Leximin Approximation: From Single-Objective to Multi-Objective

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

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 $\epsilon \geq 0$ 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*. For this problem, assuming sub-modular objectives functions, the single-objective egalitarian welfare can be approximated, with only a multiplicative error, to an optimal $1 - \frac{1}{e} \approx 0.632$ factor w.h.p. We show how to extend the approximation to leximin, over all the objective functions, to a multiplicative factor of $\frac{(e-1)^2}{e^2-e+1} \approx 0.52$ w.h.p or $\frac{1}{3}$ deterministically.

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 second-smallest 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 [25, 2, 1, 3, 20, 22]). 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 *approximates* 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. [22, 25, 1, 20], 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:

$$\begin{aligned} \text{lex max min} \quad & \{f_1(x) = x_1, f_2(x) = x_2\} \\ s.t. \quad & (1)\ x_1 + x_2 \leq 1, \quad (2)\ x \in \mathbb{R}^2_+ \end{aligned}$$

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 minimum z_t, but there is no guarantee that this would lead to a good approximation.

Contributions This paper studies the problem of leximin optimization in multi-objective optimization problems, focusing on problems 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 [21] for exact leximin-optimization. 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.

Lastly, 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 [16]. 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-objective[1] optimization problems, finding a leximin[2] 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 [19]. 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 [25], giveaway lotteries [2], portioning with ordinal preferences[1], cake sharing [3], multi-commodity flow networks [20], and location problems [23]. 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 [13], fair combinatorial auctions [4], package upgrade-ability [7], allocating papers to referees [10, 18], 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 find an aggregation function, which takes a utility vector and returns a number such that a solution is leximin-preferred 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 [19, 22]. Still, there are interesting cases in which such functions can be found. For example, Yager [26] 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. [14, 13]) and lack characteristics that we emphasize within the context of error.

Most similar to our work is the recent paper by Henzinger et al. [13]. 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

[1] Multi-objective is also called *multi-criteria* (for example in [8]).

[2] Leximin is also called *Max-Min fairness* (for example in [20]), *Lexicographic Min-Max* (for example in [21]), *Lexicographic max-ordering* (for example in [8]) and *Leximax* (for example in [13]).

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 (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 $f \colon S \to \mathbb{R}$ 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 (minimum) objective value, that is $f(x^*) \geq f(x)$ ($f(x^*) \leq f(x)$) for any other solution $x \in S$.

A $(1-\beta, \epsilon)$-*approximation algorithm* for a single-objective *maximization* problem (S, f) is one that returns a solution $x \in S$ that *approximates* the optimal solution x^* from below. That is, $f(x) \geq (1-\beta) \cdot f(x^*) - \epsilon$ for $\beta \in [0, 1)$ and $\epsilon \geq 0$ (that describe the allowed multiplicative and additive error factors respectively).

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

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.

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

$$\begin{aligned} \max \quad & \{f_1(x), f_2(x), \ldots f_n(x)\} \\ s.t. \quad & x \in S \end{aligned}$$

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* $f_i \colon S \to \mathbb{R}$. 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 multiset of objective values achieved from a solution $x \in S$ is denoted by $\mathbf{V}(x) = \{f_i(x)\}_{i=1}^n$, and the j'th smallest objective value by $\mathcal{V}_j^{\uparrow}(x)$, i.e.,

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

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

$$\begin{aligned} \forall j < k \colon \quad & \mathcal{V}_j^{\uparrow}(y) = \mathcal{V}_j^{\uparrow}(x) \\ & \mathcal{V}_k^{\uparrow}(y) > \mathcal{V}_k^{\uparrow}(x) \end{aligned}$$

Two solutions, x, y, are *leximin equivalent* if $\mathbf{V}(x) = \mathbf{V}(y)$. The leximin order is a *total* order, and strict between any two solutions that yield *different* utility multisets ($\mathbf{V}(x) \neq \mathbf{V}(y)$). 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.

$$\begin{aligned} \text{lex max min} \quad & \{f_1(x), f_2(x), \dots f_n(x)\} \\ s.t. \quad & x \in S \end{aligned}$$

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 $(u_1, \dots, u_n)$. A first potential definition is that any solution in which the sorted utility vector is at least $(0.9 \cdot u_1, \dots, 0.9 \cdot u_n)$ 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 $(0.9 \cdot m_1, 0.9 \cdot m_2, \dots, 0.9 \cdot m_n)$, 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 $\epsilon \geq 0$ be multiplicative and additive approximation factors, respectively. As we focus on maximization problems, we say that a value v_2 is (α, ϵ)-*preferred* over another value v_1 if $v_2 > \frac{1}{\alpha}(v_1 + \epsilon)$. That is, if v_1 is smaller than any approximation of v_2.

The approximate leximin order The first step is defining the following *partial* order[3]: a solution y is (α, ϵ)-*leximin-preferred* over a solution x, denoted $y \succ_{(\alpha,\epsilon)} x$, if there exists an integer $1 \leq k \leq n$ such that the smallest $(k-1)$ objective values of y are *at least* those

[3] A proof that the approximate leximin order is a strict partial order can be found in appendix A.

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

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

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:

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

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 $\theta(\alpha) = 1 - \alpha$.

Observation 2. *Let $0 \leq \theta(\alpha_1) \leq \theta(\alpha_2) < 1$ 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 $\theta(\alpha)$ and ϵ increase, the relation becomes *more partial*: when $\theta(\alpha) = 0$ 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 utility multisets $\mathbf{V}(x) = \{1, 10, 15\}, \mathbf{V}(y) = \{1, 40, 60\}, \mathbf{V}(z) = \{2, 20, 30\}$. 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 $\theta(\alpha)$ 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 $1 \leq k \leq 3$ that meets the requirements. For $k = 1$, as $2 \not> \frac{1}{0.75}(1 + 1)$, the requirement for k does not hold. However, for $k = 2$, it does. As $2 \geq 1$, 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 $2 \not> \frac{1}{0.75}(1 + 1)$, 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 $\mathcal{R}_{(\alpha,\epsilon)}$.

α \ ϵ	0	1	15	45
1	{(z,x),(z,y),(y,x)}	{(z,x),(y,x)}	{(y,x)}	{}
0.75	{(z,x),(z,y),(y,x)}	{(z,x),(y,x)}	{(y,x)}	{}
0.5	{(y,x)}	{(y,x)}	{}	{}
0.25	{}	{}	{}	{}

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 $\succ_{(\alpha,\epsilon)}$. 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 ($\theta(\alpha) = \epsilon = 0$) 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 $0 \leq \theta(\alpha_1) \leq \theta(\alpha_2) < 1$ and $0 \leq \epsilon_1 \leq \epsilon_2$. 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 $x \not\succ_{(1,0)} x^*$ for any solution $x \in S$. This holds if and only if $x \not\succ x^*$ for any solution $x \in S$ (by Lemma 1). Thus, by definition, x^* is also leximin optimal. □

Lemma 4. *Let* $0 \leq \theta(\alpha_1) \leq \theta(\alpha_2) < 1$*,* $0 \leq \epsilon_1 \leq \epsilon_2$*, 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, $y \not\succ_{(\alpha_1,\epsilon_1)} x$ for any solution $y \in S$. Observation 2 implies that $y \not\succ_{(\alpha_2,\epsilon_2)} x$ 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* $\theta(\alpha) \in [0, 1)$ *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 $0 \leq \theta(\alpha) < 1$ and $\epsilon \geq 0$. □

Using the example given previously, we shall now demonstrate that as the error parameters $\theta(\alpha)$ 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 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)-leximin-approximation. 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.

Algorithm 1 The Ordered Outcomes Algorithm

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

4 Approximation Algorithm

We now present an algorithm for computing a leximin approximation. The algorithm is an adaptation of one of the algorithms of Ogryczak and Śliwiński [21] 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, $t = 1, \ldots, n$, 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:

$$\begin{array}{lll} \max & z_t & \text{(P1)} \\ s.t. & \text{(P1.1)}\ x \in S & \\ & \text{(P1.2)}\ \mathcal{V}_\ell^{\uparrow}(x) \geq z_\ell & \ell = 1, \ldots, t-1 \\ & \text{(P1.3)}\ \mathcal{V}_t^{\uparrow}(x) \geq z_t & \end{array}$$

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 $\mathsf{OP}(z_1, \ldots, z_{t-1})$, 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 $t = 1, \ldots, n$, as described in Algorithm 1.

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. [21] suggests a way to "linearize" the program in two steps. First, we replace (P1) with a the following program, that considers sums instead of individual values (where again the variables are z_t and x):

$$\begin{array}{lll} \max & z_t & \text{(P2)} \\ s.t. & \text{(P2.1)}\quad x \in S & \\ & \text{(P2.2)}\quad \sum_{i \in F'} f_i(x) \geq \sum_{i=1}^{|F'|} z_i & \forall F' \subseteq [n],\ |F'| < t \\ & \text{(P2.3)}\quad \sum_{i \in F'} f_i(x) \geq \sum_{i=1}^{t} z_i & \forall F' \subseteq [n],\ |F'| = t \end{array}$$

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[4]). Similarly, (P2.3) says that the sum of any t objectives

[4] A formal proof of this claim is given in Appendix B.2

(equivalently: the sum of the smallest t objectives) is at least the sum of the first $t-1$ constants z_i, plus the variable z_t.

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 problem (y_ℓ and $m_{\ell,j}$ for all $1 \leq \ell \leq t$ and $1 \leq j \leq n$):

$$
\begin{array}{lll}
\max & z_t & \text{(P3)} \\
s.t. & \text{(P3.1)}\ x \in S & \\
& \text{(P3.2)}\ \ell y_\ell - \sum_{j=1}^{n} m_{\ell,j} \geq \sum_{i=1}^{\ell} z_i & \ell = 1,\ldots,t-1 \\
& \text{(P3.3)}\ t y_t - \sum_{j=1}^{n} m_{t,j} \geq \sum_{i=1}^{t} z_i & \\
& \text{(P3.4)}\ m_{\ell,j} \geq y_\ell - f_j(x) & \ell = 1,\ldots,t,\ \ j = 1,\ldots,n \\
& \text{(P3.5)}\ m_{\ell,j} \geq 0 & \ell = 1,\ldots,t,\ \ j = 1,\ldots,n
\end{array}
$$

The importance of the problems (P2) and (P3) for leximin is shown by the following theorem (that combines Theorem 4 in [22] and Theorem 1 in [21]):

Theorem. *If Algorithm 1 is applied with a solver for* (P2) *or* (P3)[5] *(instead of for* (P1)*), the algorithm still outputs a leximin-optimal solution.*

We shall later see that our main result (Theorem 8) extends and implies their theorem.

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)-leximin-approximation[6].

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.

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

Lemma 6. *Let $1 \leq t \leq n$ 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 $1 \leq \ell \leq t$ and $1 \leq j \leq n$ 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. 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 7. *There exist $\alpha \in (0,1]$, $\epsilon \geq 0$ 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.*

[5] If the algorithm uses a solver for (P3), it takes only the assignment of the variables x and z_t , ignoring the auxiliary variables.

