Exact Methods for Computing All Lorenz Optimal Solutions to Biobjective Problems

Lucie Galand, Thibaut Lust

Exact Methods for Computing all Lorenz Optimal Solutions to Biobjective Problems

Lucie Galand[1] and Thibaut Lust[2]

[1] PSL, Université Paris-Dauphine LAMSADE
CNRS UMR 7243 F-75775, Paris Cedex 16, France
lucie.galand@dauphine.fr
[2] Sorbonne Universités, UPMC Université Paris 06
CNRS, LIP6, UMR 7606, F-75005, Paris, France
thibaut.lust@lip6.fr

Abstract. This paper deals with biobjective combinatorial optimization problems where both objectives are required to be well-balanced. Lorenz dominance is a refinement of the Pareto dominance that has been proposed in economics to measure the inequalities in income distributions. We consider in this work the problem of computing the Lorenz optimal solutions to combinatorial optimization problems where solutions are evaluated by a two-component vector. This setting can encompass fair optimization or robust optimization. The computation of Lorenz optimal solutions in biobjective combinatorial optimization is however challenging (it has been shown intractable and NP-hard on certain problems). Nevertheless, to our knowledge, very few works address this problem. We propose thus in this work new methods to generate Lorenz optimal solutions. More precisely, we consider the adaptation of the well-known two-phase method proposed in biobjective optimization for computing Pareto optimal solutions to the direct computing of Lorenz optimal solutions. We show that some properties of the Lorenz dominance can provide a more efficient variant of the two-phase method. The results of the new method are compared to state-of-the-art methods on various biobjective combinatorial optimization problems and we show that the new method is more efficient in a majority of cases.

Keywords: Multiobjective Combinatorial Optimization, Fairness, Lorenz dominance, Two-phase method

1 Introduction

In many decision problems, a decision (or a solution) has to be evaluated with respect to several dimensions. In multicriteria decision making the dimensions reflect several aspects to take into account (one aspect per criterion). In multiagent decision making they reflect the point of view of several agents, and they can reflect several scenarios that can occur in robust decision making. We consider in this paper a general framework where a solution is evaluated with respect to a component vector, which could be a vector of criteria, a vector of scenarios

or a vector of agents' utilities. Since there is generally not a solution that optimizes all the components, one has to determine compromise solutions. In this setting, the concept of *Pareto dominance* enables to focus on the subset of solutions to a decision problem for which one cannot make a component better off without worsening another component. However the number of *Pareto-optimal* (P-optimal) solutions to a decision problem can be very large, which could make a final choice of one (or a few) solution(s) among the P-optimal ones difficult for a decision maker. The notion of Lorenz dominance has been proposed in economics to measure the inequalities in income distributions. It refines the Pareto dominance by selecting only the better distributed solutions. Furthermore, it has been used for characterizing equitable solutions in multicriteria optimization [1, 2] and robust solutions in decision under uncertainty [3]. It has also been studied within the framework of convex-cone theory [4] in multiobjective programming [5]. The *Lorenz-optimal* (L-optimal) solutions can be determined with a two-stage procedure that first generates all the P-optimal solutions and second selects only the L-optimal ones among them. But the efficiency of the two-stage procedure depends on the efficiency of the procedure that generates the P-optimal solutions. Besides the number of L-optimal solutions can be very small compared to the number of P-optimal solutions, which would make the two-stage procedure quite inadequate. In the last decade, some procedures have been proposed to deal with the direct determination of the L-optimal solutions in combinatorial optimization (see e.g. the works of Perny *et al.* [3], Baatar and Wiecek [5], Moghaddam *et al.* [6], and Endriss [7]), which is generally a difficult problem (NP-complete and intractable [3, 7]). Nevertheless, the amount of works related to Lorenz optimization is quite small compared to the amount of works related to Pareto optimization in combinatorial optimization. The aim of this work is therefore to study the direct determination of L-optimal solutions in combinatorial optimization. More precisely, we propose in this paper to adapt one of the most famous method proposed in biobjective optimization, namely the *two-phase method* [8], to Lorenz optimization. The two-phase method is a generic approach that enables to determine the P-optimal solutions by computing first the subset of P-optimal solutions that optimize a weighted sum, and second the other P-optimal solutions. It has been widely applied on various problems of biobjective combinatorial optimization (see e.g. the works of Visée *et al.* [9], Ehrgott and Skriver [10], Przybylski *et al.* [11], and Raith and Ehrgott [12]). In this paper, we propose two variants of the adaptation of the two-phase method to Lorenz optimization and we study the efficiency of these procedures on two biobjective combinatorial optimization problems: the biobjective shortest path problem and the biobjective set covering problem.

In Section 2, we formally define the Lorenz dominance and we present the problem of Lorenz optimization in multi-objective combinatorial optimization. Section 3 is devoted to some characterizations of L-optimal solutions. In Section 4, we present a straight adaptation of the two-phase method to Lorenz optimization, and a variant that uses the characterization results of Section 3 to improve its efficiency. We present in Section 5 some numerical experiments,

and discuss the efficiency of the adapted two-phase method compared to some state-of-the-art methods. We conclude in Section 6.

2 Multi-objective Combinatorial Optimization

2.1 Notations and Definitions

A multi-objective combinatorial optimization (MOCO) problem can be formulated as follows:

$$\begin{aligned} \text{“}\min_{x}\text{”} \ & f(x) = Cx = (f_1(x), f_2(x), \ldots, f_p(x))^T \\ \text{s.t. } & Ax \leq b \\ & x \in \{0,1\}^n \end{aligned}$$

where $A \in \mathbb{R}^{m \times n}$, $b \in \mathbb{R}^m$, $C \in \mathbb{R}^{p \times n}$ and the quotation marks means that we want to minimize a vector and not a single scalar value. A feasible solution x is a binary vector of n variables that satisfies the m constraints of the problem. Each solution x is evaluated by p objective functions f_k, $k = 1, \ldots, p$ such that $f_k(x)$ is the value of solution x for objective k. The feasible set in decision space is given by $\mathcal{X} = \{x \in \{0,1\}^n : Ax \leq b\}$. One compares the solutions according to their evaluation in $\mathbb{R}^p$, called the *objective space*. The feasible set in the objective space, that is the evaluation of the feasible set, is given by $\mathcal{Y} = f(\mathcal{X}) = \{f(x) : x \in \mathcal{X}\} \subset \mathbb{R}^p$. An element of the set $\mathcal{Y}$ is called a *cost-vector* or a *point*. W.l.o.g. we consider through the paper that the p objective functions have to be minimized.

