (P3.4) \quad & m_{\ell,i} \geq 0 \quad \forall \ell \in [t], \forall i \in [n]
 \end{aligned} \tag{P3}$$

Constraints (P3.2–4) introduce $t(n+1) \leq n(n+1)$ auxiliary variables: y_ℓ and $m_{\ell,i}$ for all $\ell \in [t]$ and $i \in [n]$. We formally prove the equivalence between the two sets of constraints in Appendix D. We show that it implies that the required solver for P2 can be derived from the same type of solver for P3.

(V) $\frac{1}{\alpha}$ -Approximately-Optimal-Sparse-Solver for P3

Input: An integer $t \in N$ and rationals $z_1, \dots, z_t$.

Output: A poly-sparse $(\mathbf{x}^A, \mathbf{y}^A, \mathbf{m}^A) \in F(\text{P3})$ such that for any $(\mathbf{x}, \mathbf{y}, \mathbf{m}) \in F(\text{P3})$:

$$\text{obj}(\text{P3}, (\mathbf{x}^A, \mathbf{y}^A, \mathbf{m}^A)) \leq \frac{1}{\alpha} \text{obj}(\text{P3}, (\mathbf{x}, \mathbf{y}, \mathbf{m}))$$

Lemma 7.1. *Given an $\frac{1}{\alpha}$ -approximately-optimal-sparse-solver for P3 (V), an $\frac{1}{\alpha}$ -approximately-optimal-sparse-solver for P2 (IV) can be obtained.*

8 Approximately-Optimal Solver for P3

P3 is a linear program, but has more than $|S|$ variables. Although $|S|$ (the number of states) may be exponential in n , P3 can be approximated in polynomial time using a variant of the ellipsoid method, similarly to Karmarkar and Karp [16], as described below. The method uses an approximate separation oracle for the dual of the linear program P3:

$$\begin{aligned}
 \max \quad & \sum_{\ell=1}^t q_\ell \sum_{i=1}^{\ell} z_i \quad s.t. \quad (D3.1) \quad \sum_{i=1}^n u_i(s_j) \sum_{\ell=1}^t v_{\ell,i} \leq 1 \quad \forall j = 1, \dots, |S| \\
 (D3.2) \quad & \ell q_\ell - \sum_{i=1}^n v_{\ell,i} \leq 0 \quad \forall \ell \in [t] \\
 (D3.3) \quad & -q_\ell + v_{\ell,i} \leq 0 \quad \forall \ell \in [t], \forall i \in [n] \\
 (D3.4) \quad & q_\ell \geq 0 \quad \forall \ell \in [t] \\
 (D3.5) \quad & v_{\ell,i} \geq 0 \quad \forall \ell \in [t], \forall i \in [n]
 \end{aligned} \tag{D3}$$

Similarly to P3, the program D3 is parameterized by an integer t , and rational numbers $(z_1, \dots, z_t)$. It has a polynomial number of variables: q_ℓ and $v_{\ell,j}$ for any $\ell \in [t]$ and $j \in [n]$; and a potentially exponential number of constraints due to (D3.1); see Appendix E for derivation.

We prove that the required solver for **P3** can be designed given the following procedure for its dual **D3**:

(VI) $\frac{1}{\alpha}$ -Approximate-Separation-Oracle for **D3**

Input: An integer $t \in N$, rationals $z_1, \dots, z_t$, and a potential assignment of the variables $(\mathbf{q}, \mathbf{v})$.

Output: One of the following regarding $(\mathbf{q}, \mathbf{v})$:

- **Infeasible:** At least one of the constraints is violated by $(\mathbf{q}, \mathbf{v})$. In this case, the oracle returns such a constraint.
- **$\frac{1}{\alpha}$ -Approximately-Feasible:** All the constraints are $\frac{1}{\alpha}$ -approximately-maintained — the left-hand side of the inequality is at least $\frac{1}{\alpha}$ times the its right-hand side.

In Appendix F, we present the variant of the ellipsoid method that, given a $\frac{1}{\alpha}$ -approximate-separation oracle for the (max.) dual program, allows us to obtain a sparse $\frac{1}{\alpha}$ -approximation to the (min.) primal program (Lemma F.1). This allows us to conclude the following:

Corollary 8.1. *Given a $\frac{1}{\alpha}$ -approximate-separation-oracle for **D3** (VI), a $\frac{1}{\alpha}$ -approximately-optimal-sparse-solver for **P3** (V) can be derived.*

9 Approximate Separation Oracle for **D3**

Now, we design the required oracle using the given approximate black-box for the utilitarian welfare.