[6] A formal proof is given in Appendix B.1

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

$$\begin{aligned} \max \quad & \{f_1(x) := x_1, f_2(x) := x_2\} \\ s.t. \quad & (1.1)\ x_1 \leq 100, \quad (1.2)\ x_1 + x_2 \leq 200, \quad (1.3)\ x \in \mathbb{R}^2_+ \end{aligned}$$

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 $(0.9, 0)$-leximin-approximation, 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 8. *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 8 is given in Appendix B.3. 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 $z_t \leq \sum_{i=1}^{t} \mathcal{V}_i^{\uparrow}(x) - \sum_{i=1}^{t-1} z_i$. 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. 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 $\left(\frac{\alpha^2}{1-\alpha+\alpha^2}, \frac{\epsilon}{1-\alpha+\alpha^2}\right)$-preferred[7] over $\mathcal{V}_k^{\uparrow}(x^*)$. 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 8 implies that if OP has only a multiplicative error ($\epsilon = 0$), the returned solution will also have only a multiplicative error, and if OP has only an additive error ($\alpha = 1$), the returned solution will also have only the same additive error ϵ.

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 single-objective 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 9. *Let $\alpha \in (0, 1]$, $\epsilon \geq 0$, $p \in (0, 1]$, 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.*

[7] See Section 3.2 for formal definition.

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.

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 $\frac{1}{3}$-approximation[8] 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 than $1 - \frac{1}{e} \approx 0.632$ [16]. However, as an α-approximation to leximin is first and foremost an α-approximation to the egalitarian welfare, 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 $A : [m] \to [n]$, 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 $u_j \colon \mathcal{A} \to \mathbb{R}_{\geq 0}$ 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 assign a positive utility to the set of all items and 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 ($u_j(b_1) \leq u_j(b_2)$ if $b_1 \subseteq b_2$), and submodular ($u_j(b_1) + u_j(b_2) \geq u_j(b_1 \cup b_2) + u_j(b_1 \cap b_2)$ for any bundles b_1, b_2); and to be given in the *value oracle model* — that is, 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 risk-neutral. 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:

$$\begin{aligned} \text{lex max min} \quad & \{E_1(d), E_2(x), \ldots E_n(d)\} \\ s.t. \quad & d \in \mathcal{D} \end{aligned}$$

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

Kawase and Sumita [16] present an approximation algorithm, which relates the problem of finding a stochastic allocation that approximates the egalitarian welfare, to the problem

[8] Throughout this section, we only discuss multiplicative approximations; so, for brevity, we use the term "α-approximation" to refer to "$(\alpha, 0)$-approximation".

of finding a *simple* allocation that approximates the *utilitarian welfare* (i.e., the sum of utilities):

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

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

Theorem 10. *Given a randomized algorithm that returns a simple allocation that β-approximates the utilitarian welfare (with success probability p). Then, Algorithm 1 can be used to obtain a stochastic allocation that approximates leximin with a multiplicative error of at most $\frac{\beta}{1-\beta+\beta^2}$ (with the same probability).*

A complete proof is given in Appendix C. 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 exponentially-many 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 10 yield 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[9, 6], and therefore:

Corollary 11. *Algorithm 1 can be used to obtain a stochastic allocation that approximates leximin with a multiplicative error at most $\frac{0.5}{1-0.5+0.5^2} = \frac{2}{3}$.*

Second, there is a randomized $(1 - \frac{1}{e})$-approximation algorithm w.h.p [24], and therefore:

Corollary 12. *Algorithm 1 can be used to obtain a stochastic allocation that approximates leximin with a multiplicative error at most $\frac{e}{e^2-e+1} \approx 0.48$ 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?

7 Acknowledgments

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,[9] 1Rock,[10] Mark L. Stone[11] and Rob Pratt.[12] Lastly, we would like to thank the reviewers in COMSOC 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*, pages 11–17, Macao, China, August 2019. International Joint Conferences on Artificial Intelligence Organization. ISBN 978-0-9992411-4-1. doi: 10.24963/ijcai.2019/2. URL https://www.ijcai.org/proceedings/2019/2.

[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] 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.

[4] Sylvain Bouveret and Michel Lemaître. Computing leximin-optimal solutions in constraint networks. *Artificial Intelligence*, 173(2):343–364, 2009. ISSN 0004-3702. doi: https://doi.org/10.1016/j.artint.2008.10.010. URL https://www.sciencedirect.com/science/article/pii/S0004370208001495.

[5] Jürgen Branke, Kalyanmoy Deb, Kaisa Miettinen, and Roman Słowiński, editors. *Multiobjective Optimization: Interactive and Evolutionary Approaches*, volume 5252 of *Lecture Notes in Computer Science*. Springer Berlin Heidelberg, Berlin, Heidelberg, 2008. doi: 10.1007/978-3-540-88908-3. URL http://link.springer.com/10.1007/978-3-540-88908-3.

[6] Niv Buchbinder, Moran Feldman, and Mohit Garg. *Deterministic ($\frac{1}{2}+\epsilon$)-Approximation for Submodular Maximization over a Matroid*, pages 241–254. 01 2019. ISBN 978-1-61197-548-2. doi: 10.1137/1.9781611975482.16. URL https://epubs.siam.org/doi/abs/10.1137/1.9781611975482.16.

[7] Miguel Cabral, Mikoláš Janota, and Vasco Manquinho. SAT-Based Leximax Optimisation Algorithms. In Kuldeep S. Meel and Ofer Strichman, editors, *25th International Conference on Theory and Applications of Satisfiability Testing (SAT 2022)*, volume 236 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages

[9] https://cstheory.stackexchange.com/questions/51206 and https://cstheory.stackexchange.com/questions/51003 and https://cstheory.stackexchange.com/questions/52353

[10] https://math.stackexchange.com/questions/4466551

[11] https://or.stackexchange.com/questions/8633

[12] https://or.stackexchange.com/questions/8980

29:1–29:19, Dagstuhl, Germany, 2022. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISBN 978-3-95977-242-6. doi: 10.4230/LIPIcs.SAT.2022.29. URL https://drops.dagstuhl.de/opus/volltexte/2022/16703.

[8] Matthias Ehrgott. *Multicriteria optimization*. Springer, Berlin ; New York, 2nd ed edition, 2005. ISBN 978-3-540-21398-7.

[9] 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.

[10] Naveen Garg, Telikepalli Kavitha, Amit Kumar, Kurt Mehlhorn, and Julián Mestre. Assigning Papers to Referees. *Algorithmica*, 58(1):119–136, September 2010. ISSN 0178-4617, 1432-0541. doi: 10.1007/s00453-009-9386-0. URL http://link.springer.com/10.1007/s00453-009-9386-0.

[11] M. Grötschel, L. Lovász, and A. Schrijver. The ellipsoid method and its consequences in combinatorial optimization. *Combinatorica*, 1(2):169–197, June 1981. ISSN 0209-9683, 1439-6912. doi: 10.1007/BF02579273. URL http://link.springer.com/10.1007/BF02579273.

[12] Martin Grötschel, László Lovász, and Alexander Schrijver. *Geometric Algorithms and Combinatorial Optimization*, volume 2 of *Algorithms and Combinatorics*. Springer Berlin Heidelberg, Berlin, Heidelberg, 1993. ISBN 978-3-642-78240-4. doi: 10.1007/978-3-642-78240-4. URL http://link.springer.com/10.1007/978-3-642-78240-4.

[13] 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.

[14] Rim Kalaı¨, 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. ISSN 03772217. doi: 10.1016/j.ejor.2012.01.056. URL https://linkinghub.elsevier.com/retrieve/pii/S037722171200094X.

[15] 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/.

[16] 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.

[17] Subhash Khot, Richard J. Lipton, Evangelos Markakis, and Aranyak Mehta. Inapproximability Results for Combinatorial Auctions with Submodular Utility Functions. *Algorithmica*, 52(1):3–18, September 2008. ISSN 0178-4617, 1432-0541. doi: 10.1007/s00453-007-9105-7. URL http://link.springer.com/10.1007/s00453-007-9105-7.

[18] 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. ISSN 2374-3468, 2159-5399. doi: 10.1609/aaai.v32i1.11484. URL https://ojs.aaai.org/index.php/AAAI/article/view/11484.

[19] Hervé Moulin. *Fair division and collective welfare*. MIT press, 2004.

[20] 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. ISSN 1553-877X. doi: 10.1109/SURV.2008.080403. URL http://ieeexplore.ieee.org/document/4738463/.

[21] 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.

[22] 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.

[23] 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. URL https://www.sciencedirect.com/science/article/pii/S0377221796001543.

[24] 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.

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

[26] 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. ISSN 00189472. doi: 10.1109/21.87068. URL http://ieeexplore.ieee.org/document/87068/.

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

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

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

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

A The Approximate Leximin Order

Unlike the leximin order, $\succ$, which is a **total** order, the approximate leximin order, $\succ_{(\alpha,\epsilon)}$ for $\alpha \in (0,1]$ and $\epsilon \geq 0$ is a **partial** order. The difference is that in partial orders, not all vectors are comparable. Consider for example the sorted vectors $(1,2)$ and $(1,3)$. According to the leximin order, $(1,3)$ is clearly preferred (as $3 > 2$), but according to many approximate leximin orders neither one is preferred over the other, for example according to the orders $\succ_{(0.6,0)}$,$\succ_{(1,1)}$ or $\succ_{(0.8,0.5)}$.

An order is a strict partial order if it is irreflexive, transitive and asymmetric. Lemma 13 proves that the order is irreflexive, Lemma 14 proves it is transitive, and Lemma 15 proves that it is asymmetric.

Let $\alpha \in (0,1]$ and $\epsilon \geq 0$.

Lemma 13. *The approximate leximin order $\succ_{(\alpha,\epsilon)}$ is irreflexive.*

Proof. Let x be a solution. We will show that $x \not\succ_{(\alpha,\epsilon)} x$. As the definition requires that one component be *strictly greater* than the other, it is trivial. □

Lemma 14. *The approximate leximin order $\succ_{(\alpha,\epsilon)}$ is transitive.*

Proof. Let x, y and z be solutions such that $x \succ_{(\alpha,\epsilon)} y$ and $y \succ_{(\alpha,\epsilon)} z$. We will prove that $x \succ_{(\alpha,\epsilon)} z$.

Since $x \succ_{(\alpha,\epsilon)} y$, there exists an integer $k_1 \in [n]$ such that:

$$\forall j < k_1 \colon \mathcal{V}_j^{\uparrow}(x) \geq \mathcal{V}_j^{\uparrow}(y)$$
$$\mathcal{V}_{k_1}^{\uparrow}(x) > \frac{1}{\alpha}\left(\mathcal{V}_{k_1}^{\uparrow}(y) + \epsilon\right)$$

And since $y \succ_{(\alpha,\epsilon)} z$, there exists an integer $k_2 \in [n]$ such that:

$$\forall j < k_2 \colon \mathcal{V}_j^{\uparrow}(y) \geq \mathcal{V}_j^{\uparrow}(z)$$
$$\mathcal{V}_{k_2}^{\uparrow}(y) > \frac{1}{\alpha}\left(\mathcal{V}_{k_2}^{\uparrow}(z) + \epsilon\right)$$

As $\alpha \in (0,1]$ and $\epsilon \geq 0$, it follows that:

$$\mathcal{V}_{k_1}^{\uparrow}(x) > \mathcal{V}_{k_1}^{\uparrow}(y), \quad \mathcal{V}_{k_2}^{\uparrow}(y) > \mathcal{V}_{k_2}^{\uparrow}(z) \tag{1}$$

Let $k = \min\{k_1, k_2\}$.

