Leximin Approximation: From Single-Objective to Multi-Objective

Eden Hartman^{a;*}, Avinatan Hassidim^{a,b}, Yonatan Aumann^a and Erel Segal-Halevi^c

^aBar-Ilan University, Israel ^bGoogle, Israel ^cAriel University, Israel

Abstract. Leximin is a common approach to multi-objective optimization, frequently employed in fair division applications. In leximin optimization, one first aims to maximize the smallest objective value; subject to this, one maximizes the second-smallest objective; and so on. Often, even the single-objective problem of maximizing the smallest value cannot be solved accurately. What can we hope to accomplish for leximin optimization in this situation? Recently, Henzinger et al. (2022) defined a notion of *approximate* leximin optimality. Their definition, however, considers only an additive approximation.

In this work, we first define the notion of approximate leximin optimality, allowing both multiplicative and additive errors. We then show how to compute, in polynomial time, such an approximate leximin solution, using an oracle that finds an approximation to a single-objective problem. The approximation factors of the algorithms are closely related: an (α, ϵ) -approximation for the single-objective problem (where $\alpha \in (0, 1]$ and $\epsilon \ge 0$ are the multiplicative and additive factors respectively) translates into an $\left(\frac{\alpha^2}{1-\alpha+\alpha^2}, \frac{\epsilon}{1-\alpha+\alpha^2}\right)$ -approximation for the multi-objective leximin problem, regardless of the number of objectives.

Finally, we apply our algorithm to obtain an approximate leximin solution for the problem of *stochastic allocations of indivisible goods*.

1 Introduction

Many real life scenarios involve more than one objective. These situations are often modeled as *multi-objective optimization problems*, which include defining the set of possible decisions, along with functions that describe the different objectives. As a concrete example, we use the context of social choice, in which the objective functions represent people's utilities. Different criteria can be used to determine optimality when considering multi-objectives. For example, the *utilitarian* criterium aims to maximize the sum of utilities, while the *egalitarian* criterium aims to maximize the least utility. This paper focuses on the *leximin* criterium, according to which one aims to maximize the least utility, and, subject to this, maximize the secondsmallest utility, and so on. In the context of social choice, the leximin criterium is usually mentioned in the context of fairness, as strives to benefit, as much as possible, the least fortunate in society. Common algorithms for finding a leximin optimal solution are iterative, optimizing one or more single-objective optimization problems at each iteration (for example [1, 2, 3, 16, 17, 21]). Often, these single-objective problems cannot be solved exactly (e.g. when they are computationally hard, or when there are numeric inaccuracies in the solver), but can be solved approximately. In this work, we define an approximate variant of leximin and show how such an approximation can be computed, given approximate single-objective solvers.

The Challenge When single-objective solvers only *approximate* the optimal value, existing methods for extending the solvers to leximin optimally may fail, as we illustrate next.

A common algorithm, independently proposed many times, e.g. [1, 16, 17, 21], is based on the notion of *saturation*, operates roughly as follows. In the first iteration, the algorithm looks for the maximum value that all objective functions can achieve simultaneously, z_1 , and then it determines which of the objective-functions are saturated — that is, cannot achieve more than z_1 given that the others do. Afterwards, in each iteration t, given that for any i < t the objective-functions that were determined saturated in the *i*'th iteration achieve at least z_i , it looks for the maximum value that all other objective-functions can achieve simultaneously, z_t , and then determines which of those functions are saturated. When all functions become saturated, the algorithm ends.

Now, the following simple example demonstrates the problem that may arise when the individual solver may return sub-optimal results. Consider the following problem:

lex max min {
$$f_1(x) = x_1, f_2(x) = x_2$$
}
s.t. (1) $x_1 + x_2 \le 1$, (2) $x \in \mathbb{R}^2_+$

As f_1 and f_2 are symmetric, the leximin optimal solution in this case is (0.5, 0.5). Now suppose that rather than finding the exact value 0.5, the solver returns the value 0.49. The optimal value of f_1 given that f_2 achieves at least 0.49 is 0.51, and vice versa for f_2 . As a consequence, none of the objective functions would be determined saturated, and the algorithm may not terminate. One could perhaps define an objective as "saturated" if its maximum attainable value is close to the maximum z_t , but there is no guarantee that this would lead to a good approximation¹.

Contributions This paper studies the problem of leximin optimization in multi-objective optimization problems, focusing on prob-

^{*} A full version of this paper can be found in https://arxiv.org/ abs/2303.12506. Contact information: eden.r.hartman@gmail.com, avinatan@cs.biu.ac.il, aumann@cs.biu.ac.il, erelsgl@gmail.com.

¹ An example will be given on request.

lems for which even the single-objective problems cannot be solved exactly in polynomial time. Our contribution is threefold.

First, a new definition of leximin approximation is presented. It captures both multiplicative and additive errors. The definition has several desirable properties, including that a leximin optimal solution is also approximately-optimal (for any approximation factor), and that the definition is equivalent to the original one in the absence of errors.

Second, an algorithm is provided that, given an approximation algorithm for a single-objective problem, computes a leximin approximation to the multi-objective problem. The algorithm was first presented by Ogryczak and Śliwiński [18] for exact leximinoptimization. In contrast to the saturation-based algorithm described in the Introduction, this algorithm always terminates, even when the single-objective solver is inaccurate. Moreover, the accuracy of the returned solution is closely correlated with the accuracy of the single-objective solver — given an (α, ϵ) -approximation algorithm for the single-objective problem (where α and ϵ describe the multiplicative and additive factors respectively), the returned solution is an $\left(\frac{\alpha^2}{1-\alpha+\alpha^2}, \frac{\epsilon}{1-\alpha+\alpha^2}\right)$ -approximation of leximin. Importantly, this holds for any number of objectives. It should also be noted that the algorithm is applicable in many cases, for example, when the feasible region is convex and all the objectives are concave (and polynomially computable).

Lastly, we consider a more challenging case. We apply our results to the problem of *stochastic allocations of indivisible goods*. When agents have submodular utilities, approximating the egalitarian value to a (multiplicative) factor better than $1 - \frac{1}{e} \approx 0.632$ is NP-hard [13]. That is, even the first-objective of leximin, i.e., maximizing the smallest objective, is NP-hard. We demonstrate that our method enables extending an approximation algorithm for the egalitarian welfare to an approximation for leximin with only a multiplicative error. In particular, we prove that a $\frac{1}{3}$ -approximation can be obtained deterministically, whereas a $\frac{(e-1)^2}{e^2-e+1} \approx 0.52$ -approximation can be obtained w.h.p.

Organization Section 2 gives preliminary knowledge and basic definitions. Section 3 presents the definition of leximin approximation. An algorithm for computing such an approximation is presented in Section 4. The problem of stochastic allocations of indivisible goods is considered in Section 5. Section 6 concludes with some future work directions.