Lemma 9.1. *Given an α -approximate black-box for the utilitarian welfare (I), a $\frac{1}{\alpha}$ -approximate-separation-oracle for **D3** (VI) can be constructed.*

10 The Main Result

Putting it all together, we obtain:

Theorem 10.1. *Given an α -approximate black-box for the utilitarian welfare (I). An α -leximin-approximation (Def. 3.2) can be computed in time polynomial in n and the running time of the black-box.*

Proof. By Lemma 9.1, given an α -approximate black-box for utilitarian welfare, a $\frac{1}{\alpha}$ -approximate-separation-oracle for **D3** can be constructed.

By Corollary 8.1, using this oracle, we can construct a $\frac{1}{\alpha}$ -approximately-optimal-sparse-solver for **P3**.

By Lemma 7.1, we can use this solver to design a $\frac{1}{\alpha}$ -approximately-optimal-sparse-solver for **P2**.

By Lemma 6.1, this solver allows us to construct an α -weak objective-feasibility-oracle for **P1**.

By Lemma 5.1, a binary search with (1) this procedure, (2) the given α -approximate black-box for utilitarian welfare, and (3) an arbitrary solution for **P1** (as follows); allows us to design an α -shallow-solver for **P1**. As an arbitrary solution, for $t = 1$, we take $\mathbf{x}^0$ defined as the S -distribution with $x_d^0 = 1$ and $x_j^0 = 0$ for $j \neq d$. However, for $t \geq 2$, we rely on the fact that the shallow solver is used within the iterative Algorithm 1, and take the solution returned in the previous iteration, $\mathbf{x}^{t-1}$, which, by Observation 4.1, is feasible for the program at iteration t as well.

By Lemma 4.2, when this shallow solver is used in Alg.1, it outputs an α -leximin approximation. $\square$

Together with Observation 3.2, this implies:

Corollary 10.2. *Given an exact black-box for the utilitarian welfare, a leximin-optimal S -distribution can be obtained in polynomial time.*

10.1 Randomized Solvers

A *randomized* black-box returns a state that α -approximates the utilitarian welfare with probability $p > 0$, otherwise returns an arbitrary state. Theorem 10.1 can be extended as follows:

Theorem 10.3. *Given a randomized α -approximate black-box for the utilitarian welfare with success probability $p \in (0, 1)$. An α -leximin-approximation can be obtained with the same success probability p in time polynomial in n and the running time of the black-box.*

11 Applications

This section provides three applications of our general reduction framework. Each application employs a different black-box for the associated utilitarian welfare.

11.1 Stochastic Indivisible Allocations

In the problem of *fair stochastic allocations of indivisible goods*, described by Kawase and Sumita [17], there is a set of m indivisible goods, G , that needs to be distributed fairly among the n agents.

The Set S . The states are the possible allocations of the goods to the agents. Each state can be described by a function mapping each good to the agent who gets it. Accordingly, $S = \{s \mid s: G \rightarrow N\}$, and $|S| = n^m$. We assume that agents care only about their own share, so we can abuse notation and let each u_i take a bundle B of goods. The utilities are assumed to be normalized such that $u_i(\emptyset) = 0$, and monotone – $u_i(B_1) \leq u_i(B_2)$ if $B_1 \subseteq B_2$. Under these assumptions, different black-boxes for the utilitarian welfare exist.

The Utilitarian Welfare. For any n constants $c_1, \dots, c_n$, the goal is to maximize the following:

$$\max_{s \in S} \sum_{i=1}^n c_i \cdot u_i(s)$$

Many algorithms for approximating utilitarian welfare are already designed for classes of utilities, which are closed under multiplication by a constant. This means that, given such an algorithm for the original utilities $(u_i)_{i \in N}$, we can use it as-is for the utilities $(c_i \cdot u_i)_{i \in N}$.