If $k = k_1$, by the definition of k_1, $\mathcal{V}_k^{\uparrow}(x) > \frac{1}{\alpha}\left(\mathcal{V}_k^{\uparrow}(y) + \epsilon\right)$. However, $\mathcal{V}_k^{\uparrow}(y) \geq \mathcal{V}_k^{\uparrow}(z)$, by definition if $k < k_2$ and by Equation (1) if $k = k_2$. 2 Therefore, $\mathcal{V}_k^{\uparrow}(x) > \frac{1}{\alpha}\left(\mathcal{V}_k^{\uparrow}(z) + \epsilon\right)$.

Otherwise, if $k = k_2$, by the definition of k_2, $\mathcal{V}_k^{\uparrow}(y) > \frac{1}{\alpha}\left(\mathcal{V}_k^{\uparrow}(z) + \epsilon\right)$. But, $\mathcal{V}_k^{\uparrow}(x) \geq \mathcal{V}_k^{\uparrow}(y)$, by definition if $k < k_1$ and by Equation (1) if $k = k_1$. Again, we can conclude that $\mathcal{V}_k^{\uparrow}(x) > \frac{1}{\alpha}\left(\mathcal{V}_k^{\uparrow}(z) + \epsilon\right)$.

In addition, for each $j < k$, since $j < k_1$ and $j < k_2$, by definition the following holds:

$$\mathcal{V}_j^{\uparrow}(x) \geq \mathcal{V}_j^{\uparrow}(y) \geq \mathcal{V}_j^{\uparrow}(z) \tag{2}$$

So, k is an integer that satisfy all the requirements, and so, $x \succ_{(\alpha,\epsilon)} z$. □

Lemma 15. *The approximate leximin order $\succ_{(\alpha,\epsilon)}$ is asymmetric.*

Proof. Let x and y be solutions such that $x \succ_{(\alpha,\epsilon)} y$. We will show that $y \not\succ_{(\alpha,\epsilon)} x$. Assume by contradiction that $y \succ_{(\alpha,\epsilon)} x$. From Lemma 14, this relation is transitive. Therefore, since $x \succ_{(\alpha,\epsilon)} y$ and $y \succ_{(\alpha,\epsilon)} x$, also $x \succ_{(\alpha,\epsilon)} x$. But, from Lemma 13, this relation is irreflexive — a contradiction. □

B Proofs Omitted From Section 4

B.1 Using an approximate solver for (P1)

Recall that (P1) is described as follows:

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

This section proves the following lemma:

Lemma 16. *Let $\alpha \in (0,1]$, $\epsilon \geq 0$, and OP be an (α,ϵ)-approximation procedure to* (P1)*. Then Algorithm 1 outputs an (α,ϵ)-leximin-approximation.*

Proof. Let x^* be the returned solution and assume by contradiction that it is *not* (α,ϵ)-approximate leximin-optimal. This means that there exists a $y \in S$ that is (α,ϵ)-leximin-preferred over it. That is, there exists an integer $k \in [n]$ such that:

$$\begin{aligned} \forall i < k\colon\ & \mathcal{V}_i^\uparrow(y) \geq \mathcal{V}_i^\uparrow(x^*) \\ & \mathcal{V}_k^\uparrow(y) > \frac{1}{\alpha}\left(\mathcal{V}_k^\uparrow(x^*) + \epsilon\right) \end{aligned}$$

Since x^* is a solution to (P1) that was solved in the iteration $t = n$, it must satisfy all its constraints, and therefore:

$$\forall i \in [n]\colon\ \ \mathcal{V}_i^\uparrow(x^*) \geq z_i \tag{3}$$

by constraint (P1.2) for $i < n$ and by constraint (P1.3) for $i = n$.

As y's smallest k values are at least as those of x^*, we can conclude that for each $i \leq k$ the i-th smallest value of y is at least z_i. Therefore, y is feasible to (P1) that was solved in the iteration $t = k$.

During the algorithm run, z_k was obtained as an (α,ϵ)-approximation to (P1) that was solved in the iteration $t = k$, and therefore, the optimal value of this problem is at most $\frac{1}{\alpha}(z_k + \epsilon)$. But, the objective value y yields in this problem is $\mathcal{V}_k^\uparrow(y)$, which is higher than this value:

$$\begin{aligned} \mathcal{V}_k^\uparrow(y) &> \frac{1}{\alpha}\left(\mathcal{V}_k^\uparrow(x^*) + \epsilon\right) \\ &\geq \frac{1}{\alpha}\left(z_k + \epsilon\right) && \text{(By Equation (3) for } i = k\text{)} \end{aligned}$$

This is a contradiction. □

B.2 Equivalence of (P2) and (P3)

Recall the problems' descriptions:

$$\begin{aligned}
\max \quad & z_t && && \text{(P2)}\\
s.t. \quad & \text{(P2.1)} \quad x \in S\\
& \text{(P2.2)} \quad \sum_{i\in F'} f_i(x) \geq \sum_{i=1}^{|F'|} z_i && \forall F' \subseteq [n],\ |F'| < t\\
& \text{(P2.3)} \quad \sum_{i\in F'} f_i(x) \geq \sum_{i=1}^{t} z_i && \forall F' \subseteq [n],\ |F'| = t
\end{aligned}$$

$$\begin{aligned}
\max \quad & z_t && && \text{(P3)}\\
s.t. \quad & \text{(P3.1)}\ x \in S\\
& \text{(P3.2)}\ \ell y_\ell - \sum_{j=1}^{n} m_{\ell,j} \geq \sum_{i=1}^{\ell} z_i && \ell = 1,\dots,t-1\\
& \text{(P3.3)}\ t y_t - \sum_{j=1}^{n} m_{t,j} \geq \sum_{i=1}^{t} z_i\\
& \text{(P3.4)}\ m_{\ell,j} \geq y_\ell - f_j(x) && \ell = 1,\dots,t,\ j = 1,\dots,n\\
& \text{(P3.5)}\ m_{\ell,j} \geq 0 && \ell = 1,\dots,t,\ j = 1,\dots,n
\end{aligned}$$

We use another equivalent representation of (P2), which is more compact and will simplify the proofs, also introduced by [21]:

$$\begin{aligned}
\max \quad & z_t && && \text{(P2-Comp)}\\
s.t. \quad & \text{(P2-Comp.1)}\ x \in S\\
& \text{(P2-Comp.2)}\ \sum_{i=1}^{\ell} \mathcal{V}_i^{\uparrow}(x) \geq \sum_{i=1}^{\ell} z_i && \ell = 1,\dots,t-1\\
& \text{(P2-Comp.3)}\ \sum_{i=1}^{t} \mathcal{V}_i^{\uparrow}(x) \geq \sum_{i=1}^{t} z_i
\end{aligned}$$

In this problem, constraints (P2.2) and (P2.3) are replaced by (P2-Comp.2) and (P2-Comp.3), respectively. (P2.2) gives, for each ℓ, a lower bound on the sum for *any* set of ℓ objective functions; whereas (P2-Comp.2) only considers the sum of the ℓ *smallest* such values, and similarly for (P2.3) and (P2-Comp.3).

This section proves that these *three* problems are *equivalent* in the following sense:

Lemma 17. *Let $t \in [n]$ and let $z_1, \dots z_{t-1} \in \mathbb{R}$. Then, (x, z_t) is feasible for* (P2) *if and only if (x, z_t) is feasible for* (P2-Comp) *if and only if there exist y_ℓ and $m_{\ell,j}$ for $\ell \in [t]$ and $j \in [n]$ such that $(x, z_t, (y_1, \dots, y_t), (m_{1,1}, \dots m_{t,n}))$ is feasible for* (P3).

It is clear that this lemma implies Lemma 6, which only claims a part of it.

We start by proving that (P2) and (P2-Comp) are equivalent. That is, (x, z_t) is feasible for (P2) if and only if (x, z_t) is feasible for (P2-Comp). First, it is clear that x satisfies constraint (P2.1) if and only if it satisfies constraint (P2-Comp.1) (as both constraints are the same, $x \in S$). To prove the other requirements, we start with the following lemma:

Lemma 18. *For any $x \in S$, any $\ell \in [n]$ and a constant $c \in \mathbb{R}$ the following two conditions are equivalent:*

$$\forall F' \subseteq [n], |F'| = \ell \colon \sum_{i \in F'} f_i(x) \geq c \tag{4}$$

$$\sum_{i=1}^{\ell} \mathcal{V}_i^{\uparrow}(x) \geq c \tag{5}$$

Proof. For the first direction, recall that the values $\mathcal{V}_1^{\uparrow}(x), \ldots, \mathcal{V}_\ell^{\uparrow}(x)$ were obtained from ℓ objective functions (those who yield the smallest value). By the assumption, the sum of any set of function with size ℓ is at least c; therefore, it is true in particular for the functions corresponding to the values $(\mathcal{V}_1^{\uparrow}(x))_{i=1}^{\ell}$. For the second direction, assume that $\sum_{i=1}^{\ell} \mathcal{V}_i^{\uparrow}(x) \geq c$. Since $\mathcal{V}_1^{\uparrow}(x), \ldots, \mathcal{V}_\ell^{\uparrow}(x)$ are the ℓ smallest values in $\mathbf{V}(x)$, we get that:

$$\forall F' \subseteq [n], \ |F'| = \ell \colon \quad \sum_{i \in F'} f_i(x) \geq \sum_{i=1}^{s} \mathcal{V}_i^{\uparrow}(x) \geq c.$$

□

Accordingly, x satisfies constraint (P2.2) — for any $\ell \in [t-1]$,

$$\forall F' \subseteq [n], |F'| = \ell \colon \sum_{i \in F'} f_i(x) \geq \sum_{i=1}^{\ell} z_i$$

if and only if it satisfies $\sum_{i=1}^{\ell} \mathcal{V}_i^{\uparrow}(x) \geq \sum_{i=1}^{\ell} z_i$, which is constraint (P2-Comp.2). Similarly, x and z_t satisfy constraint (P2.3),

$$\forall F' \subseteq [n], |F'| = t \colon \sum_{i \in F'} f_i(x) \geq \sum_{i=1}^{t} z_i$$

if and only if $\sum_{i=1}^{t} \mathcal{V}_i^{\uparrow}(x) \geq \sum_{i=1}^{t} z_i$, which is constraint (P2-Comp.3). That is, x ans z_t satisfy all the constraints of (P2) if and only if they satisfy all the constraints of (P2-Comp).

Now, we will prove that that (P2-Comp) and (P3) are equivalent, that is, (x, z_t) is feasible for (P2-Comp) if and only if there exist y_ℓ and $m_{\ell,j}$ for $\ell \in [t]$ and $j \in [n]$ such that $(x, z_t, (y_1, \ldots, y_t), (m_{1,1}, \ldots m_{t,n}))$ is feasible for (P3).

