Exploring New Horizons in Multi-Objective Combinatorial Optimization: From Pareto Generation to Interactive Methods

Thibaut Lust

Sorbonne Université
Faculté des Sciences et Ingénierie

Thesis entitled

Exploring New Horizons in Multi-Objective Combinatorial Optimization: From Pareto Generation to Interactive Methods

Presented by Thibaut Lust

for the diploma

Habilitation à Diriger des Recherches

Clarisse Dhaenens	Professor, Université de Lille	Reviewer
Jin-Kao Hao	Professor, Université d'Angers	Reviewer
Vincent Mousseau	Professor, Université Paris-Saclay	Examinator
Luís Paquete	Associate Professor, University of Coimbra	Examinator
Patrice Perny	Professor, Sorbonne Université	Examinator
Daniel Vanderpooten	Professor, Université Paris Dauphine-PSL	Reviewer

Defense date: 21 March 2024

Remerciements

J'ai le grand plaisir de pouvoir remercier les personnes qui ont rendu possible la finalisation de cette thèse d'Habilitation à Diriger des Recherches[1].

Je remercie tous les membres du jury : Clarisse Dhaenens, Jin-Kao Hao et Daniel Vanderpooten, en tant que rapporteurs, Vincent Mousseau et Luís Paquete, en tant qu'examinateurs. Merci beaucoup pour votre tavail d'évaluation et d'appréciation, travail parfois ingrat, mais tellement important pour le développement de la recherche scientifique. Je remercie plus particulièrement Patrice Perny, examinateur, mais aussi responsable de l'équipe Décision à laquelle j'appartiens depuis mon recrutement en 2012 à Sorbonne Université. Son exigence, sa culture scientifique, son enthousiasme et ses qualités pédagogiques m'ont grandement fait évoluer au sein de l'équipe, ce qui m'a permis, de venir -enfin !- à bout ce manuscrit.

Je remercie également le Professeur Jacques Teghem, mon directeur de thèse, de m'avoir initié à la recherche et à l'optimisation multi-objectifs.

Je remercie l'ensemble de mes collaboratrices et collaborateurs scientifiques, auprès de qui j'ai pu partager la joie de la découverte scientifique.

Travailler dans un laboratoire vous permet aussi de rencontrer de nombreuses personnes, même si vous ne travaillez pas directement avec elles. Depuis 2012, j'ai pu partager beaucoup de choses, avec de nombreux doctorants, chercheurs et enseignants. Impossible de citer tout le monde, et j'espère que personne n'est actuellement en train de lire ces remerciements en cherchant désespérément son nom sans jamais le trouver, c'est juste un oubli malencontreux ! Parmi toutes ces personnes, je remercie particulièrement Lucie et Olivier, pour leur gentillesse et leur accueil lors de mon arrivée à Paris. Je remercie également Yasmina, Julien, et Paul, "les anciens", qui furent de super compagnons de travail et de sortie lors de mes premières années au LIP6 ; Bruno, Fanny, Pierre, Safia, de l'équipe RO, pour leur sagesse ; Anne-Elisabeth, David, Hugo I, Hugo II, Nadjet, Parham, Siao-Leu, "les anciens doctorants" pour leur sens de l'humour aiguisé et leur enthousiasme sans faille ; Cassandre, Gaspard, Mahdi, Margot et Marvin du fameux bureau 401, pour leur déraison et leur amour du buisson ardent. Et sans oublier évidemment Clara, de ce même bureau, "the community manager", merci beaucoup pour ton écoute et tes bonnes trouvailles en tous sens. Un grand merci aussi à Nawal, inclassable, pour ta joie, ta passion, et ton rire communicatif !

Je remercie également les enseignants du Département des Activités Physiques et Sportives qui m'ont permis de pouvoir nager le papillon, de m'initier à l'apnée, de courir un peu plus vite, de battre Marvin au badminton, de souffrir terriblement au cours de préparation physique générale avec Paul et Olivier, de me détendre avec Fanny au cours de sophrologie, et de parfaire mes coups au tennis (ce qui ne m'a pas empêché de subir une cinglante défaite contre Bruno). Pouvoir se défouler physiquement dans de bonnes conditions me fut indispensable pendant ce long travail de rédaction.

Je remercie ma famille, et plus particulièrement mes parents d'être toujours présents à mes côtés, même de loin, et de m'avoir toujours laissé une liberté totale dans mes choix, et ce depuis que je suis "petit".

Enfin, un énorme merci à mes deux amours de toujours, Virginie, la plus cool et la plus patiente des femmes, et Louis, mon meilleur campagnon de jeu, toujours prêt à vouloir gagner contre son Papa !

[1] Même si ce manuscrit a été écrit dans la langue d'Alan Turing, j'ai préféré écrire ces quelques lignes dans la langue d'Henri Poincaré (mon arrière-arrière-arrière grand-père selon l'arbre généalogique des Docteurs en Mathématiques ! (`https://genealogy.math.ndsu.nodak.edu/`), pour, on peut le dire, essentiellement me faciliter la tâche dans l'écriture de cette section très attendue !

Foreword

The objective of this manuscript is to provide a coherent survey of the work we have done on multi-objective combinatorial optimization since we defended our PhD thesis in December 2009.

We started working on multi-objective combinatorial optimization during our PhD thesis, which was about the development of new metaheuristics to solve multi-objective combinatorial optimization problems. More precisely, the goal was to develop new heuristic methods that generate good approximations of Pareto-optimal sets in small running times. The methods have been applied to the multi-objective traveling salesman problem, the multi-objective multidimensional knapsack problem, and an application in radiotherapy treatment.

Thenceforth, during a short research stay at the Université Pierre et Marie Curie (the former Sorbonne Université) and a post-doctoral stay at the Université Lumière Lyon 2, we started working on new dominance relations and rank-dependent aggregators, mainly the Lorenz dominance (with Lucie Galand) and the Choquet integral (with Antoine Rolland). We also have accomplished a post-doctoral stay at the Université Catholique de Louvain in Belgium, where we worked on the multi-objective scheduling of the operating theater and on a machine learning project aiming to predict academic success. This last subject is somewhat outside our main area of research and will not be discussed in this manuscript.

In parallel, we have continued working on the adaptation of metaheuristics to multi-objective problems, and more particularly in some ways to improve the two-phase Pareto local search that we have developed during our PhD thesis. Some very-large scale neighborhood search has been developed to solve the multi-objective set covering problem (with Daniel Tuyttens). A new data structure has been developed to accelerate the operation of non-dominance checking (with Andrzej Jaszkiewicz). We also have worked on different applications with some researchers met during a research stay at the Universal Federal Fluminense (Niteroi, Brazil).

During the last years, we have mainly worked on interactive methods (preferences of the DM are learned during the search) to solve multi-objective combinatorial problems. A new exact method has been developed (with Nawal Benabbou) and local search and genetic algorithms have been adapted. A large part of this work has been realized in the context of the supervision of the PhD thesis of Cassandre Leroy at Sorbonne Université. The title was "Incremental elicitation combined with heuristic methods for solving multi-objective combinatorial optimization problems". We were co-supervising her thesis with Nawal Benabbou, and the Director was Patrice Perny. The defense took place on December 5, 2022.

Our research is mainly based on developing new algorithms to solve combinatorial problems. Many of the methods developed are heuristics, and their quality is evaluated through intensive experiments on academic problems. Due to limited number of pages, we won't be able to present all experimental results, only few results, tables and figures will be shown. We prefer to focus on showing the ideas behind the methods, often through small didactic examples. Detailed experimental results could be found in the related papers, that are all available on our personal web page[2]. In the same spirit, most of the proofs associated with the various properties developed in this manuscript will not be detailed.

We also point out that between January 2016 and January 2018, we took a sabbatical leave for spousal follow-up in Ireland. After a 6-month research stay at IBM Dublin, we worked as a data scientist for the Technological University of Dublin in Ireland. We worked mainly on data clustering, synthetic data generation (i.e., creating "random" data-sets that resemble actual company data), and on a visualization tool that combines the opinions expressed in customer feedback and spatio-temporal data. All these results-oriented works will not be exposed in this manuscript.

Finally, we want to specify that, as requested by Sorbonne Université, this document must be of about fifty pages, must present all our main research results, and situate it in relation to international literature. A perspective on further research should also be included.

[2] `https://webia.lip6.fr/~lustt/`

Contents

Main acronyms and notations

- MOCO: Multi-Objective Combinatorial Optimization
- MCDA: Multiple-Criteria Decision Analysis
- p: number of objectives
- $\mathcal{P}$: set of objective numbers, i.e., $\{1, \ldots, p\}$
- x: feasible solution
- y: point in objective space (image of a feasible solution)
- $f_i(x)$: objective function (number i)
- $\mathcal{X}$: feasible set of a MOCO problem, or set of alternatives of a MCDA problem
- $\mathcal{Y}$: image of $\mathcal{X}$ in the objective space
- $\succ_P$: Pareto dominance
- $\succeq_P$: weak Pareto dominance
- $>_P$: strict Pareto dominance
- X_E: Pareto-optimal set
- Y_E: Pareto front
- X_N: non-dominated set or Pareto archive
- Y_N: set of mutually Pareto non-dominated points
- X_L: Lorenz-optimal set
- X_C: Choquet-optimal set
- DM: Decision-Maker
- WS: Weighted Sum
- OWA: Ordered Weighted Average
- PLS: Pareto Local Search
- TSP: Traveling Salesman Problem
- BOTSP: Bi-Objective TSP
- MOTSP: Multi-Objective TSP
- MOKP: Multi-Objective Knapsack Problem
- MOSTP: Multi-Objective Spanning Tree Problem

1 Introduction

Combinatorial optimization problems are ubiquitous in our society. They appear in many applications: production, scheduling, logistics, timetables design, transport, and more. Combinatorial optimization consists in determining an optimal solution from a finite discrete set of feasible solutions (i.e., solutions that respect the constraints of the combinatorial optimization problem under study). A function, defined by a decision-maker (DM), is generally used to evaluate the solutions, which can correspond to a cost, a gain, an utility or any other criterion (or objective). A good knowledge of the problem is essential for the choice of the criterion to be optimized. Indeed, a poor definition of the criterion can lead to the generation of solutions that do not correspond to the preferences of the DM. However, many real world decision contexts require taking into account more than one criterion: some criteria evaluating quantitative issues (performance, duration, cost, etc.) and sometimes other measuring qualitative issues (environmental aspects, customer opinions, etc.). These criteria are often conflicting and heterogeneous, but considering explicitly them enables the DM(s) to appreciate the possible trade-offs and to progress towards the definition of a best compromise solution.

A common way to solve multi-objective combinatorial optimization problems is to generate all compromise solutions. A compromise solution is a solution such that the value of the solution for one objective cannot be improved without negatively affecting at least one of the other objectives. These solutions are more formally called *Pareto-optimal solutions*, a notion that will be properly defined in the next section. Multi-objective optimization is an important area of research [81], but more complex than single-objective optimization, given that the notion of optimality is no longer direct.

During our doctoral thesis research, we primarily assumed that the DM had no established preferences regarding the criteria or certain solutions. Therefore, the main approach we studied was the generation of all Pareto-optimal solutions, since it is impossible to differentiate between these solutions. The set of Pareto-optimal solutions can then be proposed to the DM, who is then free to choose the solution that best matches her preferences.

Multi-objective combinatorial optimization problems are difficult to solve. The generation of the set of Pareto-optimal solutions becomes increasingly burdensome with the number of criteria, even for medium size instances of multi-objective problems. This is mainly due to the following two reasons. First, the number of Pareto-optimal solutions grows exponentially with the number of criteria [81]. Second, for most multi-objective combinatorial optimization problems, the associated decision problem is NP-complete, even if the underlying single objective problem can be solved in polynomial time. Owing to these major computational difficulties, exact algorithms aiming at generating the entire Pareto-optimal set can only be used for small-size instances of multi-objective problems. In the case of larger instances, heuristic approaches are commonly employed to yield high-quality approximations within a practical processing time. The heuristics are usually generated from more high-level concepts: the metaheuristics.

Metaheuristics for solving multi-objective combinatorial problems

Metaheuristics are general concepts [105, 126, 240] used for heuristic solving of combinatorial optimization problems, including both intensification techniques to explore promising regions of the search space and diversification techniques to explore the search space as much as possible. Metaheuristics are often employed to generate heuristics for a specific combinatorial optimization problem. Metaheuristics, being initially developed to solve single-objective optimization problems, had to be adapted to deal with multi-objective problems. The first adaptations date from the 1990s, with the multi-objective simulated annealing [224, 251].

In our PhD thesis, we have developed a powerful adaptation of local search for solving multi-objective combinatorial optimization problems: the two-phase Pareto local search (2PPLS). The method has been applied to several multi-objective combinatorial optimization problems: the traveling salesman problem, the knapsack problem and to a combinatorial optimization problem encountered in radiotherapy, which amounts to decomposing an integer non-negative matrix into a linear combination of binary matrices [84]. After our thesis, we have made a few improvements to the method (in particular the use of a specific data structure for handling the problem of dominance checking), and solved other applications. All these results will be summarized in Section 2. This section will also contain a small survey about the methods (exact and heuristic) used to solve multi-objective combinatorial optimization problems.

Lorenz dominance

For some problems, the Pareto dominance is not sufficiently restrictive. Let's take the example of assigning objects to certain agents. Each agent expresses her interest in each of the items (through a utility function), and the goal is to find an allocation such that all the agents are satisfied with the solution. We can model this

problem as a multi-objective combinatorial optimization problem: the utility received by each agent corresponds to one distinct objective. In order to satisfy all agents, it is important to generate fair solutions, i.e., solutions in which no agent appears to be disadvantaged. However, Pareto dominance does not incorporate the notion of equity at all. Many unfair solutions will be generated, i.e., solutions that are good for some agents but bad for the others. One way to integrate fairness in multi-objective combinatorial optimization is to use Lorenz dominance. Lorenz dominance is a refinement of Pareto dominance, which allows generating solutions that establish a good compromise between the total values of the utilities received by the agents, and the minimal value of the utility received by one of the agents. We have developed a new exact method to generate all Lorenz optimal solutions of bi-objective combinatorial problems. The ideas behind this method will be presented in Section 3.

Choquet integral

Another way to deal with multi-objective optimization problems is to aggregate the different objectives in order to obtain a problem with a single criterion to be optimized. Though, the problem with this type of approach is that, first, it exists multiple aggregation functions, and it is not easy to select the most appropriate one. Second, an aggregation function often requires some parameters, like weights associated to each criterion, which are not easy to set. As a result, there is a high risk that the resulting solution will not fully match the DM's preferences.

In Section 4, we present an original approach using an aggregation function, the Choquet integral, but without the previously mentioned disadvantages. Indeed, the Choquet integral [109] is a general aggregation function, that can model many different aggregators, like the weighted sum, the operators OWA [261] or WOWA [248]. Nevertheless, the use of such aggregation function complicates the search for preferred solutions, on the one hand in the learning of the preferential parameters, generally more sophisticated than a set of weights (a capacity for the Choquet integral, which is a set function, with an exponential number of parameters according to the number of objectives), and on the other hand in the search for optimal solutions since the non-linearity of these aggregation functions leads to solving NP-hard combinatorial optimization problems, even when the mono-objective version of these problems is polynomial [102]. To avoid the need to determine the parameters of the Choquet integral beforehand, our approach consists in generating the set of solutions that are optimal for at least one set of parameters of the Choquet integral. This approach has two advantages: the parameters of the Choquet integral do not need to be determined and, in the end, a set of solutions of smaller size comparing to the set of Pareto-optimal solutions is proposed to the DM.

Interactive methods and preference learning

A last possibility studied in this manuscript to solve a multi-objective combinatorial optimization problem is to *interact* with the DM along the process of generation of the solutions [193]. Indeed, it may suffice to partially learn the preferences of the DM to determine her preferred solution among all the feasible solutions of the considered multi-objective problem. Furthermore, these preferences can be learned during the solving process. Consequently, interactive methods handle both the learning of preferences and the resolution of the resulting optimization problem, relying on a partial and dynamic learning of the preferences of the DM in order to direct the search straightly toward favored solutions [185, 215].

Learning a DM's preferences is a fundamental area of artificial intelligence (AI), particularly when AI is used to assist human beings in their decision-making. If we take the example of deciding on an itinerary for a road-trip, the various criteria to be taken into account (distance, cost, landscapes crossed, environmental aspects, etc.) and the colossal number of possible options often make decision-making difficult and tedious for human beings. In the context of these problems, AI can help the DM by accelerating her decision-making by efficiently learning her subjective preferences. This involves asking the DM the right questions (for example, by asking to compare certain solutions) and considerably reducing the number of possible options, by focusing on those solutions that represent a good compromise in the eyes of the DM. Note that in multi-objective combinatorial optimization problems, the set of options to be compared is not explicitly defined (as the size of this set is exponential, like for the road trip problem), but only implicitly given through the structure of the problem (e.g., a graph representing all the roads between cities).

A crucial issue in interactive methods is to limit the number of queries presented to the DM, so that determining her preferred solutions is not a tedious process. Another challenge is to guarantee the quality of the solution proposed. Section 5 will focus on interactive methods and preference elicitation, for solving classic multi-objective combinatorial problems, and problems under matroid constraints, with exact and heuristic methods.

2 Exact and heuristic methods for solving multi-objective combinatorial optimization problems

2.1 Multi-objective combinatorial optimization

We first define a general multi-objective combinatorial optimization (MOCO) problem. Let's consider a discrete finite set of n elements, $E = \{e_1, e_2, \ldots, e_n\}$ and p cost functions $c(e)$ that associate to each element $e \in E$ a multidimensional cost vector $(c_1(e), c_2(e), \ldots, c_p(e))$ with $c_i : E \to \mathbb{R}, \forall i \in \mathcal{P} = \{1, \ldots, p\}$. The elements of E are generally linked to a combinatorial structure (e.g., a graph, a tree, a matroid, etc.). Let $\mathcal{X}$ a feasible set, defined by a set of constraints on the set E, such that $\mathcal{X}$ is a subset of the power set of E ($\mathcal{X} \subseteq 2^E = \{0,1\}^n$). For example, for the spanning tree problem, E represents the set of edges of a graph, while $\mathcal{X}$ is the set including all the spanning trees of the graph. Note that $\mathcal{X}$ is generally not explicitly given (due to its exponential size) but implicitly given through the structure and/or constraints on E. Each solution $x \in \mathcal{X}$ is evaluated through p objective functions f_i, $\forall i \in \mathcal{P}$, that associates to x a real value, i.e., $f_i : \mathcal{X} \to \mathbb{R}$, for $i \in \mathcal{P}$. Usually the value associated with the objective i of a feasible solution x is simply equal to the sum of the costs of the elements present in x, i.e., $f_i(x) = \sum_{e \in x} c_i(e)$, for $i \in \mathcal{P}$. The cost vector associated with a feasible solution x is denoted by $y(x) = (f_1(x), \ldots, f_p(x)) \in \mathbb{R}^p$. We therefore have that the image of the admissible set in the objective space is defined by $\mathcal{Y} = \{y(x) : x \in \mathcal{X}\} \subset \mathbb{R}^p$. Note that in this manuscript, we will consider that all objective functions are commensurable (i.e., same unit and same scale). Then a general MOCO problem comes down to the following problem[3]:

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

This means we are trying to find a solution x that optimizes all the objective functions at the same time. Unfortunately, this problem is often unsolvable. Indeed, the objective functions are generally conflicting, i.e., a solution optimal for one objective function is not necessarily optimal for the others. Therefore, in MOCO, solutions are habitually compared through their images in the objective space (also called *points*) with the Pareto dominance relation.

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

For the sake of simplicity, we will use the dominance relation for solutions as well, i.e., a solution x Pareto dominates a solution x' if and only if the point corresponding to x Pareto dominates the point corresponding to x': $x \succ_P x' \Leftrightarrow y(x) \succ_P y(x')$.

We can now define a Pareto-optimal solution.