Results. When the utilities are additive, it is known that a leximin-optimal solution can be found in polynomial time by introduce variables describing the allocation probability of each agent–item pair, and then solve it using an iterative linear-programming algorithm (e.g., [24, 20]). However, this result can also be derived independently from our approach, since utilitarian welfare maximization is computationally easy—it can be solved in polynomial time by greedily assigning each item to the agent who values it most. Together with Corollary 10.2:

Corollary 11.1. *For additive utilities, a leximin-optimal S -distribution can be obtained in poly. time.*

When the utilities are submodular, approximating leximin to a factor better than $(1 - \frac{1}{e})$ is NP-hard [17].⁵ However, as there is a deterministic $\frac{1}{2}$ -approximation algorithm for the utilitarian welfare [10], by Theorem 10.1:

⁵Kawase and Sumita [17] prove that approximating the egalitarian welfare to a factor better than $(1 - \frac{1}{e})$ is NP-hard. However, since an α -leximin-approximation is first-and-foremost an α -approximation to the egalitarian welfare, the same hardness result applies to leximin as well.

Corollary 11.2. *For submodular utilities, a $\frac{1}{2}$ -leximin-approximation can be found in polynomial time.*

There is also a randomized $(1 - \frac{1}{e})$ -approximation algorithm for the case where utilities are submodular, with high success probability [23]. Thus, by Theorem 10.3:

Corollary 11.3. *For submodular utilities, a $(1 - \frac{1}{e})$ -leximin-approximation can be obtained with high probability in polynomial time.*

11.2 Giveaway Lotteries

In *giveaway lotteries*, described by Arbiv and Aumann [2], there is an event with a limited capacity, and groups who wish to attend it - but only-if all the members of the group can attend together. Here, each group of people is an agent. We denote the size of group i by $w_i \in \mathbb{N}_{\geq 0}$ and the event capacity by $W \in \mathbb{N}_{\geq 0}$. It is assumed that $w_i \leq W$ for $i \in N$ and $\sum_{i \in N} w_i > W$.⁶

The Set S . Each state describes a set of the groups that can attend the event together: $S = \{s \subseteq N \mid \sum_{i \in s} w_i \leq W\}$. Here, $|S|$ is only bounded by 2^n . The utility of group $i \in N$ from a state s is 1 if they being chosen according to s (i.e., if $i \in s$) and 0 otherwise.

The Utilitarian Welfare. For any n constants $c_1, \dots, c_n$, the goal is to maximize the following:

$$\max_{s \in S} \sum_{i=1}^n c_i \cdot u_i(s) = \max_{s \in S} \sum_{i \in s} c_i$$

This is just a knapsack problem with n item (one for each group), where the weights are the group sizes w_i (as we only look at the legal packing $s \in S$), and the values are the constants c_i .

Result. It is well known that there is an FPTAS for the Knapsack problem. By Theorem 10.1:

Corollary 11.4. *There exists an FPTAS for leximin for the problem of giveaway lotteries.*

11.3 Participatory Budgeting Lotteries

The problem of *fair lotteries for participatory budgeting*, was described by Aziz et al. [4]. Here, the n agents are voters, who share a common budget $B \in \mathbb{R}_{>0}$ and must decide which projects from a set P to fund. Each voter, $i \in N$, has an *additive* utility over the set of projects, u_i ; while the projects have costs described by $\text{cost}: P \rightarrow \mathbb{R}_{>0}$.⁷

The set S . The states are the subsets of projects that fit in the given budget: $S = \{s \subseteq P \mid \text{cost}(s) \leq B\}$. The size of S in this problem is only bounded by $2^{|P|}$.

The Utilitarian Welfare. For any n constants $c_1, \dots, c_n$, the goal is to maximize the following:

$$\begin{aligned} \max_{s \in S} \sum_{i=1}^n c_i \cdot u_i(s) &= \max_{s \in S} \sum_{i=1}^n \sum_{p \in s} c_i \cdot u_i(p) && \text{(Additivity)} \\ &= \max_{s \in S} \sum_{p \in s} \left(\sum_{i=1}^n c_i \cdot u_i(p) \right) \end{aligned}$$

This can also be seen as a knapsack problem where: the items are the projects, the weights are the costs, and the value of item $p \in P$ is $\sum_{i=1}^n c_i \cdot u_i(p)$.

Result. The existence of a FPTAS for the Knapsack problem together with Theorem 10.1, give:

Corollary 11.5. *There is an FPTAS for leximin for participatory budgeting lotteries.*

⁶Arbiv and Aumann [2] provide an algorithm to compute a leximin-optimal solution. However, their algorithm is polynomial only for a unary representation of the capacity.