1.1 Related Work

This paper is related to a large body of research, which can be classified into three main fields: multi-objective optimization problems, approximation variants of known solution concepts, and algorithms for finding optimal leximin solutions.

In general multi-objective² optimization problems, finding a leximin³ optimal solution is quite common goal [8], which is still an open challenge. Studies on this topic are usually focused on a specific problem and leverages its special characteristics — the structure of the *feasible region* and *objective-functions* that describe the concrete problem at hand. In this paper, we focus on the widely studied domain of resource allocation problems [15]. In that context, as leximin maximization is an extension of egalitarian welfare maximization, it is usually mentioned when fairness is desired.

There are cases where a leximin optimal solution can be calculated in polynomial time, for example in: fair allocation of divisible items [21], giveaway lotteries [2], portioning with ordinal preferences [1], cake sharing [3], multi-commodity flow networks [16], and location problems [19]. However, even when algorithms are theoretically polynomial, they can still be inaccurate in practice, for example due to numeric round-off errors.

In other cases, calculating a leximin optimal solution is NP-hard, for example in: representative cohort selection [11], fair combinatorial auctions [4], package upgrade-ability [7], allocating papers to referees [10, 14], and stochastic allocations of indivisible goods (Section 5 in this paper). However, to our knowledge, studies of this kind typically suggest non-polynomial algorithms and heuristics for solving small instances of the general problem and empirically evaluate their efficiency, rather than suggesting polynomial-time approximation algorithms.

Another approach to leximin optimization is to represent the leximin ordering by an aggregation function. Such a function takes a utility vector and returns a number, such that a solution is leximinpreferred over another if-and-only-if its aggregate number is higher. Finding such a function will of course reduce the problem to solving only one single-objective optimization problem. Unfortunately, it is known that no aggregate function can represent the leximin ordering in *all* problems [15, 17]. Still, there are interesting cases in which such functions can be found. For example, Yager [22] suggested that the *ordered weighted averaging (OWA)* technique can be used when there is a lower bound on the difference between any two possible utilities. However, it is unclear how (and whether) approximating the aggregate function would translate to approximating leximin.

To the best of our knowledge, other general approximations of leximin exist but they are less common. They are usually mentioned in the context of robustness or noise (e.g. [11, 12]) and lack characteristics that we emphasize within the context of errors.

Most similar to our work is the recent paper by Henzinger et al. [11]. This paper presents several approximation variants of leximin for the case of *additive* errors in the single-objective problems. Their motivation is different than ours — they use approximation as a method to improve efficiency and ensure robustness to noise. However, one of their definitions, (ϵ -tradeoff Leximax) fits our motivation of achieving the best possible leximin-approximation in the presence of errors. In fact, our approximation definition can be viewed as a generalization of their definition to include both multiplicative and additive errors. It should also be noted that the authors mention multiplicative approximation in the their Future Work Section.

2 Preliminaries

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

Single-objective optimization A single-objective maximization (resp. minimization) problem is a tuple (S, f) where S is the set of all feasible solutions to the problem (usually $S \subseteq \mathbb{R}^m$ for some $m \in \mathbb{N}$) and $f: S \to \mathbb{R}_{\geq 0}$ is a function describing the objective value of a solution $x \in S$. The goal in such problems is to find an *optimal* solution, that is, a feasible solution $x^* \in S$ that has the maximum (resp. minimum) objective value, that is $f(x^*) \geq f(x)$ (resp. $f(x^*) \leq f(x)$) for any other solution $x \in S$.

A (α, ϵ) -approximation algorithm for a single-objective maximization problem (S, f) is one that returns a solution $x \in S$,

 $[\]frac{1}{2}$ Multi-objective is also called *multi-criteria* (for example in [8]).

³ Leximin is also called *Max-Min fairness* (for example in [16]), *Lexico-graphic Min-Max* (for example in [18]), *Lexicographic max-ordering* (for example in [8]) and *Leximax* (for example in [11]).

which approximates the optimal solution x^* from below. That is, $f(x) \ge \alpha \cdot f(x^*) - \epsilon$ for $\alpha \in (0, 1]$ and $\epsilon \ge 0$ (that describe the multiplicative and additive approximation factors respectively).

Similarly, a $(1 + \beta, \epsilon)$ -approximation algorithm for a singleobjective minimization problem is one that returns a feasible solution x that approximates the optimal solution x^* from above. That is, $f(x) \le (1 + \beta) \cdot f(x^*) + \epsilon$ for $\beta \ge 0$ and $\epsilon \ge 0$.

A *p*-randomized approximation algorithm, for $p \in (0, 1]$, is one that returns a solution $x \in S$ such that, with probability p, the objective value f(x) is approximately-optimal.

The term "with high probability" (w.h.p.) is used when the success probability is at least 1 - 1/poly(X) where X describes the input size.

Multi-objective optimization A *multi-objective maximization* problem [5] can be described as follows:

$$\max \{f_1(x), f_2(x), \dots f_n(x)\}$$

s.t. $x \in S$

Where $S \subseteq \mathbb{R}^m$ for some $m \in \mathbb{N}$ is the *feasible region* and f_1, f_2, \ldots, f_n are *n* objective-functions $f_i \colon S \to \mathbb{R}_{\geq 0}^4$. An example application is group decision making: some *n* people have to decide on an issue that affects all of them. The set of possible decisions is *S*, and the utility each person *i* derives from a decision $x \in S$ is $f_i(x)$.

Ordered outcomes notation The *j*'th smallest objective value obtained by a solution $x \in S$ is denoted by $\mathcal{V}_{i}^{\dagger}(x)$, i.e.,

$$\mathcal{V}_1^{\uparrow}(x) \leq \mathcal{V}_2^{\uparrow}(x) \leq \cdots \leq \mathcal{V}_n^{\uparrow}(x).$$

The corresponding *sorted* utility vector is denoted by $\mathbf{V}^{\uparrow}(x) = (\mathcal{V}^{\uparrow}_{1}(x), \dots, \mathcal{V}^{\uparrow}_{n}(x)).$

The leximin order A solution y is considered *leximin-preferred* over a solution x, denoted $y \succ x$, if there exists an integer $1 \le k \le n$ such that the smallest (k - 1) objective values of both are equal, whereas the k'th smallest objective value of y is higher:

$$orall j < k \colon \quad \mathcal{V}_j^{\uparrow}(y) = \mathcal{V}_j^{\uparrow}(x) \ \mathcal{V}_k^{\uparrow}(y) > \mathcal{V}_k^{\uparrow}(x)$$

Two solutions, x, y, are *leximin equivalent* if $\mathbf{V}^{\uparrow}(x) = \mathbf{V}^{\uparrow}(y)$. The leximin order is a *total* order, and strict between any two solutions that yield *different* sorted utility vectors ($\mathbf{V}^{\uparrow}(x) \neq \mathbf{V}^{\uparrow}(y)$). A *maximum* element of the leximin order is a solution over which *no* solution is preferred (including solutions that yield the same utilities).

