Optimizing a linear function over the Lorenz-efficient set of multi-objective combinatorial optimization problems

Mohammed Bachir Bederina[a], Djamal Chaabane[a], Thibaut Lust[b]

[a]*University of Sciences and Technology Houari Boumediene, AMCD-RO Laboratory, DGRSDT, Algiers, Algeria*
[b]*Sorbonne Université, CNRS, LIP6, Paris, France*

Abstract

This paper addresses multi-objective combinatorial optimization problems, where all objectives need to be effectively balanced. For this purpose, we use the Lorenz dominance, an enhancement of Pareto dominance introduced in economics to assess income distribution inequalities. Moreover, in addition to the search for a balanced solution, we are looking for a solution optimizing an additional linear function, allowing to model the preferences of the decision-maker and to select the most suitable solution among all Lorenz-efficient solutions. We propose two new exact methods to solve this problem, and apply it to the assignment and knapsack problems. Results show that the new methods are much more efficient than novel methods that we also propose for enumerating the Lorenz-efficient set.

Keywords: Combinatorial optimization, Multi-objective, Fairness, Integer linear programming, Lorenz dominance

1. Introduction

In many decision problems, a solution must be assessed across multiple dimensions. In multi-criteria decision making, these dimensions represent various aspects to consider, with each aspect corresponding to a criterion. In multi-agent decision making, they represent the perspectives of different agents, while in robust decision making, they may reflect various possible scenarios. This paper presents a general framework in which a solution is evaluated based on a component vector, which could consist of a vector of criteria, a vector of scenarios, or a vector of agents' utilities.

Consider, for instance, scenarios involving decisions made by multiple agents, each with a unique utility function over various potential choices. Such situations are common in real-world applications. For example, multi-agent scheduling problems involve each agent completing a set of jobs on shared machines. Each job has a specified processing time, and each agent assigns a utility value to jobs completed on time [17]. Another classic case is the multi-agent assignment problem, where items are distributed among agents. A globally optimal solution, such as one that maximizes the total utility, may not be preferable due to potential unfairness. It is therefore crucial that any solution emerges from a decision-making process involving all agents. This prompts the question: how can a good compromise solution be defined and identified?

To address this question, one can consider integrating an equity criterion into the proposed solutions. Recently, a substantial body of recent research investigates fair outcome definitions, particularly for resource allocation scenarios. These scenarios require decision-makers to distribute resources (divisible or indivisible) among various agents, guided by an appropriate equity principle. Numerous fairness criteria have been investigated in the literature (egalitarian criterion [8], Nash product [11], ordered weighted average [22], proportionality [33], and envy-freeness [37]), for a detailed review, see Amanatidis et al. [2].

In this paper, beyond seeking a fair solution, an additional criterion is taken into account, modelled by a linear function, to decide which of the fair solutions (when the criterion of fairness does not allow a single solution to be determined) is the best. Depending on the context, this linear function may model the preferences of a decision-maker (e.g., the owner of the machines in the case of the multi-agent scheduling problem) or an economic criterion (e.g., the total cost of assigning the objects to the agents in the multi-agent assignment problem). In this case, a fair solution will not necessarily be optimal for the linear function, and an optimal solution for the linear function will not be necessarily fair. The goal is thus to find a solution that achieves a good compromise between fairness and the value obtained for the auxiliary linear function. We deal with this problem as follows: we use a dominance relation that integrate fairness: the Lorenz dominance. Then the problem comes down to finding a Lorenz-efficient solution (i.e., a non-dominated solution according to the Lorenz dominance) with the best value for the additional linear function. This problem has been widely studied in the context of Pareto dominance, i.e., the search for a solution that optimizes a linear function under the constraint that the solution is Pareto-efficient [1, 9, 19, 24, 35]. With the exception of our prior study on multi-agent combinatorial optimization problems [5], the investigation of this problem with respect to Lorenz dominance is, to our knowledge, unprecedented. Some research explores trade-offs between solution fairness and efficiency, see e.g., Bertsimas et al. [7], but the problem addressed here is different: a solution is evaluated by an auxiliary linear function, independent of the objective functions.

A method for solving this problem would be, to first, generate all Lorenz-efficient solutions, and then to select, among all Lorenz-efficient solutions, a solution optimizing the auxiliary linear function. However, the two-step approach has not proved effective with Pareto dominance [1], and it is preferable to develop methods that directly seek to optimize the linear function. Despite this pitfall, since Lorenz dominance is more restrictive than Pareto dominance, there are generally fewer Lorenz-efficient solutions than Pareto-efficient solutions. It is therefore interesting to study two-step methods in the context of Lorenz dominance.

The main contributions are two-fold: first, we propose two new methods to enumerate the Lorenz-efficient solutions of multi-objective combinatorial optimization problems: the first method is based on the method of Sylva and Crema [34] and the second on the method of Lokman and Köksalan [25], both methods being initially developed for Pareto dominance. Secondly, we directly optimize the auxiliary linear function, and we also propose two new methods. The methods are compared on instances of the multi-objective knapsack and multi-agent assignment problems. The results show that the execution times of the first method are quite sensitive to the number of variables, while the computational times of the second method highly depend on the number of objectives.

This paper is structured as follows. Section 2 introduces Lorenz dominance within the context of multi-objective combinatorial optimization problems. In Section 3, we present the problem of generating a Lorenz-efficient set of multi-objective combinatorial optimization problems: two new methods are presented. In 4, we deal with the problem of optimizing a linear function over the Lorenz-efficient set and two new methods are proposed. Finally, in Section 5, we report results for two multi-objective combinatorial optimization problems: the assignment and the knapsack problems.

2. Fairness and Lorenz Dominance

2.1. Multi-objective combinatorial optimization

We define a combinatorial multi-objective optimization problem as follows. Given $\mathcal{X}$, a feasible set, defined by a set of linear constraints on n binary decision variables, a solution $x \in \mathcal{X}$ is evaluated through different linear objective functions, that associate to a solution x a utility vector $y(x) = (f_1(x), f_2(x), \ldots, f_p(x)) \in \mathbb{R}^p$, where $f_i(x)$ corresponds to the objective function i (a number p of objective functions is considered). We aim to maximize the p objective functions:

$$\underset{x \in \mathcal{X}}{\text{maximize}} \big(f_1(x), \ldots, f_p(x)\big)$$

As, in general, there does not exist a solution that is optimal for all objective functions, a dominance relation is needed to compare solutions. Typically, the solutions are compared through the Pareto dominance relation, defined as follows.

Definition 1. *Weak Pareto dominance relation: we say that a point* $u = (u_1, \ldots, u_p) \in \mathbb{R}^p$ weakly Pareto dominates *a point* $v = (v_1, \ldots, v_p) \in \mathbb{R}^p$ *if, and only if,* $u_k \geq v_k, \forall\, k \in \{1, \ldots, p\}$. *We denote this relation by* $u \succeq_P v$.

Definition 2. *Pareto dominance relation: we say that a point* $u = (u_1, \ldots, u_p) \in \mathbb{R}^p$ Pareto dominates *a point* $v = (v_1, \ldots, v_p) \in \mathbb{R}^p$ *if, and only if,* $u_k \geq v_k, \forall\, k \in \{1, \ldots, p\} \wedge \exists\, k \in \{1, \ldots, p\} : u_k > v_k$. *We denote this relation by* $u \succ_P v$.

Definition 3. *Pareto-efficient solution: a feasible solution* $x^* \in \mathcal{X}$ *is called* Pareto-efficient *if there is no other feasible solution* $x \in \mathcal{X}$ *such that* $y(x) \succ_P y(x^*)$.

The set of all Pareto-efficient solutions is called the Pareto-efficient set, denoted by X_E and its representation in objective space is called the Pareto front, denoted by Y_E.

However, using the standard Pareto dominance to discriminate between the solutions proves insufficient, since a Pareto-efficient solution might not be balanced according to all the objective functions, as shown in the following example.

Example 1. *We consider in this example an instance of the multi-objective knapsack problem. The following table displays the utilities (score between 1 and 9) given by three agents (a_1, a_2, a_3) to five objects (o_1, o_2, o_3, o_4, o_5).*

	o_1	o_2	o_3	o_4	o_5
a_1	4	3	9	6	6
a_2	6	7	6	3	7
a_3	3	9	1	9	4

Each object has a weight w_i, given in the next table:

	o_1	o_2	o_3	o_4	o_5
w	7	5	7	3	7

The goal is to find a subset of items that satisfies all the agents and such that the total weight of the items selected is less than or equal to the capacity W of the knapsack, here considered equal to 14. The objective functions correspond to the total utility received by each of the agents. As there are three agents, there are $p = 3$ objective functions to be maximized.
By proceeding by a simple enumeration of all feasible solutions, it is easy to show that this problem presents six Pareto-efficient solutions, given is the following table, where $x^i_j = 1$ means that the object o_j is selected in the solution x^i (and $x^i_j = 0$ means that the object o_j is not selected):

Solution	$y(x)$
$x^1 = (0,1,0,1,0)$	$(9,10,18)$
$x^2 = (0,1,0,0,1)$	$(9,14,13)$
$x^3 = (0,1,1,0,0)$	$(12,13,10)$
$x^4 = (0,0,0,1,1)$	$(12,10,13)$
$x^5 = (0,0,1,0,1)$	$(15,13,5)$
$x^6 = (0,0,1,1,0)$	$(15,9,10)$

However, among these solutions there are some solutions that are not fair: for example, agent a_3 will not be happy with solution x^5 (the total utility received by agent a_3 is equal to 5, while she would get a total utility of 18 with solution x^1), and agent a_2 will not be happy with solution x^6 since she would get two items that she has rated as the worst. We can therefore see that it would be necessary to use a more restrictive relation that incorporates fairness.

2.2. Fairness in multi-objective optimization

Ensuring fairness among multiple objective functions is a challenging issue, as fairness can be interpreted in various ways. The concept of fairness has been widely studied in economics and social choice theory, leading to the development of different definitions. In economics, several measures of inequality in outcome distributions have been proposed, such as the Gini index [13] and the Atkinson index [3]. These indices are designed to quantify the extent of inequality in a distribution. In the field of welfare economics, the notion of fairness is often addressed through social welfare functions, which are used to evaluate the overall goodness of alternatives based on individual utilities [30]. These functions aim to aggregate the preferences of individuals into a collective measure of social well-being, allowing for a comparison of different allocation outcomes.

The traditional utilitarian criterion evaluates a solution based on the sum of values obtained by the different objective functions. However, it is indifferent to how this total utility is distributed among the objectives [31], potentially leading to unfair outcomes. This is because it permits compensation between highly satisfied and poorly satisfied objectives. For instance, a solution with a utility vector of $(20, 1)$ would be deemed superior to one with $(10, 10)$, despite the latter offering a more balanced distribution of satisfaction. To address this issue, the classic egalitarian criterion (or its lexicographical refinement) assesses a solution based on the utility of the least satisfied objective, aligning with the max-min principle. However, this approach considers only the lowest utility value, which can lead to the rejection of high-quality solutions. For example, a solution with a utility vector of $(10, 10, 10)$ would be preferred over one with $(9, 20, 20)$, even if the latter considerably benefits two objectives while only slightly disadvantaging the less satisfied one.