Definition 2. *Pareto-optimal solution: a feasible solution* $x^* \in \mathcal{X}$ *is called* Pareto-optimal *(or* efficient*) if there is no other feasible solution* $x \in \mathcal{X}$ *such that* $y(x) \succ_P y(x^*)$.

For these solutions, it is not possible to improve the value of one of the criteria without deteriorating at least one other criterion.

The set of all Pareto-optimal solutions is called the Pareto-optimal set, and is denoted by X_E. The image in the objective space of a Pareto-optimal solution is called a *Pareto* non-dominated point. The set of Pareto non-dominated points corresponds to the Pareto front, and is denoted by Y_E. Note that we can have that two or more different solutions can correspond to the same point in the objective space. Such solutions will be called *equivalent* solutions. If an exact method is used to generate X_E, in theory, the method needs to generate all equivalent Pareto-optimal solutions (a Pareto-optimal set which contains all equivalent solutions is sometimes called a *maximal* Pareto-optimal set). However, in practice, we are generally satisfied to obtain a unique corresponding solution for each Pareto non-dominated point (a *minimal* Pareto-optimal set). Note that it is the same as in single-objective optimization, where only one optimal solution is usually sought.

We now define two auxiliary dominance relation.

Definition 3. *Weak Pareto dominance relation: we say that a point* $u = (u_1, ..., u_p) \in \mathbb{R}^p$ weakly Pareto dominates *a point* $v = (v_1, ..., v_p) \in \mathbb{R}^p$ *if, and only if,* $u \succ_P v$ *or* $u = v$. *We denote this relation by* $u \succeq_P v$.

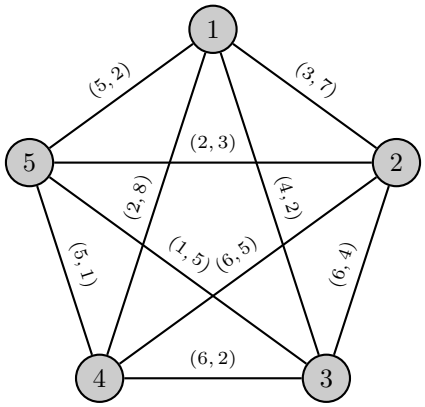
Definition 4. *Strict Pareto dominance relation: we say that a point* $u = (u_1, ..., u_p) \in \mathbb{R}^p$ strictly Pareto dominates *a point* $v = (v_1, ..., v_p) \in \mathbb{R}^p$ *if, and only if,* $u_k < v_k, \forall\, k \in \mathcal{P}$. *We denote this relation by* $u >_P v$.

[3] the case of maximization can be reduced to minimization by a simple change of sign.

Definition 5. *Weak Pareto-optimal solution: a feasible solution $x^* \in \mathcal{X}$ is called* weak Pareto-optimal *(or* weakly-efficient*) if there is no other feasible solution $x \in \mathcal{X}$ such that $y(x) >_P y(x^*)$.*

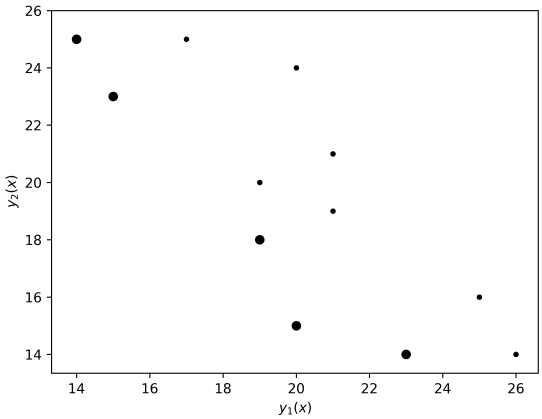
The set of all weak Pareto-optimal solutions is called the weak Pareto-optimal set, and is denoted by X_{wE}.

Example 1. *We consider in this example the bi-objective traveling salesman problem (BOTSP). Let V be a set of n vertices, denoted by $\{1, \ldots, n\}$, and E a set of edges, denoted by $\{(i,j) : i \neq j, i, j \in \{1, \ldots, n\}\}$, connecting the vertices two by two. Two cost functions associate to each edge e a two-dimensional cost vector $(c_1(e), c_2(e))$. A feasible solution to this problem is a cycle that visits each vertex exactly once (Hamiltonian cycle). The value associated with the objective i of a feasible solution x is equal to the sum of the costs of the edges present in x, i.e., $f_i(x) = \sum_{e \in x} c_i(e)$, for $i \in \{1, 2\}$. Let's consider the following instance of this problem, with $n = 5$, represented by the following graph:*



For the symmetric TSP (the problem is symmetric when the cost from a vertex i to a vertex j is equal to the cost from the vertex j to the vertex i), there are exactly $\frac{(n-1)!}{2}$ feasible solutions. For this instance with $n = 5$ nodes, there are thus 12 feasible solutions. The feasible solutions and their evaluation according to the two cost functions are given in the following table.

Solution	*Evaluation*
$x^1 = (1,2,3,4,5)$	$(25, 16)$
$x^2 = (1,2,3,5,4)$	$(17, 25)$
$x^3 = (1,2,4,3,5)$	$(21, 21)$
$x^4 = (1,2,4,5,3)$	$(19, 20)$
$x^5 = (1,2,5,3,4)$	$(14, 25)$
$x^6 = (1,2,5,4,3)$	$(20, 15)$
$x^7 = (1,3,2,4,5)$	$(26, 14)$
$x^8 = (1,3,2,5,4)$	$(19, 18)$
$x^9 = (1,3,4,2,5)$	$(23, 14)$
$x^{10} = (1,3,5,2,4)$	$(15, 23)$
$x^{11} = (1,4,2,3,5)$	$(20, 24)$
$x^{12} = (1,4,3,2,5)$	$(21, 19)$



For example, the solution $x^1 = (1,2,3,4,5)$ corresponds to the following cycle: $1 \to 2 \to 3 \to 4 \to 5 \to 1$. Note that the representation of a cycle is not unique: this same cycle could have been represented by the solution $(3,4,5,1,2)$ or even by the solution $(5,4,3,2,1)$ (given the symmetric cost).

On the right of the table, we have represented in the objective space the 2-dimensional points corresponding to the evaluation of each feasible solution. We see that the Pareto front Y_E is composed of 5 Pareto non-dominated points: $\{(14,25), (15,23), (19,18), (20,15), (23,14)\}$ (represented by the 5 large filled circles). The Pareto-optimal set X_E is thus equal to $\{x^5, x^{10}, x^8, x^6, x^9\}$. Note that there is also one weak Pareto-optimal solution: x^7 $(y(x^7) = (26,14))$.

In MOCO, there exists an important classification of the Pareto-optimal solutions: *supported* Pareto-optimal solutions and *unsupported* Pareto-optimal solutions. The images of the supported solutions in the

objective space are located on the convex hull of the Pareto front, and the images of the unsupported solutions are located inside the convex hull of the Pareto front. More precisely, we can characterize these solutions as follows [81]:

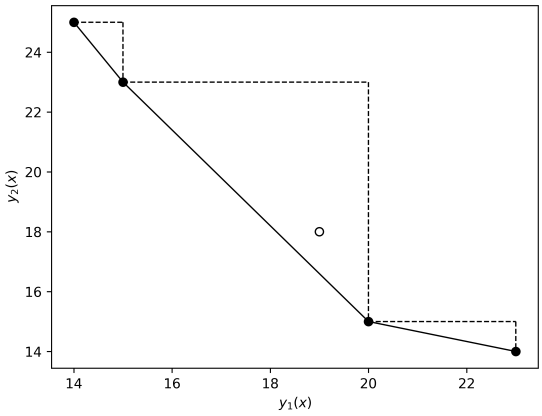
Definition 6. *Supported Pareto-optimal solution: a solution x is a supported Pareto-optimal solution if and only if there exists a strictly positive weight vector λ ($\lambda_k > 0, \forall\, k \in \mathcal{P}$) such that x is an optimal solution to the weighted sum (WS) single-objective problem:* $\min_{x\in\mathcal{X}} \sum_{k=1}^{p} \lambda_k\, f_k(x)$.

A Pareto-optimal solution which is not supported is simply called a *Pareto unsupported optimal solution.*

Example 2. *For the BOTSP instance of Example 1, it can be shown that there exists 4 supported Pareto-optimal solutions (given in the table below), and one unsupported optimal solution (x^8). On the right of the table, we have represented some of the edges of the convex hull of the Pareto front (black line). We see that all supported Pareto-optimal points (represented by filled points) are located on these edges. We have also drawn right triangles (dotted lines) between each consecutive supported points. An unsupported point is always located inside these triangles, and we see that it is the case for the point $(19, 18)$ corresponding to the image of the unsupported Pareto-optimal solution x^8. We note that the unsupported Pareto-optimal solution x^8 establishes the best compromise between the two objective functions, and it is therefore essential not to restrict the search to supported solutions only.*

In the table, we have also added the values of λ_1 (the first component of the weight vector (λ_1, λ_2) of the WS, with $\lambda_2 = 1 - \lambda_1$) for which a supported solution is optimal according to the corresponding WS. We see that the solution x^{10} is only optimal for a small range of values ($[\frac{8}{13}, \frac{2}{3}]$) (which represents only about 5% of the possible values). There is thus less chance to generate this solution if a WS with a random weight set was used.

Supported solutions	*Evaluations*	$\lambda_1{}^*$
$x^5 = (1, 2, 5, 3, 4)$	$(14, 25)$	$[\frac{2}{3}, 1[$
$x^{10} = (1, 3, 5, 2, 4)$	$(15, 23)$	$[\frac{8}{13}, \frac{2}{3}]$
$x^6 = (1, 2, 5, 4, 3)$	$(20, 15)$	$[\frac{1}{4}, \frac{8}{13}]$
$x^9 = (1, 3, 4, 2, 5)$	$(23, 14)$	$]0, \frac{1}{4}]$



Note that in bi-objective optimization, the set of supported Pareto-optimal solutions can be easily computed with a *dichotomic search* [9, 56] which gives the different weighting vectors that allow to generate all supported solutions (more details will be given in Section 2.2.6).

2.2 Exact methods for solving MOCO problems

2.2.1 Main difficulties

MOCO problems are hard to solve. We point out three major difficulties:

1. $\mathcal{NP}$-hardness of these problems even when the associated single-objective problem is of polynomial complexity: e.g., the multi-objective assignment, the multi-objective spanning tree and the multi-objective shortest path problems are $\mathcal{NP}$-hard [85]. The decision problem related to a multi-objective problem is as follows: "Given $d \in \mathbb{R}^p$, does there exist $x \in \mathcal{X}$ such that $(f_1(x), \ldots, f_p(x)) \succeq_P d$?" For many MOCO problems, the decision problem is NP-complete, even in the bi-objective case.

2. Intractability: as there is not generally a unique optimal solution when multiple objectives are involved, the number of Pareto-optimal solutions turns out to be a crucial point in assessing the difficulty of the problem. It leads us to the notion of *intractability* [81].

Definition 7. *A MOCO problem is called* intractable *if the number of Pareto-optimal solutions can be exponential in the size of the instance.*

Unfortunately, most MOCO problems are intractable [81]. Indeed, for many MOCO problems, it is possible to create instances such that each feasible solution corresponds to a distinct non-dominated Pareto point.

3. The presence of unsupported Pareto-optimal solutions (i.e., not localized on the edges of the convex hull of the Pareto front), more difficult to generate than supported Pareto-optimal solutions, as these solutions are not optimal for any WS. Moreover, generally, the number of unsupported solutions increases exponentially with the instance size, while the number of supported has polynomial growth.

In the following, we briefly review the state of the art of existing exact methods for solving MOCO problems (for a full survey, see the recent paper of Halffmann et al. [118]).

2.2.2 Adaptation of single-objective polynomial algorithms

We start by examining a few adaptations of single-objective polynomial algorithms to multi-objective optimization. Unfortunately, it is not trivial to adapt single-objective algorithms to the multi-objective case. We give some examples for the multi-objective spanning tree and the multi-objective shortest path.

Multi-objective spanning tree

In 1985, Corley [59] was the first to try to adapt the Prim algorithm to solve the multi-objective spanning tree problem. We remind the problem and the idea of the Prim algorithm. From a graph $\mathcal{G} = (V, E)$ with a set V of n vertices and a set E of evaluated edges (such that the graph is connected), the goal is to find a subset of edges such that the graph is connected and that the total sum of the cost associated to each edge is minimized. The algorithm operates by building a tree one vertex at a time. From an arbitrary starting vertex, at each step the minimal cost edge connecting the tree to another vertex is added, if no cycle is formed, until $(n-1)$ edges have been selected to form a spanning tree. The idea of Corley to adapt the single-objective Prim algorithm is simple: instead of choosing the minimal cost edge, all Pareto non-dominated edges are considered to be added to the vertex, and a forest is built, from which each tree is considered as new starting point for applying the algorithm. Unfortunately, it has been shown that Pareto dominated spanning trees may be returned by the algorithm (see, e.g., [151] for an example). Hamacher and Ruhe [119] tried to improve the adaptation of Corley by eliminating the dominated trees at each step of the algorithm. Unfortunately, it does not work as well, as some Pareto-optimal spanning trees may be omitted (see e.g., [142] for an example). Another option to solve the multi-objective spanning tree problem is to adapt the Kruskal algorithm [149], and using the property discovered by Serafini [223] in 1987 stipulating that there exists a topological order of the edges of the graph $\mathcal{G}$ such that the greedy algorithm applied to this order yields a Pareto-optimal spanning tree. Unfortunately, the problem of identifying the appropriate topological orders is not easy to solve [81].

Multi-objective shortest path

For the multi-objective shortest path, we have that although a Pareto-optimal path is always composed of efficient sub-paths between vertices along the path, the composition of efficient sub-paths does not yield necessarily a Pareto-optimal path (see e.g., [81] for an example).

2.2.3 Connectedness of Pareto-optimal solutions

One can also have the idea to use neighborhood search to generate all Pareto-optimal solutions. It comes from the observation that Pareto-optimal solutions can be truly closed in the decision space, differing by only a few number of variable values. We first define the notion of neighborhood:

Definition 8. *A neighborhood is a mapping function $\mathcal{N} : \mathcal{X} \to 2^{\mathcal{X}}$ that assigns to any solution $x \in \mathcal{X}$ a set of solutions $\mathcal{N}(x) \subset 2^{\mathcal{X}}$. $\mathcal{N}(x)$ is called the neighborhood of x, and a solution $x' \in \mathcal{N}(x)$ is called a neighbor of x.*

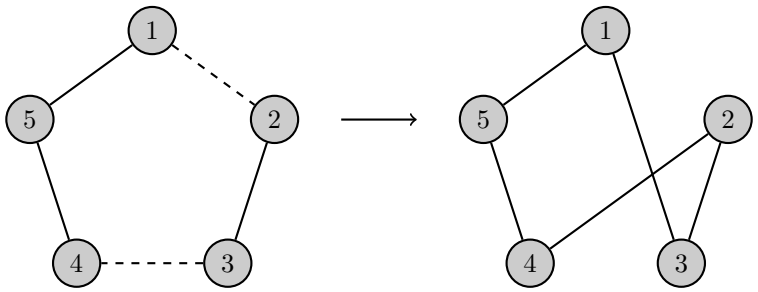
We now define the notion of adjacency graph following the definition of Gorski et al. [106].

Definition 9. *The adjacency graph $\mathcal{G} = (V, E)$ of the Pareto-optimal solutions of a given MOCO problem is defined as follows: the set of vertices V represents the Pareto-optimal solutions of the MOCO problem. An edge is introduced between all pairs of vertices which are neighbors with respect to the considered definition of neighborhood of the MOCO problem. These edges form the set E.*

We can now define the notion of connectedness of Pareto-optimal solutions.

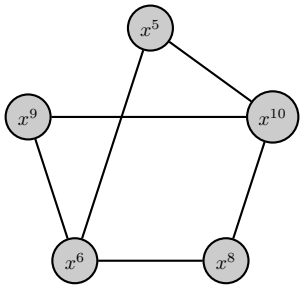
Definition 10. *The set X_E of all Pareto-optimal solutions of a given MOCO problem is said to be connected, if and only if, its corresponding adjacency graph $\mathcal{G}$ is connected.*

Example 3. *A popular neighborhood for the TSP is the 2-opt neighborhood. This neighborhood works as follows: from a solution, two edges are removed, and are replaced by two other edges to form a new solution. It is illustrated below for an instance with 5 vertices. Starting from the solution (1,2,3,4,5) the edges (1,2) and (3,4) are removed and replaced by the edges (1,3) and (2,4) (there exists only one possibility to form a new solution).*



All combinations of two edges are considered. In this way, a set of new solutions can be generated. For the 2-opt neighborhood, its size is in $\mathcal{O}(n^2)$ ($\frac{(n)(n-3)}{2}$) more precisely).

In the figure below, we show the connectdness graph of the instance of the bi-objective TSP of Example 1. The vertices represent the 5 Pareto-optimal solutions, and an edge between two vertices mean that the two solutions associated to the two vertices are neighbors according to the 2-opt neighborhood. We remark that, for this small instance, the graph is connected, which means that it would be possible to generate all the Pareto-optimal solutions by using the 2-opt neighborhood, if it was applied from one of the Pareto-optimal solutions.



Unfortunately, Ehrgott and Klamroth [83] and Gorski et al. [106] found instances for many MOCO problems (i.e., shortest path, minimum cost flow, minimum spanning tree, knapsack and assignment problems) for which the set X_E is not connected. However, using neighborhood search can be used to provide good approximations of the Pareto-optimal set, as it will be shown in Section 2.4.1.

In the following, we present some general techniques to generate all Pareto-optimal solutions of MOCO problems. We first focus on classic techniques and then on more specific techniques.

2.2.4 ϵ-constraint method

The ϵ-constraint method is a one of the first method developed for solving MOCO problems. The method has been introduced by Haimes et al. [117] in 1971. In the ϵ-constraint method, one of the objective functions is optimized while the other objective functions are put as constraints, in the following way:

$$\begin{cases} \underset{x \in \mathcal{X}}{\text{minimize}} & f_k(x) \\ \text{subject to} & f_i(x) \leq \epsilon_i, i \in \mathcal{P} \setminus \{k\} \end{cases}$$

The ϵ-constraint method has the advantage of being adaptable to any MOCO problems following a suitable adaptation of the constraints. On the other hand, this method modifies the structure of the original problem by adding constraints, which can make the problem more difficult to solve. However, many improvements have been made to the method (especially by partitioning the search space into distinct zones) [33, 139, 154, 156, 181, 183, 241, 270]. We have ourselves applied the method to a multi-objective scheduling problem in the operating room, taking into account a balanced allocation of nurses' skills [179].