Leximin optimal A *leximin optimal* solution is a maximum element of the leximin order. Given a feasible region S, as the order is determined only by the utilities, we denote this optimization problem as follows.

lex max min {
$$f_1(x), f_2(x), \dots f_n(x)$$
}
s.t. $x \in S$

3 Approximate Leximin Optimality

In this section, we present our definition of leximin approximation in the presence of multiplicative and additive errors, in the context of multi-objective optimization problems.

3.1 Motivation: Unsatisfactory Definitions

Which solutions should be considered approximately-optimal in terms of leximin? Several definitions appear intuitive at first glance. As an example, suppose we are interested in approximations with an allowable multiplicative error of 0.1. Denote the utilities in the leximin-optimal solution by (u_1, \ldots, u_n) . A first potential definition is that any solution in which the sorted utility vector is at least $(0.9 \cdot u_1, \ldots, 0.9 \cdot u_n)$ should be considered approximately-optimal. For example, if the utilities in the optimal solution are (1, 2, 3), then a solution with utilities (0.9, 1.8, 2.7) is approximately-optimal. However, allowing the smallest utility to take the value 0.9 may substantially increase the maximum possible value of the second (and third) smallest utility - e.g. a solution that yields utilities (0.9, 1000, 1000) might exist. In that case, a solution with utilities (0.9, 1.8, 2.7) is very far from optimal. We expect a good approximation notion to consider the fact that an error in one utility might change the optimal value of the others.

The following, second attempt at a definition, captures this requirement. An approximately-optimal solution is one that yields utilities at least $(0.9 \cdot m_1, 0.9 \cdot m_2, \ldots, 0.9 \cdot m_n)$, where m_1 is the maximum value of the smallest utility, m_2 is the maximum value of the second-smallest utility *among all solutions whose smallest utility is at least* $0.9 \cdot m_1$; m_3 is the maximum value of the third-smallest utility among all solutions whose smallest utility is at least $0.9 \cdot m_1$; and their second-smallest utility is at least $0.9 \cdot m_2$; and so on. In the above example, to be considered approximately-optimal, the smallest utility should be at least 0.9 and the second-smallest should be at least 900. Thus, a solution with utilities (0.9, 1.8, 2.7) is not considered approximately-optimal. Unfortunately, according to this definition, even the leximin-optimal solution — with utilities (1, 2, 3) is not considered approximately-optimal. We expect a good approximation notion to be a relaxation of leximin-optimality.

3.2 Our Definition

Let $\alpha \in (0, 1]$ and $\epsilon \ge 0$ be multiplicative and additive approximation factors, respectively.

Comparison of values As we focus on maximization problems, given two values $v_2 \ge v_1 \ge 0$, we say that v_1 approximates v_2 if $v_1 \ge \alpha \cdot v_2 - \epsilon$. In this case, v_1 is an approximate replacement for v_2 . However, when $v_1 < \alpha \cdot v_2 - \epsilon$, we say that v_2 is (α, ϵ) -substantially-higher than v_1 . In this case, v_1 is smaller than any (α, ϵ) -approximation of v_2 .

The approximate leximin order The first step is defining the following *partial* order⁵: a solution y is (α, ϵ) -*leximin-preferred* over a solution x, denoted $y \succ_{(\alpha,\epsilon)} x$, if there exists an integer $1 \le k \le n$ such that the smallest (k-1) objective values of y are at *least* those

⁴ Note that the number of objectives in multi-objective optimization is commonly assumed to be constant. However, in this paper, we use a more general setting in which the number of objectives is a parameter of the problem.

⁵ A proof that the approximate leximin order is a strict partial order can be found in Appendix A of the full version.

of x, and the k'th smallest objective value of y is (α, ϵ) -substantiallyhigher than the k'th smallest objective value of x, that is:

$$\begin{split} \forall j < k \colon \quad \mathcal{V}_{j}^{\uparrow}(y) \geq \mathcal{V}_{j}^{\uparrow}(x) \\ \mathcal{V}_{k}^{\uparrow}(y) > \frac{1}{\alpha} \left(\mathcal{V}_{k}^{\uparrow}(x) + \epsilon \right) \end{split}$$

A maximal element of this order is a solution over which no solution is (α, ϵ) -leximin-preferred. For clearity, we define the corresponding relation set as follows:

$$\mathcal{R}_{(\alpha,\epsilon)} = \{ (y,x) \mid x, y \in S, \ y \succ_{(\alpha,\epsilon)} x \}$$

Before describing the approximation definition, we present two observations about this relation that will be useful later, followed by an example to illustrate how it works. The proofs are straightforward and are omitted.

The first observation is that the leximin order is equivalent to the approximate leximin order for $\alpha = 1$ and $\epsilon = 0$ (that is, in the absence of errors).

Lemma 1. Let $x, y \in S$. Then, $y \succ x \iff y \succ_{(1,0)} x$

The second observation relates different approximate leximin orders according to their *error* factors. Notice that, for additive errors, ϵ also describes the error size; whereas for multiplicative errors, one minus α describes it. Throughout the remainder of this section, we denote the multiplicative *error factor* by $\overline{\alpha} = 1 - \alpha$.

Observation 2. Let $0 \leq \overline{\alpha_1} \leq \overline{\alpha_2} < 1$ and $0 \leq \epsilon_1 \leq \epsilon_2$. Then, $y \succ_{(\alpha_2, \epsilon_2)} x \Rightarrow y \succ_{(\alpha_1, \epsilon_1)} x$.

One can easily verify that it follows directly from the definition as $\frac{1}{\alpha_2} \geq \frac{1}{\alpha_1}$. Accordingly, by considering the relation sets $\mathcal{R}_{(\alpha_1,\epsilon_1)}$ and $\mathcal{R}_{(\alpha_2,\epsilon_2)}$, we can conclude that $\mathcal{R}_{(\alpha_2,\epsilon_2)} \subseteq \mathcal{R}_{(\alpha_1,\epsilon_1)}$. This means that as the *error* parameters $\overline{\alpha}$ and ϵ increase, the relation becomes *more partial*: when $\overline{\alpha} = 0$ and $\epsilon = 0$ it is a total order, any two elements that yield different utilities appear as a pair in $\mathcal{R}_{(1,0)}$; but as they increase, the set $\mathcal{R}_{(\alpha,\epsilon)}$ potentially becomes smaller, as fewer pairs are comparable.