Other aggregation functions, e.g., the Ordered Weighted Average (OWA) [38], can promote solutions where objective values are more balanced. However, they require additional preference information in the form of appropriate weighting parameters.

2.3. Lorenz dominance

In this work, we use Lorenz dominance to compare solutions, as it elegantly refines Pareto dominance to incorporate fairness. Originally introduced in economics to measure income inequality, Lorenz dominance extends Pareto dominance by favoring solutions with better-distributed utilities. Broadly speaking, Lorenz dominance identifies Pareto-efficient solutions that achieve well-balanced compromises among objective functions while preserving high-performance outcomes. It generalizes both the utilitarian and egalitarian criteria and has been applied to characterize equitable solutions in multi-objective optimization [20, 21] and robust decision-making under uncertainty [28]. Additionally, it has been explored within convex-cone theory [39], multi-objective programming [4], and metaheuristics [12, 23].

The Lorenz dominance relies on the construction of particular vectors, called *generalized Lorenz vectors*, obtained as follows.

Definition 4. *The* generalized Lorenz vector *(or simply* Lorenz vector*) of* $y \in \mathbb{R}^p$ *is the vector* $L(y) \in \mathbb{R}^p$ *defined by:* $L(y) = (y_{(1)}, y_{(1)} + y_{(2)}, \ldots, y_{(1)} + y_{(2)} + \ldots + y_{(p)})$*, where* $(y_{(1)}, y_{(2)}, \ldots, y_{(p)})$ *represents the components of* y *sorted from the worst to the best (i.e.,* $y_{(1)} \leq y_{(2)} \leq \ldots \leq y_{(p)}$ *in the case of maximization and* $y_{(1)} \geq y_{(2)} \geq \ldots \geq y_{(p)}$ *in the case of minimization).*

Definition 5. *Lorenz dominance relation: we say that a point* $u = (u_1, \ldots, u_p) \in \mathbb{R}^p$ Lorenz dominates *a point* $v = (v_1, \ldots, v_p) \in \mathbb{R}^p$*, if, and only if,* $L(u) \succ_P L(v)$*. We denote this relation by* $u \succ_L v$*.*

Definition 6. *Lorenz-efficient solution: a feasible solution* $x^* \in \mathcal{X}$ *is called* Lorenz-efficient *if there is no other feasible solution* $x \in \mathcal{X}$ *such that* $y(x) \succ_L y(x^*)$*. The image of a Lorenz-efficient solution in objective space is called a Lorenz-non-dominated point.*

The *Lorenz-efficient set* denoted by X_L contains all Lorenz-efficient solutions. Note that in the Lorenz-efficient set, we can have two different solutions x_1 and x_2 with the same utility vector, i.e., $y(x_1) = y(x_2)$, but generally we are only interested by generating *a minimal complete set* X_{L_m}, i.e., for each Lorenz-non-dominated point, only one corresponding efficient solution is retained.

Example 2. *The following table presents the Lorenz vector for each Pareto-efficient solution of the multi-objective knapsack problem from Example 1. A Pareto dominance comparison of these vectors shows that only four are non-dominated (bolded), corresponding to the Lorenz-efficient solutions: (*x^1*,* x^2 x^3 *and* x^4*). Examining the Lorenz vectors reveals that solution* x^1 *is optimal for the sum of utilities (last component of the Lorenz vector). Solutions* x^3

and x^4 are the best for the max min criterion (given by the first component of the Lorenz vector), while solution x^2 represents a good compromise between these objectives.

Solution	$y(x)$	$L(y(x))$
$x^1 = (0,1,0,1,0)$	$(9,10,18)$	$(\mathbf{9},\mathbf{19},\mathbf{37})$
$x^2 = (0,1,0,0,1)$	$(9,14,13)$	$(\mathbf{9},\mathbf{22},\mathbf{36})$
$x^3 = (0,1,1,0,0)$	$(12,13,10)$	$(\mathbf{10},\mathbf{22},\mathbf{35})$
$x^4 = (0,0,0,1,1)$	$(12,10,13)$	$(\mathbf{10},\mathbf{22},\mathbf{35})$
$x^5 = (0,0,1,0,1)$	$(15,13,5)$	$(5,18,33)$
$x^6 = (0,0,1,1,0)$	$(15,9,10)$	$(9,19,34)$

Through this example, we observe the clear advantage of using Lorenz dominance: it produces solutions that represent compromises between utilitarian and egalitarian criteria.

The Lorenz dominance is closely tied to the principle of *Pigou-Dalton transfers*. In social choice theory, a Pigou-Dalton transfer is a redistribution of income from a wealthier individual to a poorer one, where the transferred amount does not exceed the initial income gap between them.

Definition 7. *Transfer principle [18]: let $y \in \mathbb{R}^p$ such that $y_i > y_j$ for some i, j. Then for all ε such that $0 \leq \varepsilon \leq y_i - y_j$, we have that $y - \varepsilon\, e_i + \varepsilon\, e_j \succsim_L y$ where e_i (resp. e_j) is the vector whose i^{th} (resp. j^{th}) component equals 1, all others being 0.*

This principle implies that for a utility vector $y \in \mathbb{R}^p$ with $y_i > y_j$, slightly increasing y_j while decreasing y_i, all while maintaining the same mean utility, leads to a more balanced distribution and thus a fairer solution. For instance, the vector $y = (20, 20)$ represents a more equitable distribution than $y' = (10, 30)$, as the latter can be transformed into the former by transferring 10 units. This principle allows comparisons between vectors with the same mean. Note that if the transfer size exceeds 20, it would instead increase inequality in the distribution of utilities. This explains why the transfer size must satisfy $\varepsilon \leq y_i - y_j$.

The generalized Lorenz extension considered here allows for the comparison of vectors with different means. This is possible due to its *Pareto-monotonicity* property [32]: if a vector y^1 Pareto dominates another vector y^2, then y^1 Lorenz dominates y^2.

Property 1. *Pareto-monotonicity:* $\forall y^1, y^2 \in \mathbb{R}^p, y^1 \succ_P y^2 \Rightarrow y^1 \succ_L y^2$.

If we look again at Example 1, consider, e.g., solutions x^6 and x^3, whose utility vectors are $(15, 9, 10)$ and $(12, 13, 10)$ respectively. We can now explain why, $(15, 9, 10)$ is Lorenz dominated by $(12, 13, 10)$ (although the Pareto dominance cannot discriminate between these two vectors). We have that $(16, 9, 10) \succ_P (15, 9, 10)$. Therefore, $(16, 9, 10) \succ_L (15, 9, 10)$ by the Pareto-monotinicity principle. Furthermore, we have that $(12, 13, 10) \succ_L (16, 9, 10)$ by the transfer principle (transfer of 4 from the first objective to the second objective). By transitivity, we have that $(12, 13, 10) \succ_L (15, 9, 10)$ and x^6 is thus Lorenz dominated by x^3.

2.4. Optimization of a linear function over the set of Lorenz-efficient solutions

We now present the primary problem studied in this paper. Consider an auxiliary linear function $\varphi : \mathcal{X} \to \mathbb{R}$, used to evaluate a solution x independently of the objective functions. The auxiliary function could for example correspond to the preferences of a decision-maker or to a cost function. The optimization of this function over the set of Lorenz-efficient solutions leads to the following problem:

$$\underset{x \in X_L}{\text{minimize}}\ \varphi(x)$$

i.e., we search among all Lorenz-efficient solutions the solution minimizing $\varphi(x)$ (note that this would not change the problem if we were trying to maximize $\varphi(x)$, since $\varphi(x)$ is independent of the p objective functions).

The challenge with this problem lies in the fact that the set X_L is not characterized, either explicitly by a solutions set or implicitly by a set of constraints. Furthermore, optimizing $\varphi(x)$ across the complete feasible solutions set produces in general a dominated Lorenz solution.

Example 3. *Let us go back to the multi-objective knapsack problem of Example 1. We consider now that selecting an item has a cost. The cost vector is the following:*

	o_1	o_2	o_3	o_4	o_5
c	*1*	*2*	*3*	*4*	*5*

The auxiliary function $\varphi(x)$ corresponds to a cost function, and is equal to $\sum_{i=1}^{5} c_i x_i$ where x_i corresponds to the binary choice of selecting an item or not. The solution minimizing $\varphi(x)$ is obviously $(0, 0, 0, 0, 0)$, but this solution is not interesting since all agents obtain zero total utility, and it is Pareto dominated by other solutions (which implies that this solution is not Lorenz-efficient). Our objective is to locate the best Lorenz-efficient solution for the cost function. For this small instance, we can simply evaluate all Lorenz-efficient solutions identified earlier (see Example 2) using the cost function. The resulting analysis, summarized in the table below, shows x^3 to be the optimal choice.

Solution	$y(x)$	*Total Cost*
$x^1 = (0, 1, 0, 1, 0)$	$(9, 10, 18)$	$\varphi(x^1) = 6$
$x^2 = (0, 1, 0, 0, 1)$	$(9, 14, 13)$	$\varphi(x^2) = 7$
$x^3 = (0, 1, 1, 0, 0)$	$(12, 13, 10)$	$\varphi(x^3) = \mathbf{5}$
$x^4 = (0, 0, 0, 1, 1)$	$(12, 10, 13)$	$\varphi(x^4) = 9$

3. Generating all Lorenz-efficient Solutions

A primary method to solve the problem of optimizing a linear function over the Lorenz-efficient set, is to enumerate the complete set of Lorenz-efficient solutions X_L, then select from X_L a solution that minimizes $\varphi(x)$.

Lorenz-efficient solutions can also be generated using a two-stage procedure: first, enumerate all Pareto-efficient solutions, and second, select the Lorenz-efficient solutions from this subset (as all Lorenz-efficient solutions are necessarily Pareto-efficient). However, the problems under consideration can involve a large number of objective functions (e.g., up to 80, as shown in Section 5). Consequently, the Pareto-efficient set becomes too large for enumeration, as nearly all feasible solutions may be Pareto-efficient in such cases.

3.1. State-of-the-Art

As far as we are aware, few existing works tackle the problem of generating all Lorenz-efficient solutions of multi-objective combinatorial problems. We briefly summarize below the main methods proposed in the literature to address this problem.

Ranking Method. The ranking method was introduced by Perny et al. [28] within a robust optimization framework. This approach operates by computing solutions in non-increasing order of their sum using a k-best algorithm. Notably, the sum of the objective values corresponds to the final component of the Lorenz vector. As a result, a solution that maximizes this sum is weakly Lorenz-efficient (i.e., there exists no solution strictly better for all components of the Lorenz vector). To ensure that the algorithm terminates appropriately, a valid stopping criterion must be established. This criterion is based on the following proposition: a vector $(y_1, \ldots, y_p)$ Lorenz-dominates another vector $(y'_1, \ldots, y'_p)$ if $\sum_{i=1}^{p} y'_i < p \cdot y_{(1)}$ where $y_{(1)} = \min(y_1, \ldots, y_p)$. Consequently, once the ranking algorithm identifies a solution that meets this stopping criterion, it can be terminated, as all Lorenz-efficient solutions will have been generated.