Definition 1 *The* Pareto dominance *relation (P-dominance for short) is defined for all* $y^1, y^2 \in \mathbb{R}^p$ *by:* $y^1 \succ_P y^2 \iff [\forall k \in \{1, \ldots, p\}, y^1_k \leq y^2_k$ *and* $y^1 \neq y^2]$

Within a feasible set $\mathcal{X}$, any element x^1 is said to be *P-dominated* when $f(x^2) \succ_P f(x^1)$ for some x^2 in $\mathcal{X}$, and *P-efficient* (or P-optimal) when there is no x^2 in $\mathcal{X}$ such that $f(x^2) \succ_P f(x^1)$. The *P-efficient set* denoted by $\mathcal{X}_P$ contains all the P-efficient solutions. The image $f(x)$ in the objective space of a P-efficient solution x is called a *Pareto-non-dominated* point. The image $\mathcal{Y}_P = f(\mathcal{X}_P)$ of the P-efficient set $\mathcal{X}_P$ in $\mathcal{Y}$, is called the *Pareto front*.

The Lorenz dominance is based on the construction of particular vectors, called *generalized Lorenz vectors*, that are obtained as follows:

Definition 2 *For all* $y \in \mathbb{R}^p$*, the* generalized Lorenz vector *of* y *is the vector* $L(y)$ *defined by:* $L(y) = (y_{(1)}, y_{(1)} + y_{(2)}, \ldots, y_{(1)} + y_{(2)} + \ldots + y_{(p)})$*, where* $y_{(1)} \geq y_{(2)} \geq \ldots \geq y_{(p)}$ *represent the components of* y *sorted in non-increasing order. The* k^{th} *component of* $L(y)$ *is* $L_k(y) = \sum_{i=1}^{k} y_{(i)}$.

Definition 3 *The* Lorenz dominance *relation (L-dominance for short) is defined for all* $y^1, y^2 \in \mathbb{R}^p$ *by:* $y^1 \succ_L y^2 \iff [L(y^1) \succ_P L(y^2)]$

The space in which the generalized Lorenz vectors of a solution x are represented is called the *Lorenz space*. Within a feasible set $\mathcal{X}$, any element x^1 is said to be *L-dominated* when $f(x^2) \succ_L f(x^1)$ for some x^2 in $\mathcal{X}$, and *L-efficient* (or L-optimal) when there is no x^2 in $\mathcal{X}$ such that $f(x^2) \succ_L f(x^1)$. The *L-efficient set* denoted by $\mathcal{X}_L$ contains all the L-efficient solutions. The image $f(x)$ in the objective space or the image $L(f(x))$ in the Lorenz space of a L-efficient solution x is called a *L-non-dominated* point. The image $\mathcal{Y}_L = f(\mathcal{X}_L)$ of the L-efficient set in $\mathcal{Y}$ is called the *Lorenz front*. The generalized Lorenz vectors of the Lorenz front are given by $L(\mathcal{Y}_L)$. The Lorenz dominance is closely related to the *Transfer Principle* [13], which means that for some cost-vector $y \in \mathbb{R}^p$ with $y_i > y_j$, slightly improving y_j to the detriment of y_i while preserving the mean of the costs would produce a better distribution of the costs, and consequently a more balanced solution. This principle enables to compare vectors with the same mean. The generalized Lorenz extension considered here enables to compare vectors with different means thanks to the *P-monotonicity* axiom [14], which means that if a cost-vector y^1 P-dominates another cost-vector y^2 then y^1 L-dominates y^2. Consequently L-optimal solutions are a subset of P-optimal solutions.

Following Definition 3, finding the L-efficient solutions to a MOCO problem boils down to finding the P-efficient solutions to the same MOCO problem where the costs are given by the generalized Lorenz vectors:

$$\text{“}\min_{x \in \mathcal{X}}\text{”} \quad L(f(x))$$

where $L(f(x)) = (f_{(1)}(x), f_{(1)}(x) + f_{(2)}(x), \ldots, f_{(1)}(x) + f_{(2)}(x) + \ldots + f_{(p)}(x))$ with $f_{(1)}(x) \geq f_{(2)}(x) \geq \ldots \geq f_{(p)}(x)$ represent the components of $f(x)$ sorted in non-increasing order. In the special case where $p = 2$, the two objective functions to be minimized are: $L_1(f(x)) = \max(f_1(x), f_2(x))$ and $L_2(f(x)) = f_1(x) + f_2(x)$. We thus look for solutions that establish a good compromise between the value of the worst performance and the sum of the costs.

Example 1 *Let us consider the point $y = (6, 3)$. All the points L-dominated by y are in the hatched area called “Lorenz worse” in the biobjective space of Figure 1 (left part). The points that L-dominate y are in the hatched area called “Lorenz better” in the same figure. To illustrate the L-dominance, the symmetric point of y, the point (3,6), is also represented in the figure by a circle. The points in the not hatched area are incomparable to y with L-dominance. The generalized Lorenz vector of the point $(6, 3)$, that is the point $(6, 9)$, is represented in the Lorenz space (right part of the figure).*

2.2 Algorithmic Issues

Intractability As there is not generally a unique optimal solution when multiple objectives are involved, the number of optimal solutions turns out to be a crucial point to evaluate the hardness of the problem. It leads us to the notion of *intractability* [15]. In our setting, a Lorenz optimization problem is intractable if the number of L-efficient solutions is exponential in the size of the instance. The biobjective shortest path problem, the biobjective spanning tree problem and the biagent Markov decision process problem have been proved intractable

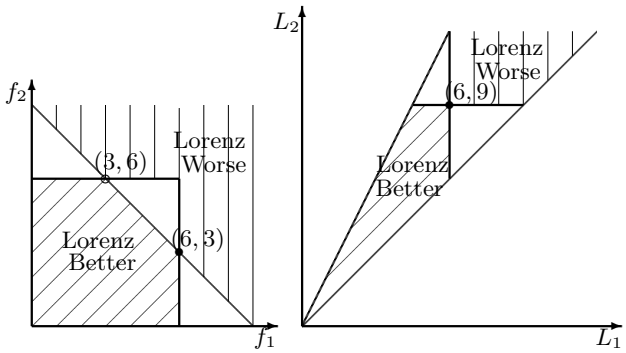

Fig. 1: Representation of the Lorenz dominance.

when looking for L-efficient solutions [3, 16]. As Ehrgott did in a Pareto optimization setting [15], one can show that even the unconstrained problem is intractable in a Lorenz optimization setting. The multi-objective unconstrained (MOUC) problem is defined as follows:

$$\text{“}\min_{x_i \in \{0,1\}^n}\text{”} \sum_{i=1}^{n} c_k^i x_i \qquad k = 1 \dots p$$

Proposition 1 *Problem MOUC is intractable for Lorenz optimization.*

Proof For $p = 2$, by setting $c_1^i = 2^{(i-1)}$ and $c_2^i = -2^i$, we obtain $\mathcal{Y} = \{(0,0), (1,-2), (2,-4), \dots, (2^n - 1, -2^{n+1} + 2)\}$. If we represent the generalized Lorenz vectors of all $y \in \mathcal{Y}$, we obtain: $L(\mathcal{Y}) = \{(0,0), (1,-1), (2,-2), \dots, (2^n - 1, -2^n + 1)\}$. All the generalized Lorenz vectors have the same sum $L_1 + L_2$ and a distinct value on the first dimension L_1 (and consequently on the second dimension L_2 as well). Thus all the generalized Lorenz vectors are P-non-dominated in the Lorenz space and then we have $\mathcal{Y} = \mathcal{Y}_L$. Furthermore, by construction, each feasible solution has a distinct image in the objective space, i.e. $|\mathcal{Y}| = |\mathcal{X}|$. As the number of feasible solution is $|\mathcal{X}| = 2^n$, we have thus $|\mathcal{X}_L| = |\mathcal{Y}_L| = 2^n$. □

NP-completeness The complexity of a Lorenz optimization problem is defined by the complexity of its decision version: given a vector $v = (v_1, \dots, v_p)$, does there exist a solution to the Lorenz optimization problem that L-dominates v? The decision version of the biobjective shortest path problem and the biobjective spanning tree problem has been proved NP-complete [3], and the decision version of the multi-agent allocation problem has been proved NP-complete for many languages [7]. Besides, one can easily show that the decision version of problem MOUC, which is obviously in NP, is also NP-complete.