2.2.5 Branch and bound method

The branch and bound method [152] is a method that works by implicit enumeration of the feasible set. As a single-objective method, it uses bounds on the optimal value of sub-problems, enabling the elimination of partial solutions by detecting that they cannot lead to optimal solutions. The simplest bounds that could be used in multi-objective optimization are the ideal and nadir points, defined as follows:

Definition 11. *The ideal point of a MOCO problem denoted as z^I is the point composed of the best coordinates of all Pareto non-dominated points, i.e., $z^I_k = \min_{y \in Y_E} y_k, \forall\, k \in \mathcal{P}$.*

Definition 12. *The nadir point of a MOCO problem denoted as z^N is the point composed of the worst coordinates of all Pareto non-dominated points, i.e., $z^N_k = \max_{y \in Y_E} y_k, \forall\, k \in \mathcal{P}$.*

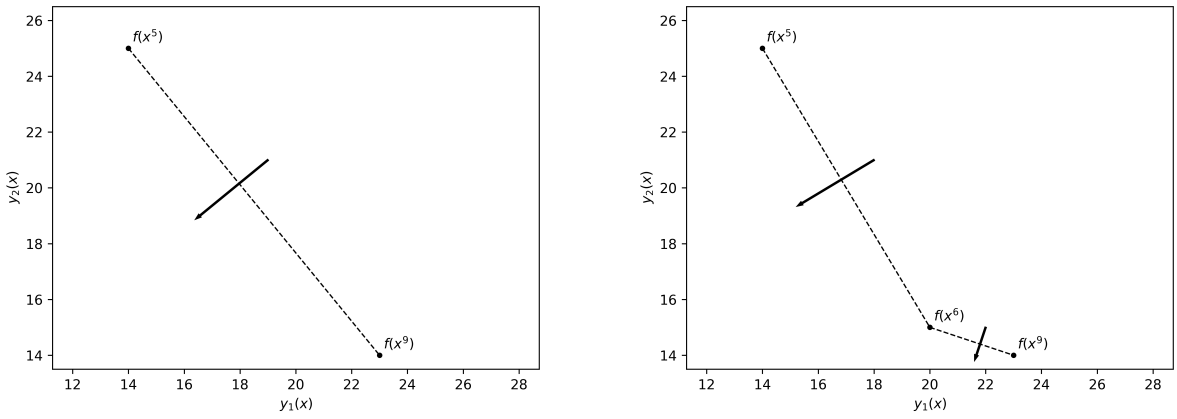
Example 4. *For the MOCO problem of Example 1, it can be easily seen that the ideal point z^I is equal to $(14, 14)$ and the nadir point z^N is equal to $(23, 25)$.*

However, to effectively adapt the branch and bound method to MOCO problems, it is necessary to use more elaborate bounds, composed of a set of points. Effective adaptations of branch and bound have been proposed by Sourd and Spanjaard [232] for solving the bi-objective spanning tree problem and by Ulungu and Teghem [250] for solving the bi-objective knapsack problem. A recent effective adaptation of branch and bound to problems with more than two objectives have been realized by Forget et al. [94] (in about one hour of computing time, they can solve instances of the knapsack problem with 50 objects and 3 objectives, or 20 objects and 5 objectives). For a full review, see the survey of Przybylski and Gandibleux [207].

2.2.6 Two-phase method

The two-phase method is a general approach for solving MOCO problems, introduced by Ulungu and Teghem [249] in 1995 for solving bi-objective problems. In the first phase, the set of supported Pareto-optimal solutions is generated and in the second phase, using the non-dominated zone of the objective space formed by the supported Pareto-optimal points (i.e., the zone where new Pareto non-dominated points could be found), enumeration techniques are used to generate the unsupported Pareto-optimal solutions. In the first phase, a series of WS problems are solved. Indeed, a supported Pareto-optimal solutions is necessarily optimal for a WS problem with strictly positive weight. For problems with two objectives, there exist an efficient algorithm for generating all sets of weights corresponding to supported Pareto-optimal points. The method, independently proposed by Cohon [56] and Aneja and Nair [9], is recursive and is based on generation of normal lines to the lines connecting two already detected adjacent supported Pareto-optimal points. The normal lines give the new weight directions that would enable to potentially get new supported Pareto-optimal points.

Example 5. *Let's go back to Example 1. Let's suppose that the solution x^5 (evaluation equal to $(14, 25)$) and x^9 (evaluation equal to $(23, 14)$) have been previously generated. Then to find new supported Pareto-optimal solutions, a new weighted set, corresponding to the normal vector to the line connecting the two points (see the figure at bottom left) is determined. A new supported Pareto-optimal solution is generated by solving a WS problem with this weight set. The solution x^6 is obtained (evaluation equal to $(20, 15)$) and two new WS problems will need to be solved, as shown in the figure at bottom right.*



Unfortunately, this method does not work for problems with more than two objectives. Indeed, as shown by Przybylski et al. [209], with at least three objectives, normal directions to the hyper-plans passing through three adjacent supported Pareto-optimal points can have negative values and therefore not leading to supported Pareto-optimal solutions by solving WS problems with these weights. Adaptations of the method to deal with problems with more than two objectives were nevertheless achieved quite recently (the first method dates back to 2010) [35, 194, 209, 210].

In the second phase, enumeration techniques are used. Branch and bound approaches were initially developed [249] but lately, the most successful approach was using k-ranking algorithms, that are methods that generate the k-best solutions to an optimization problem (see e.g., [252] for a k-best algorithm for the TSP). It works by discovering unsupported Pareto-optimal solutions in ascending order of value obtained by WS problems [208, 211, 233].

2.2.7 Tchebychev aggregation functions

In this section, we consider the weighted Tchebychev norm for generating the Pareto-optimal set of a MOCO problem.

Definition 13. *The weighted Tchebychev norm is defined as follows:*

$$f_T(x) = \max_{i \in \mathcal{P}} \lambda_i |f_i(x) - z^r| \text{ with } \lambda_i > 0 \text{ and } \sum_{i=1}^{p} \lambda_i = 1$$

The point z^r is a reference point, generally the ideal point.

The use of this norm was suggested by Bowman [39] in 1976. However, by optimizing this norm we can generate only weakly Pareto-optimal solutions. Therefore, Steuer and Choo [234] proposed the augmented weighted Tchebychev norm, where the sum of the values of all objective functions multiplied by some constant coefficient ϵ has been added.

Definition 14. *The augmented weighted Tchebychev norm is defined as follows:*

$$f_{AT}(x) = \max_{i \in \mathcal{P}} \lambda_i |f_i(x) - z^r| + \epsilon \sum_{i=1}^{p} |f_i(x)| \text{ with } \lambda_i \geq 0 \text{ and } \sum_{i=1}^{p} \lambda_i = 1, \epsilon \geq 0, \lambda_i + \epsilon > 0, i \in \mathcal{P}$$

This new norm has mainly been used in interactive methods (see Section 5). To our knowledge, it is only in 2012 with the work of Dächert et al. [78] that the method has been successfully applied for solving MOCO problems. They have showed that it is not possible to use one single coefficient ϵ to generate the Pareto-optimal set, and it is necessary to adapt the coefficient according to the problem instance and the choice of the reference point. The method was then improved by Holzmann and Smith [125] by the use of a weighted augmented term, which allows simplifying the determination of ϵ.

2.3 Approximation algorithms with performance guarantee

An alternative to exact methods are approximation algorithms with performance guarantee. Approximation algorithms are mainly based on the ϵ-dominance relation, defined as follows.

Definition 15. *ϵ-dominance relation: we say that a point $u = (u_1, ..., u_p) \in \mathbb{R}^p$ ϵ-*Pareto dominates *a point $v = (v_1, ..., v_p) \in \mathbb{R}^p$ if, and only if, $u_k \leq (1 + \epsilon)v_k, \forall\, k \in \mathcal{P}$ and $\epsilon > 0$. We denote this relation by $u \succeq_\epsilon v$.*

We can now define the notion of $(1 + \epsilon)$-approximation of a Pareto-optimal set.

Definition 16. *A set X of feasible solutions is called an $(1+\epsilon)$-approximation if, for every feasible solution $x \in \mathcal{X}$, there exists a solution in X that ϵ-Pareto dominates x.*

Finding $(1 + \epsilon)$-approximations has gained a lot of attention since the seminal work of Papadimitriou and Yannakakis [196], showing that any multi-objective problems admits an ϵ-Pareto set of fully polynomial cardinality (under very weak assumptions). Moreover, they give a complete characterization of multi-objective problems for which $(1 + \epsilon)$-approximations can be obtained in polynomial time according to the size of the instances. Following these results, researchers have undertaken numerous studies on approximation methods designed to address specific problems like knapsack, shortest path, matching, traveling salesman, and others. Additionally, efforts have been directed towards diminishing the size of approximations, exploring approximations consisting of supported solutions only, and investigating approximations with exact values for at least one of the objectives.

However, we have not carried out any research on this subject, and although it is a very interesting one, we will not go into the methods and issues involved. We recommend that interested readers refer to the recent survey of Herzel et al. [123].

2.4 Metaheuristics for solving MOCO problems

Due to the relative inefficiency of exact methods to solve high-size instances of MOCO problems, many metaheuristics (MHs) have been adapted with the aim of getting good approximations of the Pareto-optimal set. One common point between all multi-objective metaheuristics (i.e., adaptation of MHs to MO problems, denoted by MOMHs) is that they all manage a *non-dominated set* (or *Pareto archive*), which is a set of potentially Pareto-optimal solutions, in the sense that the solutions in the set are solutions that are not dominated by any other solutions found so far. However, we cannot prove that the solutions are Pareto-optimal, since generally MOMHS do not explore all the search space and cannot provide guarantees that the solutions generated are Pareto-optimal. Also, in the Pareto archive, if two different solutions correspond to the same point, usually only one solution is kept in the archive. We define more formally a Pareto archive below. To do this, we first need to define the mutually non-dominated relation between two solutions.

Definition 17. *Mutually non-dominated relation: we say that two solutions are mutually non-dominated or non-dominated w.r.t. each other if neither one of the two solutions dominates the other one.*

Definition 18. *Non-dominated set (or Pareto archive) (X_N): a set of distinct feasible solutions such that any pair of solutions in the set are mutually non-dominated, i.e $\forall x \in X_N, \nexists\, x' \in X_N \mid x' \succ x$. The representation of X_N in the objective space is denoted by Y_N.*

A solution belonging to a non-dominated set will be called a non-dominated solution or a potentially Pareto-optimal solution. In MOMHs, the Pareto archive is constantly updated with new solutions, in order to keep only mutually non-dominated solutions. This is an important operation that needs to be performed efficiently. Different ways to manage the Pareto archive will be presented in Section 2.5.

The goal of any metaheuristic is to return the best possible approximation of the Pareto-optimal set, denoted by $\widetilde{X}_E$, according to the available computation time. Different indicators can be used to measure the quality of the approximation. If the Pareto-optimal set is available, one can measure, e.g., the proportion of Pareto-optimal solutions generated and the average distance between the Pareto front and the approximation. However, the Pareto-optimal set is not always available. In this case, the most popular quality indicator is the hypervolume [268], which is in line with the Pareto dominance relation. Its calculation is not trivial as the number of objectives increases, but a number of efficient methods have recently been developed [113, 131]. For a full review of quality indicators, see the survey of Zitzler et al. [269].

2.4.1 Pareto local search

Pareto Local Search (PLS) is a powerful adaptation of simple local search [127] based on improving moves to multi-objective optimization, developed independently by many authors [10, 82, 197, 239] at the beginning of the 2000s. The idea of PLS can also be found in older papers related to the multi-objective spanning tree problem [7, 119]. Its principle is very simple: starting from an initial population of non-dominated solutions (i.e., a non-dominated set, that can be composed of one single feasible solution), a neighborhood function is applied to each solution of the population with the aim of generating new non-dominated solutions. The neighborhood is then applied to each new non-dominated solution found, until no more improvement is possible, i.e., a Pareto local optimal set has been found. A Pareto local optimum set is a set of solutions that has no improving solutions in its neighborhood. A Pareto archive of unlimited size is used to store all non-dominated solutions found.

Two main advantages of PLS over more elaborated methods is that, first, PLS does not need any numeric parameters, and second, it always ends [10] (i.e., there is no need to introduce stopping rules).

There are slightly different versions of PLS according to the way the neighborhood is applied. There are versions where the neighborhood is only applied from non-dominated solutions [197, 239] and there are versions where the neighborhood can sometimes be applied from dominated solutions [10, 82, 128]. We will present here a version where the neighborhood can occasionally be applied from dominated solutions. The general algorithm of our version of PLS works as follows (see Algorithm 1). Two entries are needed: an initial population P composed of non-dominated solutions and a neighborhood function $\mathcal{N}(x)$. The method returns an approximation $\widetilde{X}_E$ of the Pareto-optimal set X_E.

First, all the solutions in P are added to the approximation $\widetilde{X}_E$. Then a solution s (the current solution) is selected from P, and the neighborhood of s is explored, i.e., for a solution s we generate all its neighbors. Let s' be a neighbor solution of s. If s does not weakly Pareto dominate s', we update the approximation $\widetilde{X}_E$ with s' (that is s' is added to $\widetilde{X}_E$ if no solution in X_E Pareto dominates s', and all solutions Pareto dominated by s' are removed from $\widetilde{X}_E$). Then if s' has been added to $\widetilde{X}_E$, s' is added to P and its neighborhood will be later explored. The solution s is removed from P and the method starts again the exploration of the neighborhood from another solution of P. The method stops naturally when P becomes empty.

Note that we can have some dominated solutions in P, as a solution previously added could be dominated by a new solution. However, this does not append frequently, since a solution is added to P only if the solution is not dominated according to the solutions in $\widetilde{X}_E$ (and thus not dominated according to the solutions in P). One way to deal with only non-dominated solutions is to also update the population P, but it could be a bit time-consuming, and allowing some dominated solutions brings diversity. Another way is to do as in the PLS version of Paquete et al. [197], where no population P is used: the solutions are directly picked up from the archive $\widetilde{X}_E$. In this way, the neighborhood is always applied from a non-dominated solution. On the other hand, this requires the use of a flag on each solution of $\widetilde{X}_E$ that states if the neighborhood of the solution has been explored or not.

Note also that in this version the neighborhood of a solution is fully explored (best improvement exploration), but some authors have studied other possibilities, like stopping the neighborhood exploration once a non-dominated neighbor has been found (first improvement exploration) [74, 161]. Liefooghe et al. [161] have also studied the way the solution is selected from P and the use of a bounded archive as well.

Algorithm 1 `Pareto Local Search (PLS)`

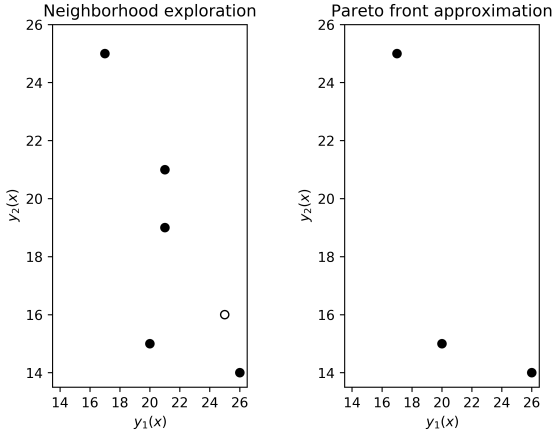
IN $\downarrow$: an instance of a MOCO problem, an initial population of mutually non-dominated solutions P, a neighborhood function $\mathcal{N}(x)$.
OUT $\uparrow$: an approximation $\widetilde{X}_E$ of the efficient set X_E.

--| Initialization of $\widetilde{X}_E$
$\widetilde{X}_E \leftarrow P$
while $P \neq \emptyset$ **do**
  --| Random selection of a solution s of P
  $s \leftarrow$ `Select`(P)
  --| Generation of all the neighbors s' of s
  for all $s' \in \mathcal{N}(s)$ **do**
    if $f(s) \not\succeq f(s')$ **then**
      if `Update`$(\widetilde{X}_E \updownarrow, s' \downarrow)$ **then**
        $P \leftarrow P \cup \{s'\}$
  $P \leftarrow P \setminus \{s\}$
return $\widetilde{X}_E$

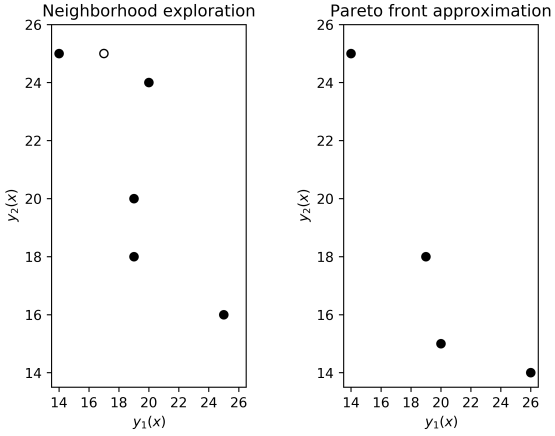
where the procedure `Update()` updates a set of non-dominated solutions $\widetilde{X}_E$ with a new solution and returns true if the new solution has been added to this set.

Example 6. *We illustrate in the following figures the main ideas of PLS, through the first iterations of the PLS method applied to the instance of the bi-objective TSP of Example 1.*

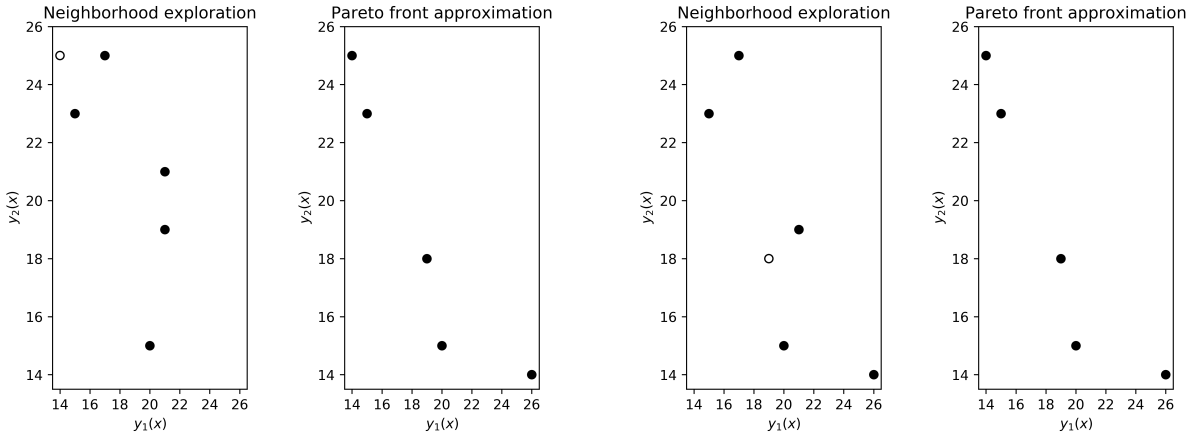
Iteration 1:



Iteration 2:



Iteration 3: **Iteration 4:**



We start PLS from one single solution: solution $x^1 = (1, 2, 3, 4, 5)$ with an evaluation equal to (25,16). The current solution x_1 is represented in the objective space by the empty black circle on the left figure of Iteration 1. Using the 2-opt neighborhood, x_1 has five neighbors, represented on the same figures with filled black circles. Three of them are non-dominated and constitutes the current approximation of the Pareto front, which are represented in the objective space by the filled black circles, on the right figure of Iteration 1. The other figures also represent the neighborhood exploration of one solution of the population, and the current approximation of the Pareto front. We see that at the end of iteration 3, the exact Pareto front has been generated (corresponding to the five Pareto-optimal solutions given in Example 1). However, the method will continue to explore the neighborhood from the solution of the population P until the population is empty.

Because of its simplicity and effectiveness, PLS serves as a crucial element in some of the best approaches for tackling MOCO problems [61, 137, 171]. PLS has also been used in other contexts, e.g., for solving scheduling problems [55, 76, 245], multi-agent problems [128] or multi-objective Markov decision processes [144].

Also, many improvements and modifications of PLS have been realized since the first versions appeared. Without going into details, we list below some of these studies:

- Inja et al. [128] have proposed the queued PLS. They studied the integration of dominated solutions in the population P in order to bring some diversity: k (between 2 and 10) candidates locally not dominated by the current solution s are randomly selected from the neighborhood. They also proposed a genetic version of PLS. The principle is as follows: mutation and re-combinations operators are applied to the local optimum set found at the end of an execution of PLS. The modified set is then used as a new starting population for PLS. This process is applied until a stopping criterion is met (limited number of iterations or computation time). The method has been applied to multi-objective coordination graph problems, which are single-state problems from the multi-agent literature in which agents must work together in order to obtain a shared (vector-valued) reward.

- Dubois-Lacoste et al. [77] have studied the *anytime* behavior of PLS. Indeed, as PLS has a natural stopping criterion, there is no guarantee that the method will obtain good results if the method was stopped earlier (e.g., because of some limited computation time or unknown available computation time). This is why it is often interesting to get a method which is able to obtain the best results possible according to the available computation time. The authors improved the anytime behavior of PLS by improving the selection of the solution in P and by modifying the neighborhood exploration.