Perny et al. [28] applied their method on the multi-objective shortest path problem, for which an efficient k-best algorithm is known. However, this method can only be effective if the number of Lorenz-efficient solutions is very small. Indeed, by enumerating the k-best solutions for the sum of the objectives, many solutions located very far from the area where Lorenz-non-dominated points are located can be produced and this greatly slows down the method.

A* Based Method. Perny and Spanjaard [27] have proposed a sophistication of A* algorithm specially designed to determine the set of robust paths in a state space graph. Robust paths correspond here to Lorenz-efficient paths. As the Bellman principle does not hold for Lorenz dominance [27, 29], the authors have proposed two pruning rules specific to Lorenz dominance. No numerical results are provided.

Two-Phase Method. Galand and Lust [15] have adapted the classic two-phase method for Pareto optimization [36] in the case of Lorenz dominance. The method is effective, has been tested on shortest path and set covering problems but it only works for $p = 2$ objective functions.

We present in the next two subsections two new methods for generating all Lorenz-non-dominated points (and their corresponding solutions, i.e., a minimal complete set X_{L_m}) of multi-objective combinatorial optimization problems.

3.2. Enumeration method 1

Our proposed method for generating X_{L_m} in multi-objective combinatorial optimization builds upon the approach of Sylva and Crema [34], for Pareto dominance. The core idea of their method is as follows: at each step, a weighted sum of all objective functions is optimized, subject to additional constraints ensuring that the obtained solution is not Pareto dominated by any previously generated solutions, meaning it must improve at least one objective compared to all prior solutions. By repeating this process, all Pareto-non-dominated points can be identified. However, this approach does not directly extend to Lorenz dominance, as optimizing a strictly positive combination of all objective functions does not necessarily yield a Lorenz-efficient solution, unlike Pareto dominance.

Instead, we need to optimize Ordered Weighted Average (OWA) functions, as Ogryczak [26] demonstrated that any solution maximizing an OWA function with strictly decreasing and strictly positive weights is Lorenz-efficient.

The OWA function has been introduced by Yager [38]:

Definition 8. *Given a vector $y \in \mathbb{R}^p$ and a weighting vector $\omega \in [0,1]^p$, the* ordered weighted average (OWA) *of y with respect to ω is defined by:* $f^{owa}(y,\omega) = \sum_{k=1}^{p} \omega_k y_{(k)}$ *where* $y_{(1)} \leq \ldots \leq y_{(p)}$.

As demonstrated in [16, 28], optimizing an OWA function with strictly decreasing and strictly positive weights in a multi-objective combinatorial optimization problem is equivalent to optimizing a strictly positive linear combination of the Lorenz vector. Specifically, we have:

$$x \in \arg\max \ \mathrm{OWA}(f(x),\omega) \Leftrightarrow x \in \arg\max \sum_{k=1}^{p} \lambda_k \, L_k(f(x))$$

with $\omega_i = \sum_{k=i}^{p} \lambda_k, \forall i \in \{1,\ldots,p\}$, and $\lambda_k > 0, \forall k \in \{1,\ldots,p\}$.

However, since Lorenz vectors are rank-dependent, an Integer Linear Program (ILP) solver cannot be directly used to obtain optimal solutions for OWA functions. To address this, the operator must be linearized. We adopt the linearization proposed by Ogryczak [26], which is based on the following linear program that extracts the Lorenz vector from any vector $y \in \mathbb{R}^p$:

$$L_k(y) = \begin{cases} \min & \sum_{i=1}^{p} \alpha_i^k y_i \\ s.t. & \\ & \sum_{i=1}^{p} \alpha_i^k = k \\ & 0 \leq \alpha_i^k \leq 1 \quad i = 1, \dots, p \end{cases}$$

Example 4. *For example, if we have $y = (12, 13, 10)$, and we want to get the second component of the Lorenz vector (i.e., for $k = 2$), we will need to solve the following linear program:*

$$L_2(y) = \begin{cases} \min & 12\alpha_1^2 + 13\alpha_2^2 + 10\alpha_3^2 \\ s.t. & \\ & \alpha_1^2 + \alpha_2^2 + \alpha_3^2 = 2 \\ & 0 \leq \alpha_i^2 \leq 1 \qquad\qquad i = 1, \dots, 3 \end{cases}$$

And it is easy to see that the optimal solution is $\alpha^2 = (1, 0, 1)$ with $L_2(y) = 22$.

Nevertheless, adopting the dual formulation is more practical, considering our requirement to maximize a linear combination of the Lorenz vectors. The corresponding dual form is given below:

$$L_k(y) = \begin{cases} \max & kr_k - \sum_{i=1}^{p} b_i^k \\ s.t. & \\ & r_k - b_i^k \leq y_i \quad i = 1, \dots, p \\ & b_i^k \geq 0 \qquad\quad i = 1, \dots, p \end{cases}$$

Therefore, by solving the following ILP (P_1) with any strictly positive weights ($\lambda_k > 0, \forall k \in \{1, \dots, p\}$), we can obtain a first Lorenz-efficient solution for any multi-objective combinatorial optimization problem (by replacing $\mathcal{X}$ by the set of constraints associated to the combinatorial optimization problem).

$$P_1 \left\{ \begin{array}{lll} \max & \sum_{k=1}^{p} \lambda_k (k r_k - \sum_{i=1}^{p} b_i^k) & \\ s.t. & & \\ & r_k - b_i^k \leq f_i(x) & i,k = 1,\ldots,p \\ & b_i^k \geq 0 & i,k = 1,\ldots,p \\ & x \in \mathcal{X} & \end{array} \right.$$

The remaining Lorenz-efficient solutions can then be obtained by repeatedly solving the same model, incorporating an additional constraint to ensure that each newly generated solution is not Lorenz dominated by any previously found solutions:

$$P_L \left\{ \begin{array}{lll} \max & \sum_{k=1}^{p} \lambda_k (k r_k - \sum_{i=1}^{p} b_i^k) & \\ s.t. & & \\ & r_k - b_i^k \leq f_i(x) & i,k = 1,\ldots,p \\ & k r_k - \sum_{i=1}^{p} b_i^k \geq (L_k^s + 1) z_k^s & k = 1,\ldots,p \\ & & s = 1,\ldots,l \\ & \sum_{k=1}^{p} z_k^s \geq 1 & s = 1,\ldots,l \\ & b_i^k \geq 0 & i,k = 1,\ldots,p \\ & z_k^s \in \{0,1\} & k = 1,\ldots,p \\ & & s = 1,\ldots,l \\ & x \in \mathcal{X} & \end{array} \right.$$

In this ILP, l is the number of Lorenz-efficient solutions generated so far, L_k^s represents the k^{th} component of the Lorenz vector associated with a Lorenz-efficient solution x^s, and z^s is a binary vector of size p, associated to each solution x^s, allowing to impose that the new solution generated is better than all previously found Lorenz-efficient solutions for at least one component of the Lorenz vector. Solving this ILP either yields a new Lorenz-efficient solution or results in infeasibility, indicating that all Lorenz-non-dominated vectors have been generated.

It is important to note that different utility vectors can correspond to the same Lorenz vector (see e.g., x^3 and x^4 in Example 2). Once a Lorenz-efficient solution x' is obtained, we thus apply Algorithm 1 to identify all other solutions that share the same Lorenz vector.

Algorithm 1: Find-Similar method: Search for solutions with same Lorenz vector $L(y(x'))$

```
Data: $x'$, $X_{L_m}$, $\mathcal{X}$, $f(x)$
Result: $X_{L_m}$
$X_s \leftarrow \{x'\}$;
Solve $P_s$ with $L(y(x'))$, $\forall x' \in X_s$ ;
while $P_s$ is feasible do
    $x' \leftarrow$ solution obtained by solving $P_s$;
    $X_s \leftarrow X_s \cup \{x'\}$;
    Solve $P_s$ with $L(y(x'))$, $\forall x' \in X_s$ ;
end
$X_{L_m} \leftarrow X_{L_m} \cup X_s$;
Return $X_{L_m}$ ;
```

Initially, the set X_s contains only the solution x'. The procedure then repeatedly solves the problem P_s, adding to X_s any additional solutions that share the same Lorenz vector as x'. This process continues until P_s becomes infeasible, indicating that all such solutions have been found.

The problem P_s is formulated as follows:

$$P_s \left\{ \begin{array}{lll} \max & \sum_{k=1}^{p} \lambda_k (k r_k - \sum_{i=1}^{p} b_i^k) & \\ s.t. & & \\ & r_k - b_i^k \leq f_i(x) & i,k = 1,\dots,p \\ & k r_k - \sum_{i=1}^{p} b_i^k = L_k & k = 1,\dots,p \\ & f_i(x) \geq (y_i^s + 1) z_i^s & i = 1,\dots,p, \\ & & s = 1,\dots,|X_s| \\ & \sum_{i=1}^{p} z_i^s \geq 1 & s = 1,\dots,|X_s| \\ & b_i^k \geq 0 & i,k = 1,\dots,p \\ & z_i^s \in \{0,1\} & i = 1,\dots,p, \\ & & s = 1,\dots,|X_s| \\ & x \in \mathcal{X} & \end{array} \right.$$

By solving iteratively P_L and the function **Find-Similar**, we can generate all Lorenz-non-dominated points and their corresponding solutions (see Algorithm 2). Note that only a minimal complete set of Lorenz-efficient solutions is produced, i.e., if there exists two Lorenz-efficient solutions with the same evaluation in the objective space, only one will be produced (but all Lorenz-non-dominated points are generated).