⁷Aziz et al. [4] study fairness properties based on *fair share* and *justified representation*.

12 Conclusion and Future Work

In this work, we establish a strong connection between leximin fairness and utilitarian optimization, demonstrated by a reduction. It is robust to errors in the sense that, given a black-box that approximates the utilitarian value, a leximin-approximation with respect to the same approximation factor can be obtained in polynomial time.

Negative Utilities. Our current technique requires to assume that the utilities are non-negative since we use the degenerate-state. For non-positive utilities, we believe that a similar reduction might still be possible by redefining the degenerate state to represent a lower bound on the worst outcome with respect to negative utilities. For example, in chores allocation, an appropriate choice for the degenerate state can be an artificial allocation in which every agent is assigned all tasks. We note that combining positive and negative utilities poses even more challenges – as even the meaning of multiplicative approximation in this case is unclear.

Nash Welfare. The Nash welfare (product of utilities) is another prominent objective in social choice, offering a compelling compromise between the efficiency of utilitarianism and the fairness of egalitarianism. From a computational perspective, maximizing Nash welfare is typically as challenging as maximizing egalitarian welfare. An interesting question is whether a similar reduction can be constructed from Nash welfare (in expectation) to utilitarian optimization.

Applications. We believe this method can also be applied to a variety of other problems, such as: Selecting a Representative Committee [15], Allocating Unused Classrooms [19], Selecting Citizens’ Assemblies [11], Cake-Cutting [3, 9], Nucleolus [8].

Best-of-both-worlds. Aziz et al. [4], who study participatory budgeting lotteries, focus on fairness that is both ex-ante (which is a guarantee on the distribution) and ex-post (which is a guarantee on the deterministic support). In this paper, we guarantee only ex-ante fairness. Can ex-post fairness be achieved alongside it? We note that our method ensures both ex-ante and ex-post efficiency – since leximin guarantees Pareto-optimality with respect to the expected utilities E_i , any state in the support is Pareto-optimal with respect to the utilities u_i .

Deterministic Setting. When the objective is to find a leximin-optimal (or approximate) *state* rather than a distribution, it remains an open question whether a black-box for utilitarian welfare can still contribute, even if for a different approximation factor.

Truthfulness. Arbiv and Aumann [2], who study giveaway lotteries, prove that a leximin-optimal solution is not only fair but also truthful in this case. Can our leximin-approximation be connected to some notion of truthfulness?

Leximin-Approximation Definitions. This paper suggests a weaker definition of leximin-approximation than the one proposed by Hartman et al. [14] (see Appendix H for more details). Can similar results be obtained with the stronger definition?

Weighted Leximin. A natural extension is to consider weighted leximin, where agents are assigned different weights that determine their relative priority in the lexicographic ordering. For instance, if agent a_1 has weight 1 and agent a_2 has weight 2, then an allocation that gives a_1 utility of 1 and a_2 a utility of 2 would be preferred over allocation that gives both utility of 1.5. Since our framework already relies on access to a weighted utilitarian black box, we expect that the reduction extends to this setting as well, though a careful analysis is left for future work.

Acknowledgments

This research is partly supported by the Israel Science Foundation grants 712/20, 1092/24, 2697/22, and 3007/24. We sincerely appreciate Arkadi Nemirovski, Yasushi Kawase, Nikhil Bansal, Shaddin Dughmi, Edith Elkind, and Dinesh Kumar Baghel for their valuable insights, helpful answers, and clarifications. We are also grateful to the following members of the stack exchange network for their very helpful answers to our technical questions: Neal Young,⁸ 1Rock,⁹ Mark L. Stone,¹⁰ Rob Pratt,¹¹ mtanneau,¹² and mhdadk.¹³ Lastly, we would like to thank the reviewers in AAAI 2025 and COMSOC 2025 for their helpful comments.