Example To illustrate, consider a group of 3 agents, that has to select one out of three options x, y, z, with sorted utility vectors $\mathbf{V}^{\uparrow}(x) = (1, 10, 15), \mathbf{V}^{\uparrow}(y) = (1, 40, 60), \mathbf{V}^{\uparrow}(z) = (2, 20, 30).$ Table 1 indicates what is $\mathcal{R}_{(\alpha, \epsilon)}$ for different choices of α and ϵ .

It is easy to verify that, indeed, $\mathcal{R}_{(1,0)}$ is a total order — $(z,x), (z,y) \in \mathcal{R}_{(1,0)}$ since 2 > 1 and $(y,x) \in \mathcal{R}_{(1,0)}$ since 1 = 1 and 40 > 10.

In accordance with Observation 2, the relation set remains the same or becomes smaller as either α decreases (and the error factor $\overline{\alpha}$ increases) or ϵ increases. As an example, we provide a partial calculation of $\mathcal{R}_{(0.75,1)}$.

First, by Observation 2, we know that $\mathcal{R}_{(0.75,1)} \subseteq \mathcal{R}_{(1,0)}$, and so, it is sufficient to consider only the pairs in $\mathcal{R}_{(1,0)}$.

Consider the pair (z, x). In order to be included in the relation set, there must be a $1 \le k \le 3$ that meets the requirements. For k = 1, as $2 \ne \frac{1}{0.75}(1+1)$, the requirement for k does not hold. However, for k = 2, it does. As $2 \ge 1$, the requirement for i < k holds; and as $20 > \frac{1}{0.75}(10+1)$, the requirement for k holds. Therefore, $(z, x) \in \mathcal{R}_{(0.75, 1)}$. Similarly, one can check that $(y, x) \in \mathcal{R}_{(0.75, 1)}$.

Next, consider the pair (z, y). For k = 1, as before, since $2 \ge \frac{1}{0.75}(1+1)$, the requirement for k does not hold. For k = 2 and k = 3, it is sufficient to notice that 20 < 40, therefore the requirements for both does not hold. And so, $(z, y) \notin \mathcal{R}_{(0.75, 1)}$.

The leximin approximation can now be defined.

Table 1: Different relation sets result from different choices of α and ϵ in the example above. Each cell contains the corresponding relation set $\mathcal{R}_{(\alpha,\epsilon)}$.

$\alpha \overset{\epsilon}{}$	0	1	15	45
1 0.75 0.5 0.25	$ \{(z,x),(z,y),(y,x)\} \\ \{(z,x),(z,y),(y,x)\} \\ \{(y,x)\} \\ \{\} \} $	$ \{(z,x),(y,x)\} \\ \{(z,x),(y,x)\} \\ \{(y,x)\} \\ \{\} \} $	$ \{(y,x)\} \\ \{(y,x)\} \\ \{\} \\ \{\} \\ \{\} \\ \{\} \\ \{\} \\ \{\} \\ \{\} \\ $	{ } { } { } { } { }

Leximin approximation We say that a solution $x \in S$ is (α, ϵ) approximately leximin-optimal if it is a maximum element of the order $\succ_{(\alpha,\epsilon)}$ in S for $\alpha \in (0,1]$ and $\epsilon \ge 0$. That is, if there is no solution in S that is (α, ϵ) -leximin-preferred over it. For brevity, we use the term *leximin approximation* to describe an approximately leximin-optimal solution.

This definition has some important properties. Lemma 3 proves that in the absence of errors ($\overline{\alpha} = \epsilon = 0$) it is equivalent to the exact leximin optimal definition. Then, Lemma 4 shows that an (α_1, ϵ_1) leximin-approximation is also an (α_2, ϵ_2) -leximin-approximation when $0 \le \overline{\alpha_1} \le \overline{\alpha_2} < 1$ and $0 \le \epsilon_1 \le \epsilon_2$. Finally, Lemma 5 proves that a leximin optimal solution is also a leximin approximation for all factors.

Lemma 3. A solution is a (1, 0)-leximin-approximation if and only if it is leximin optimal.

Proof. By definition, a solution x^* is a (1, 0)-leximin-approximation if and only if $x \neq_{(1,0)} x^*$ for any solution $x \in S$. This holds if and only if $x \neq x^*$ for any solution $x \in S$ (by Lemma 1). Thus, by definition, x^* is also leximin optimal.

Lemma 4. Let $0 \le \overline{\alpha_1} \le \overline{\alpha_2} < 1$, $0 \le \epsilon_1 \le \epsilon_2$, and $x \in S$ be an (α_1, ϵ_1) -leximin-approximation. Then x is also an (α_2, ϵ_2) -leximin-approximation.

Proof. Since x is an (α_1, ϵ_1) -leximin-approximation, by definition, $y \not\geq_{(\alpha_1, \epsilon_1)} x$ for any solution $y \in S$. Observation 2 implies that $y \not\geq_{(\alpha_2, \epsilon_2)} x$ for any solution $y \in S$. This means, by definition, that x is an (α_2, ϵ_2) -leximin-approximation.

Lemma 5. Let $x^* \in S$ be a leximin optimal solution. Then x^* is also an (α, ϵ) -leximin-approximation for any $\overline{\alpha} \in [0, 1)$ and $\epsilon \geq 0$.

Proof. By Lemma 3, x^* is an (1, 0)-leximin-approximation. Thus, according to Lemma 4, x^* is also an (α, ϵ) -leximin-approximation for any $0 \le \overline{\alpha} < 1$ and $\epsilon \ge 0$.

Using the example given previously, we shall now demonstrate that as the error parameters $\overline{\alpha}$ and ϵ increase, the quality of the approximation decreases. Consider table 1 once again.

If the corresponding relation set for α and ϵ is the total order $\{(z, x), (z, y), (y, x)\}$, the only solution over which no other solution is (α, ϵ) -leximin-preferred is z. Therefore, z is the only (α, ϵ) -leximin-approximation for these factors. Indeed, it is the only group decision that maximizes the welfare of the agent with the smallest utility. If the corresponding relation set is either $\{(z, x), (y, x)\}$ or $\{(y, x)\}$, as no solution is (α, ϵ) -leximin-preferred over z and y, both are (α, ϵ) -leximin-approximations. For example, for $\alpha = 0.5$ and $\epsilon = 0$, z still maximizes the utility of the poorest agent (2), and y gives the poorest agent a utility of 1, which is acceptable as it is half the maximum possible value (2), and subject to giving the poorest agent at least 1, maximizes the second-smallest utility (40). In contrast, while x, too, gives the poorest agent utility 1, its

Algorithm 1 The Ordered Outcomes Algorithm				
1: for $t = 1$ to n do				
2: $(x_t, z_t) \leftarrow OP(z_1, \dots, z_{t-1})$				
3: end for				
4: return x_n (with objective values $f_1(x_n), \ldots, f_n(x_n)$).				

second-smallest utility is 10, which is less than half the maximum possible in this case (40), and therefore, x is not a (α, ϵ) -leximinapproximation. Lastly, if the relation set is the empty set, then no solution is (α, ϵ) -leximin-preferred over the other, and all are (α, ϵ) leximin-approximations.