Proposition 2 *Given a vector $v = (v_1, \dots, v_p)$, deciding whether there exists a solution to problem MOUC that L-dominates v is NP-complete.*

Proof We use a reduction from the partition problem. Consider an instance of this problem with a finite set $A = \{a^1, \dots, a^n\}$ and a size $s(a^i) \in \mathbb{N}$ for each element a^i in A. We construct a biobjective MOUC instance by setting $c_1^i = s(a^i)$

and $c_2^i = -2s(a^i)$, and we ask if there exists a solution that L-dominates the vector $(\sum_i s(a^i)/2 + \varepsilon, -\sum_i s(a^i) + \varepsilon)$ for some small $\varepsilon > 0$. Answering this question amounts to solving the partition problem. □

Other difficulties In addition to the previous complexity results, the determination of Lorenz optimal solutions can encounter another algorithmic issue. It has indeed been shown for some Lorenz optimization problems that one cannot resort directly to an approach based on the Lorenz optimality of partial solutions, like dynamic programming or greedy procedures [16, 17].

2.3 State-of-the-Art

To our knowledge, only a few works address the problem of Lorenz optimization for MOCO problems. We now briefly list the methods proposed in the literature.

Ranking method. The ranking method has been proposed by Perny *et al.* [3] in a robust optimization setting. This method works simply by computing the solutions in nondecreasing order of their sum using a k-best algorithm (that is an algorithm that generates the k best solutions to an optimization problem). This enumeration can be stopped when all the L-efficient solutions have been generated. Note that the sum of the costs of a L-dominated solution can be lower than the sum of the costs of a L-efficient solution, thus the ranking method actually computes a superset of the set $\mathcal{X}_L$. As one cannot know in advance the number k of solutions to be enumerated, one has to define a valid stopping criterion that ensures that all L-efficient solutions have been generated. This method can be used with any number of objectives, but its efficiency strongly relies on the efficiency of the k-best algorithm.

ε-Constraint based method. This method, proposed by Baatar and Wiecek [5], is based on the classic ε-constraint procedure for Pareto optimization [18]. It generates L-efficient solutions in the nondecreasing order of their sum. In addition, it uses Euclidean norm optimization to ensure to determine an L-efficient solution for a given sum of the costs (and not a L-dominated one). Each solution is computed by solving two mathematical programs with appropriate constraints and objective functions. This method works with any number of objectives, but solving such mathematical programs can be inefficient in practice. Besides it generates only one L-efficient solution per Lorenz vector which implies that it computes the set $L(\mathcal{Y}_L)$ but not the Lorenz front $\mathcal{Y}_L$.

Dynamic Programming based method. One cannot use directly a dynamic programming procedure to generate the L-non-dominated points to a MOCO problem (see Section 2.2). However, since dynamic programming can be used with P-dominance, and since L-optimal solutions are also P-optimal, Perny *et al.* have proposed to adapt a multi-objective dynamic programming based procedure to Lorenz optimization by using a valid additional dominance rule [3].

3 Supported Solutions

3.1 Definitions

In multi-objective optimization, there exists an important classification of the P-efficient solutions: *supported* P-efficient solutions (SP solutions) and *non-supported* P-efficient solutions (NSP solutions). The images of the SP solutions in the objective space are located on the convex hull of the Pareto front and the images of the NSP solutions are located inside the convex hull of the Pareto front. More precisely, we can characterize these solutions as follows [15]:

Supported P-efficient solutions: a solution x is supported P-efficient iff there exists a vector λ ($\lambda_k > 0, \forall\, k \in \{1,\dots,p\}$) such that x is an optimal solution to the weighted sum single-objective problem: $\min_{x\in\mathcal{X}} \sum_{k=1}^{p} \lambda_k\, f_k(x)$. The set of SP solutions is denoted by $\mathcal{X}_{SP}$ and the set of supported P-non-dominated points, called SP points, by $\mathcal{Y}_{SP}(= f(\mathcal{X}_{SP}))$. Note that in biobjective optimization, the set $\mathcal{X}_{SP}$ can be easily computed with a *dichotomic search* [19, 20] which gives the different weighting vectors that allow to generate all the SP points.

Non-supported P-efficient solutions: P-efficient solutions that are not supported. The set of NSP solutions is denoted by $\mathcal{X}_{NSP}$ and the set of non-supported P-non-dominated points, called NSP points, by $\mathcal{Y}_{NSP}(= f(\mathcal{X}_{NSP}))$.

One can easily transpose these notions to Lorenz dominance by applying the definitions of SP and NSP solutions in the Lorenz space. In that respect, we define *supported L-efficient solutions* (SL solutions) as follows: a solution x is supported L-efficient iff there exists a vector λ ($\lambda_k > 0, \forall\, k \in \{1,\dots,p\}$) such that x is an optimal solution to the weighted sum single-objective problem defined on the generalized Lorenz vector of $f(x)$: $\min\limits_{x\in\mathcal{X}} \sum_{k=1}^{p} \lambda_k\, L_k(f(x))$, where $f_{(1)}(x) \geq f_{(2)}(x) \geq \ldots \geq f_{(p)}(x)$. Note that $\sum_{k=1}^{p} \lambda_k\, L_k(f(x)) = (\sum_{k=1}^{p} \lambda_k) f_{(1)}(x) + (\lambda_2 + \ldots \lambda_p) f_{(2)}(x) + \ldots + \lambda_p f_{(p)}(x)$. Let w be a weight vector defined by $w_k = \sum_{i=k}^{p} \lambda_i$, then $\sum_{k=1}^{p} \lambda_k\, L_k(f(x)) = w_1 f_{(1)}(x) + w_2 f_{(2)}(x) + \ldots + w_p f_{(p)}(x)$. Such an aggregation function is well-known in fair optimization, it corresponds to a particular family of Ordered Weighted Averages (OWA), where the weights are strictly decreasing. The OWA function has been introduced by Yager [21]:

Definition 4 *Given a vector $y \in \mathbb{R}^p$ and a weighting vector $w \in [0,1]^p$, the* ordered weighted average (OWA) *of y with respect to w is defined by: $f^{owa}(y,w) = \sum_{k=1}^{p} w_k y_{(k)}$ where $y_{(1)} \geq \ldots \geq y_{(p)}$.*