References

- [1] Jacob D Abernethy, Robert Schapire, and Umar Syed. Lexicographic optimization: Algorithms and stability. In *International Conference on Artificial Intelligence and Statistics*, pages 2503–2511. PMLR, 2024. URL <https://api.semanticscholar.org/CorpusID:269502696>.
- [2] Tal Arbiv and Yonatan Aumann. Fair and Truthful Giveaway Lotteries. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(5):4785–4792, June 2022. ISSN 2374-3468, 2159-5399. doi: 10.1609/aaai.v36i5.20405. URL <https://ojs.aaai.org/index.php/AAAI/article/view/20405>.
- [3] Yonatan Aumann, Yair Dombb, and Avinatan Hassidim. Computing socially-efficient cake divisions. In *International conference on Autonomous Agents and Multi-Agent Systems, AAMAS*, pages 343–350. IFAAMAS, 2013. URL <http://dl.acm.org/citation.cfm?id=2484976>.
- [4] Haris Aziz, Xinhang Lu, Mashbat Suzuki, Jeremy Vollen, and Toby Walsh. Fair lotteries for participatory budgeting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 9469–9476, 2024. URL <https://doi.org/10.1609/aaai.v38i9.28801>.
- [5] Nikhil Bansal and Maxim Sviridenko. The santa claus problem. In *Proceedings of the Thirty-Eighth Annual ACM Symposium on Theory of Computing, STOC '06*, page 31–40, New York, NY, USA, 2006. Association for Computing Machinery. ISBN 1595931341. doi: 10.1145/1132516.1132522. URL <https://doi.org/10.1145/1132516.1132522>.
- [6] Xiaohui Bei, Xinhang Lu, and Warut Suksompong. Truthful Cake Sharing. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(5):4809–4817, June 2022. doi: 10.1609/aaai.v36i5.20408. URL <https://ojs.aaai.org/index.php/AAAI/article/view/20408>.
- [7] Hao Cheng, Shufeng Kong, Yanchen Deng, Caihua Liu, Xiaohu Wu, Bo An, and Chongjun Wang. Exploring leximin principle for fair core-selecting combinatorial auctions: Payment rule design and implementation. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, pages 2581–2588. ijcai.org, 2023. doi: 10.24963/IJCAI.2023/287. URL <https://doi.org/10.24963/ijcai.2023/287>.
- [8] Edith Elkind, Leslie Ann Goldberg, Paul W. Goldberg, and Michael J. Wooldridge. On the computational complexity of weighted voting games. *Ann. Math. Artif. Intell.*, 56(2):109–131, 2009. doi: 10.1007/S10472-009-9162-5. URL <https://doi.org/10.1007/s10472-009-9162-5>.
- [9] Edith Elkind, Erel Segal-Halevi, and Warut Suksompong. Mind the gap: Cake cutting with separation. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI*, pages 5330–5338. AAAI Press, 2021. doi: 10.1609/AAAI.V35I6.16672. URL <https://doi.org/10.1609/aaai.v35i6.16672>.

⁸<https://cstheory.stackexchange.com/questions/51206> and <https://cstheory.stackexchange.com/questions/51003> and <https://cstheory.stackexchange.com/questions/52353>

⁹<https://math.stackexchange.com/questions/4466551>

¹⁰<https://or.stackexchange.com/questions/8633> and <https://or.stackexchange.com/questions/11007> and <https://or.stackexchange.com/questions/11311>