We start with the following lemma:

Lemma 19. *For any $x \in S$ and any constant $c \in \mathbb{R}$ (where c does not depend on j),*

$$\sum_{j=1}^{n} \max(0, c - f_j(x)) = \sum_{j=1}^{n} \max(0, c - \mathcal{V}_j^{\uparrow}(x)).$$

Proof. Let $(\pi_1, \ldots, \pi_n)$ be a permutation of $\{1, \ldots, n\}$ such that $f_{\pi_i}(x) = \mathcal{V}_i^{\uparrow}(x)$ for any $i \in [n]$ (notice that such permutation exists by the definition of $\mathcal{V}^{\uparrow}()$). That is, the value that f_{π_i} obtains is the π_i-th smallest one in the multiset of all values $\mathbf{V}(x)$. Since each element in the sum $\sum_{j=1}^{n} \max(0, c - f_j(x))$ is affected by j only through $f_j(x)$, the permutation π allows us to conclude the following:

$$\begin{aligned}\sum_{j=1}^{n} \max(0, c - f_j(x)) &= \sum_{j=\pi_1}^{\pi_n} \max(0, c - f_j(x)) \\ &= \sum_{j=1}^{n} \max(0, c - f_{\pi_i}(x)) = \sum_{j=1}^{n} \max(0, c - \mathcal{V}_j^{\uparrow}(x)).\end{aligned}$$

□

Lemma 20 below proves the first direction of the equivalence between (P2-Comp) and (P3):

Lemma 20. *Let (x, z_t) be a feasible solution to* (P2-Comp). *Then there exist y_ℓ and $m_{\ell,j}$ for $\ell \in [t]$ and $j \in [n]$ such that $(x, z_t, (y_1, \ldots, y_t), (m_{1,1}, \ldots m_{t,n}))$ is feasible for* (P3).

Proof. For any $\ell \in [t]$ and $j \in [n]$ define y_ℓ and $m_{\ell,j}$ as follows:

$$y_\ell := \mathcal{V}_\ell^\uparrow(x)$$
$$m_{\ell,j} := \max(0, \mathcal{V}_\ell^\uparrow(x) - f_j(x))$$

First, since x satisfies constraint (P2-Comp.1), it is also satisfies constraint (P3.1) of (as both constraints are the same and include only x).

In addition, based on the choice of y and m, it is clear that $m_{\ell,j} \geq 0$ and $m_{\ell,j} \geq \mathcal{V}_\ell^\uparrow(x) - f_j(x) = y_\ell - f_j(x)$ for any $\ell \in [n]$ and $j \in [n]$. Therefore, this assignment satisfies constraints (P3.4) and (P3.5).

To show that this assignment also satisfies constraints (P3.2) and (P3.3), we first prove that for any $\ell \in [n]$ this assignment satisfies the following equation:

$$\ell y_\ell - \sum_{j=1}^{n} m_{\ell,j} = \sum_{i=1}^{\ell} \mathcal{V}_i^\uparrow(x) \tag{6}$$

By the choice of m, $\sum_{j=1}^n m_{\ell,j} = \sum_{j=1}^n \max(0, \mathcal{V}_\ell^\uparrow(x) - f_j(x))$, and therefore, by Lemma 19, it equals to $\sum_{j=1}^n \max(0, \mathcal{V}_\ell^\uparrow(x) - \mathcal{V}_j^\uparrow(x))$. Since $\mathcal{V}_\ell^\uparrow(x)$ is the ℓ-th smallest objective, it is clear that $\mathcal{V}_\ell^\uparrow(x) - \mathcal{V}_j^\uparrow(x) \leq 0$ for any $j > \ell$, and $\mathcal{V}_\ell^\uparrow(x) - \mathcal{V}_j^\uparrow(x) \geq 0$ for any $j \leq \ell$. And so, $\sum_{j=1}^n m_{\ell,j} = \ell \cdot \mathcal{V}_\ell^\uparrow(x) - \sum_{i=1}^{\ell} \mathcal{V}_i^\uparrow(x)$:

$$\begin{aligned}
&\sum_{j=1}^{n} m_{\ell,j} = \sum_{j=1}^{n} \max(0, \mathcal{V}_\ell^\uparrow(x) - \mathcal{V}_j^\uparrow(x)) \\
&= \sum_{j=1}^{\ell} \max(0, \mathcal{V}_\ell^\uparrow(x) - \mathcal{V}_j^\uparrow(x)) + \sum_{j=\ell+1}^{n} \max(0, \mathcal{V}_\ell^\uparrow(x) - \mathcal{V}_j^\uparrow(x)) \\
&= \sum_{j=1}^{\ell} (\mathcal{V}_\ell^\uparrow(x) - \mathcal{V}_j^\uparrow(x)) + \sum_{j=\ell+1}^{n} 0 = \ell \cdot \mathcal{V}_\ell^\uparrow(x) - \sum_{i=1}^{\ell} \mathcal{V}_i^\uparrow(x)
\end{aligned}$$

We can now conclude Equation (6):

$$\begin{aligned}
\ell y_\ell - \sum_{j=1}^{n} m_{\ell,j} &= \ell \cdot \mathcal{V}_\ell^\uparrow(x) - \ell \cdot \mathcal{V}_\ell^\uparrow(x) + \sum_{i=1}^{\ell} \mathcal{V}_i^\uparrow(x) \\
&= \sum_{i=1}^{\ell} \mathcal{V}_i^\uparrow(x).
\end{aligned}$$

Now, since x satisfies constraint (P2-Comp.2), $\sum_{i=1}^{\ell} \mathcal{V}_i^\uparrow(x) \geq \sum_{i=1}^{\ell} z_i$ for any $\ell \in [t-1]$. Therefore, by Equation (6), $\ell \cdot y_\ell - \sum_{j=1}^n m_{\ell,j} \geq \sum_{i=1}^{\ell} z_i$ for any $\ell \in [t-1]$ and this assignment satisfies constraint (P3.2). Similarly, as x and z_t satisfy constraint (P2-Comp.3), $\sum_{i=1}^{t} \mathcal{V}_i^\uparrow(x) \geq \sum_{i=1}^{t} z_i$, and so by Equation (6), $t \cdot y_t - \sum_{j=1}^n m_{t,j} \geq \sum_{i=1}^{t} z_i$. This means that it also satisfies constraints (P3.3). □

Finally, Lemma 21 below proves the second direction of the equivalence:

Lemma 21. *Let* $(x, z_t, (y_1, \ldots, y_t), (m_{1,1}, \ldots m_{t,n}))$ *be a feasible solution to* (P3)*. Then,* (x, z_t) *is feasible for* (P2-Comp)*.*

Proof. It is easy to see that since x satisfies constraint (P3.1), it is also satisfies constraint (P2-Comp.1) (as both are the same). To show that it also satisfies constraints (P2-Comp.2) and (P2-Comp.3), we start by proving that for any $\ell \in [n]$:

$$\sum_{i=1}^{\ell} \mathcal{V}_i^{\uparrow}(x) \geq \ell y_\ell - \sum_{j=1}^{n} m_{\ell,j} \tag{7}$$

Suppose by contradiction that $\sum_{i=1}^{\ell} \mathcal{V}_i^{\uparrow}(x) < \ell y_\ell - \sum_{j=1}^{n} m_{\ell,j}$.

For any $j \in [n]$ and any $\ell \in [n]$, $m_{\ell,j} \geq y_\ell - f_j(x)$ by constraint (P3.4), and also $m_{\ell,j} \geq 0$ by constraint (P3.5). Therefore, $m_{\ell,j} \geq \max(0, y_\ell - f_j(x))$. And so, by Lemma 19, for any $\ell \in [t]$, the sum of $m_{\ell,j}$ over all $j \in [n]$ can be described as follows:

$$\sum_{j=1}^{n} m_{\ell,j} \geq \sum_{j=1}^{n} \max(0, y_\ell - f_j(x)) = \sum_{j=1}^{n} \max(0, y_\ell - \mathcal{V}_j^{\uparrow}(x)) \tag{8}$$

Therefore, $\sum_{i=1}^{\ell} \mathcal{V}_i^{\uparrow}(x) < \ell y_\ell - \sum_{j=1}^{n} \max(0, y_\ell - \mathcal{V}_j^{\uparrow}(x))$. Which means that:

$$\ell y_\ell - \sum_{i=1}^{\ell} \mathcal{V}_i^{\uparrow}(x) - \sum_{j=1}^{n} \max(0, y_\ell - \mathcal{V}_j^{\uparrow}(x)) > 0$$

However, we will now see that the value of this expression is at most 0, which is a contradiction:

$$\begin{aligned}
&\ell y_\ell - \sum_{i=1}^{\ell} \mathcal{V}_i^{\uparrow}(x) - \sum_{j=1}^{n} \max(0, y_\ell - \mathcal{V}_j^{\uparrow}(x)) \\
&= \sum_{i=1}^{\ell} y_\ell - \sum_{i=1}^{\ell} \mathcal{V}_i^{\uparrow}(x) - \sum_{j=1}^{n} \max(0, y_\ell - \mathcal{V}_j^{\uparrow}(x)) \\
&\leq \sum_{i=1}^{\ell} (y_\ell - \mathcal{V}_i^{\uparrow}(x)) - \sum_{j=1}^{\ell} \max(0, y_\ell - \mathcal{V}_j^{\uparrow}(x)) - \sum_{j=\ell+1}^{n} 0 \quad \text{(Since max with 0 is at least 0)} \\
&= \sum_{j=1}^{\ell} \Big((y_\ell - \mathcal{V}_j^{\uparrow}(x)) - \max(0, y_\ell - \mathcal{V}_j^{\uparrow}(x)) \Big) \\
&\leq 0 \qquad \text{(Since each element is at most 0)}
\end{aligned}$$

This is a contradiction; so (7) is proved.

Now, by constraint (P3.2), $\ell y_\ell - \sum_{j=1}^{n} m_{\ell,j} \geq \sum_{i=1}^{\ell} z_i$ for any $\ell \in [t-1]$. Therefore, by (7), also $\sum_{i=1}^{\ell} \mathcal{V}_i^{\uparrow}(x) \geq \sum_{i=1}^{\ell} z_i$, which means that x satisfies constraint (P2-Comp.2). Similarly, by constraint (P3.3), $t y_t - \sum_{j=1}^{n} m_{t,j} \geq \sum_{i=1}^{t} z_i$, and so by (7), also $\sum_{i=1}^{t} \mathcal{V}_i^{\uparrow}(x) \geq \sum_{i=1}^{t} z_i$. This means that x and z_t satisfy constraint (P2-Comp.3). □

B.3 Proof of Theorem 8

This section is dedicated to proving Theorem 8: 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.

Based on Lemma 17, it is sufficient to prove the theorem for (P2-Comp).