It has been shown that any solution minimizing an OWA function endowed with strictly decreasing and strictly positive weights is L-efficient [22]. One can then define the SL solutions as follows:

Definition 5 *The* SL solutions *are the solutions that minimize an OWA function for some strictly decreasing and strictly positive weighting vector.*

However, in general there exist L-efficient solutions that do not optimize any OWA functions. We call these solutions *non-supported L-efficient* solutions (NSL solutions). The set of SL (resp. NSL) solutions is denoted by $\mathcal{X}_{SL}$ (resp. $\mathcal{X}_{NSL}$)

and the set of supported (resp. non-supported) L-non-dominated points, called SL (resp. NSL) points, by $\mathcal{Y}_{SL}$ (resp. $\mathcal{Y}_{NSL}$). Unfortunately, there are no trivial relations between the sets $\mathcal{X}_{SP}$ and $\mathcal{X}_{SL}$, or between the sets $\mathcal{Y}_{SP}$ and $\mathcal{Y}_{SL}$, as illustrated in the following example.

Example 2 *Let us consider the following 4 solutions:* x^1 *with* $f(x^1) = (6, 18)$, x^2 *with* $f(x^2) = (9, 16)$, x^3 *with* $f(x^3) = (12, 14)$ *and* x^4 *with* $f(x^4) = (20, 2)$. *The generalized Lorenz vectors of the 4 solutions are all P-non-dominated:* $L(f(x^1)) = (18, 24)$, $L(f(x^2)) = (16, 25)$, $L(f(x^3)) = (14, 26)$ *and* $L(f(x^4)) = (20, 22)$. *It means that the 4 solutions are L-efficient. In this simple example, we have a SP solution that is not SL (*x^1*), a solution that is neither SP nor SL (*x^2*), a SL solution that is not SP (*x^3*) and a solution that is both SP and SL (*x^4*).*

3.2 Biobjective case

Even though there is no trivial relation between the supported Pareto and Lorenz efficient sets, we show in this section that, in the case of two objectives, there are interesting properties on the location of the SL solutions with respect to the location of some SP solutions in the objective space. Before presenting the properties, let us introduce some notation for the biobjective case.We denote by $O^i \subset \mathbb{R}^2_{\geq}$ the space in the positive orthant of the objective space where all the points y are such that $y_i \leq y_j$ for $j \neq i$. The *bisector* is the line $y_1 = y_2$ in the objective space. Any point in O^1 (resp. O^2) is said to be *above* (resp. *below*) the bisector. Let us also denote by x^* a L-efficient solution that minimizes the sum of the costs (it is an optimal solution to leximin $\big(f_1(x) + f_2(x), \max(f_1(x), f_2(x))\big)$, where an optimal solution to leximin (y_1, y_2) is an optimal solution w.r.t. y_1 that minimizes y_2 among all the optimal solutions w.r.t. y_1), and y^* its image in the objective space. In this section, we suppose, w.l.o.g., that $y^*_2 > y^*_1$, i.e. $y^* \in O^1$. We will also use the notion of "betweenness" in the sequel: we say that a point y is *between* two points y^i and y^j $(i \neq j)$ in the objective space when $y^i_1 \leq y^j_1 \Rightarrow y^i_1 \leq y_1 \leq y^j_1$. Furthermore, two SP points y^1 and y^2 are said to be *adjacent* when there is no other solution x^3 in $\mathcal{X}_{SP}$ such that $f(x^3)$ is between y^1 and y^2.

Property 1 *All SP solutions* x *such that* $f(x) \in O^1$ *and* $f_2(x) < y^*_2$ *are SL.*

Proof Let x^1 be a solution in $\mathcal{X}_{SP}$ such that $x^1 \in O^1$ and $f_2(x^1) < y^*_2$. As x^1 is in $\mathcal{X}_{SP}$, $\exists \lambda$ s.t. $\lambda_1 f_1(x^1) + \lambda_2 f_2(x^1) \leq \lambda_1 f_1(x) + \lambda_2 f_2(x), \forall x \in \mathcal{X}$. Moreover, as $f_2(x^1) < y^*_2$, $\lambda_2 > \lambda_1$ (otherwise x^* would be better than x^1 for the weighted sum). This implies that for any y in O^1, $\lambda_1 y_1 + \lambda_2 y_2 = f^{owa}(y, \lambda)$, and therefore for any solution $x \in \mathcal{X}$ s.t. $f(x) \in O^1$, $f^{owa}(f(x^1), \lambda) \leq f^{owa}(f(x), \lambda)$ (1). Let us now consider the case where the image of a solution $x \in \mathcal{X}$ is not in O^1, which means that $f_1(x) > f_2(x)$. As $\lambda_2 > \lambda_1$, we have $\lambda_2(f_1(x) - f_2(x)) > \lambda_1(f_1(x) - f_2(x))$, that is $f^{owa}(f(x), \lambda) = \lambda_2 f_1(x) + \lambda_1 f_2(x) > \lambda_1 f_1(x) + \lambda_2 f_2(x)$. As x^1 is in $\mathcal{X}_{SP}$ and in O^1, we have thus $f^{owa}(f(x^1), \lambda) = \lambda_2 f_2(x^1) + \lambda_1 f_1(x^1) < f^{owa}(f(x), \lambda)$ (2). From (1) and (2), we have that solution x^1 is in $\mathcal{X}_{LS}$. □

Note that all solutions x such that $f(x)$ is in O^1 and $f_2(x) > y_2^*$ are L-dominated by y^* since $L(y^*) = (y_2^*, y_1^*+y_2^*) \succ_P (f_2(x), f_1(x)+f_2(x)) = L(f(x))$.

The next property enables us to locate in the objective space the SL points that are NSP according to the bisector. We say that two points are located *on opposite sides of the bisector* if one is above the bisector and the other one is below.

Property 2 *The image of a solution that is SL but not SP is between the two adjacent SP points located on opposite sides of the bisector.*

Proof Let x^1 and x^2 be two P-efficient solutions such that their images $y^1 = f(x^1)$ and $y^2 = f(x^2)$ are two adjacent SP points such that $y_1^1 \leq y_1^2$ (and thus $y_2^1 \geq y_2^2$), and y^1 and y^2 are both either in O^1, either in O^2. Let us assume for example that they are in O^1. Let x^3 be a NSP solution whose image $y^3 = f(x^3)$ is between y^1 and y^2 in the objective space. As points y^1, y^2 and thus y^3 are in O^1, we have $f^{owa}(y^i, \lambda) = \lambda_1 y_2^i + \lambda_2 y_1^i$ for any $i = 1, 2$ or 3. Since x^1 and x^2 are SP and x^3 is NSP, there is always a weight λ such that $f^{owa}(y^1, \lambda)$ or $f^{owa}(y^2, \lambda)$ is less than $f^{owa}(y^3, \lambda)$. Thus solution x^3 cannot be L-efficient. Obviously, we come to the same conclusions when y^1 and y^2 are in O^2. □