Algorithm 2: Generate a minimal complete Lorenz-efficient set X_{L_m} **(Enumeration method 1)**

```
Data: $\mathcal{X}$, $f(x)$
Result: $X_{L_m}$
Solve $P_1$ to obtain the first solution $x^1$;
$X_{L_m} \leftarrow$ Find-Similar($x^1$, $X_{L_m}$, $\mathcal{X}$, $f(x)$);
$it \leftarrow 1$;
Solve $P_L$;
while $P_L$ is feasible do
    $it \leftarrow it + 1$;
    $x^{it} \leftarrow$ solution returned by $P_L$;
    $X_{L_m} \leftarrow$ Find-Similar($x^{it}$, $X_{L_m}$, $\mathcal{X}$, $f(x)$);
    Solve $P_L$;
end
Return $X_{L_m}$
```

Example 5. *Let us consider the multi-objective knapsack problem of Example 1. We will apply Algorithm 2 to generate X_{L_m}. First, the problem P_1 is solved in order to get an initial Lorenz-optimal solution. By using the positive weight vector $\lambda = (0.1, 0.2, 0.7)$, we obtain the Lorenz-optimal solution $x^1 = (0, 1, 0, 1, 0)$ with objective vector $y(x^1) = (9, 10, 18)$ and Lorenz vector $L(y(x^1)) = (9, 19, 37)$. To possibly find other solutions sharing the same Lorenz vector, we apply* ***Find-Similar****, which does not yield any additional solutions, indicating that there exists no other solution with the same Lorenz vector value $(9, 19, 37)$.*

Subsequently, solving problem P_L returns $x^2 = (0, 1, 0, 0, 1)$ with $y(x^2) = (9, 14, 13)$ and $L(y(x^2)) = (9, 22, 36)$. Similarly in this case, ***Find-Similar*** *does not deliver any solution, indicating that no other solution exists with a Lorenz vector equal to $(9, 22, 36)$.*

In the next iteration, P_L produces $x^3 = (0, 1, 1, 0, 0)$ with $y(x^3) = (12, 13, 10)$ and $L(y(x^3)) = (10, 22, 35)$. In this case, ***Find-Similar*** *produces the solution $x^4 = (0, 0, 0, 1, 1)$ with $y(x^4) = (12, 10, 13)$ and $L(y(x^4)) = (10, 22, 35) = L(y(x^3))$. Finally, solving P_L results in infeasibility, confirming that no additional Lorenz-non-dominated points exist. A minimal complete set X_{L_m} of Lorenz-efficient solutions is thus:*

$$X_{L_m} = \left\{ x^1 = (0, 1, 0, 1, 0),\ x^2 = (0, 1, 0, 0, 1),\ x^3 = (0, 1, 1, 0, 0),\ x^4 = (0, 0, 0, 1, 1) \right\}.$$

(which corresponds, for this small example, to the set of all Lorenz-efficient solutions X_L).

3.3. Enumeration method 2

The second method builds upon the method proposed by Lokman and Köksalan [25] in 2013, which was originally designed for enumerating Pareto-efficient solutions in multi-objective combinatorial optimization problems. Unlike traditional methods, such as the one by Sylva and Crema [34], which eliminates previously explored and dominated regions by introducing additional binary variables and constraints (thereby increasing problem complexity), this new approach divides the problem into multiple subproblems of equal size, each defined by distinct combinations of bounds.

The method begins by generating a solution optimizing one of the components of the Lorenz vector. Here we choose to optimize the last component of the Lorenz vector, i.e., $L_p(y(x))$. To ensure that the solution returned is Lorenz-efficient (and not only weakly Lorenz-efficient), the remaining Lorenz components ($L_i(y(x))$, for $i = 1, \dots, p-1$), are multiplied by a small positive constant ε and added in the objective function. This formulation, denoted as P_1', guarantees that the resulting solution is Lorenz-efficient.

$$P_1' \left\{ \begin{array}{lll} \max & pr_p - \sum\limits_{i=1}^{p} b_i^p + \varepsilon[(1r_1 - \sum\limits_{i=1}^{p} b_i^1) + & \\ & \cdots + ((p-1)r_{p-1} - \sum\limits_{i=1}^{p} b_i^{p-1})] & \\ s.t. & & \\ & r_k - b_i^k \leq f_i(x) & i,k = 1,\dots,p \\ & b_i^k \geq 0 & i,k = 1,\dots,p \\ & x \in \mathcal{X} & \end{array} \right.$$

After solving P_1', a Lorenz-efficient solution, called x^1, is obtained. Subsequently, the function **Find-Similar** is solved to possibly generate additional solutions that have the same Lorenz vector as $L(y(x^1))$.

For the remaining iterations, a new problem, called $P_L(u)$, is solved to identify a new Lorenz-efficient solution whose the Lorenz vector differs from those previously generated, where u is a vector of size $(p-1)$ containing the indices of the previously generated solutions, i.e., $u_k \in \{0, \dots, N\}$ $\forall k = 1, \dots, p-1$, with N equal to the number of Lorenz-optimal solutions with different Lorenz vectors generated so far (each solution is indexed by a number, starting from 1; the index 0 corresponds to a special case explained below). Each component u_k allows to determine a lower bound $g_k(u)$ to be applied to the k^{th} Lorenz component $L_k(y(x))$. If $u_k = 0$, no constraint is imposed on the k^{th} Lorenz component, and the corresponding bound constraint is simply $L_k(y(x)) \geq 0$. Otherwise, if $u_k = m > 0$, the new solution is required to be superior to a previously identified Lorenz-optimal solution x^m for the k^{th} Lorenz component, which is enforced by setting the lower bound $g_k(u) = L_k(y(x^m)) + \varepsilon$ (where $\varepsilon > 0$ ensures strict improvement over this solution).

More precisely, the bound $g_k(u)$ associated to the k^{th} Lorenz component $L_k(y(x))$ is defined as follows:

$$g_k(u) = \begin{cases} 0 & u_k = 0 \quad \textbf{(No bound)} \\ L_k(y(x^{u_k})) + \varepsilon & u_k \neq 0 \quad \textbf{(Strict improvement)} \end{cases}$$

To illustrate how it works, consider a four-objective problem with the vector $u = (2, 1, 0)$. This vector can be interpreted as follows:

- For the first Lorenz component, a lower bound is applied using the value of the second Lorenz-optimal solution previously found x^2, incremented by a small positive constant ε to ensure strict improvement: $g_1(u) = L_1(y(x^2)) + \varepsilon$.

- For the second Lorenz component, the bound is based on the first Lorenz-optimal solution found x^1, again incremented by ε: $g_2(u) = L_2(y(x^1)) + \varepsilon$.

- For the third Lorenz component, $u_{p-1} = u_3 = 0$ indicates that no lower bound is imposed, and thus $g_3(u) = 0$.

These bounds guide the search process by excluding previously explored dominance regions while ensuring that new Lorenz-nondominated points are discovered. For further methodological details, see Lokman et al. [24, 25].

Given a vector u, the problem $P_L(u)$ to be solved in the following:

$$
P_L(u)\left\{
\begin{array}{lll}
\max & pr_p - \sum\limits_{i=1}^{p} b_i^p + \varepsilon[(1r_1 - \sum\limits_{i=1}^{p} b_i^1) + & \\
 & \cdots + ((k-1)r_{k-1} - \sum\limits_{i=1}^{p} b_i^{k-1})] & \\
s.t. & & \\
 & r_k - b_i^k \leq f_i(x) & i,k = 1,\ldots,p \\
 & b_i^k \geq 0 & i,k = 1,\ldots,p \\
 & kr_k - \sum\limits_{i=1}^{p} b_i^k \geq g_k(u) & k = 1,\ldots,p-1 \\
 & x \in \mathcal{X} &
\end{array}
\right.
$$

In this problem, the goal is to maximize the last component of the Lorenz vector $(L_p(y(x)))$, with some constraints on the remaining Lorenz vector components, identified by the bounds $g_k(u)$, as described earlier.

At each iteration, a collection $\mathbf{U}$ of vectors u is defined to ensure that the union of the search space of each subproblem $P_L(u)$ covers exactly the search space where new Lorenz-efficient solutions can be found (i.e., not dominated by any previously identified Lorenz-efficient solutions). This mechanism ensures that each subproblem explores a specific region of the search space by applying selective bound constraints, enabling an efficient and non-redundant search.

Note that at iteration it, the algorithm solves a number of subproblems to identify the $(it+1)^{\text{th}}$ Lorenz-efficient solution. For a multi-objective problem with p objectives, the number of subproblems that may be solved in the worst case at iteration it is in the order of $O(it^{p-2})$, due to the combinatorial generation of lower bound vectors for the $(p-1)$ objectives (excluding the one being maximized). Consequently, the total number of subproblems that may be solved to generate N non-dominated Lorenz points is $O(N^{p-1})$ in the worst case. However, in practice, this number is often significantly smaller due to the reuse of previously solved subproblems and early detection of infeasibility. For example, in the case of $p = 3$, the algorithm solves at most $it + 1$ subproblems at iteration it, resulting in a worst-case total of $\sum_{it=0}^{N}(it+1) = \frac{(N+1)(N+2)}{2}$ subproblems, which is $O(N^2)$.

Each subproblem $P_L(u)$ is solved (for each $u \in \mathbf{U}$) and the solutions are saved in a set $X_{\mathbf{U}}$. Then among all solutions in $X_{\mathbf{U}}$ the solution x^* that maximizes the p^{th} Lorenz component value $L_p(y(x))$ is selected. Other solutions, infeasible subproblems and previously explored bound combinations are recorded to prevent redundant computations in subsequent iterations. Then, all solutions sharing the same Lorenz vector values as x^* are generated by solving **Find-Similar**. These steps are repeated iteratively until all subproblems $P_L(u)$ becomes infeasible, indicating that a minimal complete set X_{L_m} has been generated.

Algorithm 3: Generate a minimal complete Lorenz-efficient set X_{L_m} **(Enumeration method 2)**

```
Data: $\mathcal{X}$, $f(x)$
Result: $X_{L_m}$
Solve $P'_1$ to obtain the first solution $x^1$;
$X_{L_m} \leftarrow$ Find-Similar($x^1$, $X_{L_m}$, $\mathcal{X}$, $f(x)$);
$it \leftarrow 1$ ;
$Stop \leftarrow False$ ;
while $Stop = False$ do
    $it \leftarrow it + 1$ ;
    Set a collection $\mathbf{U}$ of vectors $u$;
    $X_{\mathbf{U}} \leftarrow \{\}$;
    foreach $u \in \mathbf{U}$ do
        Solve $P_L(u)$;
        if $P_L(u)$ is feasible then
            $x \leftarrow$ solution returned by $P_L(u)$;
            $X_{\mathbf{U}} \leftarrow X_{\mathbf{U}} \cup \{x\}$
        end
    end
    if $X_{\mathbf{U}} = \emptyset$ then
        $Stop \leftarrow True$;
    end
    else
        Select $x^{it} = \arg\max_{\forall x \in X_{\mathbf{U}}} L_p(y(x))$ ;
        $X_{L_m} \leftarrow$ Find-Similar($x^{it}$, $X_{L_m}$, $\mathcal{X}$, $f(x)$);
    end
end
Return $X_{L_m}$;
```

Example 6. *In this example, we consider the multi-objective knapsack prob-*

lem described in Example 1. After solving the problem P'_1, we obtain the Lorenz-efficient solution $x^1 = (0, 1, 0, 1, 0)$, with a utility vector $y(x^1)$ equal to $(9, 10, 18)$ and a Lorenz vector $L(y(x^1))$ equal to $(9, 19, 37)$. We obtain that no other solution with the same Lorenz vector exists when applying ***Find-Similar***.

In the next iteration, the problem $P_L(u)$ is addressed by generating new combinations of bounds based on previously identified Lorenz-optimal solutions. This results in the creation of two subproblems (the number of subproblems corresponds to the number of Lorenz-efficient solutions found so far plus one).