We start by observing that the value of the variable z_t is completely determined by the variable x. This is because z_t only appears in the last constraint, which is equivalent to $z_t \leq \sum_{i=1}^{t} \mathcal{V}_i^{\uparrow}(x) - \sum_{i=1}^{t-1} z_i$. Therefore, for every x that satisfies the first two constraints, it is possible to satisfy the last constraint by setting z_t to any value which is at most $\sum_{i=1}^{t} \mathcal{V}_i^{\uparrow}(x) - \sum_{i=1}^{t-1} z_i$. Moreover, as the program aims to maximize z_t, it will necessarily set z_t to be equal to that expression, since z_t is maximized when the constraint holds with equality. This is summarized in the observation below:

Observation 22. *For any $t \geq 1$ and any constants $z_1, \ldots, z_{t-1}$, every vector x that satisfies constraints (P2-Comp.1) and (P2-Comp.2) is a part of a feasible solution (x, z_t) for $z_t = \sum_{i=1}^{t} \mathcal{V}_i^{\uparrow}(x) - \sum_{i=1}^{t-1} z_i$. Moreover, the objective value obtained by a feasible solution x to the problem* (P2-Comp) *solved in iteration t is $z_t = \sum_{i=1}^{t} \mathcal{V}_i^{\uparrow}(x) - \sum_{i=1}^{t-1} z_i$.*

Based on Observation 22, we can now slightly abuse the terminology and say that a solution x is "feasible" in iteration t if it satisfies constraints (P2-Comp.1) and (P2-Comp.2) of the program solved in iteration t.

We denote $x^* := x_n =$ the solution x attained at the last iteration ($t = n$) of the algorithm. Since x^* is a feasible solution of (P2-Comp) in iteration n, and as each iteration only adds new constraints to (P2-Comp.2), it follows that x^* is also a feasible solution of (P2-Comp) in any iteration $1 \leq t \leq n$.

Observation 23. *x^* is a feasible solution of* (P2-Comp) *in any iteration $1 \leq t \leq n$.*

Lastly, as the value obtained as (α, ϵ)-approximation for this problem is the *constant* z_t, the optimal value is at most $\frac{1}{\alpha}(z_t + \epsilon)$. Consequently, the objective value of any feasible solution is at most this value. Since x^* is feasible for any iteration t (Observation 23) and since the objective value corresponding to x^* is $\sum_{i=1}^{t} \mathcal{V}_i^{\uparrow}(x^*) - \sum_{i=1}^{t-1} z_i$ (Observation 22), we can conclude:

Observation 24. *The objective value obtained by x^* to the problem* (P2-Comp) *that was solved in iteration t is at most $\frac{1}{\alpha}(z_t + \epsilon)$. That is:*

$$\sum_{i=1}^{t} \mathcal{V}_i^{\uparrow}(x^*) - \sum_{i=1}^{t-1} z_i \leq \frac{1}{\alpha}\left(z_t + \epsilon\right).$$

We start with Lemmas 25-27, which establish a relationship between the k-th least objective value obtained by x^* and the difference between the sum of the $(k-1)$ least objective values obtained by x^* and the sum of the $(k-1)$ first z_i values. Theorem 8 then uses this relation to prove that the existence of another solution that would be $\left(\frac{\alpha^2}{1-\alpha+\alpha^2}, \frac{\epsilon}{1-\alpha+\alpha^2}\right)$-leximin-preferred over x^* would lead to a contradiction.

For clarity, throughout the proofs, we denote the multiplicative error factor by $\beta = 1-\alpha$.

Lemma 25. *For any $1 \leq k \leq n$,*

$$\beta \sum_{i=1}^{k} \mathcal{V}_i^{\uparrow}(x^*) - \beta \sum_{i=1}^{k-1} z_i \geq \sum_{i=1}^{k} \mathcal{V}_i^{\uparrow}(x^*) - \sum_{i=1}^{k} z_i - \epsilon$$

Proof. By Observation 24,

$$\begin{aligned}
&\sum_{i=1}^{k} \mathcal{V}_i^{\uparrow}(x^*) - \sum_{i=1}^{k-1} z_i \leq \frac{1}{\alpha}\left(z_k + \epsilon\right) = \frac{1}{1-\beta}\left(z_k + \epsilon\right) \\
&\Rightarrow (1-\beta)\left(\sum_{i=1}^{k} \mathcal{V}_i^{\uparrow}(x^*) - \sum_{i=1}^{k-1} z_i\right) \leq z_k + \epsilon \\
&\Rightarrow \sum_{i=1}^{k} \mathcal{V}_i^{\uparrow}(x^*) - \sum_{i=1}^{k-1} z_i - \beta\sum_{i=1}^{k} \mathcal{V}_i^{\uparrow}(x^*) + \beta\sum_{i=1}^{k-1} z_i \leq z_k + \epsilon \\
&\Rightarrow \sum_{i=1}^{k} \mathcal{V}_i^{\uparrow}(x^*) - \sum_{i=1}^{k} z_i - \epsilon \leq \beta\sum_{i=1}^{k} \mathcal{V}_i^{\uparrow}(x^*) - \beta\sum_{i=1}^{k-1} z_i.
\end{aligned}$$

□

Lemma 26. *For any* $1 \leq k \leq n$,

$$\sum_{i=1}^{k} \beta^i \mathcal{V}_{k-i+1}^{\uparrow}(x^*) \geq \sum_{i=1}^{k} \mathcal{V}_i^{\uparrow}(x^*) - \sum_{i=1}^{k} z_i - \frac{1}{1-\beta}\epsilon$$

Proof. The proof is by induction on k. For $k = 1$ the claim follows directly from Lemma 25 as $\frac{1}{1-\beta} \geq 1$ for any $\beta \in [0, 1)$. Assuming the claim is true for $1, \ldots k-1$, we show it is true for k:

$$\begin{aligned}
&\sum_{i=1}^{k} \beta^i \mathcal{V}_{k-i+1}^{\uparrow}(x^*) = \beta\mathcal{V}_k^{\uparrow}(x^*) + \sum_{i=2}^{k} \beta^i \mathcal{V}_{k-i+1}^{\uparrow}(x^*) \\
&= \beta\mathcal{V}_k^{\uparrow}(x^*) + \sum_{i=1}^{k-1} \beta^{i+1} \mathcal{V}_{k-(i+1)+1}^{\uparrow}(x^*) \\
&= \beta\mathcal{V}_k^{\uparrow}(x^*) + \beta\sum_{i=1}^{k-1} \beta^i \mathcal{V}_{(k-1)-i+1}^{\uparrow}(x^*) \\
&\geq \beta\mathcal{V}_k^{\uparrow}(x^*) + \beta\left(\sum_{i=1}^{k-1} \mathcal{V}_i^{\uparrow}(x^*) - \sum_{i=1}^{k-1} z_i - \frac{1}{1-\beta}\epsilon\right) && \text{(By induction assumption)} \\
&= \beta\sum_{i=1}^{k} \mathcal{V}_i^{\uparrow}(x^*) - \beta\sum_{i=1}^{k-1} z_i - \frac{\beta}{1-\beta}\epsilon \\
&\geq \sum_{i=1}^{k} \mathcal{V}_i^{\uparrow}(x^*) - \sum_{i=1}^{k} z_i - \epsilon - \frac{\beta}{1-\beta}\epsilon && \text{(By Lemma 25)} \\
&= \sum_{i=1}^{k} \mathcal{V}_i^{\uparrow}(x^*) - \sum_{i=1}^{k} z_i - \frac{1}{1-\beta}\epsilon
\end{aligned}$$

□

Lemma 27. *For all* $1 < k \leq n$,

$$\frac{\beta}{1-\beta}\mathcal{V}_k^{\uparrow}(x^*) \geq \sum_{i=1}^{k-1} \mathcal{V}_i^{\uparrow}(x^*) - \sum_{i=1}^{k-1} z_i - \frac{1}{1-\beta}\epsilon$$

Proof. First, notice that since $k \geq (k-1)-i+1$ for any $1 \leq i \leq k$ and as the function $\mathcal{V}_i^{\uparrow}()$ represents the i-th smallest objective value, also:

$$\forall 1 \leq i \leq k\colon \quad \mathcal{V}_k^{\uparrow}(x^*) \geq \mathcal{V}_{(k-1)-i+1}^{\uparrow}(x^*) \tag{9}$$

In addition, consider the geometric series with a first element 1, a ratio β, and a length $(k-1)$. As $\beta < 1$, its sum can be bounded in the following way:

$$\sum_{i=1}^{k-1} \beta^{i-1} = \frac{1-\beta^{k-1}}{1-\beta} < \lim_{k\to\infty} \frac{1-\beta^{k-1}}{1-\beta} = \frac{1}{1-\beta} \tag{10}$$

Now, the claim can be concluded as follows:

$$\begin{aligned}
&\frac{\beta}{1-\beta}\mathcal{V}_k^{\uparrow}(x^*) = \beta\left(\frac{1}{1-\beta}\mathcal{V}_k^{\uparrow}(x^*)\right) \\
&> \beta\left(\sum_{i=1}^{k-1}\beta^{i-1}\mathcal{V}_k^{\uparrow}(x^*)\right) && \text{(By Equation (10))} \\
&\geq \beta\left(\sum_{i=1}^{k-1}\beta^{i-1}\mathcal{V}_{(k-1)-i+1}^{\uparrow}(x^*)\right) && \text{(By Equation (9))} \\
&= \sum_{i=1}^{k-1}\beta^{i}\mathcal{V}_{(k-1)-i+1}^{\uparrow}(x^*) \\
&\geq \sum_{i=1}^{k-1}\mathcal{V}_i^{\uparrow}(x^*) - \sum_{i=1}^{k-1} z_i - \frac{1}{1-\beta}\epsilon && \text{(By Lemma 26 for } (k-1)\geq 1)
\end{aligned}$$

□

We are now ready to prove the Theorem 8.

Proof of Theorem 8. Recall that the claim is that x^* is a $\left(\frac{\alpha^2}{1-\alpha+\alpha^2}, \frac{\epsilon}{1-\alpha+\alpha^2}\right)$-leximin-approximation.

For brevity, we define the following constant:

$$\Delta(\alpha) = \frac{1}{1-\alpha+\alpha^2}$$

Accordingly, we need to prove that x^* is a $\big(\alpha^2\cdot\Delta(\alpha), \epsilon\cdot\Delta(\alpha)\big)$-approximation.

As $\alpha = 1-\beta$, it is easy to verify that:

$$\Delta(\alpha) = \frac{1}{1-\beta+\beta^2} \tag{11}$$

Now, suppose by contradiction that x^* is *not* $\big(\alpha^2\cdot\Delta(\alpha), \epsilon\cdot\Delta(\alpha)\big)$-approximate leximin-optimal. By definition, this means there exists a solution $y \in S$ that is $\big(\alpha^2\cdot\Delta(\alpha), \epsilon\cdot\Delta(\alpha)\big)$-leximin-preferred over it. That is, there exists an integer $1 \leq k \leq n$ such that:

$$\begin{aligned}
&\forall j < k\colon \mathcal{V}_j^{\uparrow}(y) \geq \mathcal{V}_j^{\uparrow}(x^*); \\
&\qquad \mathcal{V}_k^{\uparrow}(y) > \frac{1}{\alpha^2\cdot\Delta(\alpha)}\left(\mathcal{V}_k^{\uparrow}(x^*) + \epsilon\cdot\Delta(\alpha)\right).
\end{aligned}$$

Since x^* was obtained in (P2-Comp) that was solved in the last iteration n, it is clear that $\sum_{i=1}^{k} \mathcal{V}_i^{\uparrow}(x^*) \geq \sum_{i=1}^{k} z_i$ (by constraint (P2-Comp.2) if $k < n$ and (P2-Comp.3) otherwise). Which implies:

$$\sum_{i=1}^{k} \mathcal{V}_i^{\uparrow}(x^*) - \sum_{i=1}^{k-1} z_i \geq z_k \tag{12}$$

Now, consider (P2-Comp) that was solved in iteration k. By Observation 23, x^* is feasible to this problem. As the $(k-1)$ smallest objective values of y are at least as high as those of x^*, it is easy to conclude that y also satisfies constraints (P2-Comp.2) of this problem; since, for any $\ell < k$:

$$\sum_{i=1}^{\ell} \mathcal{V}_i^{\uparrow}(y) \geq \sum_{i=1}^{\ell} \mathcal{V}_i^{\uparrow}(x^*) \geq \sum_{i=1}^{\ell} z_i$$

Therefore, by Observation 22, y is also feasible to this problem.