Property 3 *Let y^2 be an SP point (image of a solution) such that y^2 is in set O^2. Then all solutions x such that $f_1(x) > y_1^2$ are not L-efficient.*

Proof It is sufficient to show that all solutions $x \in \mathcal{X}_P$ such that $f_1(x) > y_1^2$ are L-dominated by a solution x^2 the image of which is y^2. Let us consider one of these solutions x. As x is P-efficient, $f_2(x) < y_2^2$, which implies that $f(x) \in O^2$, then $L_1(f(x)) = f_1(x) > y_1^2 = L_1(y^2)$. To be L-efficient x must therefore satisfy $L_2(f(x)) < L_2(y^2)$, that is $f_1(x)+f_2(x) > y_1^2+y_2^2$ (1). As solution x^2 is SP, there exists a weighting vector λ such that $\lambda_1 y_1^2 + \lambda_2 y_2^2 \leq \lambda_1 f_1(x) + \lambda_2 f_2(x)$ (2) and $\lambda_1 y_1^2 + \lambda_2 y_2^2 \leq \lambda_1 f_1(x^*) + \lambda_2 f_2(x^*)$ (3). From (1) and (2) and since $f_1(x) > y_1^2$, one obtains $\lambda_1 > \lambda_2$. And from (3) and the fact that solution x^* minimizes the sum of the costs, we obtain $\lambda_2 \geq \lambda_1$, which leads to a contradiction. Therefore, we have $L_2(f(x)) \geq L_2(y^2)$. Thus, solution x^2 L-dominates solution x. □

The image in the objective space of all L-efficient solutions in O^2 are thus between y^* and any SP point in O^2. Let x^{max} be an SP solution such that $f(x^{max}) = y^{max}$ is in O^2 and minimizes the cost of the first criterion among all the P-non-dominated points in O^2. Then, from the previous properties, we can deduce that all L-non-dominated points are between y^* and y^{max}. We show in the next section how we can use these properties to compute all the L-non-dominated points of $\mathcal{Y}_\mathcal{L}$, without generating the entire set $\mathcal{Y}_\mathcal{P}$.

4 New Methods

4.1 Straight Adaptation of the Two-Phase Method

The two-phase method has been developed by Ulungu and Teghem [8] to find the P-efficient solutions to MOCO problems with two objectives. We describe

only at a high level how this method can be adapted in order to generate all the L-non-dominated points of a biobjective optimization problem. The adaptation closely follows the original method. As the name of the method suggests it, the method works in two distinct phases:

Phase 1: generation of all the SL solutions ($\mathcal{X}_{SL}$). This consists in applying the first phase of the original two-phase method for Pareto optimization in the Lorenz space instead of the objective space. This amounts to optimizing OWA functions with different weights until all the SL solutions have been detected. The weight sets w used in the different OWA functions are defined by $w_k = \sum_{i=k}^{p} \lambda_i$ ($k \in \{1, \ldots, p\}$) from the weight sets λ defined in the Lorenz space and computed by the dichotomic search (see Section 3.1).

Phase 2: generation of $\mathcal{X}_{NSL}$. The SL points generated at Phase 1 are used to reduce the search space, since for biobjective problems, NSL solutions are always located, in the Lorenz space, in the interior of the right triangle defined by two adjacent SL points $L(y^1)$ and $L(y^2)$ and the point $(L_1(y^2), L_2(y^1))$ (when $L_1(y^1) < L_1(y^2)$). The exploration of the triangles can be performed with a branch and bound algorithm or with a k-best algorithm.

Even if this straight adaptation of the two-phase method is theoretically interesting, the main drawback is in the first phase: the OWA function that has to be optimized is non-linear and therefore even generating only the SL solutions could be computationally expensive. We propose in the next section a new method where the optimization of OWA functions is avoided.

4.2 Supported Pareto-Efficient Solutions based Method

From Properties 1 and 3, we have that all L-non-dominated points are between the points y^* and y^{max} (see Section 3.2). We use this property to define a new two-phase method based on the computation of some SP solutions. This new method is called *method SP* in the sequel.

Phase 1. The first phase of method SP consists in generating all the SP points located between y^* and y^{max}. From Property 1, we know indeed that all these solutions, except perhaps solution x^{max}, are SL. In order to do so, one first finds a solution x^1 that optimizes L_2. Three cases can then occur:

1. $f_1(x^1) = f_2(x^1)$: in this case, $f(x^1)$ is the only L-non-dominated point of the problem since it optimizes both L_2 and L_1: the method SP stops and returns solution x^1.
2. $f_1(x^1) < f_2(x^1)$ (i.e. $f(x^1) \in O^1$): in this case, we perform a dichotomic search between the points $f(x^1)$ and $f(x^2)$ where x^2 optimizes $\min_{x \in \mathcal{X}}(f_2(x))$. Note that we only need to compute the point y^{max} in O^2, and consequently the search is mainly performed in O^1.
3. $f_1(x^1) > f_2(x^1)$ (i.e. $f(x^1) \in O^2$): analogous to case 2.

For the cases 2 and 3, the SP point y^{max} is also stored. Note that at the end of the first phase, we have generated a subset of $\mathcal{Y}_{SL}$, that is the set of points that are also in $\mathcal{Y}_{SP}$. However, from Properties 2 and 3, we know that

the remaining SL points are between the two adjacent SP points of Property 2 (in fact one is y^{max}), which are now known.

Phase 2. Let Y be the set of the points generated at Phase 1. To each point y of Y is associated an area in the objective space containing all the points that are not L-dominated by y (called the *L-non-dominance zone* of y). The intersection of all the non-dominance zones of the points in Y defines the search zone to be explored at Phase 2. Two illustrations of search zone are given in Figure 2 in the objective space. The black squares represent the SP points generated in Phase

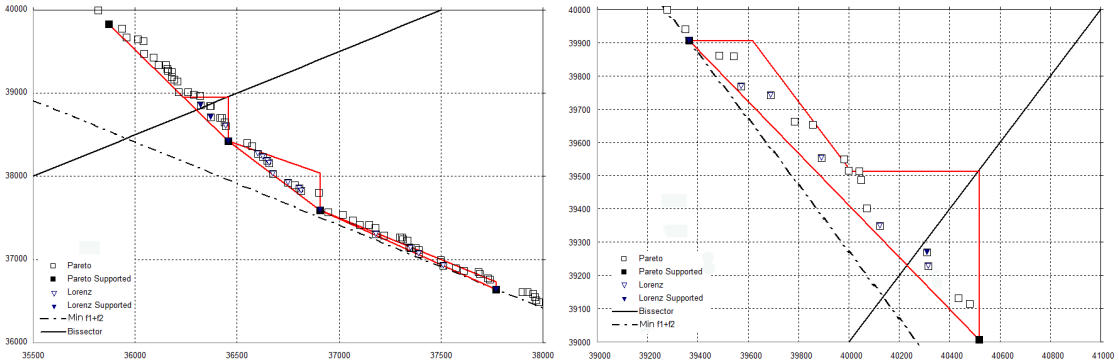

Fig. 2: Search zonee: in the objective space (left) and in the Lorenz space (right).