- Jaszkiewicz [132] have studied adaptations of PLS to deal with many-objective combinatorial optimization problems ($p \geq 3$). Indeed, for problems with at least 3 objectives, PLS is quite inefficient because there can be many non-dominated solutions in the neighborhood, which results in slow convergence of the method. Moreover, the process of updating large Pareto archives with new solutions could become time-consuming. For these reasons, Jaszkiewicz proposed a new mechanism for selecting solutions whose neighborhood is explored. Only some "promising" solutions, i.e., solutions that are the best for weighted Tchebychev scalarizing functions, are selected for neighborhood exploration. Moreover, a partial exploration of the neighborhood is considered. The method has been experimented on the multi-objective TSP (MOTSP).

- Drugan and Thierens [73, 74] have studied the use of stochastic perturbation operators (mainly mutation operations), to restart PLS when a Pareto local optimum set has been reached. The method has been experimented on bi-objective quadratic assignment problems.

2.4.2 Two-phase Pareto local search

The Two-Phase Pareto Local Search (2PPLS) has been developed during our PhD thesis [171]: the method simply consists in applying PLS from a high-quality initial population. In our first version of 2PPLS the method was only designed for solving bi-objective combinatorial problems and the initial population was composed of a good approximation of the supported Pareto-optimal solutions. An adaptation of the exact Aneja and Nair method (see Section 2.2.1) has been developed, in order to be able to use a heuristic instead of an exact method to solve the single-objective problems resulting from the linear weighted aggregation of the objectives.

The method has been applied to two classic MOCO problems:

1. The bi-objective TSP, for which we obtained state-of-the-art results (at the time of the publication of the results). We have also introduced some speed-up techniques to improve the neighborhood (composed of 2-opt moves) exploration and to be able to solve high-size instances (until 1000 cities) [171].

2. The multi-objective knapsack problem, for which also state-of-the-art results have been obtained, thanks to the use of very large scale neighborhood technique, which allows an efficient exploration of the neighborhood of a solution [176].

Some improvements to the method have been developed by some authors, we briefly list them below:

- Cornu et al. [61] have developed the "Perturbed Decomposition Algorithm" (PDA). The method combines intelligently different ideas from 2PPLS, decomposition and data perturbation. A decomposition-based approach divides a MOCO problem into a set of equally distributed aggregated sub-problems [265]. These sub-problems are simultaneously optimized in a collaborative manner. Each sub-problem is characterized by a weight that determines a distinct search direction. Additionally, each sub-problem maintains its best solution based on the chosen aggregation function. The data perturbation technique consists in generating single-objective instances using linear aggregation, with some perturbations that bring some small modifications to the data. In this way, when a single-objective solver is used to solve the perturbed instance, the solution generated will depend on the perturbations, and this allows to get a diversified set of solutions for one single-objective instance (and unsupported Pareto-optimal solutions can be generated even if a linear aggregation of the objectives combined with an exact solver is used). We have ourselves introduced the data perturbation technique for multi-objective optimization [175]. With this new method, they obtained new state-of-the-art results for the MOTSP.

- Shi et al. [226, 227] have proposed PPLS/D for Parallel Pareto Local Search Based on Decomposition. In this method, the original search space is divided into some sub-regions and PLS is applied independently in each of these sub-regions. The method is applied to unconstrained binary quadratic programming problems and to the MOTSP with at most four objectives.

- We have made a few improvements to 2PPLS ourselves [133]: instead of using the heuristic adaptation of Aneja and Nair in the first phase, we simply generate a specified number L of single-objective problems (with a weighted linear aggregation of the objectives) to be solved, which allows bringing more flexibility and allows setting the best value of CPU time spent during the first phase by setting an appropriate value for L. We also have studied the integration of solutions obtained by weighted linear aggregations, during the execution of PLS. All these improvements have been tested on the MOTSP.

Application to the bi-objective pollution-routing problem

We have applied 2PPLS to solve a variant of the vehicle routing problem that arises in the context of green logistics, called the bi-objective pollution-routing problem (bPRP) [62]. The two conflicting objectives considered are the minimization of the CO_2 emissions and the costs related to driver's wages. We present in more details the problem below.

Let $\mathcal{G} = (\mathcal{V}, \mathcal{A})$ be a complete and directed graph with a set $V = \{0, \ldots, n\}$ of vertices and a set $A = \{(i, j) : i, j \in \mathcal{V}; i \neq j\}$ of arcs. The depot is represented by vertex 0 whereas the set of customers is denoted by $V' = V \setminus \{0\}$. Each customer $i \in V'$ has a non-negative demand q_i, a time interval $[a_i, b_i]$ when it can be served, and a service time t_i. The travel distance between a pair of vertices i and j is given by d_{ij}, $(i, j) \in A$. A set $K = \{1, \ldots, r\}$ of homogeneous vehicles with capacity Q is available at the depot. Drivers are assumed to be paid per unit of time, and CO_2 emissions are assumed to be proportional to vehicle fuel consumption, which in turn is dependent on environment and traffic-related parameters such as vehicle type, speed, load, acceleration and congestion [17, 67]. Let ν_{ij} and f_{ij} be the vehicle speed and the vehicle load

on arc (i, j), respectively. The amount of CO_2 emissions associated with travel from i to j can be computed as follows:

$$F_{ij}(\nu_{ij}, f_{ij}, d_{ij}) = \xi(\mu NV + w\gamma\alpha_{ij}\nu_{ij} + \gamma\alpha_{ij}f_{ij}\nu_{ij} + \beta\gamma\nu_{ij}^3)d_{ij}/\nu_{ij}, \quad (1)$$

where ξ and γ are constants related to fuel properties, β and w are associated with vehicle characteristics and α_{ij} is a constant that depends on road characteristics and vehicle acceleration. Moreover, μ is the engine friction factor, N is the engine speed, V is the engine displacement. The equation (1) is based on the comprehensive emissions model described by Barth et al. [14] and Barth and Boriboonsomsin [13]. The parameter values adopted in the bPRP can be found in [68, 148].

The bPRP aims at designing a set of routes by deciding the arcs to be included in the solution as well as their associated vehicle speeds, in order to minimize the two aforementioned objectives, while respecting time window constraints. Hence, if we consider f_c and f_d as the cost associated with fuel consumption and labor activities, respectively, the bPRP objective functions can be expressed as follows:

$$\text{minimize } f_1(x) = f_c \sum_{(i,j)\in S} F_{ij}(\nu_{ij}, f_{ij}, d_{ij}) \quad (2)$$

$$\text{minimize } f_2(x) = f_d \sum_{i\in\mathcal{V}'} s_i, \quad (3)$$

where x is a feasible solution (set of routes), S is the set of arcs in x, $F_{ij}(.)$ corresponds to the amount of CO_2 emissions as given in Eq. (1), and s_i represents the total time spent on a route that has the vertex $i \in \mathcal{V}'$ as the last visit before returning to the depot.

Note that we can consider the bPRP as a multi-objective mixed integer linear program (MOMILP) since one has to perform binary decisions, i.e., by deciding whether an arc must be included in the solution or not, as well as continuous decisions, i.e., by deciding the vehicle speed over each selected arc. Motivated by real-life aspects, the selected speeds must respect lower (V_{MIN}) and upper (V_{MAX}) bounds, which come from the problem definition. According to Bektaş and Laporte [17], the speed at which a vehicle travels on arc (i, j) is imposed by traffic regulations. Nevertheless, it is important to emphasize that in the PRP definition, traffic conditions are not taken into account. Hence, the vehicles are allowed to travel at any speed within a given interval.

For solving the single-objective PRPs obtained from the linear weighted aggregation of the objectives, necessary to the first phase of 2PPLS, we have used the algorithm proposed by Kramer et al. [148], which is to the best of our knowledge the best heuristic method available for solving the single-objective PRP. Their multi-start matheuristic, called ILS-SP-SOA, combines iterated local search (ILS) [167] with a set partitioning (SP) approach and a speed optimization algorithm (SOA).

Different neighborhoods have been used in the second phase of 2PPLS. Indeed, for this problem, it is not enough to just use a 2-opt neighborhood to optimize the routes, since it is also necessary to determine which vehicles will deliver which client. Therefore, other neighborhoods have been added, such as the shift operator (a customer is moved from one route to another one) or the swap operator (two customers in two different routes are switched).

Extensive computational experiments over existing benchmark instances show that this adaptation of 2PPLS leads to better results in less CPU time when compared to those obtained by state-of-the-art methods (see [62] for more details).

2.4.3 Variable neighborhood search

In 2PPLS, the method generally stops when a Pareto local optimum set is obtained. We have proposed a new strategy to escape from Pareto local optimal set [178], based on the variable neighborhood search technique (VNS) [120]. Once a Pareto local optimum set has been found according to a neighborhood, we increase the size of the neighborhood in order to generate new potentially Pareto-optimal solutions and to escape from the Pareto local optimum set. The pseudocode of 2PPLS with VNS is given by Algorithm 2.

Compared to Algorithm 1, the algorithm needs different neighborhood functions $\mathcal{N}_k(x)$, identified by their size k, with $k \in \{k_{\min}, k_{\min}+1, \ldots, k_{\max}\}$, with k_{min} the minimal size and k_{max} the maximal size. As PLS, the method needs as entry an initial population P composed of non-dominated solutions.

The method starts by exploring the neighborhood of each solution s of the population P. The neighborhood structure initially used is the smallest ($k = k_{\min}$). If a neighbor s' is not weakly dominated by the current solution s, we update the archive $\widetilde{X}_E$ with the solution s'. If s' has been added to $\widetilde{X}_E$, s' is added to an auxiliary population P_a for future exploration. We then associate a value to the solution s, equal to k, that is the neighborhood size. Once the neighborhood of all the solutions s in P has been explored, if the

Algorithm 2 PLS with VNS

IN ↓: an instance of a MOCO problem, an initial population of mutually non-dominated solutions P, neighborhood functions $\mathcal{N}_k(x)$ ($k \in \{k_{\min}, \ldots, k_{\max}\}$).
OUT ↑: an approximation $\widetilde{X}_E$ of the efficient set X_E.

```
--| Initialization of X̃_E
X̃_E ← P
--| Initialization of the neighborhood size
k ← k_min
--| Initialization of an auxiliary population
P_a ← ∅
repeat
  while P ≠ ∅ do
    --| Generation of all neighbors s' of each solution s ∈ P
    for all s ∈ P do
      for all s' ∈ N_k(s) do
        if f(s) ⋦ f(s') then
          if Update(X̃_E ↕, s' ↓) then
            P_a ← P_a ∪ {s'}
      k(s) ← k
  if P_a ≠ ∅ then
    --| P is composed of the new generated solutions
    P ← P_a
    --| Reinitialization of P_a
    P_a ← ∅
    --| We start again with the smallest neighborhood structure
    k ← k_min
  else
    --| We use a larger neighborhood structure
    k ← k + 1
    --| We use as population the solutions of X̃_E that are not already Pareto local optimum for N_k(x)
    P ← {x ∈ X̃_E | k(x) < k}
until k > k_max
return X̃_E
```

auxiliary population is not empty, we start again the neighborhood exploration with the solution s in P_a, while maintaining the smallest neighborhood structure. Otherwise, if P_a is empty, that means that a Pareto local optimum set according to the neighborhood of size k has been reached. We thus increase the size of the neighborhood k by one unity ($k \leftarrow k+1$). The neighborhood exploration is then re-iterated from the solutions in $\widetilde{X}_E$ that are not already Pareto local optimum for the neighborhood of size k. Note that, in general, a solution Pareto local optimum for the neighborhood of size k is not necessary Pareto local optimum for the neighborhood of size $(k-1)$. That is why, after considering a larger neighborhood, we always restart the search with the smallest neighborhood structure.

Application to the multi-objective set covering problem

In this section, we show how to adapt PLS+VNS to the MO version of the set covering problem. In the MO set covering problem (MOSCP), we have a set of m rows (or items), and each row can be covered by a subset of n columns (or sets). To each column j, p costs c_l^j ($l \in \mathcal{P}$) are associated. The MOSCP consists in determining a subset of columns, among the n columns, such that all the rows are covered by at least one column and that the total costs are minimized.

More precisely, the MOSCP is defined as follows:

$$\text{(MOSCP)} \begin{cases} \text{minimize} & f_l(x) = \sum_{j=1}^{n} c_l^j x_j & l \in \mathcal{P} \\ \text{s.t.} & \sum_{j=1}^{n} t_{ij} x_j \geq 1 & i \in \{1, \ldots, m\} \\ & x_j \in \{0,1\} & j \in \{1, \ldots, n\} \end{cases}$$

with x the decision vector, formed of the binary variables x_j ($x_j = 1$ means that the column j is in the solution) and t, a binary covering matrix, such that t_{ij} is equal to 1 if the column j covers the row i and equal to 0 otherwise. It is assumed that all coefficients c_l^j are non-negative integer. The data associated to the MOSCP are thus a cost matrix of size (n x p) and a covering matrix of size (m x n). There are many applications related to the SCP [80]. For example, a classic application in the selection of a set of employees to accomplish a set of tasks. Each task needs some skills from the employees, and each employee has a set of skills. The goal is to select the smallest subset of employees such that all tasks can be realized by at least one employee with the required skill.

As the single-objective version of the SCP is $\mathcal{NP}$-Hard [103], the MOSCP is $\mathcal{NP}$-Hard too. We have thus applied PLS+VNS to the problem to generate a good approximation of the Pareto-optimal set. With local search, the larger the neighborhood, the better the quality of the local optimum obtained. However, by increasing the size of the neighborhood, the time to explore the neighborhood becomes higher. Therefore, using a larger neighborhood does not necessary give rise to a more effective method. If we want to keep reasonable running times while using a large neighborhood, an efficient strategy has to be implemented in order to explore the neighborhood.

We have thus used a large neighborhood search (LNS) [204, 225] technique. LNS was introduced by Shaw [225] in 1998 to solve the vehicle routing problem. With LNS, the neighborhood exploration consists of two parts: a destroy method and a repair method. The destroy method destructs some parts of the current solution, while the repair method rebuilds the destroyed solution. The LNS belongs to the class of neighborhood search known as very large scale neighborhood search (VLSNS) [4]. These two neighborhood exploration techniques often cause some confusion. Contrary to LNS, in VLSNS, the neighborhood is usually restricted to a neighborhood that can be searched efficiently. Ahuja et al. [4] defines three methods used in VLSNS to explore the neighborhood: variable depth methods, network flow based improvement methods and methods based on restriction to subclasses solvable in polynomial time. Therefore, VLSNS is not only a large neighborhood (furthermore the definition of large is imprecise), but essentially a neighborhood that uses a specific method to explore efficiently a large neighborhood.

VLSNS and LNS are very popular in single-objective optimization [4, 204]. For example, the Lin-Kernighan heuristic [162], one of the best heuristics for solving the single-objective TSP, is based on VLSNS. On the other hand, there is almost no study of LNS/VLSNS for solving MOCO problems. To our knowledge, the only known result is the LS of Angel et al. [10], which integrates a dynasearch neighborhood (the neighborhood is explored with dynamic programming) to solve the BOTSP.

For our adaptation of VNS to solve the MOSCP, as the size of the neighborhood will be kept relatively small, a VLSNS is not necessary. Nevertheless, we have retained the destruction and repair methods from LNS. Starting from a current solution, called x^c, the aim of LNS is to produce a set of neighbors of high quality, in a reasonable CPU time. The general technique that we have used for solving the MOSCP is the following:

1. **Destroy method:** Identification of a set of variables candidates to be removed from x^c (set $\mathcal{O}$)
2. **Repair method:**
   - Identification of a set of variables, not in x^c, candidates to be added (set $\mathcal{I}$)
   - Creation of a residual MO problem formed by the variables belonging to $\{\mathcal{O} \cup \mathcal{I}\}$, and the constraints not anymore fulfilled (that is the items not anymore covered, in the case of the MOSCP).
   - Resolution of the residual problem: a set of potentially Pareto-optimal solutions of this problem is produced. The potentially Pareto-optimal solutions of the residual problem are then merged with the unmodified variables of x^c to produce the neighbors.

The details about the adaptation of LNS to the MOSCP can be found in the related paper [178].

The method has been applied to different size instances, from small instances (100 columns, 10 rows) to bigger instances (1000 columns, 100 rows), with 2 objectives. The results obtained by PLS+VNS gives a proportion of Pareto non-dominated points generated included between 21% and 92%, with a running time always less than 45 seconds. We have also applied the method to solve three-objective instances. However, the method needs about 30 min to generate 11039 potentially Pareto-optimal solutions of an instance with 60 rows and 600 columns, while for the instance of the same size, with two objectives, only 25 seconds were needed, to generate 221 solutions. We see that increasing the number of objectives considerably increases the number of potentially Pareto-optimal solutions to generate, and therefore the computational time. One way to limit the computational time is to limit the number of potentially Pareto-optimal solutions generated. In order to do so, we have proceeded in the following way: we maintain a hypergrid in the objective space, dynamically updated according to the minimum and maximum values of the solutions for each objective,

and we measure the *mean density of the hypergrid*, calculated as follows. It is equal to the number of non-dominated points generated divided by the number of hypercubes of the hypergrid that contain at least one non-dominated point. The hypergrid allows thus to obtain information concerning the distribution of the solutions in the objective space and to stop the search once enough solutions have been generated. Here, we will simply stop the method when the mean density of the hypergrid attains a certain threshold. Using this technique, we are able to control the running time of PLS+VNS, and getting approximations with a good distribution of the non-dominated points. More details about the experiments and the results can be found in [178]. At the time of the publication (2014), we obtained state-of-the-art results. Note that since, the results have been improved by Weerasena et al. [256], by using a heuristic adaptation of the branch and bound method.

Application to the multi-objective open-pit mining operational planning problem

We have also applied PLS+VNS to the multi-objective open-pit mining operational planning problem [55]. In this particular problem, there is a set of mining pits, a set of trucks and a set of load equipment's. The main objective is to create a final product by blending ores sourced from various mining pits. The key consideration is to minimize three conflicting factors: deviations in production and quality targets, along with optimizing the number of trucks required during the production process. Through computational experiments, it has been demonstrated that PLS+VNS method exhibits clear superiority compared to the adaptation of NSGA-II [64].

Application to the bi-objective direct marketing campaign

A last problem on which we have adopted PLS+VNS appears in cross-selling campaigns [54]. The objective is to present the appropriate products to customers in order to maximize the expected profit. However, this must be achieved while adhering to purchasing constraints defined by investors. The two objectives are the maximization of the total profit generated by the promotion campaign and the minimization of risk-adjusted return, which is assessed using the Sharpe ratio—a measure of reward-to-variability.

2.5 ND-Tree: a data structure and algorithm for the dynamic non-dominance problem

In this section, we consider the dynamic non-dominance problem [218], i.e., the problem of updating a Pareto archive with a new candidate solution x. We formally define this problem as follows. Let's consider a candidate solution x and a Pareto archive X_N. The problem is to update X_N with x and consists in the following operations. If x is weakly dominated by at least one solution in X_N, x is discarded and X_N remains unchanged. Otherwise, x is added to X_N. Moreover, if some solutions in X_N are dominated by x, all these solutions are removed from X_N, in order to keep only mutually non-dominated solutions (see Algorithm 3).

Algorithm 3 `DynamicNonDominance`

IN $\downarrow$: a new candidate solution x
IN-OUT $\updownarrow$: a Pareto archive X_N

if $(\nexists\, x' \in X_N \mid x' \succeq x)$ **then**
  $X_N \leftarrow X_N \cup \{x\}$
  for all $(x' \in X_N \,|\, x \succ x')$ **do**
    $X_N \leftarrow X_N \backslash \{x'\}$
return X_N

Note that if there already exists a solution x' in X_N with the same evaluation according to the p objective functions (i.e., $f_i(x) = f_i(x'), \forall i \in \mathcal{P}$), x is not added to X_N since the weakly Pareto dominance relation is used in the algorithm.

The dynamic non-dominance appears in many MOMHs, but also in exact methods, e.g., in multi-objective dynamic programming approaches when it is necessary to update partial solutions obtained at different states [16, 138]. The time needed to update a Pareto archive generally increases with a growing number of objectives and a growing number of solutions. In some cases, it may become a crucial part of the total running time of the multi-objective method and, it is essential to use an efficient method to update a Pareto archive.