¹¹<https://or.stackexchange.com/questions/8980>

¹²<https://or.stackexchange.com/questions/11910>

¹³<https://or.stackexchange.com/questions/11308>

- [10] Marshall L Fisher, George L Nemhauser, and Laurence A Wolsey. *An analysis of approximations for maximizing submodular set functions—II*, pages 73–87. Springer Berlin Heidelberg, Berlin, Heidelberg, 1978. ISBN 978-3-642-00790-3. doi: 10.1007/BFb0121195. URL <https://doi.org/10.1007/BFb0121195>.
- [11] Bailey Flanigan, Paul Gözl, Anupam Gupta, Brett Hennig, and Ariel D Procaccia. Fair algorithms for selecting citizens’ assemblies. *Nature*, 596(7873):548–552, 2021.
- [12] Bailey Flanigan, Jennifer Liang, Ariel D. Procaccia, and Sven Wang. Manipulation-robust selection of citizens’ assemblies. In Michael J. Wooldridge, Jennifer G. Dy, and Sriraam Natarajan, editors, *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 9696–9703. AAAI Press, 2024. doi: 10.1609/AAAI.V38I9.28827. URL <https://doi.org/10.1609/aaai.v38i9.28827>.
- [13] Rupert Freeman, Sujoy Sikdar, Rohit Vaish, and Lirong Xia. Equitable allocations of indivisible goods. In Sarit Kraus, editor, *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI-2019*, pages 280–286. ijcai.org, 2019. doi: 10.24963/IJCAI.2019/40. URL <https://doi.org/10.24963/ijcai.2019/40>.
- [14] Eden Hartman, Avinatan Hassidim, Yonatan Aumann, and Erel Segal-Halevi. Leximin approximation: From single-objective to multi-objective. In *ECAI 2023*, pages 996–1003. IOS Press, 2023. URL <https://doi.org/10.3233/FAIA230371>.
- [15] Monika Henzinger, Charlotte Peale, Omer Reingold, and Judy Hanwen Shen. Leximax Approximations and Representative Cohort Selection. In L. Elisa Celis, editor, *3rd Symposium on Foundations of Responsible Computing (FORC 2022)*, volume 218 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 2:1–2:22, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISBN 978-3-95977-226-6. doi: 10.4230/LIPIcs.FORC.2022.2. URL <https://drops.dagstuhl.de/opus/volltexte/2022/16525>.
- [16] Narendra Karmarkar and Richard M. Karp. An efficient approximation scheme for the one-dimensional bin-packing problem. In *23rd Annual Symposium on Foundations of Computer Science (sfcs 1982)*, pages 312–320, Chicago, IL, USA, November 1982. IEEE. doi: 10.1109/SFCS.1982.61. URL <http://ieeexplore.ieee.org/document/4568405/>.
- [17] Yasushi Kawase and Hanna Sumita. On the Max-Min Fair Stochastic Allocation of Indivisible Goods. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(02):2070–2078, April 2020. ISSN 2374-3468, 2159-5399. doi: 10.1609/aaai.v34i02.5580. URL <https://ojs.aaai.org/index.php/AAAI/article/view/5580>.
- [18] Jon Kleinberg, Yuval Rabani, and Eva Tardos. Fairness in routing and load balancing. *Journal of Computer and System Sciences*, 63(1):2–20, 2001. ISSN 0022-0000. doi: <https://doi.org/10.1006/jcss.2001.1752>. URL <https://www.sciencedirect.com/science/article/pii/S0022000001917520>.
- [19] David Kurokawa, Ariel D. Procaccia, and Nisarg Shah. Leximin allocations in the real world. *ACM Trans. Economics and Comput.*, 6(3-4):11:1–11:24, 2018. doi: 10.1145/3274641. URL <https://doi.org/10.1145/3274641>.
- [20] Włodzimierz Ogryczak and Tomasz Śliwiński. On direct methods for lexicographic min-max optimization. In Marina Gavrilova, Osvaldo Gervasi, Vipin Kumar, C. J. Kenneth Tan, David Taniar, Antonio Laganá, Youngsong Mun, and Hyunseung Choo, editors, *Computational Science and Its Applications - ICCSA 2006*, pages 802–811, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg. ISBN 978-3-540-34076-8.
- [21] Włodzimierz Ogryczak, Michał Pióro, and Artur Tomaszewski. Telecommunications network design and max-min optimization problem. *Journal of telecommunications and information technology*, 4:43–53, 2004.
- [22] Włodzimierz Ogryczak. On the lexicographic minimax approach to location problems. *European Journal of Operational Research*, 100(3):566–585, 1997. ISSN 0377-2217. doi: [https://doi.org/10.1016/S0377-2217\(96\)00154-3](https://doi.org/10.1016/S0377-2217(96)00154-3). URL <https://www.sciencedirect.com/science/article/pii/S0377221796001543>.
- [23] Jan Vondrak. Optimal approximation for the submodular welfare problem in the value oracle model. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 67–74, Victoria

British Columbia Canada, May 2008. ACM. ISBN 978-1-60558-047-0. doi: 10.1145/1374376.1374389.
URL <https://dl.acm.org/doi/10.1145/1374376.1374389>.

- [24] Stephen J. Willson. Fair division using linear programming. preprint, Departement of Mathematics, Iowa State University, 1998.

Eden Hartman
Bar-Ilan University, Israel
Email: eden.r.hartman@gmail.com

Yonatan Aumann
Bar-Ilan University, Israel
Email: aumann@cs.biu.ac.il

Avinatan Hassidim
Bar-Ilan University and Google, Israel
Email: avinatan@cs.biu.ac.il

Erel Segal-Halevi
Ariel University, Israel
Email: erelsgl@gmail.com