1. The search zone is represented by the triangles and polygon drawn between two adjacent SP points in the figures. Suppose that the set $Y = \{y^1, y^2, \ldots, y^t\}$ is ordered w.r.t. L_2, that is for any $i = 1, \ldots, t-1$ we have $L_2(y^i) \leq L_2(y^{i+1})$. By Property 1, the $t-1$ points $y^1, \ldots, y^{t-1}$ are all either above the bisector, or below. Suppose, w.l.o.g that they are below the bisector (as in left part of Figure 2). The point y^t is thus the only point in Y that could be above the bisector. Besides, note that y^t is also the only point in Y that may be L-dominated. Since two points y^i and y^{i+1} $(i < t-2)$ in Y are adjacent, any L-non-dominated point between these two points in the objective space relies inside the triangle defined by the three points y^i, y^{i+1} and $p^i = (L_1(y^i), L_2(y^{i+1}) - L_1(y^i))$. Let Z^i denote such a triangle. When the points y^{t-1} and y^t are on opposite sides of the bisector, any L-non-dominated point between these two points in the objective space relies inside a zone that could be a triangle or a particular polygon, as the polygon in Figure 2 (right part). Let Z^{t-1} denote this zone (triangle or polygon). When the points y^{t-1} and y^t are on the same side of the bisector, the zone Z^{t-1} is simply a triangle defined as the other triangles Z^i $(i < t-2)$. The search zone in the objective space to be explored is therefore defined by the $t-2$ triangles Z^i $(i < t-2)$ and the polygon Z^{t-1}.

The exploration of each of the zones Z^i $(i < t)$ consists in enumerating solutions with respect to the weighted sum of their costs with the weighting vector w^i defined such that points y^i and y^{i+1} have the same weighted sum. The enumeration is performed with a k-best algorithm (for an unknown k) and can be stopped as soon as the value of the weighted sum of a point is greater than the

following upper bound: $U^i = \max_{y^j, y^{j+1} \in Y^i} w_1^i p_1^j + w_2^i p_2^j = \max_{y^j, y^{j+1} \in Y^i} (w_2^i - w_1^i) L_1(y^j) + w_1^i L_2(y^{j+1})$, where y^j and y^{j+1} are consecutive in the ordered set Y^i of the already detected points in Z^i ordered w.r.t. L_2. In the case where the bisector crosses the zone Z^{t-1} to be explored, one can easily show that a valid upper bound for Z^{t-1} is: $U^{t-1} = \max \left\{ \max_{y^j, y^{j+1} \in Y_1} (w_2^{t-1} - w_1^{t-1}) L_1(y^j) + w_1^{t-1} L_2(y^{j+1}), L_1(y^k) \right\}$, where y^k is the point that minimizes L_1 in Y^{t-1}. Once all the L-non-dominance zones Z^i have been explored, the method SP can stop, all the L-non-dominated points have been detected.

5 Experimental Results

We have applied the method SP to the biobjective shortest path problem (BOSPP) and to the biobjective set covering problem (BOSCP). The experiments have been run on an Intel Xeon CPU E5-2430 at 2.20GHz for BOSPP, and on an Intel Core i7-3820 CPU at 3.60GHz for BOSCP.

Biobjective Shortest Path Problem. Given a digraph $G = (V, E)$, where V is a set of vertices and $E \subseteq V \times V$ is a set of arcs, and two nodes s and t in V, one looks for the L-non-dominated points, images of feasible shortest paths from s to t in G. The value of each arc e in E is given by a vector of two costs: (c_1^e, c_2^e). It is assumed that all the costs c_i^e are non-negative. In our experiments, we solve BOSPP on *layered digraphs* with randomly generated costs. A layered digraph is a graph in which the set V is partitioned into l subsets, called *layers*, $L_1, L_2, \ldots, L_l$, such that all arcs of E are between consecutive layers. To this *layered* digraph we add a vertex s and $|L_1|$ arcs from s to every vertex of L_1, and a vertex t and $|L_k|$ arcs from every vertex of L_l to t. Therefore, any path from s to t in such a graph is made of $l+1$ arcs exactly. In our instances, all the layers have the same size and there is an arc (v, v') from any vertex v of a layer L_i to any vertex v' of the layer L_{i+1}. For each instance size, 4 different kinds of objectives, A, B, C and UN, are defined. The costs of the instances of type A, B and C are drawn uniformly at random from $[1, 100]$, and the costs of type B are positively correlated and those of type C are negatively correlated (see the instances of type A, B and C proposed by Bazgan *et al.* [23]). In addition, we consider a new type UN where the first cost is randomly generated, and the second cost is also randomly generated, but with a normal distribution. We use the following normal distribution: the mean is 50 and the variance is 20. We compare the running times of the method SP to the running times of two other methods proposed by Perny *et al.* [3] for Lorenz optimization for the multi-objective shortest path problem: the ranking method, called Rkg, and an extension of dynamic programming to Lorenz optimization, called DP (see Section 2.3). For method Rkg and for the Phase 2 of method SP, a k-best algorithm is needed. We have used a modified version of Eppstein's algorithm [24] proposed by Jiménez and Marzal [25]. The results obtained are summarized in Tables 1, 2, 3 and 4. At each row of the tables is given the size of the graph $l \times v$ where l is the number of layers and v the number of vertices per layer, the average number of P-non-dominated points (#P) and L-non-dominated points (#L), and the average CPU running times in seconds of

the three tested methods on 20 instances of the same type and the same size. Note that the number of P-non-dominated points is computed by applying a multi-objective dynamic programming algorithm (see e.g. the algorithms of Stewart and White [26], and Mandow and Pérez-de-la-Cruz [27]). The running time of this algorithm is not indicated in the tables, but it is always greater than the running time of method DP. The symbol '/' in the tables means that the average running time is more than 15 minutes. The results show that the efficiency of

Size			CPU(s)		
$l \times v$	$\#P$	$\#L$	Rkg	SP	DP
20×20	90.8	2.9	0	0.01	0.08
20×50	101.1	2.6	0.01	0.06	0.46
20×100	91.4	2.5	0.05	0.46	1.17
30×20	171.4	3.2	0	0.02	0.29
30×50	161.4	3.1	0.01	0.14	1.23
30×100	147.7	2.3	0.09	0.73	5.12
50×20	349.8	4.7	0	0.04	1.19
50×50	307.5	2.6	0.04	0.44	6.75
50×100	276.7	3.1	0.18	1.45	22.6

Table 1: Type A

Size			CPU(s)		
$l \times v$	$\#P$	$\#L$	Rkg	SP	DP
20×20	11.4	4.9	1.28	0.01	0
20×50	17.1	9.7	/	0.1	0.04
20×100	20	13.2	/	0.73	0.15
30×20	20.9	11.7	/	0.02	0.01
30×50	28.7	17.5	/	0.29	0.13
30×100	30.3	20.7	/	1.63	0.38
50×20	42.1	21.5	/	0.14	0.06
50×50	60.4	36.55	/	1.23	0.34
50×100	55.7	35.7	/	4.65	1.21