The first subproblem, $P_L(u^1)$, is defined using the combination vector $u^1 = (0, 1)$. The value 0 indicates that no lower bound is applied to the first component $L_1(y(x))$. The value 1 in the second component of the combination vector indicates that a lower bound is applied to the second component of the Lorenz vector $L_2(y(x))$, based on the value of the second component of the Lorenz vector corresponding to the first solution found (x^1). Specifically, the bound is set to $L_2(y(x^1))+\varepsilon$, where $\varepsilon > 0$ is a small positive constant introduced to ensure strict improvement. We use here $\varepsilon = 1$ as integer values are used. This results in the bound vector $g(u^1) = (0, L_2(y(x^1)) + 1) = (0, 19 + 1) = (0, 20)$.

The second subproblem, $P_L(u^2)$, is defined by the combination vector $u^2 = (1, 0)$, leading to the bound vector $g(u^2) = (L_1(y(x^1)) + 1,\ 0) = (10 + 1,\ 0) = (11, 0)$. Here, the lower bound is applied to the first Lorenz vector component, while no bound is imposed on the second.

Solving these two subproblems yields the solution $x^2 = (0, 1, 0, 0, 1)$ with objective vectors $y(x^2) = (9, 14, 13)$ with $L(y(x^2)) = (9, 22, 36)$, and the solution $x^4 = (0, 0, 0, 1, 1)$ with $y(x^4) = (12, 10, 13)$ and $L(y(x^4)) = (10, 22, 35)$. Among these, we select x^2 because it has the highest value in the third (i.e., p-th) Lorenz component vector ($L_3(x^2) = 36$). The alternative solution x^4 is stored for future iterations to help avoid redundant computations. Solving ***Find-Similar*** *for x^2 gives that no other solution shares the same Lorenz vector.*

In the following iteration, three new bound combinations are considered: $u^1 = (0, 2)$, yielding $g(u^1) = (0, 23)$; $u^2 = (1, 0)$, with $g(u^2) = (10, 0)$; and $u^3 = (2, 0)$, with $g(u^3) = (10, 0)$. The first subproblem, defined by u^1, is explicitly solved and found to be infeasible. For the second and third subproblems, resolution is not necessary: solving these subproblems with the given bounds would lead back to a solution found in a previous iteration. Specifically, using these bounds yields the already identified solution $x^4 =$

$(0,0,0,1,1)$, with $y(x^4) = (12,10,13)$ and the corresponding Lorenz vector $L(y(x^4)) = (10,22,35)$. This occurs because the bound vector $g(u^2) = (10,0)$ satisfies $g(u^2) \leq (10,22)$, which are the first two components of $L(y(x^4))$, which implies that x^4 is a solution to problems $P_L(u^2)$ and $P_L(u^3)$. The solution $x^4 = (0,0,0,1,1)$, is selected and added to the set of Lorenz-optimal solutions, denoted X_{L_m}. Subsequently, applying the ***Find-Similar*** *procedure reveals another solution, $x^3 = (0,1,1,0,0)$, with the same Lorenz vector: $y(x^3) = (12,13,10)$, $L(y(x^3)) = L(y(x^4)) = (10,22,35)$. Both solutions are stored as part of the set of distinct feasible solutions sharing the same Lorenz vector.*

In the final iteration, the generated combinations vectors are $u^1 = (0,2)$, $u^2 = (1,3)$, $u^3 = (2,3)$, and $u^4 = (3,0)$, with the bounds vectors $g(u^1) = (0,23)$, $g(u^2) = (10,23)$, $g(u^3) = (10,23)$, and $g(u^4) = (11,0)$. The subproblem corresponding to $g(u^1) = (0,23)$ is marked as infeasible without solving $P_L(u^1)$, since it was already evaluated in a previous iteration. For $g(u^2) = (10,23)$, infeasibility is deduced without resolution, as $g(u^2) \geq g(u^1)$ and $g(u^1)$ was already found infeasible. Similarly, $g(u^3) = (10,23)$ is identical to $g(u^2)$, and is also considered infeasible without solving the corresponding subproblem. For $g(u^4) = (11,0)$, the subproblem $P_L(u^4)$ is solved and found infeasible. Since all generated subproblems are infeasible, no solution can be found at this iteration. As a result, the algorithm terminates, having successfully enumerated all Lorenz-efficient solutions.

$$X_{L_m} = \left\{x^1 = (0,1,0,1,0),\ x^2 = (0,1,0,0,1),\ x^3 = (0,1,1,0,0),\ x^4 = (0,0,0,1,1)\right\}.$$

4. Optimizing a linear function over the Lorenz-efficient set

We now study the problem of optimizing a linear function over the set of Lorenz-efficient solutions. As the set of Lorenz-efficient solutions can be large (many multi-objective combinatorial optimization problems have been shown to be intractable[1] for Lorenz dominance, see, e.g.,[15, 28, 29]), to efficiently identify an optimal Lorenz-efficient solution, it is more practical to develop methods that only partially enumerate X_L. To achieve this, we have adapted the approaches developed by Jorge [19] and Lokman [24] for Pareto dominance to the context of Lorenz dominance. Our proposed methods rely

[1]A problem is intractable if it has an exponential number of optimal solutions.

on a Lorenz efficiency test and introduces constraints specific to Lorenz dominance to reduce the search space.

4.1. Adaptation of the method of Jorge ***(φ-optimal method 1)***

The process starts by establishing a lower bound v^l, for the auxiliary linear function $\varphi(x)$. This bound is generated by solving the relaxed problem P_r, which involves optimizing the linear function $\varphi(x)$ across the feasible set $\mathcal{X}$.

$$P_r \begin{cases} \min & \varphi(x) \\ s.t. & \\ & x \in \mathcal{X} \end{cases}$$

Solving the relaxed problem P_r yields a solution $x^r \in \mathcal{X}$, with the best possible value for $\varphi(x)$. However, this solution is generally not Lorenz-efficient. To determine if a solution is Lorenz-efficient, we have developed a new Lorenz efficiency test, building on the test for Pareto dominance proposed by Ecker and Kouada [14]. This efficiency test is conducted using the following ILP, referred to as E_L:

$$E_L \begin{cases} \max & \sum\limits_{k=1}^{p} \phi_k \\ s.t. & \\ & kr_k - \sum\limits_{i=1}^{p} b_i^k - \phi_k = L_k \quad & k = 1, \ldots, p \\ & r_k - b_i^k \leq f_i(x) & i, k = 1, \ldots, p \\ & b_i^k \geq 0 & i, k = 1, \ldots, p \\ & \phi_k \geq 0 & k = 1, \ldots, p \\ & x \in \mathcal{X} & \end{cases}$$

where L_k denotes the k-th component of the Lorenz vector corresponding to the solution under evaluation for Lorenz efficiency. The efficiency test proceeds by seeking a solution $x \in \mathcal{X}$ that dominates a given Lorenz vector L. This involves identifying positive values $\phi_k, \forall k \in \{1, \ldots, p\}$, such that the Lorenz vector of x is component-wise greater than or equal to L, with strict inequality holding for at least one component. If such a solution x exists, then the optimal value of E_L, given by $\sum_{k=1}^{p} \phi_k$, will be strictly greater than zero, and the corresponding solution returned by E_L will constitute a new Lorenz-efficient solution. If no such x exists, the solution corresponding to the Lorenz vector L is confirmed to be Lorenz-efficient.

If the solution x^r is found to be non-Lorenz-efficient, we attemp to generate an alternative solution that minimizes the linear function $\varphi(x)$, under the constraint that it is not Lorenz-dominated by any of the previously identified Lorenz-efficient solutions. This task is formulated as the following ILP, denoted P_L^{φ}, which resembles P_L but adopts $\varphi(x)$ as its objective function.

$$P_L^{\varphi}\left\{\begin{array}{lll}
\min & \varphi(x) & \\
s.t. & & \\
& r_k - b_i^k \leq f_i(x) & i,k = 1,\ldots,p \\
& kr_k - \sum_{i=1}^{p} b_i^k \geq (L_k^s + 1) z_k^s & k = 1,\ldots,p; \\
& & s = 1,\ldots,l \\
& \sum_{k=1}^{p} z_k^s \geq 1 & s = 1,\ldots,l \\
& b_i^k \geq 0 & i,k = 1,\ldots,p \\
& z_k^s \in \{0,1\} & k = 1,\ldots,p; \\
& & s = 1,\ldots,l \\
& \varphi(x) < v^u & \\
& \varphi(x) \geq v^l & \\
& x \in \mathcal{X} &
\end{array}\right.$$

where v^u and v^l represent bounds on the linear function $\varphi(x)$, used to narrow down the search space. If problem P_L^{φ} is feasible, it generates a new solution that is not Lorenz-dominated by any of the previously identified Lorenz-efficient solutions. Since P_L^{φ} only considers a subset of X_L, its returned solution is not guaranteed to be Lorenz-efficient. A Lorenz efficiency test is therefore conducted by solving E_L.

Additionally, as mentioned in Section 3.2, different solutions can share the same Lorenz vector value. Therefore, after generating each Lorenz-efficient solution, we check whether there are any solutions with the same Lorenz vector but with a better value for the auxiliary function $\varphi(x)$. This is done by solving P_s^φ, as outlined below:

$$
P_s^\varphi \begin{cases}
\min & \varphi(x) \\
s.t. & \\
 & kr_k - \sum_{i=1}^{p} b_i^k = L_k \quad k = 1, \ldots, p \\
 & r_k - b_i^k \leq f_i(x) \qquad i, k = 1, \ldots, p \\
 & b_i^k \geq 0 \qquad\qquad\quad i, k = 1, \ldots, p \\
 & x \in \mathcal{X}
\end{cases}
$$

In summary, the algorithm, detailed in Algorithm 4, alternates between searching for a solution optimizing $\varphi(x)$ that is not Lorenz dominated by previously generated solutions and performing Lorenz efficiency tests.