The value obtained during the algorithm run as an approximation for this problem is z_k. This means that the optimal value is at most $\frac{1}{\alpha}(z_k + \epsilon)$. As y is feasible in this problem, and since the objective value obtained by y in this problem is $\sum_{i=1}^{k} \mathcal{V}_i^{\uparrow}(y) - \sum_{i=1}^{k-1} z_i$ (by Observation 22), this implies the following:

$$\sum_{i=1}^{k} \mathcal{V}_i^{\uparrow}(y) - \sum_{i=1}^{k-1} z_i \leq \frac{1}{(1-\beta)}(z_k + \epsilon) \tag{13}$$

If $k = 1$, we get that the objective value obtained by y is $\mathcal{V}_1^{\uparrow}(y)$. In addition, $\mathcal{V}_1^{\uparrow}(x^*) \geq z_1$ by Equation (12). However, as $0 < \alpha \leq 1$, then $\Delta(\alpha) \geq 1$ but $\alpha \cdot \Delta(\alpha) \leq 1$. It follows that:

$$\mathcal{V}_1^{\uparrow}(y) > \frac{1}{\alpha^2 \cdot \Delta(\alpha)} \left(\mathcal{V}_1^{\uparrow}(x^*) + \epsilon \cdot \Delta(\alpha) \right) \geq \frac{1}{\alpha}(z_1 + \epsilon)$$

In contradiction to Equation (13).

Therefore, $k > 1$. We start by showing that the following holds:

$$\mathcal{V}_k^{\uparrow}(y) > \frac{1}{1-\beta} \left(\mathcal{V}_k^{\uparrow}(x^*) + \beta \sum_{i=1}^{k-1} \mathcal{V}_i^{\uparrow}(x^*) - \beta \sum_{i=1}^{k-1} z_i + \epsilon \right) \tag{14}$$

Consider $\mathcal{V}_k^{\uparrow}(y)$, by the definition of y for k we get that:

$$\begin{aligned}
&\mathcal{V}_k^{\uparrow}(y) > \frac{1}{\alpha^2 \cdot \Delta(\alpha)} \left(\mathcal{V}_k^{\uparrow}(x^*) + \epsilon \cdot \Delta(\alpha) \right) \\
&= \frac{1}{\alpha} \left(\frac{1 - \beta + \beta^2}{1-\beta} \mathcal{V}_k^{\uparrow}(x^*) + \frac{1}{\alpha}\epsilon \right) && \text{(By Equ. (11) and } \beta\text{'s def.)} \\
&= \frac{1}{\alpha} \left(\mathcal{V}_k^{\uparrow}(x^*) + \frac{\beta^2}{1-\beta} \mathcal{V}_k^{\uparrow}(x^*) + \frac{1}{\alpha}\epsilon \right) \\
&\geq \frac{1}{\alpha} \left(\mathcal{V}_k^{\uparrow}(x^*) + \beta \Big[\sum_{i=1}^{k-1} \mathcal{V}_i^{\uparrow}(x^*) - \sum_{i=1}^{k-1} z_i - \frac{1}{1-\beta}\epsilon \Big] + \frac{1}{\alpha}\epsilon \right) && \text{(By Lemma 27)} \\
&\frac{1}{1-\beta} \left(\mathcal{V}_k^{\uparrow}(x^*) + \beta \sum_{i=1}^{k-1} \mathcal{V}_i^{\uparrow}(x^*) - \beta \sum_{i=1}^{k-1} z_i + \epsilon \right) && \text{(By } \beta\text{'s def.)}
\end{aligned}$$

But, we shall now prove that this means, once again, that the objective value of y, which is $\sum_{i=1}^{k} \mathcal{V}_i^{\uparrow}(y) - \sum_{i=1}^{k-1} z_i$, is higher than $\frac{1}{1-\beta}(z_k + \epsilon)$, in contradiction to Equation (13):

$$\sum_{i=1}^{k} \mathcal{V}_i^{\uparrow}(y) - \sum_{i=1}^{k-1} z_i = \sum_{i=1}^{k-1} \mathcal{V}_i^{\uparrow}(y) - \sum_{i=1}^{k-1} z_i + \mathcal{V}_k^{\uparrow}(y)$$

By the definition of y for $i < k$:

$$\geq \sum_{i=1}^{k-1} \mathcal{V}_i^{\uparrow}(x^*) - \sum_{i=1}^{k-1} z_i + \mathcal{V}_k^{\uparrow}(y)$$

By Equation (14):

$$> \sum_{i=1}^{k-1} \mathcal{V}_i^{\uparrow}(x^*) - \sum_{i=1}^{k-1} z_i + \frac{1}{1-\beta}\mathcal{V}_k^{\uparrow}(x^*) + \frac{\beta}{1-\beta}\sum_{i=1}^{k-1} \mathcal{V}_i^{\uparrow}(x^*) - \frac{\beta}{1-\beta}\sum_{i=1}^{k-1} z_i + \frac{1}{1-\beta} \cdot \epsilon$$

Since $1 + \frac{\beta}{1-\beta} = \frac{1}{1-\beta}$, this equals to:

$$= \frac{1}{1-\beta}\sum_{i=1}^{k-1} \mathcal{V}_i^{\uparrow}(x^*) - \frac{1}{1-\beta}\sum_{i=1}^{k-1} z_i + \frac{1}{1-\beta}\mathcal{V}_k^{\uparrow}(x^*) + \frac{1}{1-\beta}\epsilon$$

$$= \frac{1}{1-\beta}\left(\sum_{i=1}^{k} \mathcal{V}_k^{\uparrow}(x^*) - \sum_{i=1}^{k-1} z_i + \epsilon\right)$$

By Equation (12):

$$\geq \frac{1}{1-\beta}(z_k + \epsilon).$$

This is a contradiction, so Theorem 8 is proved. □

C Proof of Theorem 10

This section proves Theorem 10: suppose we are given a randomized algorithm that returns a simple allocation that approximates the utilitarian welfare with multiplicative error β (with success probability p). Then, Algorithm 1 can be used to obtain a stochastic allocation that approximates leximin with a multiplicative error of at most $\frac{\beta}{1-\beta+\beta^2}$ (with the same probability).

As we saw in Section 4, an approximation to leximin can be obtained by providing a procedure OP to approximate (P3) (Theorem 8), which, under these particular settings,

becomes:

$$\begin{aligned}
&\max \quad z_t \quad s.t. && && \text{(C1)}\\
&\text{(C1.1.1)} \ \sum_{A\in\mathcal{A}} p_d(A) = 1 && &&\\
&\text{(C1.1.2)} \ p_d(A) \geq 0 && \forall A \in \mathcal{A} &\\
&\text{(C1.2)} \ \ell y_\ell - \sum_{j=1}^{n} m_{\ell,j} \geq \sum_{i=1}^{\ell} z_i && \ell = 1,\ldots,t-1 &\\
&\text{(C1.3)} \ t y_t - \sum_{j=1}^{n} m_{t,j} \geq \sum_{i=1}^{t} z_i && &\\
&\text{(C1.4)} \ m_{\ell,j} \geq y_\ell - \sum_{A\in\mathcal{A}} p_d(A)\cdot u_j(A) && \ell = 1,\ldots,t, \ \ j = 1,\ldots,n &\\
&\text{(C1.5)} \ m_{\ell,j} \geq 0 && \ell = 1,\ldots,t, \ \ j = 1,\ldots,n &
\end{aligned}$$

Here the variables are $p_d(A)$ for any simple allocation $A \in \mathcal{A}$, z_t, and y_ℓ and $m_{\ell,j}$ for all $\ell \in [t]$ and $j \in [n]$; and the values $z_1, \ldots z_{t-1}$ are constants. Notice that it is a *linear program* that has a polynomial number of constraints thanks to (P3) representation, but an exponential number of variables (since there is a variable $p_d(A)$ for each simple allocation). So, it is unclear how to approach it directly in polynomial time. In addition, it means that the output size is exponential in n. To deal with this issue, the solutions are considered in *sparse form* — a list of the variables with positive values, along with their values. Accordingly, if a solution has only a polynomial number of variables with positive values it can be represented by a polynomial size. We will later see that the procedure described in this section returns such a solution in polynomial time.

With $t = 1$, (C1) can be viewed as the problem of egalitarian welfare maximization, indeed, Kawase and Sumita [16] who studied this problem, considered a slightly simpler representation. We now show how their dual-based technique can be applied to approximate (C1) for any $t \geq 1$ while maintaining the same approximation factor.

To begin, consider the following program (C2), which is the result of modifying (C1) in three ways. First, changing the objective-function to $\min 1/z_t$ instead of $\max z_t$. Second, replacing all the original variables and constants, except z_t, with new ones that are smaller by a factor z_t (that is, $p'_A = p_d(A)/z_t$ for all $A \in \mathcal{A}$, , $y'_\ell = y_\ell/z_t, m'_{\ell,j} = m_{\ell,j}/z_t$ for $\ell \in [t]$ and $j \in [n]$, and $z'_i = z_i/z_t$ for $i \in [t-1]$). And third, dividing all the constraints by z_t ($z_t > 0$ since $z_t \geq z_1$ for any $t \geq 1$ and $z_1 > 0$).

$$\min \quad 1/z_t \quad s.t. \tag{C2}$$

$$\text{(C2.1.1)} \quad \sum_{A\in\mathcal{A}} p'_A = 1/z_t$$

$$\text{(C2.1.2)} \quad p'_A \geq 0 \qquad \forall A \in \mathcal{A}$$

$$\text{(C2.2)} \quad \ell y'_\ell - \sum_{j=1}^{n} m'_{\ell,j} \geq \sum_{i=1}^{\ell} z'_i \qquad \ell = 1,\ldots,t-1$$

$$\text{(C2.3)} \quad t y'_t - \sum_{j=1}^{n} m'_{t,j} \geq \sum_{i=1}^{t-1} z'_i + 1$$

$$\text{(C2.4)} \quad m'_{\ell,j} \geq y'_\ell - \sum_{A\in\mathcal{A}} p'_A \cdot u_j(A) \qquad \ell = 1,\ldots,t,\ \ j = 1,\ldots,n$$

$$\text{(C2.5)} \quad m'_{\ell,j} \geq 0 \qquad \ell = 1,\ldots,t,\ \ j = 1,\ldots,n$$

The programs (C1) and (C2) are related in the following way:

Lemma 28. *There exists a bijection mapping each solution of* (C1) *with objective value* V *to a unique solution of* (C2) *with objective value* $1/V$.