Table 2: Type B

method SP compared to the two other methods depends on the number of P-non-dominated points ($\#P$) and on the proportion of L-non-dominated points. For instances A, the costs are rather balanced and $\#L$ is small compared to $\#P$. Method SP is quite efficient on these instances but method Rkg is a bit faster. It means that the ranking method has to enumerate only a small number of feasible solutions. This comes from the fact that the costs are well-balanced. Method DP is clearly less efficient on these instances. The instances B are also balanced, but $\#P$ is significantly smaller than for instances A. This means that a lot of feasible solutions are P-dominated, and as the costs of all the feasible solutions are quite close, the method Rkg is particularly inefficient on these instances, it can only solve very small instances, whereas the method SP is quite efficient. The exploration of several zones instead of one is thus much more appropriate on these instances. Note that the method DP is very efficient when $\#P$ and $\#L$ are small (even if the proportion of L-non-dominated points is high). Instances C

Size			CPU(s)		
$l \times v$	$\#P$	$\#L$	Rkg	SP	DP
20×10	1053.8	3.5	0.01	0.02	2.23
20×20	1806.5	3.75	0.03	0.01	23
20×50	/	3114	120.84	18.65	/
30×10	2005.1	4.6	0.05	0.01	11.52
30×20	3930.5	5	0.2	0.05	165.75
30×30	/	42.6	6.64	0.26	/
50×10	5140.9	3.6	0.24	0.02	215.5
50×20	/	7.15	1.46	0.1	/
50×30	/	5119.7	/	137.16	/

Table 3: Type C

Size			CPU(s)		
$l \times v$	$\#P$	$\#L$	Rkg	SP	DP
20×20	99.8	8.6	0.47	0.02	0.1
20×50	109.3	6.1	0.08	0.09	0.59
20×100	106.8	6.2	0.1	0.53	1.91
30×20	171.1	15.2	110.7	0.04	0.18
30×50	184.9	10.4	8.27	0.19	1.41
30×100	172.1	8.7	2.24	0.94	6.18
50×20	376.4	28	/	0.09	1.43
50×50	362.2	20.2	/	0.83	8.92
50×100	346.2	13.2	108.1	2.23	28.6

Table 4: Type UN

and UN are unbalanced, and we can observe in Tables 3 and 4 that the method

SP is clearly the most efficient on these instances. It comes from the fact that this method explores the objective space by taking into account the form of the convex hull of the images of the solutions, contrary to method Rkg. When the convex hull of the images of the solutions is not symmetric with respect to the bisector, the method SP reveals thus to be much more suitable. Besides, one can observe that the method DP is penalized by an important number of P-non-dominated points. It is particularly inefficient on instances C for which the number of P-non-dominated points is large.

Biobjective Set Covering Problem. We have a set of m rows (or items), and each row can be covered by a subset of n columns (or sets), each column j has two costs c_l^j $(l = 1, 2)$. A feasible solution to the BOSCP is a subset of columns, among the n columns $(j = 1, \ldots, n)$ such that all the rows are covered by at least one column. Our aim is to find the L-non-dominated points. We have used all the instances generated by Gandibleux *et al.* [28], from the size 100×10 (100 columns, 10 rows) to the size 1000×200 (1000 columns, 200 rows). For each size instance, two different kinds of objectives A, B are defined. In the case of instances of type A, the costs of each objective are randomly generated with a uniform distribution. For the type B, the costs of the first objective is also randomly generated with a uniform distribution and the ones of the second objective are made dependent in the following way: $c_2^j = c_1^{n-j+1}, \forall j = 1, \ldots, n$. As done with BOSPP, we also consider the new type UN with the following normal distribution: the mean is equal to the mean value of the first cost and the variance is equal to half the mean. Two types of instance are considered: type UN-A (first cost corresponds to the first cost of type A instance) and type UN-B (first cost corresponds to the first cost of type B instance). We compare the running times of the method SP with the running times of the ranking method Rkg. We also give, for the type A and type B instances, the number of P-non-dominated points ($\#P$) and the CPU time needed for generating these points. The results are from the method of Florios and Mavrotas [29], based on the ε-constraint method (on an Intel core 2 quad CPU at 3.00 GHz). In both methods, a k-best algorithm is necessary to enumerate the k-best solutions. Contrary to the shortest path problem, to our knowledge, no k-best algorithm has been previously developed. We have thus used the commercial CPLEX solver and implemented a procedure based on incumbent callback with solution injection (we inject in the search tree the current best solutions generated to get the next best solution), in order to enumerate the k-best solutions. The results are given in Tables 5, 6, 7 and 8. For each type of instance, we report the name of the instance (same name as used by Gandibleux *et al.*), $\#P$ (when available), $\#L$, and the CPU times in seconds of the tested methods. From Tables 5 and 6, we see that the $\#L$ represent only a small part of $\#P$: there are only between 1 and 6 L-non-dominated points. The running time of both methods are small and quite lower than the running time of the Pareto generation. For these instances, it is thus particularly interesting to apply a method dedicated to the generation of L-non-dominated points. However, the running time of the method Rkg is almost always slightly lower than the new method SP.

			CPU(s)		
#	#P	#L	Rkg	SP	Pareto
11	39	1	**0.019**	0.15	8.64
41	107	1	**0.026**	0.074	18.01
42	208	4	**0.072**	0.085	35.83
43	46	3	**0.15**	0.30	12.80
61	257	6	**0.76**	0.81	83.66
62	98	2	**0.27**	0.99	58.10
81	424	4	**0.20**	0.57	148.66
82	132	3	**0.33**	0.83	116.51
101	157	1	**0.59**	1.25	375.84
102	83	1	**0.37**	1.04	104.76
201	274	2	**9.20**	27.15	6850

Table 5: Type A

			CPU(s)		
#	#P	#L	Rkg	SP	Pareto
11	43	3	**0.017**	0.047	6.26
41	108	2	**0.049**	0.051	16.63
42	276	2	0.24	**0.15**	52.04
43	28	1	**0.023**	0.18	7.51
61	338	2	**0.17**	0.38	114.01
62	99	1	**0.082**	0.58	60.20
81	354	4	**0.25**	2.09	130.26
82	88	2	**0.06**	0.26	38.16
101	141	5	**1.52**	2.20	225.50
102	86	1	**0.29**	1.62	211.48
201	282	6	**19.61**	22.07	4278

Table 6: Type B

		CPU(s)	
#	#L	Rkg	SP
11	2	**0.016**	0.15
41	3	**0.062**	0.13
42	5	0.26	**0.23**
43	3	**0.17**	0.71
61	5	2.39	**0.40**
62	3	1.24	**0.83**
81	1	**0.067**	0.27
82	3	3.95	**1.30**
101	3	7.70	**6.35**
102	4	2.06	3.53
201	5	209.07	**9.40**

Table 7: Type UN-A

		CPU(s)	
#	#L	Rkg	SP
11	1	**0.014**	0.036
41	3	**0.067**	0.18
42	3	**0.059**	0.11
43	6	1.20	**0.63**
61	4	**0.34**	0.70
62	3	**0.73**	0.84
81	4	20.71	**0.41**
82	3	**1.40**	2.41
101	2	**0.61**	1.79
102	2	**0.87**	2.59
201	10	1085.22	**87.7**

Table 8: Type UN-B

From Tables 7 and 8, the running times of both methods Rkg and SP are comparable for most of the instances, except for the last instance 201 (with 1000 columns and 200 rows). For the type UN-A (resp. UN-B), Rkg needs 209.07s (resp. 1085.22s) while SP only needs 9.40s (resp. 87.7s). We see thus that, as soon as the two objectives are unbalanced, the CPU time needed by method Rkg can be very high compared to method SP.