A small variation of the dynamic problem is the *static* non-dominance problem, where one has to find the set of non-dominated solutions X_N among a set of solutions X. In general, static problems can be solved more

effectively than their dynamic counterparts since they have access to richer information [97, 114, 150, 205]. However, in many multi-objective methods, it is not possible to store all candidate solutions and then solving the static non-dominance problem. Indeed, many MOMHs use the Pareto archive during the run of the algorithm, i.e., the Pareto archive is not just the final output of the algorithm. For example, PLS works directly with the Pareto archive and only solutions from the archive are added to a population for future neighborhood exploration. In the MOMH developed by Deb et al. [65], one of the parents is selected from the Pareto archive. Therefore, in such methods, computation of the Pareto archive cannot be postponed at the end of the algorithm. Also, saving all solutions generated can be very demanding in terms of memory.

2.5.1 Brief state of the art

Here, we present some of the methods and data structures proposed in the literature for dealing with the dynamic non-dominance problem. This review is not supposed to be exhaustive. Other methods can be found in [89, 220] and reviews in [5, 188, 219]. We describe linear list, quad-tree and one recent method, M-Front [72]. Note that, in the following, a solution that weakly dominates another will sometimes be called a *covering* solution.

Linear List

With this structure, a new solution is compared to all solutions in the list until a covering solution is found, or all solutions are checked. The solution is only added if it is non-dominated w.r.t. all solutions in the list, that is, in the worst case we need to browse the whole list before adding a solution. The complexity in terms of number of solutions comparison is thus in $\mathcal{O}(N)$ with N the size of the list. When only two objectives are considered, we can use the following specific property: if we sort the list according to one objective (let's say the first), the non-dominated list is also sorted according to the second objective. Therefore, in broad terms, updating the list can be efficiently done in the following way. We first determine the potential position i of the new candidate solution in the sorted list according to its value for the first objective, with a binary search. If the new solution is not dominated by the preceding one in the list (if there is one), it is certain that the new solution is not dominated by any solutions of the list (as the solutions after position i have a higher value for the first objective) and can be inserted at position i. If the new solution has been added, we need to check if there are some dominated solutions: we browse the next solutions in the list, until a solution with a better evaluation according to the second objective is found. Any solutions discovered with a lower evaluation score for the second objective must be eliminated, as they are dominated by the new solution.

The worst-case complexity is still in $\mathcal{O}(N)$ since it can happen that a new solution has to be compared to all the other solutions (in the special case where we add a new solution in the first position and all the solutions in the sorted list are dominated by this new solution). But on average, experiments have showed that the behavior of this structure for handling bi-objective updating problems is much better than the simple list.

Quad-tree

The use of quad-tree for storing potentially Pareto-optimal solutions was proposed by Habenicht [116] and further developed by Sun and Steuer [236] and Mostaghim and Teich [187]. In quad-tree, a tree is used to store the solutions and solutions are located in both internal nodes and leaves. Each node may have p^2 children corresponding to each possible combination of results of comparisons on each objective, where a solution can either be better or worse. In the case of mutually non-dominated points, $(p^2 - 2)$ children are possible, since the combinations corresponding to dominating or covered solutions are not necessary. Quad-tree allows for a fast checking if a new solution is dominated or covered. A weak point of this data structure is that when an existing solution is deleted, its whole subtree must be reinserted into the structure. As a result, eliminating a dominated solution is generally costly.

M-Front

M-Front has been proposed by Drozdík et al. [72] in 2015. The general idea of M-Front is as follows. There are p sorted lists of solutions on each objective. Then reference points are used to reduce the number of comparisons needed on each sorted list. A reference point is a point closed to the new candidate point. To find reference points, M-Front uses the k-d tree data structure. The k-d tree is a binary tree, in which each intermediate node divides the space into two parts based on a value of one objective. While going down the tree, the algorithm cycles over particular objectives, selecting one objective for each level.

2.5.2 ND-Tree-based update

We have proposed a new method, called ND-Tree-based update, for the dynamic non-dominance problem [134]. The method is based on a dynamic division of the objective space into hyper-rectangles, which allows avoiding many comparisons of objective function values. The main data structure on which is based the method is a tree, called a ND-Tree.

In the presentation of the method, we will directly use the representations of the solutions in objective space, i.e., points. We recall that the image of a Pareto archive X_N in the objective space is denoted by Y_N. Of course, to each point is associated a solution but considering only points is sufficient (and more simple) for the presentation of the method.

We first define the notions of (approximate) local ideal and nadir points of a subset of points.

Definition 19. *The local ideal point of a subset $\mathcal{S} \subseteq Y_N$ denoted as $z^I(\mathcal{S})$ is the point in the objective space composed of the best coordinates of all points belonging to $\mathcal{S}$, i.e., $z^I_k(\mathcal{S}) = \min_{y \in \mathcal{S}} y_k, \forall\, k \in \mathcal{P}$. A point $\widehat{z}^I(\mathcal{S})$ such that $\widehat{z}^I(\mathcal{S}) \succeq z^I(\mathcal{S})$ will be called an approximate local ideal point.*

Naturally, the (approximate) local ideal point weakly dominates all points in $\mathcal{S}$.

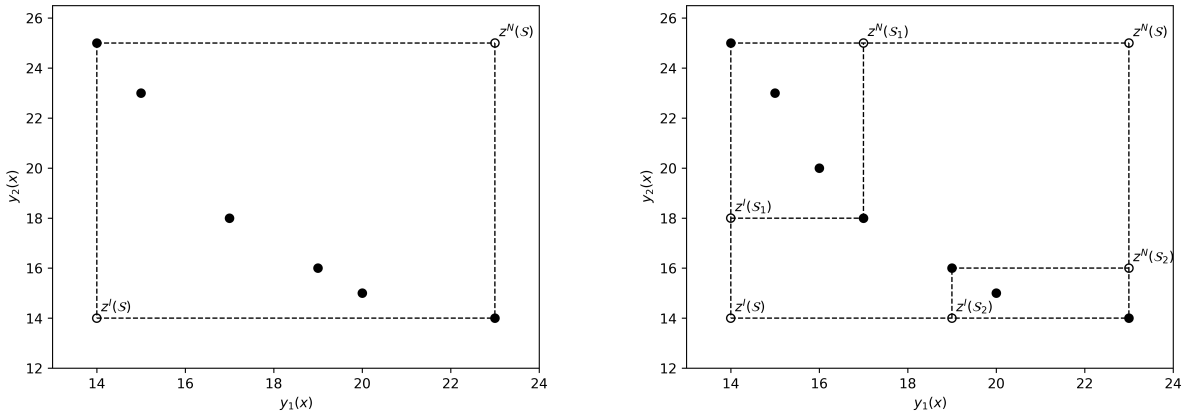
Definition 20. *The local nadir point of a subset $\mathcal{S} \subseteq Y_N$ denoted as $z^N(\mathcal{S})$ is the point in the objective space composed of the worst coordinates of all points belonging to $\mathcal{S}$, i.e., $z^N_k(\mathcal{S}) = \max_{y \in \mathcal{S}} y_k, \forall\, k \in \mathcal{P}$. A point $\widehat{z}^N(\mathcal{S})$ such that $\widehat{z}^N(\mathcal{S}) \preceq z^N(\mathcal{S})$ will be called an approximate local nadir point.*

Naturally, the (approximate) local nadir point is weakly dominated by all points in $\mathcal{S}$.

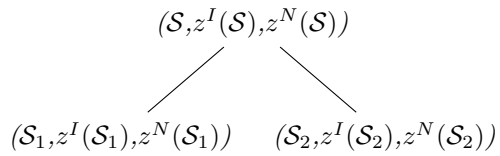
The primary concept behind the ND-Tree-based update method is to split the whole set of non-dominated points into subsets of points located close in the objective space, similarly to the clustering process in machine learning [87]. Each subset of points is represented by a local ideal point and a local nadir point, and thanks to different properties related to these points, it will be possible to avoid many comparisons when a new point is added to the Pareto archive.

This is illustrated through the following example, for a 2-objective problem.

Example 7. *Let's consider a set $\mathcal{S} = \{(14,25),(15,23),(17,18),(19,16),(20,15),(23,14)\}$ composed of 6 non-dominated points. The representation of $\mathcal{S}$ in the objective space is given in the figure below (on the left). We have also represented the local ideal point $z^I(\mathcal{S}) = (14,14)$ and the local nadir point $z^N(\mathcal{S}) = (23,25)$. At this time, the ND-Tree is only composed of one node, to which is associated a triplet equal to ($\mathcal{S}$,$z^I(\mathcal{S})$,$z^N(\mathcal{S})$). Let's consider now that a new non-dominated point, equal to $(16,20)$ must be added, and that the maximal size of the set of points associated to a node is equal to 6. As now the number of non-dominated points is equal to 7, we cannot associate the new point to the current node, and it is necessary to split the set of points. On the figure below (on the right) we have split the set of points into two subsets, $\mathcal{S}_1 = \{(14,25),(15,23),(16,20),(17,18)\}$ and $\mathcal{S}_2 = \{(19,16),(20,15),(23,14)\}$. Each subset has a local ideal and a local nadir point. We have that $z^I(\mathcal{S}_1) = (14,18)$, $z^N(\mathcal{S}_1) = (17,25)$, $z^I(\mathcal{S}_2) = (19,14)$ and $z^N(\mathcal{S}_2) = (23,16)$.*



The representation of the ND-Tree data structure is now the following:



with $S = S_1 \cup S_2$. As soon as a subset of points attains its predefined maximal size, the subset will be divided again into smaller subsets, and new nodes will be added to the ND-Tree.

More formally, the ND-Tree structure is defined in the subsequent way.

Definition 21. *The ND-Tree data structure is a tree with the following properties:*

1. *With each node n is associated a triplet constituted of a set of non-dominated points $\mathcal{S}(n)$, an approximate ideal point $\widehat{z}^I(\mathcal{S}(n))$ and an approximate nadir point $\widehat{z}^N(\mathcal{S}(n))$.*
2. *For each leaf node, $\mathcal{S}(n)$ is equal to a list $\mathcal{L}(n)$ of non-dominated points.*
3. *For each internal node n, $\mathcal{S}(n)$ is equal to the union of the disjoint sets associated with all the children of n.*
4. *If n' is a child of n, then $\widehat{z}^I(\mathcal{S}(n)) \succeq \widehat{z}^I(\mathcal{S}(n'))$ and $\widehat{z}^N(\mathcal{S}(n')) \succeq \widehat{z}^N(\mathcal{S}(n))$.*

We now present how the ND-Tree is used to update efficiently a Pareto archive. The core concept is to take advantage of the local ideal points and nadir points to limit the number of comparisons. Local ideal points and nadir points are used as follows. Consider a subset $\mathcal{S} \subseteq Y_N$ composed of mutually non-dominated points, limited by an approximate local ideal point $\widehat{z}^I(\mathcal{S})$ and an approximate local nadir points $\widehat{z}^N(\mathcal{S})$. In other words, all points in $\mathcal{S}$ are contained in the axes-parallel hyper-rectangle defined by $\widehat{z}^I(\mathcal{S})$ and $\widehat{z}^N(\mathcal{S})$. We now define the following simple properties that allow to compare a new candidate point y to the whole set $\mathcal{S}$. These properties are given by considering approximate ideal and nadir points, but of course, there are still valid for exact ideal and nadir points.

Property 1. *If y is weakly dominated by the approximate nadir point $\widehat{z}^N(\mathcal{S})$, then y is weakly dominated by each point in $\mathcal{S}$ and thus can be rejected.*

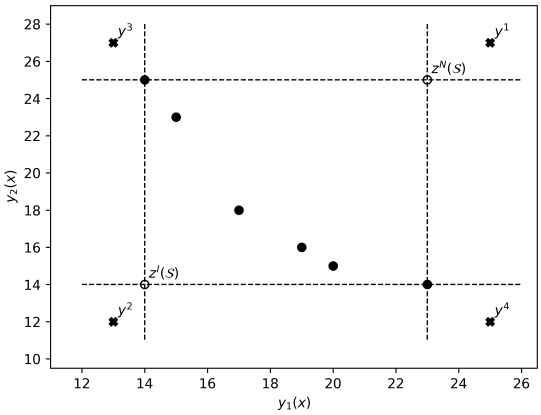
Property 2. *If y weakly dominates the approximate ideal point $\widehat{z}^I(\mathcal{S})$, then each point in $\mathcal{S}$ is weakly dominated by y and all points of $\mathcal{S}$ can be discarded.*

Proof. These two properties are a straightforward consequence of the transitivity of the weakly dominance relation. □

Property 3. *If y is non-dominated w.r.t. both the approximate nadir point $\widehat{z}^N(\mathcal{S})$ and approximate ideal point $\widehat{z}^I(\mathcal{S})$, then y is non-dominated w.r.t. each point in $\mathcal{S}$.*

Proof. If y is non-dominated w.r.t. $\widehat{z}^N(\mathcal{S})$ then there is at least one objective on which y is worse than $\widehat{z}^N(\mathcal{S})$ and thus worse than each point in $\mathcal{S}$ on the same objective. If y is non-dominated w.r.t. $\widehat{z}^I(\mathcal{S})$ then there is at least one objective on which y is better than $\widehat{z}^I(\mathcal{S})$ and thus better than each point in $\mathcal{S}$ on the same objective. So, there is at least one objective on which y is better and at least one objective on which y is worse than each point in $\mathcal{S}$. □

Example 8. *In the following figure, we illustrate the three properties in the bi-objective case. We consider the same set $\mathcal{S}$ as in Example 7 and a candidate point $y^1 = (25, 27)$. We see that y^1 is weakly dominated by the nadir point $z^N(\mathcal{S})$ and thus dominated by each point in $\mathcal{S}$; the point y^1 can thus be rejected (Property 1). Let's now consider the candidate point $y^2 = (13, 12)$. This point dominates the local point $z^I(\mathcal{S})$ and thus all points of $\mathcal{S}$ can be discarded (Property 2). Finally, if we consider the points $y^3 = (13, 27)$ or $y^4 = (25, 12)$, we see that there are both non-dominated with respect to $z^N(\mathcal{S})$ and $z^I(\mathcal{S})$, and thus non-dominated to all points in $\mathcal{S}$ (Property 3).*



If none of the above properties holds, i.e., y is neither weakly dominated by $\widehat{z}^N(\mathcal{S})$, does not weakly dominate $\widehat{z}^I(\mathcal{S})$, nor is non-dominated w.r.t. both $\widehat{z}^N(\mathcal{S})$ and $\widehat{z}^I(\mathcal{S})$, then three situations are possible:

1. y is non-dominated w.r.t. all points in $\mathcal{S}$.
2. y is weakly dominated by some points in $\mathcal{S}$.
3. y dominates some points in $\mathcal{S}$.

This can be shown by giving cases for each of the situations.

Consider for example a set $\mathcal{S} = \{(1,1,1),(0,2,2),(2,2,0)\}$. We have $z^I(\mathcal{S}) = (0,1,0)$ and $z^N(\mathcal{S}) = (2,2,2)$. Consider the three following points: $y^1 = (0,3,0)$, $y^2 = (1,1,2)$ and $y^3 = (1,1,0)$. For each of this point, it is easy to check that none of the three properties holds, i.e., none of the points is weakly dominated by $z^N(\mathcal{S})$, none of the points weakly dominates $z^I(\mathcal{S})$, and none of the points is non-dominated w.r.t both $z^N(\mathcal{S})$ and $z^I(\mathcal{S})$. The situation 1 occurs for y^1: y^1 is non-dominated w.r.t all points in $\mathcal{S}$. The situation 2 occurs for y^2: y^2 is dominated by (1,1,1) (and thus also weakly dominated) and the situation 3 occurs for y^3: y^3 dominates (2,2,0).

However, if one of the properties is satisfied by the new candidate point y, it allows comparing y to all points in the set $\mathcal{S}$, by just comparing it to the local ideal and nadir points of $\mathcal{S}$, without the need for further comparisons to individual points belonging to $\mathcal{S}$. Comparisons to individual points of $\mathcal{S}$ are only necessary if none of the three properties hold. Intuitively, the closer the approximate local ideal and nadir points, the more likely that one of the properties is satisfied. That is why it is interesting to split the whole set of non-dominated points into subsets of points located close in the objective space.

We now present the main principles of the ND-Tree-based update method. We initially aim to determine whether the new point y is weakly dominated or non-dominated w.r.t. all points in Y_N by going through the nodes of the ND-Tree and skipping children (and their subtrees) for which Property 3 holds. For a node n, if the new point is dominated by $\widehat{z}^N(\mathcal{S}(n))$ it is immediately rejected (Property 1). If $\widehat{z}^I(\mathcal{S}(n))$ is weakly dominated, the node and its whole subtree is deleted (Property 2). Otherwise, if $\widehat{z}^I(\mathcal{S}(n)) \succeq y$ or $y \succeq \widehat{z}^N(\mathcal{S}(n))$, the node needs to be analyzed. If n is an internal node, we call the algorithm recursively for each child. If n is a leaf node, y may be dominated by or dominate some points of the list $\mathcal{L}(n)$ associated to the node n and it is necessary to browse the whole list $\mathcal{L}(n)$. If a point dominating y is found, y is rejected, and if a point dominated by y is found, the point is deleted from $\mathcal{L}(n)$.

If after updating ND-Tree the new point y was found to be non-dominated, it is inserted by adding it to a close leaf. To find a proper leaf, we start from the root and always select a child with the closest distance to y. As a distance measure, we use the Euclidean distance to the *middle point*, i.e., a point lying in the middle of the line segment connecting the approximate ideal and the approximate nadir points.

Once we have reached a leaf node, we add the point y to the list $\mathcal{L}(n)$ of the node and possibly update the ideal and nadir points of the node n. However, if the size of $\mathcal{L}(n)$ becomes larger than the maximum allowed size for the list, we need to split the node into a predefined number of children. To create children that contain points that are more similar to each other than to those in other children, we use a simple clustering heuristic based on Euclidean distance [122].

The approximate local ideal and nadir points are updated only when a point is added. We do not update them when points are removed, since it is a more time-consuming operation. This is why we deal with

approximate local ideal and nadir points.

Computational complexity

We have shown that the ND-Tree-based update method has still a $\mathcal{O}(N)$ time complexity. Indeed, there are some particular cases where we need to compare the new point to each intermediate node of the tree. We are not aware of any result showing that the worst-case time complexity of any algorithm for the dynamic non-dominance problem may be lower than $\mathcal{O}(N)$ in terms of point comparisons. So, our method does not improve the worst-case complexity but according to our experiments performs significantly better in practical cases (see the following section).

Computational experiments

We have compared the performances of the ND-Tree-based update method (simply called ND-Tree in the following) with three other methods in two different cases:

1. Results for artificially generated sets which allow us to easily control the number of points in the sets and the quality of the points. We used convex, non-convex and clustered set of points, with 100 000 points.
2. Results for sets generated by a MOMH, namely MOEA/D [265] for the MO knapsack problem. MOEA/D was run for at least 100 000 iterations and the first 100 000 points generated by the algorithm were stored.

We have used the simple list, the sorted list (only for the bi-objective case), quad-tree, M-Front and ND-Tree to dynamically update the Pareto archive by considering the points of the different sets. For ND-Tree, we have used 20 as the maximum size of the list associated to a leaf and $p+1$ as the number of children (p is equal to the number of objectives). These values of the parameters were found to perform well in many cases.

ND-Tree performed the best in terms of CPU time for all artificial test sets with three and more objectives. In some cases the differences to other methods are of two orders of magnitude, and in some cases the difference to the second-best method is of one order of magnitude. ND-Tree also behaves very predictably, its execution time increasing slowly as the number of objectives and the number of non-dominated points increase. For bi-objective instances, the sorted list remains the best choice.

We give in Table 1 the results from the comparisons with instances generated by MOEA/D. The results confirm that the observations made for artificial sets also hold in the case of real sets. ND-Tree is the fastest method for three and more objectives. Note that in this table, it is the sorted list (and not the simple list) which is used for 2-objective instances.

p	$\lvert Y_N \rvert$	List	Quad-Tree	M-Front	ND-Tree
2	140	9	265	57	24
3	1789	529	793	211	89
4	5405	3011	2813	885	272
5	10126	6930	5136	2903	478
6	16074	12689	8615	7765	804

Table 1: Comparison of running times (in ms) of ND-Tree with the list (sorted list for 2 objectives), quad-tree and M-Front.