Egalitarian generalization Another property of our definition is that a leximin-approximation also approximates the optimal egalitarian welfare to the same approximation factors. Formally:

Observation 6. Let x be an (α, ϵ) -leximin-approximation. Then, the egalitatian value of x (i.e., $\min_{i \in [n]} f_i(x) = \mathcal{V}_1^{\uparrow}(x)$) is an (α, ϵ) -approximation of the optimal egalitarian value (i.e., $\max_{y \in S} \min_{i \in [n]} f_i(y)$).

Approximation Algorithm 4

We now present an algorithm for computing a leximin approximation. The algorithm is an adaptation of one of the algorithms of Ogryczak and Śliwiński [18] for finding exact leximin optimal solutions.

4.1 Preliminary: exact leximin-optimal solution

Following the definition of leximin, the core algorithm for finding a leximin optimal solution is iterative, wherein one first maximizes the least objective function, then the second, and so forth. In each iteration, t = 1, ..., n, it looks for the value that maximizes the t-th smallest objective, z_t , given that for any i < t the *i*-th smallest objective is at least z_i (the value that was computed in the *i*-th iteration). The core, single-objective optimization problem is thus:

$$\begin{array}{ll} \max & z_t & (\mathbf{P1}) \\ s.t. & (\mathbf{P1.1}) \ x \in S \\ & (\mathbf{P1.2}) \ \mathcal{V}_{\ell}^{\uparrow}(x) \geq z_{\ell} & \forall \ell \in [t-1] \\ & (\mathbf{P1.3}) \ \mathcal{V}_{\ell}^{\uparrow}(x) \geq z_t \end{array}$$

where the variables are the scalar z_t and the vector x, whereas z_1, \ldots, z_{t-1} are constants (computed in previous iterations).

Suppose we are given a procedure $OP(z_1, \ldots, z_{t-1})$, which, given z_1, \ldots, z_{t-1} , outputs (x, z_t) that is the exact optimal solution to (P1). Then, the *leximin* optimal solution is obtained by iterating this process for t = 1, ..., n, as described in Algorithm 1. The algorithm first maximizes the smallest objective $\mathcal{V}_1^{\uparrow}(x)$, and puts the result in z_1 . Then it maximizes the second-smallest objective $\mathcal{V}_2^{\uparrow}(x)$, subject to $\mathcal{V}_1^{\uparrow}(x)$ being at least z_1 , and puts the result in z_2 ; and so on.

Since constraints (P1.2) and (P1.3) are not linear with respect to the objective-functions, it is difficult to solve the program (P1) as is. Thus, [18] suggests a way to "linearize" the program in two steps.

First, we replace (P1) with the following program, that considers sums instead of individual values (where again the variables are z_t and x):

$$\begin{array}{ll} \max & z_t & (\text{P2}) \\ s.t. & (\text{P2.1}) & x \in S \end{array} \end{array}$$

P2.2)
$$\sum_{i \in F'} f_i(x) \ge \sum_{i=1}^{|F'|} z_i \qquad \forall F' \subseteq [n], \ |F'| < t$$

 $(\mathbf{D}\mathbf{A})$

(P2.3)
$$\sum_{i \in F'} f_i(x) \ge \sum_{i=1}^{t-1} z_i + z_t \quad \forall F' \subseteq [n], \ |F'| = t$$

Here, constraints (P1.2) and (P1.3) are replaced with constraints (P2.2) and (P2.3), respectively. Constraint (P2.2) says that for any $\ell < t$, the sum of any ℓ objectives is at least the sum of the first ℓ constants z_i (equivalently: the sum of the smallest ℓ objectives is at least the sum of the first ℓ constants z_i^6). Similarly, (P2.3) says that the sum of any t objectives (equivalently: the sum of the smallest tobjectives) is at least the sum of the first t - 1 constants z_i , plus the variable z_t .

Suppose (P1) is replaced with (P2) in Algorithm1. Then, in the first iteration, the algorithm still maximizes the smallest objective $\mathcal{V}_{1}^{\uparrow}(x)$, and puts the result in z_1 . In the second iteration, it maximizes the *difference* between the sum of the two smallest objectives $\mathcal{V}_1^{\uparrow}(x)$ + $\mathcal{V}_{2}^{\uparrow}(x)$ and z_{1} , subject to $\mathcal{V}_{1}^{\uparrow}(x)$ being at least z_{1} , and puts the result in z_2 . Since z_1 is the maximum value of $\mathcal{V}_1^{\uparrow}(x)$, being at least z_1 becomes being exactly z_1 , which means that, as was for (P1), the algorithm actually maximizes $\mathcal{V}_2^{\uparrow}(x)$, subject to $\mathcal{V}_1^{\uparrow}(x)$ being at least z_1 . Similarly for any iteration $1 < t \leq n$, as the sum $\sum_{i=1}^{\ell} z_i$ is the maximum value of $\sum_{i=1}^{\ell} \mathcal{V}_i^{\uparrow}(x)$ for all $1 \leq \ell < \overline{t}$, it can be concluded that the algorithm actually maximizes $\mathcal{V}_t^{\uparrow}(x)$, subject to $\mathcal{V}_{\ell}^{\uparrow}(x)$ being at least z_{ℓ} for $1 \leq \ell < t$ (as if (P1) was used). Accordingly, the algorithm still finds a leximin-optimal solution.

While (P2) is linear with respect to the objective-functions, it has an exponential number of constraints. To overcome this challenge, auxiliary variables were used in the second program (y_{ℓ} and $m_{\ell,j}$ for all $1 \le \ell \le t$ and $1 \le j \le n$):

$$\begin{array}{ll} \max & z_t & (P3) \\ s.t. & (P3.1) \ x \in S \\ & (P3.2) \ \ell y_\ell - \sum_{j=1}^n m_{\ell,j} \ge \sum_{i=1}^\ell z_i & \forall \ell \in [t-1] \\ & (P3.3) \ t y_t - \sum_{j=1}^n m_{t,j} \ge \sum_{i=1}^{t-1} z_i + z_t \\ & (P3.4) \ m_{\ell,i} \ge y_\ell - f_i(x) & \forall \ell \in [t], \ \forall j \in [n] \end{array}$$

$$(P3.5) \quad m_{\ell,j} \ge 0 \qquad \qquad \forall \ell \in [t], \quad \forall j \in [n]$$

The meaning of the auxiliary variables in (P3) is explained in the proof of Lemma 7 below.