Algorithm 4: Generate an optimal Lorenz-efficient solution **(φ-optimal method 1)**

```
Data: $\varphi(x)$, $\mathcal{X}$, $f(x)$
Result: $x^* = \arg\min_{x \in X_L} \varphi(x)$
Solve $P_r$ to get the solution $x^r$;
$v^l \leftarrow \varphi(x^r)$ ;
Test the Lorenz efficiency of $x^r$ by solving $E_L$;
if $x^r$ is Lorenz-efficient then
    Return $x^r$;
end
$x^{0\prime} \leftarrow$ solution returned by $E_L$;
Solve $P_s^{\varphi}$ to find if there is a solution better than $x^{0\prime}$ for $\varphi(x)$;
$x^0 \leftarrow$ solution returned by $P_s^{\varphi}$;
$x^* \leftarrow x^0$, $v^u \leftarrow \varphi(x^0)$;
$it \leftarrow 1$;
while $v^u > v^l$ do
    Solve $P_L^{\varphi}$;
    if $P_L^{\varphi}$ is infeasible then
        Return $x^*$ ;
    end
    else
        $x^t \leftarrow$ solution returned by $P_L^{\varphi}$;
        if $\varphi(x^t) > v^l$ then
            $v^l \leftarrow \varphi(x^t)$ ;
        end
        Test the Lorenz efficiency of $x^t$ by solving $E_L$;
        if $x^t$ is Lorenz-efficient then
            $x^* \leftarrow x^t$;
            Return $x^*$;
        end
        $x^{it\prime} \leftarrow$ solution returned by $E_L$;
        Solve $P_s^{\varphi}$ to find if there is a solution better than $x^{it\prime}$ for $\varphi(x)$;
        $x^{it} \leftarrow$ solution returned by $P_s^{\varphi}$;
        if $\varphi(x^{it}) < v^u$ then
            $v^u \leftarrow \varphi(x^{it})$ ;
            $x^* \leftarrow x^{it}$;
        end
    end
    $it \leftarrow it + 1$;
end
Return $x^*$;
```

Example 7. *Considering the multi-objective knapsack problem from Example 3, we first solve Problem P_r, obtaining the relaxed solution $x^r = (0,0,0,0,0)$ with $y(x^r) = (0,0,0)$. This sets the lower bound $v^l \leftarrow \varphi(x^r) = 0$. The Lorenz efficiency test E_L reveals that this solution is dominated by the Lorenz-optimal solution $x^{0\prime} = (0,0,0,1,1)$, where $y(x^{0\prime}) = (12,10,13)$, $L(y(x^{0\prime})) = (10,22,35)$, and $\varphi(x^{0\prime}) = 9$.*

Next, solving Problem P_s^{φ} yields a different solution $x^0 = (0,1,1,0,0)$ with $y(x^0) = (12,13,10)$. Although x^0 has the same Lorenz vector $L(y(x^0)) = (10,22,35)$ as $x^{0\prime}$, it provides an improved objective value $\varphi(x^0) = 5$. Consequently, we update the best solution as $x^* \leftarrow x^0$ and set the upper bound $v^u \leftarrow 5$.

After that, we solve Problem P_L^{φ} to search for a non-dominated solution with a better $\varphi(x)$ value. The problem proves infeasible, terminating the search with the optimal solution $x^* = (0,1,1,0,0)$ and $\varphi(x^*) = 5$. The method was able to find an optimal solution x^* by generating only two out of the four Lorenz-efficient solutions.

4.2. Adaptation of the method of Lokman ***(φ-optimal method 2)***

In this new approach, we adapt the method developed by Lokman [24], initially developed for Pareto dominance, to Lorenz dominance. As seen in the second enumeration strategy, rather than introducing additional variables and constraints at each iteration, which increases model complexity, this method solves multiple subproblems of the same size defined by bounds on the Lorenz components.

Similar to the previous approach, this method begins by computing a lower bound on $\varphi(x)$. This bound, denoted as v^l, is determined by solving the relaxed problem P_r, where $\varphi(x)$ is optimized over the feasible set $\mathcal{X}$.

The solution x^r returned by P_r is typically not Lorenz-efficient. To assess its optimality, the Lorenz efficiency test E_L is solved. This test either confirms the solution's Lorenz efficiency or identifies a new Lorenz-efficient solution $x^{it\prime}$ that dominates x^r. Subsequently, the problem P_s^{φ} is solved to determine whether another Lorenz-efficient solution x^{it} exists with the same Lorenz vector but with a better value for $\varphi(x)$. The value $\varphi(x^{it})$ is set as an upper bound v^u for the problem.

The search for a new solution continues as long as the lower bound is smaller than the upper bound ($v^l < v^u$) and the problem $P_L^{\varphi}(u)$ remains feasible (i.e., solutions not dominated by previous ones still exist in the search space).

In the next iteration, the problem $P_L^{\varphi}(u)$ is solved.

$$P_L^{\varphi}(u)\begin{cases}
\min & pr_p - \sum_{i=1}^{p} b_i^p + \varepsilon[(1r_1 - \sum_{i=1}^{p} b_i^1) + \\
& \cdots + ((k-1)r_{k-1} - \sum_{i=1}^{p} b_i^{k-1})] \\
s.t. & \\
& r_k - b_i^k \leq f_i(x) \qquad\qquad i,k = 1,\ldots,p \\
& kr_k - \sum_{i=1}^{p} b_i^k \geq g_k(u) \qquad k = 1,\ldots,p-1; \\
& \qquad\qquad\qquad\qquad s = 1,\ldots,l \\
& b_i^k \geq 0 \qquad\qquad i,k = 1,\ldots,p \\
& z_k^s \in \{0,1\} \qquad\qquad k = 1,\ldots,p; \\
& \qquad\qquad\qquad\qquad s = 1,\ldots,l \\
& \varphi(x) < v^u \\
& \varphi(x) \geq v^l \\
& x \in \mathcal{X}
\end{cases}$$

This mathematical model depends on the vector u, where each u defines a vector of bounds $g(u)$ applied to the Lorenz component values $L_k(y(x)),\ k = 1,\ldots p-1$ (as described in Section 3.3).

After solving all the subproblems $P_L^{\varphi}(u)$ for all combinations of bounds $u \in \mathbf{U}$, the solution x^t yielding the best value for $\varphi(x)$ is selected from all solutions returned by the different subproblems. This solution is used to update the lower bound ($v^l \leftarrow \varphi(x^t)$). The Lorenz efficiency test E_L is then performed to verify whether x^t is Lorenz-efficient or to identify a new Lorenz optimal solution $x^{it'}$ that dominates x^t. After finding the solution $x^{it'}$, the problem P_s^{φ} is solved in order to find a solution x^{it} that maximizes the value of $\varphi(x)$ while maintaining the same Lorenz vector ($L(y(x^{it})) = L(y(x^{it\prime})))$. If the solution x^{it} improves the value of $\varphi(x)$, it is set as the new current optimal solution ($x^* \leftarrow x^{it}$), and the upper bound is updated accordingly ($v^u \leftarrow \varphi(x^*)$).

The algorithm continues iterating through the same process until the search space is fully explored. This occurs when the lower bound exceeds the upper bound ($v^l \geq v^u$) or when all subproblems $P_L^{\varphi}(u)$ are infeasible. Upon termination, the algorithm returns a Lorenz-efficient solution x^* optimal for $\varphi(x)$.

Algorithm 5: Generate a Lorenz-efficient solution optimizing $\varphi(x)$ **(φ-optimal method 2)**

```
Data: $\varphi(x)$, $\mathcal{X}$, $f(x)$
Result: $x^* = \arg\min_{x \in X_L} \varphi(x)$
Solve $P_r$ to get the solution $x^r$;
$v^l \leftarrow \varphi(x^r)$ ;
Test the Lorenz efficiency of $x^r$ by solving $E_L$;
if $x^r$ is Lorenz-efficient then
    Return $x^r$;
end
$x^{0\prime} \leftarrow$ solution returned by $E_L$;
Solve $P_s^\varphi$ to find if there is a solution better than $x^{0\prime}$ for $\varphi(x)$;
$x^0 \leftarrow$ solution returned by $P_s^\varphi$;
$x^* \leftarrow x^0$, $v^u \leftarrow \varphi(x^0)$;
$it \leftarrow 1$;
while $v^u > v^l$ do
    Set a collection $\mathbf{U}$ of vectors $u$;
    Solve $P_L^\varphi(u)$, for each combination vector $u \in \mathbf{U}$;
    if $P_L^\varphi(u)$ is infeasible $\forall u \in \mathbf{U}$ then
        Return $x^*$ ;
    end
    else
        $x^t = \arg\min_{\forall u \in \mathbf{U}} \varphi(x^u)$ ;
        if $\varphi(x^t) > v^l$ then
            $v^l \leftarrow \varphi(x^t)$ ;
        end
        Test the Lorenz efficiency of $x^t$ by solving $E_L$;
        if $x^t$ is Lorenz-efficient then
            Return $x^t$;
        end
        $x^{it\prime} \leftarrow$ solution returned by $E_L$;
        Solve $P_s^\varphi$ to check if a solution better than $x^{it\prime}$ exists for $\varphi(x)$;
        $x^{it} \leftarrow$ solution returned by $P_s^\varphi$;
        if $\varphi(x^{it}) < v^u$ then
            $v^u \leftarrow \varphi(x^{it})$ ;
            $x^* \leftarrow x^{it}$;
        end
    end
    $it \leftarrow it + 1$;
end
Return $x^*$;
```

Example 8. *Considering the multi-objective Knapsack problem from Example 3, the relaxed problem P_r is first solved, yielding the solution $x^r = (0,0,0,0,0)$ with objective vector $y(x^r) = (0,0,0)$, Lorenz vector $L(y(x^r)) = (0,0,0)$, and $\varphi(x^r) = 0$. We set the lower bound v^l to 0.*

To test the Lorenz efficiency of this solution, we solve the Lorenz efficiency test E_L. The test shows that x^r is Lorenz dominated by the solution $x^{0\prime} = (0,0,0,1,1)$, with $y(x^{0\prime}) = (12,10,13)$, $L(y(x^{0\prime})) = (10,22,35)$ and $\varphi(x^{0\prime}) = 9$. To check whether a solution with a Lorenz vector similar to $L(y(x^{0\prime}))$ but with an improved $\varphi(x)$ value exists, we solve Problem P_s^{φ}. The resulting solution, $x^0 = (0,1,1,0,0)$, yields the objective vector $y(x^0) = (12,13,10)$ and an objective value of $\varphi(x^0) = 5$. We update the best solution found so far to $x^* \leftarrow x^0$ and set the upper bound to $v^u \leftarrow 5$.

Then, two combinations vectors are generated: $u^1 = (0,1)$ and $u^2 = (1,0)$, with bounds vectors $g(u^1) = (0,23)$ and $g(u^2) = (11,0)$. Each bound vector is used to define lower bounds on the first and second Lorenz component value for the subproblems $P_L^{\varphi}(u^1)$ and $P_L^{\varphi}(u^2)$, respectively. Both subproblems are found to be infeasible, indicating that there is no other Lorenz-efficient solution with a better value for $\varphi(x)$.

The optimal solution found is thus $x^* = (0,1,1,0,0)$ with $\varphi(x^*) = 5$.

5. Numerical Experiments

The algorithms were tested on two multi-objective combinatorial optimization problems: the Multi-Objective Knapsack Problem (MOKP) and the Multi-Agent Assignment Problem (MAAP). The MOKP is formulated as follows:

$$\text{MOKP}\left\{\begin{array}{lll} \max & f_k(x) = \sum_{i=1}^{n} \mu_i^k x_i & k = 1,\ldots,p \\ \text{s.t.} & & \\ & \sum_{i=1}^{n} w_i x_i \leq W & \\ & x_i \in \{0,1\} & i = 1,\ldots,n \end{array}\right.$$

where n is the number of items, p is the number of objectives, and $f_k(x)$ denotes the k^{th} objective function, for $k = 1,\ldots,p$. The utility vector associated with each item i is denoted by $\mu_i = (\mu_i^1,\ldots,\mu_i^p)$, where μ_i^k represents the utility of item i with respect to objective k. The weight of item i is given by w_i, and W denotes the total capacity of the knapsack. The auxiliary cost function is defined as $\varphi(x) = \sum_{i=1}^{n} c_i x_i$, where c_i is the cost associated with item i. The objective is to identify a Lorenz-efficient solution that minimizes the total cost defined by the auxiliary function.