These good results have been recently confirmed by Fieldsend [90] in 2020. He compared ND-Tree with 7 data structures, including the list, quad-tree and M-Front. He used artificial sets using normal distribution for the objective vectors. He also studied the integration of the data structure into a MOMH, namely the PAES algorithm of Knowles and Corne [141], but rather than using a bounded archive, an unbounded archive was used. For both experiments, he found that ND-Tree generally performed better.

Recent improvements

- Very recently, Lang [153] has presented different techniques for improving the efficiency of the ND-Tree-based update method. The main idea is to rebuild the ND-tree at intervals, with new strategies for

updating the lower and upper bounds in each node. Indeed, one drawback of the ND-Tree structure is that the tree is not re-balanced during its use, and after certain iterations it could be very unbalanced, which could increase the number of comparisons required on the nodes. Lang showed that these improvements can lead to significant speedups over using the original method, in particular for higher dimensions.

- Nan et al. [189] have used ND-Tree inside a MO genetic algorithm and delays the update of the Pareto archive. The main idea is to reverse the order of solutions with respect to their generated time when updating the archive. The experimental results suggest that the ND-Tree approach assisted by the proposed reverse strategy is much faster than the original ND-Tree approach in obtaining the final archive.
- Fieldsend [91] have studied how to modify ND-Tree such that the structure becomes an active source of parent solutions to directly exploit during an optimization run for a genetic algorithm.
- In his PhD thesis, Cornu [60] have developed two new data structures for updating a Pareto archive:
  - a self-balancing binary search tree (AVL) dedicated to the bi-objective case, called AVL-Archive.
  - a self-balancing k-ary search tree, called NDR*-Archive.

  Contrary to ND-Tree, these two data structures have balanced methods, and they take advantage of a principle often present when storing data in an online environment: the temporal and spatial locality. Temporal locality refers to the reuse of specific data within a relatively small-time duration, while spatial locality refers to the use of data elements within relatively close storage locations. The two data structures perform really well and outperforms ND-Tree. Unfortunately, the results were not published outside the thesis and did not have the impact they should have had.

3 Lorenz dominance in multi-objective combinatorial optimization

In the preceding sections, we have considered the Pareto dominance relation to discriminate between the feasible solutions of a multi-objective problem. Despite this, the number of feasible solutions not dominated according to the Pareto dominance relation can be very large (in particular when the number of objectives is higher than two), which could make a final choice of one (or a few) solution(s) among the non-dominated ones difficult for a DM.

In this section, we will thus consider a more restrictive dominance relation than the Pareto dominance relation, the Lorenz dominance, which is particularly well suited to deal with multi-agent combinatorial optimization problems.

3.1 Lorenz dominance

In many real-life decision problems, there is not one single DM, but several DMs (or agents) whose preferences must be taken into account. Multi-agent combinatorial optimization problems deals with problems where several agents are involved. When the preferences are cardinal, the agents express their preferences over the alternatives through *utility functions*, each agent having her own utility function to maximize. Therefore, a solution is evaluated by a vector of utilities, where a component represents the utility of an agent for this solution.

Using the Pareto dominance to discriminate between the solutions proves insufficient, since a Pareto-optimal solution cannot be fair to all agents (an agent might be completely disadvantaged compared to the others), as shown in the following example.

Example 9. *The following table displays the utilities (score between 1 and 9) given by four agents (a_1, a_2, a_3 and a_4) to four items (i_1, i_2, i_3, i_4). We look for the best assignment possible of the items to the agents (an agent can receive only one item, and an item can be assigned to only one agent).*

	i_1	i_2	i_3	i_4
a_1	4	8	2	1
a_2	8	6	5	2
a_3	9	4	4	7
a_4	3	6	1	1

There are 24 assignments possible (4!). Let's consider that a feasible assignment is evaluated through a four-component vector u, where each component u_i corresponds to the utility of the item assigned to the agent a_i. Let's note a feasible assignment as a 4-tuple, where the value of the element at position i represents the index of the item assigned to the agent a_i.
Among all the feasible assignments, there are 8 Pareto-optimal assignments, given in the following table.

Solution	*Evaluation*
$x_1 = (1, 3, 4, 2)$	$(4, 5, 7, 6)$
$x_2 = (2, 1, 4, 3)$	$(8, 8, 7, 1)$
$x_3 = (2, 3, 1, 4)$	$(8, 5, 9, 1)$
$x_4 = (2, 3, 4, 1)$	$(8, 5, 7, 3)$
$x_5 = (3, 1, 4, 2)$	$(2, 8, 7, 6)$
$x_6 = (3, 2, 1, 4)$	$(2, 6, 9, 1)$
$x_7 = (3, 4, 1, 2)$	$(2, 2, 9, 6)$
$x_8 = (4, 3, 1, 2)$	$(1, 5, 9, 6)$

However, among these solutions there are many solutions that are not fair: solution x_8 gives the worst item possible for agent a_1, same for solution x_7 for agent a_2, and same for solutions x_2, x_3 and x_6 for agent a_4. We can therefore see that it would be necessary to use more restrictive relations that incorporate fairness.

Ensuring fairness between several agents reveals to be a tricky question. The notion of fairness has been widely investigated in economics and social choice and has lead to different definitions. In economics, some measures of inequality of outcome distributions have been proposed, as the *Gini index* [70] or the *Atkinson index* [11]. In welfare economics, one refers to social welfare functions to evaluate the relative goodness of the alternatives with respect to the individual utilities [221].

The classic *utilitarian* criterion consists in evaluating a solution according to the sum of the utilities of the agents. However, it is insensitive to the distribution of the total sum of the individual's utilities [222] and can provide unfair solutions since it allows compensations between strongly satisfied agents and poorly

satisfied ones. For example, a solution with a utility vector equal to $(200, 1)$ is better than a solution with the utility vector $(100, 100)$. The classic *egalitarian* criterion (or its lexicographical refinement) overcomes this drawback by evaluating a solution with respect to the utility of its least satisfied agent (which is equivalent to the *max min criterion*). However, by focusing only on the utility of one agent (the least satisfied), high quality solutions can be eliminated by this aggregation function. For example, a solution with a utility-vector $(100, 100, 100, 100)$ is better than a solution with a utility-vector $(99, 200, 200, 200)$, while this last solution is much better for three of the agents and just a little worse for the first agent. Some other aggregation functions, such as the *Ordered Weighting Average (OWA)* [261], enable to favor solutions for which the utilities of the agents are well-balanced, but they require an additional preferential information (appropriate weights).

An elegant refinement of the Pareto dominance to deal with fairness is the Lorenz dominance. 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. Roughly speaking, the Lorenz dominance enables to select all Pareto-optimal solutions that realize well-balanced compromises between the utilities of the agents, while not eliminating high-performance solutions. It encompasses the utilitarian and the egalitarian criterion. It has been used to characterize equitable solutions in multi-objective optimization [146, 147] and robust solutions in decision under uncertainty [200]. It has also been studied within the framework of convex-cone theory [264] in multi-objective programming [12], and in metaheuristics [49, 159].

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

Definition 22. *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., $y_{(1)} \geq y_{(2)} \geq \ldots \geq y_{(p)}$ in the case of minimization and $y_{(1)} \leq y_{(2)} \leq \ldots \leq y_{(p)}$ in the case of maximization).*

Definition 23. *Lorenz dominance relation: we say that a point $u = (u_1, ..., u_p) \in \mathbb{R}^p$* Lorenz dominates *a point $v = (v_1, ..., v_p) \in \mathbb{R}^p$ if, and only if $L(u) \succ_P L(v)$. We denote this relation by $u \succ_L v$.*

Definition 24. *Weakly Lorenz dominance relation: we say that a point $u = (u_1, ..., u_p) \in \mathbb{R}^p$* weakly Lorenz dominates *a point $v = (v_1, ..., v_p) \in \mathbb{R}^p$ if, and only if $L(u) \succeq_P L(v)$. We denote this relation by $u \succeq_L v$.*

Definition 25. *Lorenz-optimal solution: a feasible solution $x^* \in \mathcal{X}$ is called* Lorenz-optimal *if there is no other feasible solution $x \in \mathcal{X}$ such that $y(x) \succ_L y(x^*)$.*

The space in which the generalized Lorenz vectors of a solution x are represented is called the *Lorenz space*. The *Lorenz-optimal set* denoted by X_L contains all the Lorenz-optimal solutions. The image of the Lorenz-optimal set in objective space is denoted by Y_L.

Example 10. *In the following table, we indicate the Lorenz vector associated to each Pareto-optimal solution of the multi-agent assignment problem of Example 9. If we compare the Lorenz vectors with the Pareto dominance relation, we can easily check that only three are Pareto non-dominated (represented in bold). Therefore, only the corresponding solutions (x_1, x_2 and x_4) are Lorenz-optimal solutions. We see that the solution x_2 is the best for the sum of the utilities (the sum is given by the last component of the Lorenz vector). Solution x_1 is the best for the* max min *criterion (given by the first component of the Lorenz vector), while solution x_4 establishes a good compromise between these two criteria.*

Solution	*Evaluation*	*Lorenz vector*
$x_1 = (1, 3, 4, 2)$	$(4, 5, 7, 6)$	$(\mathbf{4}, \mathbf{9}, \mathbf{15}, \mathbf{22})$
$x_2 = (2, 1, 4, 3)$	$(8, 8, 7, 1)$	$(\mathbf{1}, \mathbf{8}, \mathbf{16}, \mathbf{24})$
$x_3 = (2, 3, 1, 4)$	$(8, 5, 9, 1)$	$(1, 6, 14, 23)$
$x_4 = (2, 3, 4, 1)$	$(8, 5, 7, 3)$	$(\mathbf{3}, \mathbf{8}, \mathbf{15}, \mathbf{23})$
$x_5 = (3, 1, 4, 2)$	$(2, 8, 7, 6)$	$(2, 8, 15, 23)$
$x_6 = (3, 2, 1, 4)$	$(2, 6, 9, 1)$	$(1, 3, 9, 18)$
$x_7 = (3, 4, 1, 2)$	$(2, 2, 9, 6)$	$(2, 4, 10, 19)$
$x_8 = (4, 3, 1, 2)$	$(1, 5, 9, 6)$	$(1, 6, 12, 21)$

We see thus through this example the interest of using the Lorenz dominance, since solutions presenting a good compromise between the utilitarian criterion and the egalitarian criterion are also generated.

We now illustrate, through a small example, the Lorenz dominance relation (in minimization), both in objective space and Lorenz space.

Example 11. *Let's consider the point $y = (6, 3)$. All the points Lorenz dominated by y are in the hatched area called "Lorenz worse" in the bi-objective space of Figure 1 (left part). The points that Lorenz dominate y are in the hatched area called "Lorenz better" in the same figure. To illustrate the Lorenz dominance, the symmetric point of y, the point (3,6), is also represented in the figure by a circle. The points that are not located in the hatched area are incomparable to y with Lorenz dominance (there are either better for $L_2(y(x))$ but have a worse value for $L_1(y(x))$ or there are worse for $L_1(y(x)$ but better for $L_2(y(x))$. 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). Note that in this space, we cannot have any point (z_1, z_2) such that $z_2 > 2 * z_1$ since for any point y we have $L_2(y) = y_1 + y_2 \leq 2 * \max(y_1, y_2) = 2 * L_1(y)$.*

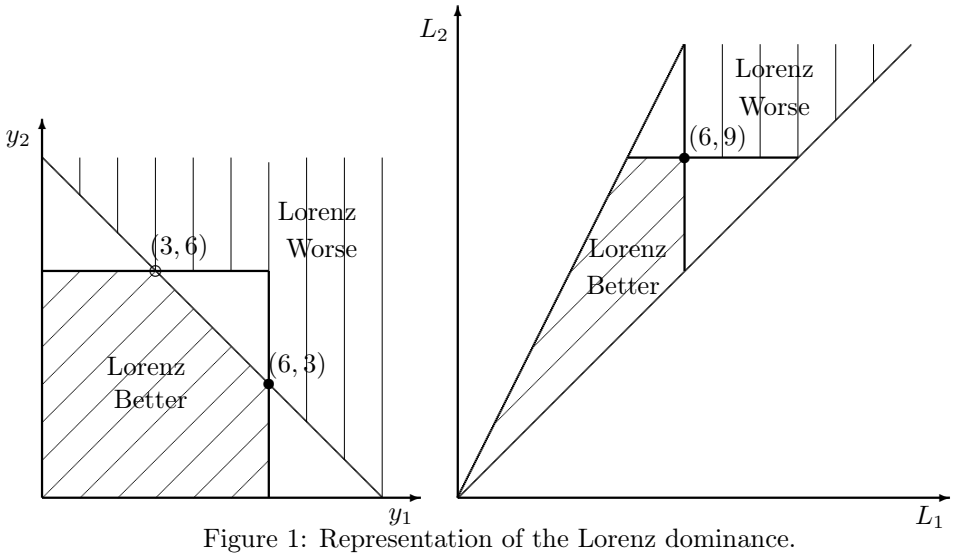


Figure 1: Representation of the Lorenz dominance.

The Lorenz dominance is closely related to the notion of *Pigou-Dalton transfers*. In social choice theory, a Pigou-Dalton transfer is an income transfer from a richer to a poorer person by an amount less than or equal to their initial income difference.

Definition 26. *Transfer principle [121]: let $y \in \mathbb{R}^p$ such that $y_i > y_j$ for some i, j. Then for all ε such that $0 \leq \varepsilon \leq y_i - y_j$, $y - \varepsilon \cdot e_i + \varepsilon\, e_j \succsim_L y$ where e_i (resp. e_j) is the vector whose i^{th} (resp. j^{th}) component equals 1, all others being 0.*

This principle means that for some cost-vector $y \in \mathbb{R}^p$ with $y_i > y_j$, slightly increasing y_j and decreasing y_i while preserving the mean of the costs would produce a better distribution of the costs, and consequently a more balanced solution. For example, the distribution of the vector $y = (20, 20)$ is better than the distribution of the vector $y' = (10, 30)$, since it can be obtained from y' by a transfer of size 10. This principle enables to compare vectors with the same mean. Note that using a similar transfer of size greater than 20 would increase the inequality between utilities distribution. This explains why the transfers must have a size $\varepsilon \leq y_i - y_j$.

The generalized Lorenz extension that we consider here enables to compare vectors with different means thanks to the *Pareto-monotonicity* property [229], which means that if a vector y^1 Pareto dominates another vector y^2 then y^1 Lorenz dominates y^2.

Definition 27. *Pareto-monotonicity: $\forall y^1, y^2 \in \mathbb{R}^p, y^1 \succ_P y^2 \Rightarrow y^1 \succ_L y^2$ (and $y^1 \succsim_P y^2 \Rightarrow y^1 \succsim_L y^2$).*

If we look again at Example 9, we can now explain why, e.g., x_3 is Lorenz dominated by x_2. We have that $y(x_2) = (8, 8, 7, 1) \succ_P (8, 7, 7, 1)$. Therefore, $y(x_2) \succ_L (8, 7, 7, 1)$ by the Pareto-monotinicity principle. Furthermore, we have that $(8, 7, 7, 1) \succ_L y(x_3) = (8, 5, 9, 1)$ by the transfer principle (transfer of 2 from the third agent to the second agent). By transitivity, we have that $y(x_2) \succ_L y(x_3)$ and x_3 is thus Lorenz dominated by x_2.

3.2 Methods for generating all Lorenz-optimal solutions

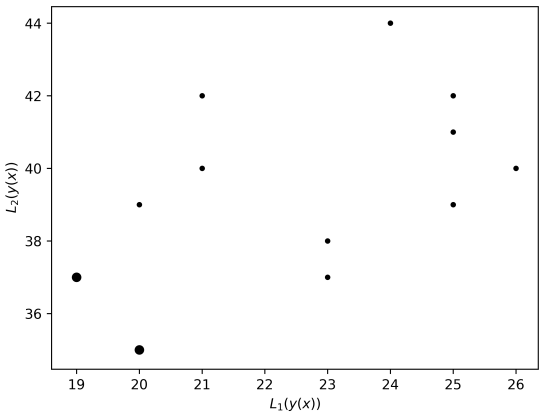
We are now interested in generating all Lorenz-optimal solutions of a MOCO problem (or a multi-agent problem where the utilities received by the agents represent the different objective functions). Following Definition 23, finding the Lorenz-optimal solutions to a MOCO problem is equivalent to identifying the Pareto-optimal solutions for the same MOCO problem, with the costs being represented by the generalized Lorenz vectors:

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

where $L\big(f_1(x),\ldots,f_p(x)\big) = (f_{(1)}(x), f_{(1)}(x)+f_{(2)}(x),\ldots,f_{(1)}(x)+f_{(2)}(x)+\ldots+f_{(p)}(x))$.
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 12. *In the following table, we give the Lorenz vectors associated to each solution of the bi-objective TSP of Example 1. We have also represented in the following figure (on the right) the Lorenz vectors in the Lorenz space. We see that there are only 2 Lorenz-optimal solutions, x^8 and x^6, with $y(x^8) = (19,18)$, $L(y(x^8)) = (19,37)$, $y(x^6) = (20,15)$ and $L(y(x^6) = (20,35)$. The solution x^8 is the best solution for $L_1(f(x))$ and x^6 is the best solution for $L_2(f(x))$.*

Solution	*Evaluation*	*Lorenz vector*
$x^1 = (1,2,3,4,5)$	$(25,16)$	$(25,41)$
$x^2 = (1,2,3,5,4)$	$(17,25)$	$(25,42)$
$x^3 = (1,2,4,3,5)$	$(21,21)$	$(21,42)$
$x^4 = (1,2,4,5,3)$	$(19,20)$	$(20,39)$
$x^5 = (1,2,5,3,4)$	$(14,25)$	$(25,39)$
$x^6 = (1,2,5,4,3)$	$(20,15)$	$(\mathbf{20},\mathbf{35})$
$x^7 = (1,3,2,4,5)$	$(26,14)$	$(26,40)$
$x^8 = (1,3,2,5,4)$	$(19,18)$	$(\mathbf{19},\mathbf{37})$
$x^9 = (1,3,4,2,5)$	$(23,14)$	$(23,37)$
$x^{10} = (1,3,5,2,4)$	$(15,23)$	$(23,38)$
$x^{11} = (1,4,2,3,5)$	$(20,24)$	$(24,44)$
$x^{12} = (1,4,3,2,5)$	$(21,19)$	$(21,40)$



The Lorenz-optimal solutions could be generated through a two-stage procedure that first generates all Pareto-optimal solutions and second selects only the Lorenz-optimal ones among them (since a Lorenz-optimal solution is necessarily a Pareto-optimal solution). But the efficiency of the two-stage procedure depends on the efficiency of the procedure that generates the Pareto-optimal solutions. Besides, the number of Lorenz-optimal solutions can be very small compared to the number of Pareto-optimal solutions, which would make the two-stage procedure inefficient. To give an example, we have used the exact Pareto-optimal set of the bi-objective knapsack instances produced by Bazgan et al. [16] to determine the proportions of Lorenz-optimal solutions among the Pareto-optimal solutions. Generally, the proportion is equal to around 5%, except for positively correlated instances, where the proportion can reach 25%. Indeed, as the two objectives are correlated, there are many solutions with close values, which gives many good candidates for fair solutions. Similar results have been obtained for the bi-objective TSP, for which the proportion was always less than 3%. It would therefore be interesting to develop methods that directly generate the Lorenz-optimal solutions.