Proof. Let $p_d(A)$ for $A \in \mathcal{A}$, z_t, and y_ℓ and $m_{\ell,j}$ for all $\ell \in [t]$ and $j \in [n]$ be a feasible solution to the program (C1) with objective value V. It can be easily verified that $p'_A = p_d(A)/z_t$ for $A \in \mathcal{A}$, z_t, and $y'_\ell = y_\ell/z_t$ and $m'_{\ell,j} = m_{\ell,j}/z_t$ for all $\ell \in [t]$ and $j \in [n]$ is a feasible solution to the program (C2) with objective value $1/V$. □

Denote this bijection by Ψ, this also implies the following:

Lemma 29. *If a solution approximates the program* (C2) *with a multiplicative error of* $\frac{\beta}{1-\beta}$. *Then the corresponding solution to* (C1) *according to the bijection* Ψ *approximates this program with a multiplicative error of* β.

Proof. Let V^* be the optimal objective value of (C1). By Lemma 28, there exists a solution to (C2) with value $1/V^*$. This solution yields the optimal value for (C2) — if there was a solution that had a value *lower* than $1/V^*$ ((C2) is a minimization problem), then the corresponding solution to (C1) (by the bijection Ψ) would have a value higher than the optimal value V^*. Now, let the value of the solution that approximates the program (C2) with a multiplicative error of $\frac{\beta}{1-\beta}$ be $1/V$. Since (C2) is a minimization problem, assuming that $1/V$ approximates $1/V^*$ with a multiplicative error of $\frac{\beta}{1-\beta}$ means that:

$$\frac{1}{V} \leq \left(1 + \frac{\beta}{1-\beta}\right)\frac{1}{V^*},$$

which implies that $V \geq (1-\beta)V^*$. As (C1) is a maximization problem, this means that V approximates this problem with multiplicative error β. By Lemma 28, V is the value of the corresponding solution to (C1) by the bijection Ψ. □

Notice that the only constraint of (C2) that includes the variable z_t, (C2.1.1), says that $\sum_{A\in\mathcal{A}} p'_A = 1/z_t$, and also that its objective function is $\min 1/z_t$. As a result, we can reduce the need for the variable z_t by removing constraint (C2.1.1) and changing the objective function to $\min \sum_{A\in\mathcal{A}} p'_A$. This change makes (C2) a *linear* program. This will allow us to approximate it using its dual, as we will see.

The following observation will be useful later:

Observation 30. *If a solution to* (C2) *is given in a sparse form — a list of the variables with nonzero value and their values, then the corresponding solution to* (C1) *in a sparse form can be computed in time polynomial to the number of nonzero variables.*

For completeness, we briefly outline the process. When given a list of variables with nonzero values, we first iterate the list and sum all variables of the form p'_A, and then set z_t to be 1 divided by this sum. After, for each variable ν' in the list, we set the corresponding variable, ν, to $z_t \cdot \nu'$.

Now, let us consider the dual program of (C2), which can be described as follows:

$$\max \quad \sum_{\ell=1}^{t-1} q_\ell \sum_{i=1}^{\ell} z_i + q_t(\sum_{i=1}^{t-1} z_i + 1) \tag{D2}$$

$$\begin{aligned}
s.t. \quad & \text{(D2.1)} \ \sum_{j=1}^{n} u_j(A) \sum_{\ell=1}^{t} v_{\ell,j} \leq 1 && \forall A \in \mathcal{A} \\
& \text{(D2.2)} \ \ell q_\ell - \sum_{j=1}^{n} v_{\ell,j} = 0 && \ell = 1, \ldots, t \\
& \text{(D2.3)} \ q_\ell - v_{\ell,j} \leq 0 && \ell = 1, \ldots, t, \ \ j = 1, \ldots, n \\
& \text{(D2.4)} \ v_{\ell,j} \geq 0 && \ell = 1, \ldots, t, \ \ j = 1, \ldots, n \\
& \text{(D2.5)} \ q_\ell \geq 0 && \ell = 1, \ldots, t
\end{aligned}$$

Here, the variables are q_ℓ and $v_{\ell,j}$ for any $\ell \in [t]$ and $j \in [n]$; and the constants are (as before) z_i for $i \in [t-1]$. Recall that $u_j(A)$ is the utility that agent j assigns to simple allocation A, as given by the value oracle. This problem has an exponential number of constraints — a constraint for each allocation (in line (D2.1)) but only a polynomial number of variables. Using the ellipsoid method [11], it could be solved in polynomial time if we had a *separation oracle* — an oracle that given a vector v either determines that v is infeasible and returns a violated constraint, or asserts that v is feasible. Unfortunately, as we shall now see, it is NP-hard to compute a separation oracle to this problem.

Lemma 31. *Computing a separation oracle to* (D2) *is NP-hard.*

Proof. We prove that a separation oracle for (D2) would allow us to compute a leximin optimal stochastic allocation. As discussed previously, computing such an allocation is NP-hard, so the same applies for computing a separation oracle for (D2).

First, we prove that such a separation oracle can be used to extract an optimal solution to (C2). Assume that the ellipsoid method was operated with the given oracle to solve (D2). Let C be the set of constraints that the oracle determined as being violated. Since the ellipsoid method operates in polynomial time, the size of the set C is also polynomial. Let V_C be the set of variables of (C2) associated with the constraints in C. By complementary slackness, the variables in V_C are the only ones that may get a *positive* value in the corresponding optimal solution to (C2). Therefore, the program (C2) with only the variables in V_C (and the other variables equal to zero) has a polynomial size, and therefore can be solved exactly.

But, by Observation 30, this would allow us to find the corresponding optimal solution to (C1) in polynomial time. This means the described process can be used as an approximation procedure to (P3) (that became (C1) under the settings of this problem) with $\beta = \epsilon = 0$. Therefore, by Theorem 8, this means we can use Algorithm 1 to obtain a leximin optimal solution[13]. □

[13] Actually, Theorem 8 says that Algorithm 1 will output a $(1, 0)$-leximin-approximation; But Lemma 3 says that such a solution is, indeed, a leximin optimal solution.

Algorithm 2 A Half-Randomized Approximate Separation Oracle to (D2)

INPUT: variables q_ℓ and $v_{\ell,j}$ for any $\ell \in [t]$ and $j \in [n]$, an α-approximation algorithm for the utilitarian welfare problem ((U1)) with success probability p.

Iterate over constraints (D2.2)-(D2.5). If one of them is violated, stop and return it.
If $p = 1$ then set $T := 1$; **else** set $T := 1 + \lceil -\log_{(1-p)}(nI) \rceil$.
repeat T **times**
  Operate the algorithm for the utilitarian welfare problem on $n, m, (u'_j)_{j=1}^n$ to obtain an allocation $\tilde{A}$ with value ν.
  if $\nu > 1$ **then**
    Return the corresponding violated constraint $\sum_{j=1}^n u_j(\tilde{A}) \sum_{\ell=1}^t v_{\ell,j} > 1$
  end if
end repeat
Return "the assignment is approximately-feasible".

In Appendix D, we present another variant of the ellipsoid method, which allows us to approximate the program (C2) given a *half-randomized approximate separation oracle* to (D2). That is, an oracle that, given a multiplicative error β, a success probability p, and a vector v, either determines that v is infeasible and returns a violated constraint; or determines that v is β-*approximately-feasible*, which means that for any constraint $a \cdot x \leq b$, the vector v satisfies $a \cdot v \leq (1+\beta) \cdot b$. When the oracle says that v is β-approximately-feasible, it is correct with probability at least p. Given such an oracle for the dual program, the ellipsoid method variant can be used to output a solution to the primal, that approximates it to the same factor with probability at least p^I, where I is an upper bound on the number of iterations in any execution of the ellipsoid method variant on the dual (if it is given a deterministic oracle). We can therefore conclude the following result:

Lemma 32. *Given a half-randomized approximate separation oracle to the problem* (D2)*, with a multiplicative error of* $\frac{\beta}{1-\beta}$ *and a success probability* p*, a stochastic allocation that approximates leximin to a multiplicative error* $\frac{\beta}{1-\beta+\beta^2}$ *can be obtained with probability* p^{nI}*.*

Proof. As described above, we can use the ellipsoid method variant of Appendix D with the given oracle to (D2) to obtain a solution to (C2), that approximates it with a multiplicative error of $\frac{\beta}{1-\beta}$ with probability p^I. Then, by Observation 30, this would allow us to find the corresponding solution to (C1), that, with probability p^I, approximates it with a multiplicative error of β. That is, the described process can be used as a randomized approximation procedure to (P3) (that became (C1) under the settings of this problem). Therefore, by Theorem 8, Algorithm 1 can be used to obtain a leximin approximation to the original problem with only a multiplicative error of $\frac{\beta}{1-\beta+\beta^2}$ with probability p^{nI} (Corollary 9). □

Now, we show that such an oracle can be designed given a randomized approximation algorithm for computing a simple allocation that approximates the utilitarian welfare. Specifically,

Lemma 33. *Given a randomized approximation algorithm for computing a simple allocation that approximates the utilitarian welfare with multiplicative error* β *and a success probability* p*, a half-randomized approximate separation oracle to* (D2) *can be designed with a multiplicative error of* $\frac{\beta}{1-\beta}$ *and a success probability at least* $\left(1 - \frac{1}{nI}(1-p)\right)$*.*

Algorithm 2 describes the oracle. It accepts as input an assignment to the variables of (D2), that is, q_ℓ and $v_{\ell,j}$ for any $\ell \in [t]$ and $j \in [n]$, and an algorithm for approximating

the maximum utilitarian welfare. It starts by verifying constraints (D2.2)-(D2.5) one by one (this is possible as their number is polynomial in n and m). If a violated constraint was found, the oracle simply returns it. Otherwise, it proceeds to check constraints (D2.1). Although the number of constraints described by (D2.1) is exponential in n, they can be treated collectively in polynomial time (as in [16]). First, notice that in order to determine whether the expression $\sum_{j=1}^{n} u_j(A) \sum_{\ell=1}^{t} v_{\ell,j}$ is at most 1 for all simple allocations ($A \in \mathcal{A}$), it is sufficient to check the allocation that maximizes this expression and compare it to 1. Define new utility functions for all $j \in [n]$ and $A \in \mathcal{A}$,

$$u'_j(A) := \sum_{\ell=1}^{t} v_{\ell,j} \cdot u_j(A)$$

The above expression can be simplified to $\sum_{j=1}^{n} u'_j(A)$. An allocation that maximizes this expression is an allocation that maximizes the utilitarian welfare (i.e., the sum of utilities) when the same sets of agents and items is considered but with different utilities[14] (u'_j instead of u_j for $j \in [n]$). Such an allocation cannot be found in polynomial time since approximating the utilitarian welfare up to a factor better than $(1 - \frac{1}{e})$ in the case of submodular utilities is known to be NP-hard [17]. However, the oracle is given an approximation algorithm to the utilitarian welfare problem as input. Therefore, an allocation $\tilde{A}$ with utilitarian value at least $(1 - \beta)$ of the optimal can be obtained with probability p. We shall now see that it is enough.