For the MAAP, we have a set of n agents and a set of n items. The goal is to assign exactly one item to one agent. Each agent has a utility for each item. For an agent i, this is given by a vector μ_i of size n. A solution is evaluated by a vector, of size n, giving the utility associated to the item assigned to each agent. We are looking for fair assignments, corresponding in this study to Lorenz-efficient assignments.

The problem is formulated as follows:

$$\text{MAAP}\begin{cases} \max f_i(x) = & \sum_{j=1}^{n} \mu_{ij} x_{ij} & i = 1, \ldots, n \\ \quad \text{s.t.} & & \\ & \sum_{i=1}^{n} x_{ij} = 1 & j = 1, \ldots, n \\ & \sum_{j=1}^{n} x_{ij} = 1 & i = 1, \ldots, n \\ & x_{ij} \in \{0, 1\} & i, j = 1, \ldots, n \end{cases}$$

The auxiliary function $\varphi(x)$ corresponds to the total cost of assignment, given by $\sum_{i=1}^{n} \sum_{j=1}^{n} x_{ij} c_{ij}$, where c_{ij} corresponds to the cost of assigning item j to agent i.

Computational experiments were carried out on a machine equipped with an Intel® Core™ i5-12500H, 2.50GHz processor. The four algorithms were implemented using Microsoft Visual Studio 2022 with the C++ programming language, and linear programs were solved using the IBM CPLEX 22.11 library. The source code used for the numerical experiments is available online.[2]

For the MOKP instances, we considered a number of items n ranging from 5 to 300, and a number of objectives p varying between 3 and 80. The utility matrix and cost vector were generated using two different value ranges: $[1, 20]$ and $[1, 40]$. For the MAAP problem, experiments were conducted with the

[2]https://github.com/BederinaMed/Optimization-Over-the-Lorenz-efficient-set

number of agents n varying from 5 to 40. Similar to the MOKP, utility and cost values were tested using the same two value ranges, $[1, 20]$ and $[1, 40]$.

The evaluation of each algorithm focused on two key metrics: computation time (in seconds) and number of Lorenz-efficient solutions generated. All reported results represent averages over 20 independent instances, with a runtime limit of 7200 seconds imposed per instance. In the tables below, the best results are shown in **bold**, and cases where an algorithm fails to return a solution within the time limit are marked with ”–”.

Range	n	p	**Enumeration method 1**		**Enumeration method 2**	
			Time(s)	# L-eff sol	Time(s)	# L-eff sol
[1,20]	4	3	**0.09**	1.3	0.11	1.3
		10	**0.17**	2.1	0.98	2.1
		20	**0.23**	1.7	29.56	1.7
		80	**6.08**	2.1	–	–
	15	3	**0.22**	2.8	0.23	2.8
		10	**0.95**	5.55	–	–
		20	**3.52**	5.8	–	–
		60	–	–	–	–
	40	3	0.97	8.55	**0.96**	8.55
		5	**20.29**	24.65	–	–
		10	**675.68**	44.3	–	–
		20	–	–	–	–
	80	3	29.94	21.05	**3.04**	21.05
	100	3	277.21	32.65	**4.88**	32.65
	150	3	–	–	**11.25**	47.40
	200	3	–	–	**18.59**	65.55
	300	3	–	–	**110.18**	88.55
[1,40]	5	3	**0.10**	1.4	0.11	1.4
		10	**0.21**	1.8	0.96	1.8
		20	**0.38**	2.4	20.20	2.4
	15	3	**0.29**	3.65	0.32	3.65
		5	**0.40**	4.8	1.68	4.8
		10	**0.92**	5.15	–	–
		20	**2.57**	7.15	–	–
	40	3	**1.40**	12.75	1.65	12.75
		5	**16.93**	21.3	–	–
		10	**383.13**	38.75	–	–
	80	3	15.67	21.2	**4.99**	21.2
	100	3	485.50	38.45	**7.51**	38.45
	150	3	–	–	**19.84**	65.75
	200	3	–	–	**57.57**	97.6
	300	3	–	–	**963.20**	250.3

Table 1: Comparison between the enumeration methods for the MOKP.

Table 1 provides a comparative analysis of Enumeration method 1 (the one based on Sylva and Crema (Algorithm 2)) and Enumeration method 2 (the one based on Lokman and Köksalan (Algorithm 3)) for generating Lorenz-efficient solutions to the MOKP, evaluated across two value ranges: $[1, 20]$ and $[1, 40]$. In both ranges, the methods consistently yield the same number of Lorenz-efficient solutions, confirming their correctness and agreement in enumeration. Note that we have also tested the ranking method proposed by Perny et al. [28] to generate the Lorenz-efficient solutions of these problems, but the method failed to solve most of the considered instances within the runtime limit. This is mainly due to the high number of objective functions considered in the instances.

For the $[1, 20]$ range, in the case of small problem instances, particularly when the number of variables n is low, Enumeration Method 1 generally performs faster, especially when the number of objectives p is high. On the other hand, Enumeration Method 2 yields comparable results when the number of objectives is low; however, as p increases, its performance deteriorates and often fails to complete within the time limit for these instances. In contrast, Enumeration Method 2 maintains tractability for larger instances, especially when $n \geq 80$, and the number of objectives is $p = 3$, while Enumeration Method 1 failed to finish within the time limit.

For the $[1, 40]$ range, which introduces greater variability in utility and weight values, both methods require longer computation times. Similar observations hold as in the previous range: Enumeration Method 2 continues to outperform Enumeration Method 1 on larger instances with $p = 3$, where the latter often becomes intractable. Conversely, Enumeration Method 1 performs better on smaller instances with a low number of variables and a high number of objectives. In general, the number of Lorenz-optimal solutions is higher in this configuration, which contributes to the increased computation times observed for both methods.

Range	n	p	**φ-optimal method 1**		**φ-optimal method 2**		**Enumeration method**	
			Time(s)	# L-eff sol	Time(s)	# L-eff sol	Time(s)	# L-eff sol
[1,20]	5	3	0.12	1.2	0.11	1.2	**0.09**	1.3
		10	0.21	1.25	0.33	1.25	**0.17**	2.1
		20	**0.19**	1.05	0.40	1.05	0.23	1.7
		80	**2.37**	1.25	118.29	1.2	6.08	2.1
	15	3	**0.20**	1.55	0.25	1.6	0.22	2.8
		10	**0.32**	2.05	1.49	2.2	0.95	5.55
		20	**1.08**	2.15	6.29	2.25	3.52	5.8
		60	**6.76**	2.6	–	–	–	–
	40	3	**0.50**	3.2	0.51	2.9	0.96	8.55
		5	**0.79**	4.05	1.85	5.3	20.29	24.65
		10	**3.58**	6.05	–	–	675.68	44.3
		20	**6.87**	2.45	–	–	–	–
	80	3	1.33	5.9	**0.82**	4.85	3.04	21.05
	100	3	2.48	8.25	**1.07**	6.6	4.88	32.65
	150	3	9.16	11.35	**2.69**	11.95	11.25	47.40
	200	3	280.43	15.9	**3.78**	15.95	18.59	65.55
	300	3	–	–	**7.62**	27.25	110.18	88.55
[1,40]	5	3	0.09	1	**0.08**	1	0.10	1.4
		10	**0.31**	1.15	0.33	1.05	0.21	1.8
		20	**0.35**	1.35	1.15	1.45	0.38	2.4
	15	3	**0.28**	1.85	0.29	1.65	0.29	3.65
		5	**0.27**	1.9	0.37	1.7	4.01	4.8
		10	**1.14**	2.15	2.49	2.8	0.92	5.15
		20	**1.23**	2.4	–	–	2.57	7.15
	40	3	**0.55**	3.55	0.62	3.35	1.40	12.75
		5	**0.66**	3.6	1.47	5	20.29	24.65
		10	**3.03**	3.45	857.52	4.3	383.13	38.75
	80	3	1.30	5.7	**1.24**	6.1	4.99	21.2
	100	3	2.47	7.45	**1.07**	6.4	7.51	38.45
	150	3	29.70	13.45	**3.56**	13.2	19.84	65.75
	200	3	–	–	**5.83**	19.4	57.57	97.6
	300	3	–	–	**14.31**	34.75	963.20	250.3

Table 2: Comparison between the φ-optimal methods for the MOKP.

Table 2 presents the performance of the two optimization-based approaches, φ-optimal method 1 and φ-optimal method 2, for identifying cost-optimal Lorenz-efficient solutions for the MOKP. These methods aim to directly explore the Lorenz space to retrieve a single cost-optimal solution without exhaustively enumerating the entire set of Lorenz-efficient solutions. For reference, the final columns of the table include the best results obtained by the enumeration methods, which compute a Lorenz-efficient set before selecting a cost-optimal solution among this set.

For small instances (e.g., $n = 5$ or 15), all methods perform similarly in terms of computation time, specially when the objective number is small. In these cases, φ-optimal method 1 is often slightly faster than φ-optimal method 2. However, as the problem size and number of objectives increase, the cost of full enumeration becomes prohibitive. On several medium and large instances, the enumeration method either requires significantly more time or fail to return results within the time limit, while φ-optimal method 2 remains efficient and robust.

Between the two φ-optimal methods, φ-optimal method 2 consistently outperforms φ-optimal method 1 on larger instances ($n \geq 40$) with a small number of objectives (e.g., 3 objectives), particularly in the $[1, 40]$ value range. It achieves faster computation times than approaches that rely on full enumeration followed by cost-based selection. In contrast, φ-optimal method 1 often identifies fewer Lorenz-efficient solutions and tends to return only a subset of those found through enumeration. Consequently, while φ-optimal method 1 remains effective for smaller problem sizes, φ-optimal method 2 proves to be more scalable and reliable across a broader range of instances. Additionally, we observe that when the value range is larger (e.g., $[1, 40]$), the number of generated solutions tends to increase, leading to longer computation times across all methods.

Range	n	**Enumeration method 1** Time(s)	**Enumeration method 1** # L-eff sol	**Enumeration method 2** Time(s)	**Enumeration method 2** # L-eff sol
[1,20]	5	**0.13**	1.35	0.21	1.35
	10	1.26	1.95	**1.20**	1.95
	15	**0.55**	2.15	2.15	2.15
	20	**3.09**	2.25	6.16	2.25
	25	**13.34**	3.6	73.70	3.6
	40	298.40	14	**272.46**	14
[1,40]	5	**0.18**	1.6	0.26	1.6
	10	**0.41**	1.75	1.23	1.75
	15	**1.13**	2.95	743.81	2.95
	18	**16.69**	4.05	–	–
	22	**201.46**	4.6	–	–

Table 3: Comparison between the enumeration methods for the MAAP.