3.2.1 Algorithmic issues

Intractability As for the Pareto dominance, we can study if there are MOCO problems for which the number of Lorenz-optimal solutions is exponential in the size of the instance. The bi-objective shortest path problem, the bi-objective spanning tree problem and the bi-agent Markov decision process problem have been proved intractable when looking for Lorenz-optimal solutions [200, 201]. As Ehrgott did in a Pareto optimization setting [81], one can show that even the following simple unconstrained problem is intractable when Lorenz dominance is considered. The multi-objective unconstrained (MOUC) problem is defined as follows:

$$\underset{x\in\{0,1\}^n}{\text{minimize}} \sum_{i=1}^{n} c_k^i x_i \qquad k\in\mathcal{P}$$

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 $\mathcal{Y} = \{(0,0),(1,-2),(2,-4),\ldots,(2^n-1,-2^{n+1}+2)\}$. If we represent the generalized Lorenz vectors of all $y\in\mathcal{Y}$, we obtain: $L(\mathcal{Y}) = \{(0,0),(1,-1),(2,-2),\ldots,(2^n-1,-2^n+1)\}$. We remark that all the generalized Lorenz vectors have the same sum

$(L_1 + L_2 = 0)$ 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 Pareto non-dominated in the Lorenz space. Then we have $Y_L = \mathcal{Y}$. 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 equal to $|\mathcal{X}| = 2^n$, we have thus $|X_L| = |Y_L| = 2^n$. □

NP-completeness The complexity of MOCO problems with Lorenz dominance is defined by the complexity of its decision version: "Given $d \in \mathbb{R}^p$, does there exist $x \in \mathcal{X}$ such that $(f_1(x), \ldots, f_p(x)) \succeq_L d$?". The decision version of the bi-objective shortest path problem and the bi-objective spanning tree problem has been proved NP-complete for Lorenz dominance [200]. Besides, one can easily show that the decision version of the MOUC problem is also NP-complete (see [98] for a proof).

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 [199, 201].

Example 13. *Let's consider an instance of the MOUC problem with $n = 3$, $k = 2$ and $c^1 = (1, -1)$, $c^2 = (-2, 2)$ and $c^3 = (2, -2)$. The cost vector c^1 Lorenz dominates the cost-vectors c^2 and c^3. However, the only Lorenz-optimal solution of this instance is $x = (0, 1, 1)$, with an evaluation equal to $(0, 0)$. We see that in this solution, $x_1 = 0$ while c^1 Lorenz dominates c^2 and c^3. Therefore, a greedy algorithm would first set $x_1 = 1$ but any solution with $x_1 = 1$ is Lorenz dominated by $x = (0, 1, 1)$.*

Example 13 shows that, even for the simple unconstrained problem, one cannot determine the Lorenz-optimal solutions from Lorenz non-dominated partial solutions.

3.2.2 Short 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. [200] in a robust optimization setting. This method works simply by computing the solutions in non-decreasing order of their sum using a k-best algorithm. Indeed, the sum of the objective values correspond to the last component of the Lorenz vector. Therefore, a solution minimizing the sum is weakly Lorenz Pareto-optimal. It is then necessary to define a valid stopping criterion for stopping the k-best algorithm. The stopping criterion is based on the following proposition: a vector $(y_1, \ldots, y_p)$ Lorenz dominates any vector $(y'_1, \ldots, y'_p)$ such that $\sum_{i=1}^{p} y'_i > p \cdot y_{(1)}$ where $y_{(1)} = \max(y_1, \ldots, y_p)$. Consequently, once the ranking algorithm generates a solution for which the stopping criterion is satisfied, the method can be stopped as all Lorenz-optimal 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. Moreover, effective k-best algorithms are only known for few combinatorial optimization problems [86].

ε-Constraint based method. This method, proposed by Baatar and Wiecek [12], is based on the classic ε-constraint procedure for generating the Pareto-optimal solutions (see Section 2.2.4) of MOCO problems. Each Lorenz-optimal solution is computed by solving two mathematical programs with appropriate constraints and objective functions. First, a solution that minimizes the sum of the objective values is generated. The best value obtained is called z_ϵ. Second, a solution that minimizes the Euclidean norm of the outcome vector is determined with the constraints that the sum of the objective values has to be equal to z_ϵ. It can be shown that this solution is Lorenz-optimal. This method works with any number of objectives, but solving such mathematical programs can be inefficient in practice. Moreover, the authors have not tested their methods on practical problems.

Dynamic programming based method. One cannot use directly a dynamic programming procedure to generate the Lorenz-optimal solutions to a MOCO problem (see Section 3.2.1). However, since dynamic programming can be used with Pareto dominance, and since Lorenz-optimal solutions are also Pareto-optimal, Perny and Spanjaard [199] have proposed to adapt a multi-objective dynamic programming based procedure to Lorenz optimization by adding a valid dominance rule.

3.2.3 Ordered weighted average

It is possible to transpose the notion of supported Pareto-optimal solutions to Lorenz dominance by applying the definitions in the Lorenz space. In that respect, we define *supported Lorenz-optimal solutions* as follows.

Definition 28. *Supported Lorenz-optimal solution: a solution x is a supported Lorenz-optimal solution if and only if there exists a strictly positive weight vector λ ($\lambda_k > 0, \forall\, k \in \mathcal{P}$) such that x is an optimal solution to the weighted sum single-objective problem defined on the Lorenz vector of $f(x)$: $\min_{x \in \mathcal{X}} \sum_{k=1}^{p} \lambda_k\, L_k(f(x))$, 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 [261]:

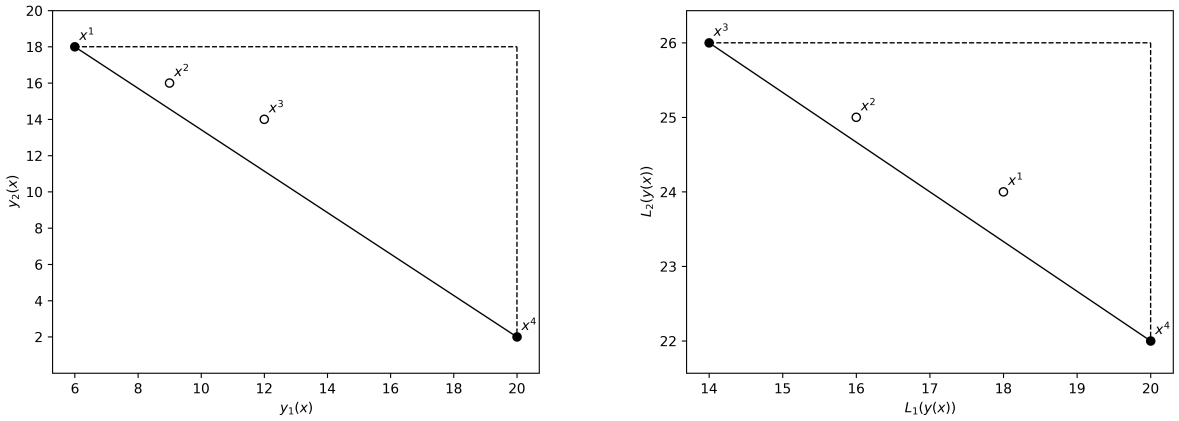
Definition 29. *Given a vector $y \in \mathbb{R}^p$ and a weighting vector $w \in [0,1]^p$, the* ordered weighted average (OWA) *of y with respect to w is defined by: $f^{owa}(y, w) = \sum_{k=1}^{p} w_k y_{(k)}$ where $y_{(1)} \geq \ldots \geq y_{(p)}$.*

Ogryczak [192] has shown that any solution minimizing an OWA function with strictly decreasing and strictly positive weights is Lorenz-optimal (and more precisely supported Lorenz-optimal). However, in general, there exist Lorenz-optimal solutions that do not optimize any OWA functions. We call these solutions *unsupported Lorenz-optimal* solutions. The geometric interpretation of the supported and unsupported Lorenz-optimal solutions is exactly the same as the geometric interpretation with Pareto dominance, but in the Lorenz space (with the Lorenz vectors) instead of the objective space.

Unfortunately, there are no trivial relations between the supported Lorenz-optimal solutions and supported Pareto-optimal solutions or between the unsupported Lorenz-optimal solutions and unsupported Pareto-optimal solutions, as illustrated in the following example.

Example 14. *Let's consider a MOCO problem presenting 4 Pareto-optimal 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 Pareto 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 Lorenz-optimal solutions. In this simple example, we have a supported Pareto-optimal solution that is not supported Lorenz-optimal solution (x^1), a solution that is unsupported in both space (x^2), a supported Lorenz-optimal solution that is an unsupported Pareto-optimal solution (x^3) and a solution that is supported in both spaces (x^4). The representation of these points in the objective space and the Lorenz space is given in the following figure.*

Objective space: **Lorenz space:**



3.3 New methods for two objectives

In this section, we present the main ideas behind the two methods we have developed for generating Lorenz-optimal solutions of bi-objective combinatorial optimization problems. The goal is to define methods that produce a set of Lorenz-optimal solutions such that to each Lorenz non-dominated point corresponds to at least one Lorenz-optimal solution in the set produced (i.e., a minimal complete set).

Straight adaptation of the two-phase method

The two-phase approach is a method used to generate Pareto-optimal solutions for MOCO problems with two objectives (see Section 2.2.6). We describe only at a high level how this method can be adapted in order to generate Lorenz-optimal solutions of bi-objective combinatorial optimization problems. 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 supported Lorenz-optimal solutions. 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 supported Lorenz-optimal solutions have been generated. The weight sets w used in the different OWA functions are defined by $w_k = \sum_{i=k}^{p} \lambda_i$ ($k \in \mathcal{P}$) from the weight sets λ defined in the Lorenz space and computed by the dichotomic search.

Phase 2: generation of all unsupported Lorenz-optimal solutions. The supported Lorenz points generated at Phase 1 are used to reduce the search space, since for bi-objective problems, unsupported Lorenz-optimal solutions are always located, in the Lorenz space, in the interior of the right triangle defined by two adjacent supported Lorenz-optimal points. 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 to be optimized is non-linear and therefore even generating only the supported Lorenz-optimal solutions could be computationally expensive. We propose in the next section another method where the optimization of OWA functions is avoided.

Supported Pareto-efficient solutions based method

Even though there is no trivial relation between the supported Pareto and Lorenz-optimal sets, we show in this section that, in the case of two objectives, there are interesting properties on the location of the Lorenz-optimal solutions with respect to the location of some supported Pareto-optimal solutions in the objective space.

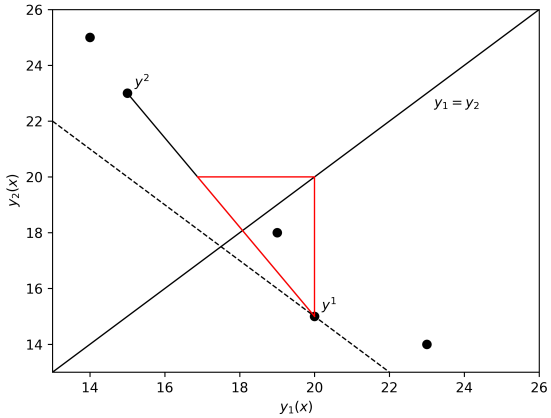
Before presenting the properties, let's introduce some notations. We denote by $O^i \subset \mathbb{R}^2_{\geq}$ the space in the positive orthant of the objective space where all the points y are such that $y_i \geq y_j$ for $j \neq i$. More formally, $O^1 = \{y \in \mathbb{R}^2_{\geq}, y_1 \geq y_2\}$ and $O^2 = \{y \in \mathbb{R}^2_{\geq}, y_2 \geq y_1\}$. The *bisector* is the line $y_1 = y_2$ in the objective space. Let's consider x^1 an optimal solution to leximin $\big(f_1(x) + f_2(x), \max(f_1(x), f_2(x))\big)$, i.e., a solution that minimizes the sum of both objective functions, and among all solutions that minimize the sum, it is a solution with a minimal value for the $\max(f_1(x), f_2(x))$ objective. This is a Lorenz-optimal solution since $L_1(f(x)) = \max(f_1(x), f_2(x))$ and $L_2(f(x)) = f_1(x) + f_2(x)$. Let's denote by y^1 the evaluation of x^1 in objective space. In this section, we suppose, w.l.o.g., that $y^1_1 > y^1_2$, i.e., $y^1 \in O^1$. Let x^2 be a supported Pareto-optimal solution corresponding to the point y^2 such that y^2 is in O^2 and minimizes the cost of the second objective among all the supported Pareto non-dominated points in O^2 (or if exceptionally there are no points in O^2, y^2 corresponds to the supported Pareto point in O^1 with the highest value for the second objective). 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 $y^i_1 \leq y_1 \leq y^j_1$.

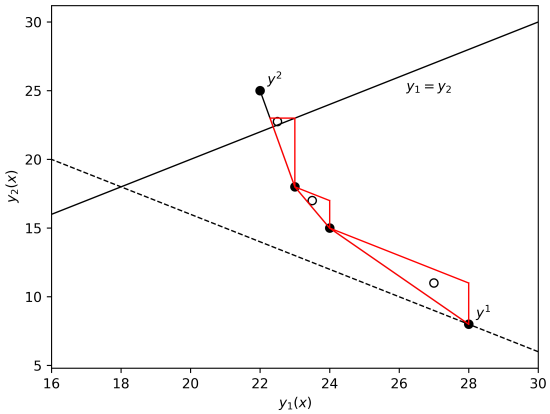
Proposition 2. *All supported Pareto-optimal solutions x such that $y(x) \in O_1$ and $y_2(x) > y^1_2$ are supported Lorenz-optimal [98].*

Proposition 3. *The image in objective space of all Lorenz-optimal solutions are between y^1 and y^2 [98].*

Example 15. *We illustrate these two properties in the following figures. In the figure on the left, we have represented the 5 Pareto non-dominated points of the instance of the bi-objective TSP of Example 1. We have also added the bisector (i.e., the line "$y_1 = y_2$"), and in dashed line, the line of equation "$y_1 + y_2 = 35$" (35 is equal to the minimal value for $f_1(x) + f_2(x)$ obtained by the point $(20, 15)$). The point y^1 corresponds to the point $(20, 15)$ and the point y^2 to the point $(15, 23)$. All the Lorenz non-dominated points should be located between these two points. Indeed, y^1 is the best solution for the sum of objectives and is Lorenz non-dominated. Therefore, new Lorenz non-dominated points can only be found on the left of y^1, near the bisector (i.e., points better for the $\max(f_1(x), f_2(x))$ objective). Beyond y^2, it is not possible to find Lorenz non-dominated points, since all the points will be worse for both objectives. The triangle, where new Lorenz non-dominated points could be found, has been represented with red lines. We see that there exists another Lorenz non-dominated point, the point $(19, 18)$ which corresponds to the best solution for the $\max(f_1(x), f_2(x))$ objective. On the right, we have represented another situation, where this time, there are more supported Pareto-optimal points located between y^1 and the bisector. The red triangles represent the zones where new Lorenz non-dominated*

points could be found (and we see that in each of this triangle there is a Pareto unsupported point which is Lorenz non-dominated). In total, there are 6 Lorenz non-dominated points (all the points except y^2).





We have used these properties to define a new two-phase method based on the computation of a subset of the supported Pareto-optimal solutions. This new method is called *method SP* in the sequel.
Phase 1. The first phase of the method SP consists in generating all the supported non-dominated points located between y^1 and y^2. From Property 2, we indeed know that all these solutions, except perhaps solution x^2 (with $f(x^2) = y^2$), are supported Lorenz-optimal. In order to do so, one first needs to find a solution x^1 that optimizes $\text{leximin}\big(f_1(x) + f_2(x), \max(f_1(x), f_2(x))\big)$. Then three cases can occur:

1. $f_1(x^1) = f_2(x^1)$: in this case, $f(x^1)$ is the only Lorenz non-dominated point of the problem, since it optimizes both L_2 and L_1: the method SP stops and returns the 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 $f(x^3)$ where x^3 optimizes $\min_{x \in \mathcal{X}}(f_1(x))$. Note that we only need to compute at most one point y^2 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 supported Pareto-optimal point y^2 is also stored.

Phase 2. Let Y be the set of points generated during Phase 1. To each point y of Y is associated an area in the objective space containing all the points that are not Lorenz dominated by y (called the *Lorenz non-dominance zone* of y). The intersection of all the Lorenz non-dominance zones of the points in Y defines the search zone to be explored in Phase 2 (see Example 15 for an illustration of these search zones).

The exploration of each of the zones consists in enumerating solutions with respect to the weighted sum with the weighting vector w defined such that two consecutive Pareto-optimal supported points have the same weighted sum. The enumeration is performed with a k-best algorithm. Once all the Lorenz non-dominance zones have been explored, the method SP can stop, as all the Lorenz non-dominated points have been detected.

3.4 Experimental results

The method has been applied to the bi-objective shortest path problem (BOSPP) and to the bi-objective set covering problem (BOSCP). For the BOSPP the method has been compared to two other methods: the ranking method and an extension of dynamic programming to Lorenz optimization (see Section 3.2.2). For the BOSCP, the method has been compared to the ranking method and to the Pareto method of Florios and Mavrotas [93], based on the ε-constraint (all Pareto-optimal solutions are then filtered to retain only the Lorenz-optimal solutions). The results showed that the proposed method was particularly efficient for "unbalanced" instances of the problems, that is, instances where the values of the two objectives do not follow the same distributions. Indeed, for these types of instances, the number of Lorenz-optimal solutions tends to be higher, that makes the SP method particularly effective compared to the ranking method. Indeed, the ranking method explores only one (large) zone of the objective space, while the SP method explores many (small) zones of the objective space. When the number of Lorenz-optimal solutions is high, the size of the zone to be explored by the ranking algorithm could be large, which could make its exploration very time-consuming. Full experimental results can be found in Galand and Lust [98].

4 Choquet integral in multi-objective combinatorial optimization

In this section we examine a general aggregation function, the Choquet integral, and its use in MOCO problems.

4.1 Presentation

The Choquet integral [52] is one of the most powerful tools in multicriteria decision-making [109, 112]. A Choquet integral can be seen as an integral on a non-additive measure (also called a capacity or a fuzzy measure). It presents extremely wide expressive capabilities and can model many specific aggregation operators, including, but not limited to, the WS, the minimum, the maximum, all the statistic quantiles, the OWA operator [261], the weighted OWA operator [248], etc.

However, this high expressiveness capability has a price: while the definition of a simple WS operator with p criteria requires $(p-1)$ parameters, the definition of the Choquet integral with p criteria requires the setting of (2^p-2) values, which can be a problem even for low values of p.

Many approaches have been studied to identify the parameters of the Choquet integral [111]. Generally, questions are asked to the DM and the information obtained is represented by linear constraints over the set of parameters. An optimization problem is then solved in order to find a set of parameters that minimizes the error according to the information given by the DM.

The approach considered in this section is quite different: we will not try to identify the parameters of the Choquet integral but given a set of solutions evaluated according to p criteria we will compute the solutions that are optimal for at least one parameter set of the Choquet integral. Therefore, the parameters of the Choquet integral will not have to be determined. Instead, a set of solutions of smaller size compared to the set of Pareto-optimal solutions (which can be very huge in the case of multi-objective or multi-agent combinatorial problems, as seen in the preceding sections) will be presented to the DM. Each solution proposed will have interesting properties since they optimize the Choquet integral for at least one set of parameters. Also, by computing all the Choquet-optimal solutions, all the solutions that optimize one of the operators that the Choquet integral can model will be generated.

The first application is multi-criteria decision-making [107]: the DM needs to choose an alternative among a set of alternatives; each alternative being evaluated according to a set of p criteria. No alternative Pareto dominates another, and therefore no alternative can be a priori rejected. However, if we plan to use the Choquet integral in order to select the best alternative according to the preferences of the DM, we can first generate the solutions that are optimal for at least one Choquet integral. This can be done in the absence of the DM. In the end, a set smaller than the Pareto-optimal set could be proposed.

Another application of the method is in MOCO. To solve a MOCO problem, three different approaches are usually followed. In the *a posteriori* approach, all the Pareto-optimal solutions are first generated (see Section 2). Then the DM is free to choose among all Pareto-optimal solutions the one that corresponds the best to her preferences. Another possibility, called the *a priori* approach, is to first ask the DM what are her preferences among all the objectives and to compute an aggregation function with specified parameters [112]. The aggregation function is then optimized and at the end, only one solution is generally proposed to the DM. A last possibility is to *interact* with the DM along the process of generation of the solutions [193] (see next Section).