Proof of Lemma 33. First, observe that when Algorithm 2 returns a violated constraint, it is always correct. This is obvious for constraints described by (D2.2)-(D2.5), since these constraints have been verified directly. For constraints described by (D2.1), it means that the algorithm found an allocation $\tilde{A}$ that satisfies $\sum_{j=1}^{n} u'_j(\tilde{A}) > 1$. By the definition of u', the constraint corresponding to this allocation is, indeed, violated:

$$\sum_{j=1}^{n} u_j(\tilde{A}) \sum_{\ell=1}^{t} v_{\ell,j} = \sum_{j=1}^{n} u'_j(\tilde{A}) > 1.$$

Let us assume that the given algorithm for the utilitarian welfare problem is deterministic (i.e., $p = 1$) and then revisit the case $p < 1$. Assume that the oracle said that the assignment is approximately-feasible. This means that the algorithm for the utilitarian welfare problem found an allocation $\tilde{A}$ with value at most 1. Since $\tilde{A}$ is approximately-optimal, the optimal utilitarian value is at most $1/(1-\beta) \cdot 1$. As this is an upper bound of the utilitarian value of any allocation, it follows that all the constraints described bu (D2.1) are $\frac{\beta}{1-\beta}$-approximately maintained — that is, for any allocation $A \in \mathcal{A}$ the following holds:

$$\sum_{j=1}^{n} u'_j(A) = \sum_{j=1}^{n} u_j(A) \sum_{\ell=1}^{t} v_{\ell,j} \leq \frac{1}{1-\beta} \cdot 1 = \left(1 + \frac{\beta}{1-\beta}\right) \cdot 1$$

We get that, in this case, the oracle is also deterministic, and that the success probability is at least $\left(1 - \frac{1}{nI}(1-p)\right) = 1$ for $p = 1$.

Assume now that $p < 1$. Then, the oracle may be incorrect when it says the assignment is approximately feasible, but only if the algorithm for the utilitarian welfare problem did not return an appropriate approximation in all $T = \lceil -\log_{(1-p)}(nI) \rceil + 1$ operations, that

[14] Notice that the utilities u'_j are normalized, monotone, submodular, and can be computed using $t \leq n$ calls to the value oracle of u_j

is, with probability at most $(1-p)^T$. Notice that $T > 1$ since $\log_{(1-p)}(nI) < 0$[15]. Now, as $T \geq -\log_{(1-p)}(nI) + 1$ and $(1-p) < 1$ we get that:

$$(1-p)^T \leq (1-p) \cdot (1-p)^{-\log_{(1-p)}(nI)} = (1-p)(nI)^{-1}$$

So, the success probability is at least $\left(1 - \frac{1}{nI}(1-p)\right)$. □

We can now prove Theorem 10.

Proof of Theorem 10. Assume we are given an algorithm that returns a simple allocation that approximates the utilitarian welfare with multiplicative error β with success probability p. By Lemma 33 this algorithm can be used to obtain an half-randomized approximate separation oracle to (D2) with a multiplicative error $\frac{\beta}{1-\beta}$ with success probability $\left(1 - \frac{1}{nI}(1-p)\right)$. By Lemma 32, with such an oracle a stochastic allocation that approximates leximin to a multiplicative error of $\frac{\beta}{1-\beta+\beta^2}$ can be obtained with probability $\left(1 - \frac{1}{nI}(1-p)\right)^{nI}$. If $p = 1$ then the success probability is 1 too (at least $\left(1 - \frac{1}{nI}(1-p)\right)^{nI} = 1$). However, if $p < 1$, then $\frac{1}{nI}(1-p) \in (0,1)$ and therefore the success probability is at least p[16]:

$$\left(1 - \frac{1}{nI}(1-p)\right)^{nI} \geq \left(1 - nI \cdot \frac{1}{nI}(1-p)\right) = p.$$

□

D Ellipsoid Method Variant for Approximation

This Appendix describes a variant of the ellipsoid method that can be used to approximate LPs that cannot be solved directly due to a large number of variables. The method combines techniques presented in [12, 11, 15].

D.1 Using Approximate Separation Oracles (multiple error)

Our goal is to solve the following linear program (the primal):

$$\begin{aligned} \min \quad & c^T \cdot x \\ s.t. \quad & A \cdot x \geq b, \quad x \geq 0; \end{aligned} \tag{P}$$

We assume that (P) has a small number of constraints, but may have a huge number of variables, so we cannot solve (P) directly. We consider its *dual*:

$$\begin{aligned} \max \quad & b^T \cdot y \\ s.t. \quad & A^T \cdot y \leq c, \quad y \geq 0. \end{aligned} \tag{D}$$

Assume that both problems have optimal solutions and denote the optimal solutions of (P) and (D) by x^* and y^* respectively. By the strong duality theorem:

$$c^T \cdot x^* = b^T \cdot y^* \tag{15}$$

While (D) has a small number of variables, it has a huge number of constraints, so we cannot solve it directly either. In this Appendix, we show that (p) can be approximated using the following tool:

[15] Since $(1-p) \in (0,1)$ and $nI > 1$ by change of base: $\log_{(1-p)}(nI) = \log(nI)/\log(1-p)$, the numerator is positive and the denominator is negative.

[16] For any $\epsilon \in (0,1)$ and $k \in \mathbb{Z}_+$: $(1-\epsilon)^k \geq 1 - k \cdot \epsilon$

Definition D.1. An *approximate separation oracle* with multiplicative error (MASO) for the dual LP is an efficient function parameterized by a constant $\beta \geq 0$. Given a vector y it returns one of the following two answers:

1. "y is infeasible". In this case, is returns a violated constraint, that is, a row $a_i^T \in A^T$ such that $a_i^T y > c_i$.
2. "y is *approximately feasible*". That means that $A^T y \leq (1+\beta) \cdot c$

Given the MASO, we apply the ellipsoid method as follows (this is just a sketch to illustrate the way we use the MASO; it omits some technical details):

- Let E_0 be a large ellipsoid, that contains the entire feasible region, that is, all $y \geq 0$ for which $A^T y \leq c$.
- For $k = 0, 1, \ldots, K$ (where K is a fixed constant, as will be explained later):
  - Let y_k be the centroid of ellipsoid E_k.
  - Run the MASO on y_k.
  - If the MASO returns "y_k is infeasible" and a violated constraint a_i^T, then make a *feasibility cut* — keep in E_{k+1} only those $y \in E_k$ for which $a_i^T y \leq c_i$.
  - If the MASO returns "y is approximately feasible", then make an *optimality cut* — keep in E_{k+1} only those $y \in E_k$ for which $b^T y \geq b^T y_k$.
- From the set $y_0, y_1, \ldots, y_K$, choose the point with the highest $b^T \cdot y_k$ among all the approximately-feasible points.

Since both cuts are through the center of the ellipsoid, the ellipsoid dilates by a factor of at least $\frac{1}{r}$ at each iteration, where $r > 1$ is some constant (see [11] for computation of r). Therefore, by choosing $K := \log_2 r \cdot L$, where L is the number of bits in the binary representation of the input, the last ellipsoid E_K is so small that all points in it can be considered equal (up to the accuracy of the binary representation).

The solution y' returned by the above algorithm satisfies the following two conditions:

$$A^T y' \leq (1+\beta) \cdot c \tag{16}$$

$$b^T y' \geq b^T y^* \tag{17}$$

Inequality 16 holds since, by definition, y' is approximately-feasible.

To prove 17, suppose by contradiction that $b^T y^* > b^T y'$. Since y^* is feasible for (D), it is in the initial ellipsoid. It remains in the ellipsoid throughout the algorithm: it is removed neither by a feasibility cut (since it is feasible), nor by an optimality cut (since its value is at least as large as all values used for optimality cuts). Therefore, it remains in the final ellipsoid, and it is chosen as the highest-valued feasible point rather than y' — a contradiction.

Now, we construct a reduced version of (D), where there are only at most K constraints — only the constraints used to make feasibility cuts. Denote the reduced constraints by $A_{red}^T \cdot y \leq c_{red}$, where A_{red}^T is a matrix containing a subset of at most K rows of of A^T, and c_{red} is a vector containing the corresponding subset of the elements of c. The reduced-dual LP is:

$$\begin{aligned} \max \quad & b^T y \\ s.t. \quad & A_{red}^T \cdot y \leq c_{red}, \quad y \geq 0 \end{aligned} \tag{RD}$$

Notice that it has the same number of variables as the program (D). Further, if we had run this ellipsoid method variant on (RD) (instead of (D)), then the result would have been exactly the same — y'. Therefore, (17) holds for the (RD) too:

$$b^T y' \geq b^T y^*_{red} \tag{18}$$

where y^*_{red} is the optimal value of (RD).

As A^T_{red} contains a subset of at most K rows of A^T, the matrix A_{red} contains a subset of *columns* of A. Therefore, the dual of (RD) has only at most K variables, which are those who correspond to the remaining columns of A:

$$\begin{aligned} \min \quad & c^T_{red} \cdot x_{red} \\ s.t. \quad & A_{red} \cdot x_{red} \geq b, \quad x_{red} \geq 0 \end{aligned} \tag{RP}$$

Since (RP) has a polynomial number of variables and constraints, it can be solved exactly by any LP solver (not necessarily the ellipsoid method). Denote the optimal solution by x^*_{red}.

Let x' be a vector which describes an assignment to the variables of (P), in which all variables that exist in (RP) have the same value as in x^*_{red}, and all other variables are set to 0. It follows that $A \cdot x' = A_{red} \cdot x^*_{red}$, therefore, since x^*_{red} is feasible to (RP)), also x' is a feasible solution to (P). Similarly, $c^T \cdot x' = c^T_{red} \cdot x^*_{red}$. We shall now see that this implies that the objective obtained by x' approximates the objective obtained by x^*:

$$\begin{aligned} & c^T \cdot x' = c^T_{red} \cdot x^*_{red} \\ & = b^T \cdot y^*_{red} && \text{(By strong duality for the reduced LPs)} \\ & \leq b^T \cdot y' && \text{(By Equation (18))} \\ & \leq (A \cdot x^*)^T y' && \text{(By the definition of (P))} \\ & = (x^*)^T (A^T \cdot y') && \text{(By properties of transpose and associativity of multiplication)} \\ & \leq (x^*)^T ((1+\beta) \cdot c) && \text{(By Equation (16))} \\ & = (1+\beta) \cdot (c^T x^*) && \text{(By properties of transpose)} \end{aligned}$$

So, x' (x^*_{red} with all missing variables set to 0) is an approximate solution to the primal LP (P) — as required.

D.2 Using Half-Randomized Approximate Separation Oracles

Here, we allow the oracle to also be *half-randomized*, that is, when it says that a solution is infeasible, it is always correct; however, when it says that a solution is approximately feasible, it is only correct with some probability $p \in [0, 1]$.

Since the ellipsoid method variant is iterative, and since the oracle calls are independent, there is a probability p^T that the oracle answers correctly in each iteration, and so, the overall process performs as before. We shall now explain why, using a half-randomized oracle, this ellipsoid method variant *always* returns a feasible solution to the primal (even if the oracle was incorrect).

First, notice that the oracle is always correct when it determines that a solution is infeasible. In addition, the construction of (RD) is entirely determined by these constraints. Therefore, by the same arguments, x' would still be a feasible solution to P (but not necessarily with an approximately-optimal objective value).

This means that given a half-randomized approximate separation oracle for the dual with error β and success probability p, this ellipsoid method variant can be used as a randomized

approximation algorithm for the primal with the same error and success probability p^I (where I is an upper bound on the number of iteration of the method on the given input).