Table 3 presents the comparative performance of Enumeration method 1 and Enumeration method 2 for the MAAP, across different problem sizes and value ranges. As in the knapsack case, both methods consistently generate the same number of Lorenz-efficient solutions, confirming their correctness. However, their performance diverges notably in terms of computation time as the instance size and utility range increase.

For smaller instances and narrower value ranges (e.g., $[1, 20]$), Enumeration Method 1 is generally faster, often completing the enumeration in under one second. As the problem size increases, particularly for $n = 40$, Enumeration Method 2 slightly outperforms Method 1. However, for the broader value range $[1, 40]$, in several large instances (e.g., $n = 18$ or 22), Enumeration Method 2 fails to return results within the time limit, whereas Enumeration Method 1 remains tractable.

		φ-optimal method 1		**φ-optimal method 2**		**Enumeration method**	
Range	n	Time(s)	# L-eff sol	Time(s)	# L-eff sol	Time(s)	# L-eff sol
[1,20]	5	**0.13**	1.05	0.17	1.1	0.13	1.35
	10	**0.59**	1.45	0.70	1.4	1.20	1.95
	15	0.58	1.2	1.89	1.65	**0.55**	2.15
	20	**1.79**	1.35	3.33	1.4	3.09	2.25
	25	**12.84**	1.8	54.32	1.85	13.34	3.6
	40	152.48	1.65	**92.31**	1.55	272.46	14
[1,40]	5	**0.11**	1.01	0.17	1.15	0.18	1.6
	10	**0.32**	1.45	1.01	1.45	0.41	1.75
	15	**1.08**	1.5	6.29	2.4	1.13	2.95
	18	**8.42**	2.3	38.48	2.4	16.69	4.05
	22	**105.07**	2.75	–	–	201.46	4.6

Table 4: Comparison between the φ-optimal approaches for the MAAP.

Table 4 presents the performance of the two optimization-based methods, φ-optimal method 1 and φ-optimal method 2 for the MAAP. As before, these results are compared to those obtained from full enumeration.

Within the $[1, 20]$ range and for smaller instances ($n = 5$ to 15), φ-optimal method 1 achieves the fastest runtimes while generating only a minimal subset of Lorenz-efficient solutions compared to the Enumeration method. It consistently outperforms both the Enumeration method and φ-optimal method 2 in terms of computation time, particularly when the number of variables is low. As the instance size increases ($n = 20$ and 25), φ-optimal method 1 maintains competitive performance, although φ-optimal method 2 begins to demonstrate greater robustness, with only a modest increase in runtime despite the growing complexity. For $n = 40$, φ-optimal method 2 outperforms both φ-optimal method 1 and the Enumeration method in terms of computation time.

For the broader value range $[1, 40]$, both φ-optimal methods are able to compute solutions within a reasonable time. In this setting, φ-optimal method 1 shows a clear advantage in runtime for several instance sizes.

6. Conclusion and Perspectives

In this paper, we studied the optimization of a linear function over the Lorenz-efficient set of multi-objective combinatorial optimization problems. The goal is to identify a fair solution—defined here as a Lorenz-efficient solution—optimal for an auxiliary linear function $\varphi(x)$, that can model an additional cost function or preferences of a decision maker. Surprisingly, this problem had not been previously addressed. We proposed two novel approaches, both inspired by existing methods developed for Pareto dominance. The first method (called φ-optimal method 1) is based on the addition of disjunctive constraints to progressively reduce the search space. The second method (called φ-optimal method 2) is based on adding bounds to the objective space. The methods were tested on two problems: the multi-objective knapsack problem and the multi-agent assignment problem. The two methods have complementary performances: the computation times of the first method are quite sensitive to the number of variables in the problem, while the second method is quite sensitive to the number of objectives. These two methods have also been adapted to generate the whole set of Lorenz-optimal solutions. For future work, scaling to larger instances could benefit from focusing the search on specific regions, thereby reducing the number of constraints in the integer linear programs [9, 35]. Another promising direction is to involve the decision-maker or the agents directly in the decision-making process and learn their preferences during problem solving, as explored in multi-objective combinatorial optimization [6, 10].

References

[1] Abbas, M., Chaabane, D., 2006. Optimizing a linear function over an integer efficient set. European Journal of Operational Research 174, 1140–1161.

[2] Amanatidis, G., Aziz, H., Birmpas, G., Filos-Ratsikas, A., Li, B., Moulin, H., Voudouris, A.A., Wu, X., 2023. Fair division of indivisible goods: Recent progress and open questions. Artificial Intelligence 322, 103965.

[3] Atkinson, A., 1970. On the measurement of inequality. Journal of Economic Theory 2, 244–263.

[4] Baatar, D., Wiecek, M., 2006. Advancing equitability in multiobjective programming. Computers & Mathematics with Applications 52, 225–234.

[5] Bederina, M., Chaabane, D., Lust, T., 2024. Generating fair solutions of minimal cost, in: ECAI 2024, IOS Press. pp. 4076–4083.

[6] Benabbou, N., Leroy, C., Lust, T., 2020. Regret-based elicitation for solving multi-objective knapsack problems with rank-dependent aggregators, in: ECAI 2020, pp. 419–426.

[7] Bertsimas, D., Farias, V., Trichakis, N., 2012. On the efficiency-fairness trade-off. Management Science 58, 2234–2250.

[8] Bezáková, I., Dani, V., 2005. Allocating indivisible goods. ACM SIGecom Exchanges 5, 11–18.

[9] Boland, N., Charkhgard, H., Savelsbergh, M., 2017. A new method for optimizing a linear function over the efficient set of a multiobjective integer program. European Journal of Operational Research 260, 904–919.

[10] Bourdache, N., Perny, P., 2019. Active preference learning based on generalized gini functions: Application to the multiagent knapsack problem, in: The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI Press. pp. 7741–7748.

[11] Caragiannis, I., Kurokawa, D., Moulin, H., Procaccia, A.D., Shah, N., Wang, J., 2019. The unreasonable fairness of maximum nash welfare. ACM Trans. Econ. Comput. 7.

[12] Chabane, B., Basseur, M., Hao, J.K., 2019. Lorenz dominance based algorithms to solve a practical multiobjective problem. Computers & Operations Research 104, 1–14.

[13] Dodge, Y., 2008. Gini Index. Springer New York. pp. 231–233.

[14] Ecker, J.G., Kouada, I., 1975. Finding efficient points for linear multiple objective programs. Mathematical Programming 8, 375–377.

[15] Galand, L., Lust, T., 2015. Exact methods for computing all Lorenz optimal solutions to biobjective problems, in: Algorithmic Decision Theory, 4th International Conference, ADT, Springer. pp. 305–321.

[16] Galand, L., Spanjaard, O., 2012. Exact algorithms for OWA-optimization in multiobjective spanning tree problems. Computers & Operations Research 39, 1540–1554.

[17] García-León, A.A., Dauzère-Pérès, S., Mati, Y., 2019. An efficient Pareto approach for solving the multi-objective flexible job-shop scheduling problem with regular criteria. Computers & Operations Research 108, 187–200.

[18] Hardy, G., Littlewood, J., Pólya, G., et al., 1952. Inequalities. Cambridge university press.

[19] Jorge, J., 2009. An algorithm for optimizing a linear function over an integer efficient set. European Journal of Operational Research 195, 98–103.

[20] Kostreva, M., Ogryczak, W., 1999. Linear optimization with multiple equitable criteria. RAIRO Operations Research 33, 275–297.

[21] Kostreva, M., Ogryczak, W., Wierzbicki, A., 2004. Equitable aggregations and multiple criteria analysis. European Journal of Operational Research 158, 362–377. Methodological Foundations of Multi-Criteria Decision Making.

[22] Lesca, J., Minoux, M., Perny, P., 2019. The fair OWA one-to-one assignment problem: NP-hardness and polynomial time special cases. Algorithmica 81, 98–123.

[23] Li, X., Chehade, H., Yalaoui, F., Amodeo, L., 2011. Lorenz dominance based metaheuristic to solve a hybrid flowshop scheduling problem with sequence dependent setup times, in: 2011 International Conference on Communications, Computing and Control Applications (CCCA), pp. 1–6.

[24] Lokman, B., 2019. Optimizing a linear function over the nondominated set of multiobjective integer programs. International Transactions in Operational Research 28, 2248–2267.

[25] Lokman, B., Köksalan, M., 2013. Finding all nondominated points of multi-objective integer programs. Journal of Global Optimization 57, 347–365.

[26] Ogryczak, W., 2000. Inequality measures and equitable approaches to location problems. European Journal of Operational Research 122, 374–391.

[27] Perny, P., Spanjaard, O., 2003. An axiomatic approach to robustness in search problems with multiple scenarios, in: Proceedings of the 19th conference on Uncertainty in Artificial Intelligence, UAI, pp. 469–476.

[28] Perny, P., Spanjaard, O., Storme, L.X., 2006. A decision-theoretic approach to robust optimization in multivalued graphs. Annals of Operations Research 147, 317–341.

[29] Perny, P., Weng, P., Goldsmith, J., Hanna, J., 2013. Approximation of Lorenz-optimal solutions in multiobjective markov decision processes, in: Proceedings of the Twenty-Ninth Conference on Uncertainty in Artificial Intelligence, UAI, AUAI Press. p. 508–517.

[30] Sen, A., 1979. Utilitarianism and welfarism. Journal of Philosophy 76, 463–489.

[31] Sen, A., 1997. On economic inequality. Clarendon Press, expanded ed.

[32] Shorrocks, A., 1983. Ranking income distributions. Economica 50, 3–17.

[33] Steinhaus, H., 1949. Sur la division pragmatique. Econometrica 17, 315–319.

[34] Sylva, J., Crema, A., 2004. A method for finding the set of nondominated vectors for multiple objective integer linear programs. European Journal of Operational Research 158, 46–55.

[35] Tamby, S., Vanderpooten, D., 2023. Optimizing over the efficient set of a multi-objective discrete optimization problem, in: 21st International Symposium on Experimental Algorithms, SEA, Schloss Dagstuhl - Leibniz-Zentrum für Informatik. pp. 9:1–9:13.

[36] Ulungu, E., Teghem, J., 1995. The two-phases method: An efficient procedure to solve biobjective combinatorial optimization problems. Foundation of Computing and Decision Science 20, 149–156.

[37] Varian, H.R., 1974. Equity, envy, and efficiency. Journal of Economic Theory 9, 63–91.

[38] Yager, R.R., 1988. On ordered weighted averaging aggregation operators in multicriteria decision making. IEEE Transactions on Systems, Man, and Cybernetics 18, 183–190.

[39] Yu, P., 1974. Cone convexity, cone extreme points and nondominated solutions in decision problems with multiobjectives. Journal of Optimization Theory and Applications 14, 319–377.