6 Conclusion

We have proposed new properties and new generic methods to generate Lorenz-optimal solutions to biobjective combinatorial optimization problems. The method has been evaluated experimentally on the biobjective shortest path problem and the biobjective set covering problem and showed good results compared to state-of-the-art methods, especially for "unbalanced instances". This work is dedicated to the efficient adaptation of the classic two-phase method to Lorenz optimization. Future work could be to efficiently adapt other classic methods proposed for Pareto optimization to Lorenz optimization. Besides, studying the location of the optimal points in the objective space is also a good starting point for developing efficient methods to generate the L-efficient solutions to MOCO problems, where the number of objectives is not limited to 2.

References

1. Kostreva, M.M., Ogryczak, W.: Linear optimization with multiple equitable criteria. RAIRO - Operations Research **33** (7 1999) 275–297
2. Kostreva, M., Ogryczak, W., Wierzbicki, A.: Equitable aggregations and multiple criteria analysis. European Journal of Operational Research **158**(2) (October 2004) 362–377
3. Perny, P., Spanjaard, O., Storme, L.X.: A decision-theoretic approach to robust optimization in multivalued graphs. Annals OR **147**(1) (2006) 317–341
4. Yu, P.: Cone convexity, cone extreme points, and nondominated solutions in decision problems with multiobjectives. Journal of Optimization Theory and Applications **14**(3) (1974) 319–377
5. Baatar, D., Wiecek, M.: Advancing equitability in multiobjective programming. Computational & Applied Mathematics **52**(1-2) (2006) 225–234
6. Moghaddam, A., Yalaoui, F., Amodeo, L.: Lorenz versus Pareto dominance in a single machine scheduling problem with rejection. In Takahashi, R., Deb, K., Wanner, E., Greco, S., eds.: Evolutionary Multi-Criterion Optimization. Volume 6576 of Lecture Notes in Computer Science. Springer Berlin Heidelberg (2011) 520–534
7. Endriss, U.: Reduction of economic inequality in combinatorial domains. In: AAMAS. (2013) 175–182
8. Ulungu, E., Teghem, J.: The two-phases method: An efficient procedure to solve biobjective combinatorial optimization problems. Foundation of Computing and Decision Science **20** (1995) 149–156
9. Visée, M., Teghem, J., Pirlot, M., Ulungu, E.: Two-phases method and branch and bound procedures to solve the bi-objective knapsack problem. Journal of Global Optimization **12** (1998) 139–155
10. Ehrgott, M., Skriver, A.: Solving biobjective combinatorial max-ordering problems by ranking methods and a two-phases approach. European Journal of Operational Research **147**(3) (2003) 657–664
11. Przybylski, A., Gandibleux, X., Ehrgott, M.: Two-phase algorithms for the biobjective assignement problem. European Journal of Operational Research **185**(2) (2008) 509–533
12. Raith, A., Ehrgott, M.: A two-phase algorithm for the biobjective integer minimum cost flow problem. Computers & Operations Research **36**(6) (2009) 1945–1954
13. Hardy, G., Littlewood, J., Pólya, G.: Inequalities. Cambridge Mathematical Library. Cambridge University Press (1952)
14. Shorrocks, A.F.: Ranking income distributions. Economica **50**(197) (1983) 3–17
15. Ehrgott, M.: Multicriteria Optimization. Second edition. Springer, Berlin (2005)
16. Perny, P., Weng, P., Goldsmith, J., Hanna, J.: Approximation of Lorenz-Optimal Solutions in Multiobjective Markov Decision Processes. In: Conference on Uncertainty in Artificial Intelligence. (2013)
17. Perny, P., Spanjaard, O.: An Axiomatic Approach to Robustness in Search Problems with Multiple Scenarios. In: Proceedings of the 19th conference on Uncertainty in Artificial Intelligence. (2003) 469–476
18. Laumanns, M., Thiele, L., Zitzler, E.: An adaptive scheme to generate the Pareto front based on the epsilon-constraint method. In Branke, J., Deb, K., Miettinen, K., Steuer, R., eds.: Practical Approaches to Multi-Objective Optimization. Number 04461 in Dagstuhl Seminar Proceedings (2005)

19. Cohon, J.: Multiobjective Programming and Planning. Academic Press, New York (1978)
20. Aneja, Y., Nair, K.: Bicriteria transportation problem. Management Science **25** (1979) 73–78
21. Yager, R.: On ordered weighted averaging aggregation operators in multicriteria decision making. In: IEEE Trans. Systems, Man and Cybern. Volume 18. (1998) 183–190
22. Ogryczak, W.: Inequality measures and equitable approaches to location problems. European Journal of Operational Research **122**(2) (2000) 374 – 391
23. Bazgan, C., Hugot, H., Vanderpooten, D.: Solving efficiently the 0-1 multi-objective knapsack problem. Computers & Operations Research **36**(1) (2009) 260–279
24. Eppstein, D.: Finding the k shortest paths. SIAM J. Computing **28**(2) (1998) 652–673
25. Jiménez, V.M., Marzal, A.: A lazy version of Eppstein's k shortest paths algorithm. In: Proceedings of the 2Nd International Conference on Experimental and Efficient Algorithms. WEA'03, Berlin, Heidelberg, Springer-Verlag (2003) 179–191
26. Stewart, B.S., White, III, C.C.: Multiobjective A*. J. ACM **38**(4) (October 1991) 775–814
27. Mandow, L., Pérez-De-la Cruz, J.L.: A new approach to multiobjective A* search. In: Proceedings of the 19th International Joint Conference on Artificial Intelligence. IJCAI'05, San Francisco, CA, USA, Morgan Kaufmann Publishers Inc. (2005) 218–223
28. Gandibleux, X., Vancoppenolle, D., Tuyttens, D.: A first making use of GRASP for solving MOCO problems. In: 14th International Conference in Multiple Criteria Decision-Making, Charlottesville (1998)
29. Florios, K., Mavrotas, G.: Generation of the exact Pareto set in multi-objective traveling salesman and set covering problems. Applied Mathematics and Computation **237** (2014) 1–19