The importance of the programs (P2) and (P3) for leximin is shown by the following theorem (that combines Theorem 4 in [17] and Theorem 1 in [18]):

Theorem. If Algorithm 1 is applied with a solver for (P2) or $(P3)^7$ (instead of for (P1)), the algorithm still outputs a leximin-optimal solution.

We shall later see that our main result (Theorem 9) extends and implies their theorem.

⁶ A formal proof of this claim is given in Appendix B.2 of the full version.

 $[\]overline{7}$ If the algorithm uses a solver for (P3), it takes only the assignment of the variables x and z_t , ignoring the auxiliary variables.

4.2 Using an approximate solver

Now we assume that, instead of an exact solver in Algorithm 1, we only have an approximate solver. In this case, the constants z_1, \ldots, z_{t-1} are only approximately-optimal solutions for the previous iterations. It is easy to see that if OP is an (α, ϵ) -approximation algorithm to (P1), then Algorithm 1 outputs an (α, ϵ) -leximinapproximation⁸.

In contrast, for (P2) and (P3), we shall see that Algorithm 1 may output a solution that is *not* an (α, ϵ) -leximin-approximation. However, we will prove that it is not too far from that — in this case, the output is always an $\left(\frac{\alpha^2}{1-\alpha+\alpha^2}, \frac{\epsilon}{1-\alpha+\alpha^2}\right)$ -leximin-approximation. To demonstrate both claims more clearly, we start by proving that

To demonstrate both claims more clearly, we start by proving that the programs (P2) and (P3) are *equivalent* in the following sense:

Lemma 7. Let $1 \le t \le n$ and let $z_1, \ldots z_{t-1} \in \mathbb{R}$. Then, (x, z_t) is feasible for (P2) if and only if there exist y_ℓ and $m_{\ell,j}$ for $1 \le \ell \le t$ and $1 \le j \le n$ such that $(x, z_t, (y_1, \ldots, y_t), (m_{1,1}, \ldots, m_{t,n}))$ is feasible for (P3).

The proof is provided in Appendix B.2 of the full version.

Since both (P2) and (P3) have the same objective function (max z_t), the lemma implies that (x, z_t) is an (α, ϵ) -approximate solution for (P2) if and only if (x, z_t) is a part of an (α, ϵ) -approximate solution for (P3). Thus, it is sufficient to prove the theorems for only one of the problems. We will prove them for (P2).

Theorem 8. There exist $\alpha \in (0, 1]$, $\epsilon \ge 0$ and *OP* that is an (α, ϵ) -approximation procedure to (P2), such that if Algorithm 1 is applied with this procedure, it might return a solution that is not an (α, ϵ) -leximin-approximation.

Proof. Consider the following multi-objective optimization problem with n = 2:

$$\max \{f_1(x) := x_1, f_2(x) := x_2\}$$

s.t. (1.1) $x_1 \le 100$, (1.2) $x_1 + x_2 \le 200$, (1.3) $x \in \mathbb{R}^2_+$

In the corresponding (P2), constraint (P2.1) will be replaced with constraints (1.1)-(1.3). The following is a possible run of the algorithm with OP that is a (0.9, 0)-approximate solver.

- In iteration t = 1, condition (P2.2) is empty, and the optimal value of z_1 is 100, so OP may output $z_1 = 0.9 \cdot 100 = 90$.
- In iteration t = 2, given $z_1 = 90$, condition (P2.2) says that each of x_1 and x_2 must be at least 90; the optimal value of z_2 under these constraints is 110, so OP may output $z_2 = 99$, for example with $x_1 = x_2 = 94.5$.
- Since n = 2, the algorithm ends and returns the solution (94.5, 94.5).

But $(x_1, x_2) = (94.5, 105.5)$ is also a feasible solution, and it is (0.9, 0)-leximin-preferred since $105.4 > \frac{1}{0.9} \cdot 94.5 = 105$. Hence, the returned solution is *not* a (0.9, 0)-leximin-approximation.

Note that, while the above solution is not a (0.9, 0)-leximinapproximation, it is for $\alpha = 0.896$. Our main theorem below shows that this is not a coincidence: using an approximate solver to (P2) or (P3) in Algorithm 1 guarantees a non-trivial leximin approximation.

Theorem 9. Let $\alpha \in (0, 1]$, $\epsilon \geq 0$, and *OP* be an (α, ϵ) -approximation procedure to (P2) or (P3). Then Algorithm 1 outputs an $\left(\frac{\alpha^2}{1-\alpha+\alpha^2}, \frac{\epsilon}{1-\alpha+\alpha^2}\right)$ -leximin-approximation.

For the above example, it guarantees an $(\frac{81}{91},0)\approx (0.89,0)$ -leximin-approximation.

A complete proof of Theorem 9 is given in Appendix B.3 of the full version. Here we provide a high level overview of the main steps.

First, we note that the value of the variable z_t is completely determined by the variable x. This is because the program aims to maximize z_t that appears only in constraint (P2.3), which is equivalent to $z_t \leq \sum_{i=1}^t \Psi_i^{\uparrow}(x) - \sum_{i=1}^{t-1} z_i$. Thus, this constraint will always hold with equality.

Next, we show that the returned solution, x^* , is feasible to all single-objective problems that were solved during the algorithm run. This allows us to relate the objective values attained by x^* and the z_i values. Since the solver used in iteration t is (α, ϵ) -approximately-optimal, it follows that the objective value attained by x^* is at most $\frac{1}{\alpha}(z_t + \epsilon)$, where z_t is the approximation obtained to that problem.

We then assume for contradiction that x^* is *not* a leximin approximation as claimed in the theorem. By definition, there exits a solution $y \in S$ and an integer $1 \leq k \leq n$ such that $\mathcal{V}_i^{\uparrow}(y) \geq \mathcal{V}_i^{\uparrow}(x^*)$ for any i < k, while $\mathcal{V}_k^{\uparrow}(y)$ is $\left(\frac{\alpha^2}{1-\alpha+\alpha^2}, \frac{\epsilon}{1-\alpha+\alpha^2}\right)$ -substantially-higher⁹ than $\mathcal{V}_k^{\uparrow}(x^*)$. Accordingly, we prove that y is feasible to the program that was solved in the k-th iteration, and that its objective value in this problem is higher than the optimal value z_t^* , which is a contradiction.

Theorem 9 implies that if OP only has a multiplicative error $(\epsilon = 0)$, the solution returned by Algorithm 1 will only have a multiplicative error as well, and if OP only has an additive error $(\alpha = 1)$, the solution returned by Algorithm 1 will have only the same additive error ϵ .

Note that there are many cases in which the required procedure (OP) can be implemented easily. For example, when the S is convex and all f_i 's are concave (and polynomially computable), using convex optimization techniques.

4.3 Using a randomized solver

Next, we assume that the solver is not only approximate but also randomized — it always returns a feasible solution to the singleobjective problem, but only with probability $p \in [0, 1]$ it is also approximately-optimal. As Algorithm 1 activates the solver n times overall, assuming the success events of different activations are independent, there is a probability of p^n that the solver returns an approximately-optimal solution in every iteration and so, Algorithm 1 performs as in the previous subsection. This leads to the following conclusion:

Corollary 10. Let $\alpha \in (0, 1]$, $\epsilon \ge 0$, $p \in (0, 1]$, and *OP* be a *p*-randomized (α, ϵ) -approximation procedure to (P2) or (P3). Then Algorithm 1 outputs an $\left(\frac{\alpha^2}{1-\alpha+\alpha^2}, \frac{\epsilon}{1-\alpha+\alpha^2}\right)$ -leximin-approximation with probability p^n .

Notice that, since the procedure OP always returns a feasible solution to the single-objective problem, Algorithm 1 always returns a feasible solution as well.

The following section applies such a solver to obtain a leximin approximation to the problem of stochastic allocations of indivisible goods w.h.p.

⁸ A formal proof is given in Appendix B.1 of the full version.

⁹ See Section 3.2 for formal definition.

5 Stochastic Allocations of Indivisible Goods

In this section, we consider a particular application of our results, for the problem of *stochastic allocations of indivisible goods*. We prove that, under the setting described bellow, a leximin approximation with *only* a multiplicative error can be obtained in polynomial time. Specifically, we prove that a $\frac{1}{3}$ -approximation¹⁰ can be obtained deterministically, whereas a $\frac{(e-1)^2}{e^2-e+1} \approx 0.52$ -approximation can be obtained w.h.p. As a reference point, it is worth noting that the problem of maximizing the egalitarian welfare in the same settings has been shown to be NP-hard to approximate to a (multiplicative) factor better than $1 - \frac{1}{e} \approx 0.632$ [13]. However, as an α -approximation to leximin is first and foremost an α -approximation to the egalitarian welfare¹¹, the same hardness result applies to our problem as well.

The setting postulates a set of n agents $1, \ldots, n$, and m items, $1, \ldots, m$, to be distributed amongst the agents. A *deterministic allocation* of the items to the agents is a mapping $A : [m] \rightarrow [n]$, determining which agent gets each item. Note that as the term "deterministic" is used in this section also when discussing algorithms, we will use the term *simple allocation* from now on. We denote by \mathcal{A} the set of simple allocations. Each agent j is associated with a function $u_j : \mathcal{A} \rightarrow \mathbb{R}_{\geq 0}$ that describes its utility from a simple allocation.

A *stochastic allocation*, d, is a distribution over the simple allocations. The set of all possible stochastic allocations is:

$$\mathcal{D} = \{ d \mid p_d \colon \mathcal{A} \to [0, 1], \sum_{A \in \mathcal{A}} p_d(A) = 1 \}$$

Agents are assumed to care only about their own share (allowing us to use the following abuse of notation in which u_j takes a bundle b of items), their utilities are assumed to be normalized $(u_j(\emptyset) = 0)$, monotone $(u_j(b_1) \le u_j(b_2)$ if $b_1 \subseteq b_2$), and submodular $(u_j(b_1) + u_j(b_2) \ge u_j(b_1 \cup b_2) + u_j(b_1 \cap b_2)$ for any bundles b_1, b_2). It is assumed that each agent assigns a positive utility to the set of all items. The utilities $(u_i)_{i=1}^n$ are assumed to be given in the value oracle model, meaning that we do not have a direct access to them, but only to an oracle that indicates the value of an agent from a given simple allocation. Lastly, the agents are assumed to be riskneutral. This means that, given a stochastic allocation d, the utility of each agent j is given by the expected value:

$$E_j(d) = \sum_{A \in \mathcal{A}} p_d(A) \cdot u_j(A)$$

The goal is to find a stochastic allocation that maximizes the set of functions E_1, \ldots, E_n . Formally, we consider the following problem:

lex max min
$$\{E_1(d), E_2(x), \dots E_n(d)\}$$

s.t. $d \in \mathcal{D}$

That is, the feasible region is the set of stochastic allocations (S = D) and the objective functions are the expected utilities ($f_i = E_i$ for any $i \in [N]$).

Kawase and Sumita [13] present an approximation algorithm, which relates the problem of finding a stochastic allocation that approximates the egalitarian welfare, to the problem of finding a *simple* allocation that approximates the *utilitarian welfare* (i.e., the sum of utilities):

$$\max \quad \sum_{i=1}^{n} u_i(A) \quad s.t. \quad A \in \mathcal{A}.$$
 (U1)

We adapt their algorithm to find an approximately leximin-optimal allocation as follows:

Theorem 11. Given a (randomized) algorithm that returns a simple allocation that α -approximates the utilitarian welfare (with success probability p). Then, Algorithm 1 can be used to obtain a $\frac{\alpha^2}{1-\alpha+\alpha^2}$ -leximin-approximation for the problem of stochastic allocations (with the same probability).

A complete proof is given in Appendix C of the full version. Here we provide an outline. We start by taking (P3) and replacing the constraint (P3.1) with the constraints describing a feasible stochastic allocation. Here we face a computational challenge: the number of variables describing a stochastic allocation is exponential in the input size, as we need a variable for each simple allocation. We address this challenge by moving to the dual of a closely related program. The dual has polynomially-many variables but exponentiallymany constraints. However, we prove that a randomized approximate separation-oracle for this program can be designed and used within a variant of the ellipsoid method to approximate (P3).

Theorem 11 yields two immediate corollaries, using known algorithms to approximate the utilitarian welfare when the agents' utility functions are monotone and submodular.

First, there are deterministic $\frac{1}{2}$ -approximation algorithms [6, 9], and therefore:

Corollary 12. Algorithm 1 can be used to obtain a $\frac{0.5^2}{1-0.5+0.5^2} = \frac{1}{3}$ -leximin-approximation for the problem of stochastic allocations.

Second, there is a randomized $(1 - \frac{1}{e})$ -approximation algorithm w.h.p [20], and therefore:

Corollary 13. Algorithm 1 can be used to obtain $\frac{(1-1/e)^2}{1-(1-1/e)+(1-1/e)^2} = \frac{(e-1)^2}{e^2-e+1} \approx 0.52$ -leximin-approximation for the problem of stochastic allocations w.h.p.

6 Conclusion and Future Work

We presented a practical solution to the problem of leximin optimization when only an approximate single-objective solver is available. The algorithm is guaranteed to terminate in polynomial time, and its approximation ratio degrades gracefully as a function of the approximation ratio of the single-objective solver.

It may be interesting to identify more problems (in addition to stochastic allocations), where an approximate egalitarian solution can be converted into an approximate leximin solution using the approach in this paper. In particular, in the problem of stochastic allocations (in Section 5), to extend the approximation algorithm for the egalitarian welfare, we had to change some steps within. What if an algorithm for egalitarian welfare is provided as a black box — could it be used to design the appropriate procedure to approximate leximin?

In the context of fair division, this study assumes that there is an access to the true valuations of the agents involved. In reality, people may lie about their valuations. Can our definition of approximate-leximin be related to some approximate version of truthfulness?

Another question is whether it is possible to obtain a better approximation factor for leximin, given an (α, ϵ) -approximation algorithm for the single-objective problem. Specifically, can an (α, ϵ) -approximation to leximin can be obtained in polynomial time? If not, what would be the best possible approximation in this case?

¹⁰ This section discusses only multiplicative approximations; so, for brevity, we use the term " α -approximation" to refer to "(α , 0)-approximation".

¹¹ See Observation 6.

Acknowledgements

This research is partly supported by the Israel Science Foundation grants 712/20 and 2697/22. We are grateful to Sylvain Bouveret for suggesting several alternative definitions and helpful insights. We are also grateful to the following members of the stack exchange network for their very helpful answers to our technical questions: Neal Young,¹² 1Rock,¹³ Mark L. Stone¹⁴ and Rob Pratt.¹⁵ Lastly, we would like to thank the reviewers in COMSOC 2023 and ECAI 2023 for their helpful comments.

References

- [1] Stéphane Airiau, Haris Aziz, Ioannis Caragiannis, Justin Kruger, Jérôme Lang, and Dominik Peters, 'Portioning Using Ordinal Preferences: Fairness and Efficiency', in *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, pp. 11–17, Macao, China, (August 2019). International Joint Conferences on Artificial Intelligence Organization.
- [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).
- [3] Xiaohui Bei, Xinhang Lu, and Warut Suksompong, 'Truthful Cake Sharing', Proceedings of the AAAI Conference on Artificial Intelligence, 36(5), 4809–4817, (June 2022).
- [4] Sylvain Bouveret and Michel Lemaître, 'Computing leximin-optimal solutions in constraint networks', *Artificial Intelligence*, **173**(2), 343– 364, (2009).
- [5] Multiobjective Optimization: Interactive and Evolutionary Approaches, eds., Jürgen Branke, Kalyanmoy Deb, Kaisa Miettinen, and Roman Słowiński, volume 5252 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008.
- [6] Niv Buchbinder, Moran Feldman, and Mohit Garg, Deterministic (¹/₂ + ε)-Approximation for Submodular Maximization over a Matroid, 241–254, 01 2019.
- [7] Miguel Cabral, Mikoláš Janota, and Vasco Manquinho, 'SAT-Based Leximax Optimisation Algorithms', in 25th International Conference on Theory and Applications of Satisfiability Testing (SAT 2022), eds., Kuldeep S. Meel and Ofer Strichman, volume 236 of Leibniz International Proceedings in Informatics (LIPIcs), pp. 29:1–29:19, Dagstuhl, Germany, (2022). Schloss Dagstuhl – Leibniz-Zentrum für Informatik.
- [8] Matthias Ehrgott, *Multicriteria optimization*, Springer, Berlin ; New York, 2nd ed edn., 2005.
- [9] Marshall L Fisher, George L Nemhauser, and Laurence A Wolsey, An analysis of approximations for maximizing submodular set functions— II, 73–87, Springer Berlin Heidelberg, Berlin, Heidelberg, 1978.
- [10] Naveen Garg, Telikepalli Kavitha, Amit Kumar, Kurt Mehlhorn, and Julián Mestre, 'Assigning Papers to Referees', *Algorithmica*, 58(1), 119–136, (September 2010).
- [11] Monika Henzinger, Charlotte Peale, Omer Reingold, and Judy Hanwen Shen, 'Leximax Approximations and Representative Cohort Selection', in 3rd Symposium on Foundations of Responsible Computing (FORC 2022), ed., L. Elisa Celis, volume 218 of Leibniz International Proceedings in Informatics (LIPIcs), pp. 2:1–2:22, Dagstuhl, Germany, (2022). Schloss Dagstuhl – Leibniz-Zentrum für Informatik.
- [12] Rim Kalai⁻, Claude Lamboray, and Daniel Vanderpooten, 'Lexicographic α-robustness: An alternative to min–max criteria', *European Journal of Operational Research*, **220**(3), 722–728, (August 2012).
- [13] 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).
- [14] Jing Wu Lian, Nicholas Mattei, Renee Noble, and Toby Walsh, 'The Conference Paper Assignment Problem: Using Order Weighted Averages to Assign Indivisible Goods', *Proceedings of the AAAI Conference* on Artificial Intelligence, **32**(1), (April 2018).

¹² https://cstheory.stackexchange.com/questions/51206 and https://cstheory. stackexchange.com/questions/51003 and https://cstheory.stackexchange. com/questions/52353

- ¹³ https://math.stackexchange.com/questions/4466551
- ¹⁴ https://or.stackexchange.com/questions/8633
- ¹⁵ https://or.stackexchange.com/questions/8980

- [15] Hervé Moulin, Fair division and collective welfare, MIT press, 2004.
- [16] Dritan Nace and Michal Pioro, 'Max-min fairness and its applications to routing and load-balancing in communication networks: a tutorial', *IEEE Communications Surveys & Tutorials*, 10(4), 5–17, (2008).
- [17] Wlodzimierz Ogryczak, Michał Pióro, and Artur Tomaszewski, 'Telecommunications network design and max-min optimization problem', *Journal of telecommunications and information technology*, 4, 43–53, (2004).
- [18] Włodzimierz Ogryczak and Tomasz Śliwiński, 'On direct methods for lexicographic min-max optimization', in *Computational Science and Its Applications - ICCSA 2006*, eds., Marina Gavrilova, Osvaldo Gervasi, Vipin Kumar, C. J. Kenneth Tan, David Taniar, Antonio Laganá, Youngsong Mun, and Hyunseung Choo, pp. 802–811, Berlin, Heidelberg, (2006). Springer Berlin Heidelberg.
- [19] Włodzimierz Ogryczak, 'On the lexicographic minimax approach to location problems', *European Journal of Operational Research*, **100**(3), 566–585, (1997).
- [20] 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*, pp. 67–74, Victoria British Columbia Canada, (May 2008). ACM.
- [21] Stephen J. Willson, 'Fair division using linear programming'. preprint, Departement of Mathematics, Iowa State University, 1998.
- [22] R.R. Yager, 'On ordered weighted averaging aggregation operators in multicriteria decisionmaking', *IEEE Transactions on Systems, Man,* and Cybernetics, 18(1), 183–190, (February 1988).