We study in this section an original approach, between the *a posteriori* approach and the *a priori* approach, that consists in trying to find the set of solutions that are potentially optimal for at least one set of parameters of an aggregation function, and more specifically in this section the Choquet integral.

Some papers already deal with the optimization of the Choquet integral of MOCO problems [95, 100, 101] but only when the Choquet integral is completely defined. Galand et al. [101] have developed a method to optimize a Choquet integral for multi-objective spanning trees problems and multi-objective knapsack problems. They present a condition (named preference for interior points) that characterizes capacities favoring well-balanced solutions. For both problems studied, they introduce a linear bound of the Choquet integral and propose a branch and bound algorithm using this bound. Multi-objective shortest path problems have also been investigated by the same authors [100]. They use a branch and bound algorithm and an enumeration algorithm, based on a ranking approach. Fouchal et al. [95] study the same problem: generating a Choquet-optimal solution given a capacity. They apply their algorithm to the multi-objective shortest path problem. They introduce Choquet dominance rules that they integrate within the label setting algorithm of Martins [180], based on dynamic programming.

4.2 Definitions

The Choquet integral has been introduced by Choquet [52] in 1953 and has been intensively studied, particularly in the field of multicriteria decision analysis, by several authors (see [107, 109, 112] for a review).

We first define the notion of capacity, on which the Choquet integral is based.

Definition 30. *A capacity is a set function* v: $2^{\mathcal{P}} \rightarrow [0,1]$ *such that:*

- $v(\emptyset) = 0$, $v(\mathcal{P}) = 1$ *(boundary conditions)*
- $\forall \mathcal{A}, \mathcal{B} \in 2^{\mathcal{P}}$, $\mathcal{A} \subseteq \mathcal{B} \Rightarrow v(\mathcal{A}) \leq v(\mathcal{B})$ *(monotonicity conditions)*

Therefore, for each subset of objectives $\mathcal{A} \subseteq \mathcal{P}$, $v(\mathcal{A})$ represents the importance of the coalition $\mathcal{A}$.

Definition 31. *The Choquet integral of a vector* $y \in \mathbb{R}^p$ *with respect to a capacity* v *is defined by:*

$$\begin{aligned} f_v^C(y) &= \sum_{i=1}^{p} \big(v(Y_{(i)}) - v(Y_{(i+1)})\big) y_{(i)} \\ &= \sum_{i=1}^{p} (y_{(i)} - y_{(i-1)}) v(Y_{(i)}) \end{aligned}$$

where $(y_{(1)}, \ldots, y_{(p)})$ *is a permutation of the components of* y *such that* $0 = y_{(0)} \leq y_{(1)} \leq \ldots \leq y_{(p)}$ *and* $Y_{(i)} = \{j \in \mathcal{P}, y_j \geq y_{(i)}\} = \{(i), (i+1), \ldots, (p)\}$ *for* $0 \leq i \leq p$ *and* $Y_{(p+1)} = \emptyset$.

Example 16. *Let's consider the following four alternatives for choosing a bike, evaluated according to three criteria (speed, weight, comfort) on a scale going from 0 (the worst value) to 20 (the best value).*

	speed	*weight*	*comfort*
Road bike	17	18	7
Mountain bike	11	9	15
Gravel bike	14	14	10
City bike	9	8	17

Let's consider that the DM desires a fast bike, not too heavy and with a minimum of comfort. Let's suppose that the following capacity v *has been elicitated:* $v(\{1\}) = 0.4, v(\{2\}) = 0.4, v(\{3\}) = 0.2, v(\{1,2\}) = 0.4, v(\{1,3\}) = 0.6, v(\{2,3\}) = 0.9$. *We have that the Choquet integral associated to the road bike is equal to* $7 + (17-7) * v(\{1,2\}) + (18-17) * v(\{2\}) = 7 + 10 * 0.4 + 0.4 = 11.4$, *that to the mountain bike is equal to* $9 + (11-9) * v(\{1,3\}) + (15-11) * v(\{3\}) = 9 + 2 * 0.6 + 4 * 0.2 = 11$, *that to the gravel bike is equal to* $10 + (14-10) * v(\{1,2\}) + (14-14) * v(\{1\}) = 11.6$ *and that to the city bike is equal to* $8 + (9-8) * v(\{1,3\}) + (17-9) * v(\{3\}) = 10.2$. *According to this capacity, the gravel bike is the best choice. Let's notice that if a WS was used, the gravel bike would never have been the first choice. Indeed, let us consider the weight set* (w_1,w_2,w_3) *with* $w_3 = (1 - w_1 - w_2)$ *and* $w_1, w_2, w_3 \geq 0$. *To be better than the road bike for the WS, it can be easily shown that the following constraint on the weights has to be satisfied:* $6w_1 + 7w_2 \leq 3$. *And to be better than the mountain bike, we have this constraint:* $8w_1 + 10w_2 \geq 5$. *It is easy to check that there are no positive values for* w_1 *and* w_2 *that respect these two constraints.*

We can also define the Choquet integral through the Möbius representation [214] of the capacity. Any set function v: $2^{\mathcal{P}} \rightarrow [0,1]$ can be uniquely expressed in terms of its Möbius representation by:

$$v(\mathcal{A}) = \sum_{\mathcal{B} \subseteq \mathcal{A}} m_v(\mathcal{B}) \quad \forall \mathcal{A} \subseteq \mathcal{P}$$

where the set function $m_v : 2^{\mathcal{P}} \rightarrow \mathbb{R}$ is called the Möbius transform or Möbius representation of v and is given by

$$m_v(\mathcal{A}) = \sum_{\mathcal{B} \subseteq \mathcal{A}} (-1)^{(a-b)} v(\mathcal{B}) \quad \forall \mathcal{A} \subseteq \mathcal{P}$$

where a and b are the cardinals of $\mathcal{A}$ and $\mathcal{B}$.

A set of 2^p coefficients $m_v(\mathcal{A})$ ($\mathcal{A} \subseteq \mathcal{P}$) corresponds to a capacity if it satisfies the boundary and monotonicity conditions [51]:

1. $m_v(\emptyset) = 0$, $\sum_{\mathcal{A} \subseteq \mathcal{P}} m_v(\mathcal{A}) = 1$

2. $\sum_{\mathcal{B}\subseteq\mathcal{A},\ i\in\mathcal{B}} m_v(\mathcal{B}) \geq 0 \quad \forall \mathcal{A} \subseteq \mathcal{P}, \forall i \in \mathcal{P}$

We can now write the Choquet integral using the Möbius coefficients. The Choquet integral of a vector $y \in \mathbb{R}^p$ with respect to a capacity v is defined as follows:

$$f_v^C(y) \quad = \quad \sum_{\mathcal{A}\subseteq\mathcal{P}} m_v(\mathcal{A}) \min_{i\in\mathcal{A}} y_i$$

The Choquet integral is a powerful tool but suffers from a high number of parameters (2^p), that are difficult to interpret. However, there are some measures that have been developed to get a better understanding of the particular values of a capacity. For example, given a capacity, the Shapley index has been proposed to summarize the importance of an individual criterion [262]. The interaction index can summarize the interactions between different criteria [111]. Moreover, they are many tools that have been developed for capacity identification in the context of the Choquet integral [110]. Note that our goal is not to elicitate a capacity for a Choquet integral, but to generate a subset of solutions such that each solution is optimal for at least one Choquet integral.

4.3 Characterization of Choquet-optimal solutions

We define now more precisely the notion of Choquet-optimal solution and Choquet-optimal set. Let's consider $\mathcal{X}$ representing a set of alternatives (coming from a multicriteria decision-making problem) or a set of feasible solutions (coming from a MOCO problem). All criteria have to be maximized. Let $\mathcal{V}$ be the collection of all capacities.

Definition 32. *A Choquet-optimal solution is a feasible solution $x_C \in \mathcal{X}$ such that:*

$$\exists v \in \mathcal{V} : x_C \in \operatorname*{argmax}_{x\in\mathcal{X}} f_v^C(y(x))$$

Definition 33. *A Choquet-optimal set X_C is defined as follows:*

$$X_C = \bigcup_{v\in\mathcal{V}} \operatorname*{argmax}_{x\in X} f_v^C(y(x))$$

As the Choquet integral is an increasing function of its arguments, we have that:

Proposition 4. $x_C \in X_C \Rightarrow x_C \in X_{wE}$.

The image of a Choquet-optimal solution in objective space is called a Choquet-optimal point, denoted by y_C, and the set of all Choquet-optimal points is denoted by Y_C. Note that Choquet-optimal solutions are sometimes called *potentially* Choquet-optimal solutions [42], in particular in the context of interactive methods, where Choquet-optimal solutions are good candidates to be the preferred solution of the DM at the end of the procedure.

A simple way to generate all Choquet-optimal solutions would be to generate the set $\mathcal{V}$ and for each capacity $v \in \mathcal{V}$ to look for the best solution for the Choquet integral. However, the size of $\mathcal{V}$ is infinite, and moreover, optimizing a Choquet integral is not trivial in the case of MOCO problems since it is a non-linear function. Therefore, we have proposed an original method to generate all Choquet-optimal solutions, which is based on WS optimal solutions, i.e., solutions that optimize a WS (with positive weights). The main idea is the following: for a set of points whose components are in the same order, the Choquet integral reduces to a simple WS. Indeed, in this case $f_v^C(y) = \sum_{i=1}^p \big(v(Y_i) - v(Y_{i+1})\big)y_i$ and as v is monotonic for the inclusion, we have $(v(Y_i) - v(Y_{i-1}) \geq 0)$. Therefore, to be a Choquet-optimal solution, it is necessary to optimize a WS in the subspace where the solution is located (e.g., $f_1(x) \geq f_2(x) \geq \ldots \geq f_p(x)$). However, it is not sufficient. For example, let us consider the following set of four alternatives evaluated according to two criteria (to be maximized), i.e., $\mathcal{Y} = \{y^1 = (1, 10), y^2 = (2, 7), y^3 = (6, 5), y^4 = (10, 1)\}$. We have that y^1 and y^2 are both optimal points for the WS if one restrict to the subspace $(f_2(x) \geq f_1(x))$ (since there are only two points in this subspace). However, the point y^2 is not Choquet-optimal: indeed to be Choquet-optimal, we need that $f_v^C(y^2) \geq f_v^C(y^1)$, i.e., $v_2 \leq 0.25$, and $f_v^C(y^2) \geq f_v^C(y^3)$, i.e., $5v_2 \geq 3 + v_1$, which is impossible if $v_2 \leq 0.25$.

More formally, let σ be a permutation on $\mathcal{P}$ and Σ the set of all permutations on $\mathcal{P}$. Let O_σ be the subset of points $y \in \mathbb{R}^p$ such that $y \in O_\sigma \iff y_{\sigma_1} \geq y_{\sigma_2} \geq \ldots \geq y_{\sigma_p}$. Let p_{O_σ} be the following application:

$$p_{O_\sigma} : \mathbb{R}^p \to \mathbb{R}^p, (p_{O_\sigma}(y))_{\sigma_i} = (\min(y_{\sigma_1}, \ldots, y_{\sigma_i})), \forall i \in \mathcal{P}$$

For example, if $p = 3$, for the permutation (2,3,1), we have:

$$p_{O_\sigma}(y) = \big(\min(y_2, y_3, y_1), y_2, \min(y_2, y_3)\big)$$

We denote by $\mathcal{P}_{O_\sigma}(\mathcal{Y})$ the set containing the points obtained by applying the application $p_{O_\sigma}(y)$ to all the points $y \in \mathcal{Y}$. As $(p_{O_\sigma}(y))_{\sigma_1} \geq (p_{O_\sigma}(y))_{\sigma_2} \geq \ldots \geq (p_{O_\sigma}(y))_{\sigma_p}$, we have $\mathcal{P}_{O_\sigma}(\mathcal{Y}) \subseteq O_\sigma$.

Then the new characterization of the Choquet-optimal points that we have proposed is the following.

Proposition 5.

$$Y_C = \bigcup_{\sigma \in \Sigma} \mathcal{Y} \cap WS(\mathcal{P}_{O_\sigma}(\mathcal{Y}))$$

where $WS(\mathcal{P}_{O_\sigma}(\mathcal{Y})$ designs the set of WS-optimal points of the set $\mathcal{P}_{O_\sigma}(\mathcal{Y})$ [173].

This proposition characterizes the points that are Choquet-optimal points as being the points that optimize a WS in the subspace O_σ, where all projections of the points $y \in \mathcal{Y}$ with $p_{O_\sigma}(y)$ have been included.

Example 17. *If we go back to Example 16, we can now easily show that all alternatives are Choquet-optimal. We first determine if the point* $(17, 18, 7)$ *is Choquet-optimal. The permutation associated to this point is* $\sigma = (2, 1, 3)$ *and so* $p_{O_\sigma}(y) = \big(\min(y_1, y_2), y_2, \min(y_1, y_2, y_3)\big)$*. By applying the application to all the other points, we obtain* $\mathcal{P}_{O_\sigma}(\mathcal{Y}) = \{(17, 18, 7), (9, 9, 9), (14, 14, 10), (8, 8, 8)\}$*. We clearly see that the point* $(17, 18, 7)$ *optimizes a WS in this set, as it has the best values for the first and second objective.*

For the points $(11, 9, 15)$ *and* $(9, 8, 17)$*, we have the following permutation and projections (they both optimize a WS in* $\mathcal{P}_{O_\sigma}(\mathcal{Y})$ *and are thus Choquet-optimal):*

$$\begin{array}{ccc||ccc} 17 & 18 & 7 & 7 & 7 & 17 \\ \mathbf{11} & \mathbf{9} & \mathbf{15} & \mathbf{11} & \mathbf{9} & \mathbf{15} \\ 14 & 14 & 10 & 10 & 10 & 10 \\ \mathbf{9} & \mathbf{8} & \mathbf{17} & \mathbf{9} & \mathbf{8} & \mathbf{17} \end{array} \quad \sigma = (3, 1, 2),\ p_{O_\sigma}(y) = \big(\min(y_1, y_3), \min(y_1, y_2, y_3), y_3\big)$$

For the point $(14, 14, 10)$*, we have the following permutation and projections:*

$$\begin{array}{ccc||ccc} 17 & 18 & 7 & 17 & 17 & 7 \\ 11 & 9 & 15 & 11 & 9 & 9 \\ \mathbf{14} & \mathbf{14} & \mathbf{10} & \mathbf{14} & \mathbf{14} & \mathbf{10} \\ 9 & 8 & 17 & 9 & 8 & 8 \end{array} \quad \sigma = (1, 2, 3),\ p_{O_\sigma}(y) = \big(y_1, \min(y_1, y_2), \min(y_1, y_2, y_3)\big)$$

4.4 Generation of Choquet-optimal solutions

From Proposition 5 we have deduced an algorithm to generate the set X_C containing all the Choquet-optimal solutions of a MOCO problem (with p objective functions to be maximized). The principle is the following.

For all the permutations σ on $\mathcal{P}$, we have to:

1. Determine the projections with the application p_{O_σ}. However, among the projections, only the Pareto non-dominated points are interesting (since if a point is Pareto-dominated, its WS is inferior to the WS of at least another point). Therefore, to determine the projections, the following MOCO problem (called P_σ, with p objective functions to maximize) has to be solved:

$$\max_{x \in \mathcal{X} \setminus \mathcal{X}_\sigma} (f_{\sigma_1}(x), \min(f_{\sigma_1}(x), f_{\sigma_2}(x)), \ldots, \min(f_{\sigma_1}(x), f_{\sigma_2}(x), \ldots, f_{\sigma_p}(x)))$$

   where $\mathcal{X}_\sigma$ is the set such that $x \in \mathcal{X}_\sigma \iff f_{\sigma_1}(x) \geq f_{\sigma_2}(x) \geq \ldots \geq f_{\sigma_p}(x)$.

2. Generate all WS-optimal solutions in $\mathcal{X}_\sigma$, while adding the points obtained from P_σ.

The main lines of the algorithm are given in Algorithm 4.

Example 18. *Let's consider a knapsack problem with two objective functions* $f_1(x)$ *and* $f_2(x)$ *to maximize, and 12 feasible solutions, evaluated as follows:* $y(x^1) = (29, 0)$*,* $y(x^2) = (28, 1)$*,* $y(x^3) = (27, 4)$*,* $y(x^4) = (24, 5)$*,* $y(x^5) = (23, 6)$*,* $y(x^6) = (21, 8)$*,* $y(x^7) = (17, 9)$*,* $y(x^8) = (15, 10)$*,* $y(x^9) = (13, 11)$*,* $y(x^{10}) = (10, 12)$*,* $y(x^{11}) = (9, 13)$*,* $y(x^{12}) = (0, 14)$*. The representation of the points in the objective space in given in Figure 2 (left part). We remark that all feasible solutions are Pareto-optimal. To determine*

the Choquet-optimal solutions, only two permutations need to be examined: $\sigma_1 = (1,2)$ *and* $\sigma_2 = (2,1)$*. Let's first consider the permutation* σ_1*. The following MOCO problem* P_{σ_1}*, in* $x \in \mathcal{X}\backslash\mathcal{X}_{\sigma_1}$*, needs to be solved:* $\max_{x\in\mathcal{X}\backslash\mathcal{X}_{\sigma_1}} (f_1(x), \min(f_1(x), f_2(x)))$*. As* $\mathcal{X}\backslash\mathcal{X}_{\sigma_1}$ *is equal to* $\mathcal{X}_{\sigma_2}$*, the problem* P_{σ_1} *simply boils down to the following single-objective problem:* $\max_{x\in\mathcal{X}_{\sigma_2}} f_1(x)$ *(as* $x \in \mathcal{X}_{\sigma_2} \Leftrightarrow f_2(x) \geq f_1(x)$*). We obtain the solution* x^{10}*, with* $f(x^{10}) = (10,12)$*. We apply* $p_{O_{\sigma_1}}$ *on* $(10,12)$ *and we obtain* $p_{O_{\sigma_1}}(10,12) = (10,10)$*. Then we generate all WS in* $\mathcal{X}_{\sigma_1}$*, with the additional point* $(10,10)$*. We see on Figure 2 (right part, blue points) that there are 4 WS-optimal points, i.e.,* $y(x^1) = (29,0)$*,* $y(x^3) = (27,4)$*,* $y(x^6) = (21,8)$ *and* $y(x^9) = (13,11)$*. Let's now consider the permutation* σ_2*. The following MOCO problem* P_{σ_2}*, in* $x \in \mathcal{X}\backslash\mathcal{X}_{\sigma_2}$*, needs to be solved:* $\max_{x\in\mathcal{X}\backslash\mathcal{X}_{\sigma_2}} (f_2(x), \min(f_1(x), f_2(x)))$*. As* $\mathcal{X}\backslash\mathcal{X}_{\sigma_2}$ *is equal to* $\mathcal{X}_{\sigma_1}$*, the problem* P_{σ_2} *simply boils down to the following single-objective problem:* $\max_{x\in\mathcal{X}_{\sigma_1}} f_2(x)$*. We obtain the solution* x^9*, with* $y(x^9) = (13,11)$*. By applying* $p_{O_{\sigma_2}}$ *on this point, we find the point* $(11,11)$*. Then we generate all WS in* $\mathcal{X}_{\sigma_2}$*, with the additional point* $(11,11)$*. We see on Figure 2 (right part, red points) that there are 3 WS-optimal points, i.e.,* $y(x^{10}) = (10,12)$*,* $y(x^{11}) = (9,13)$ *and* $y(x^{12}) = (0,14)$*. On this example, we have thus two Choquet-optimal points that are not WS-optimal points:* $y(x^{10}) = (10,12)$ *and* $y(x^9) = (13,11)$*. Note that* x^9 *is OWA-optimal (as this solution is optimal for* $\max\min(f_1(x), f_2(x))$*) but the solution* x^{10} *is not OWA-optimal (as the point* $(10,12)$ *is Lorenz dominated by the point* $(13,11)$*).*

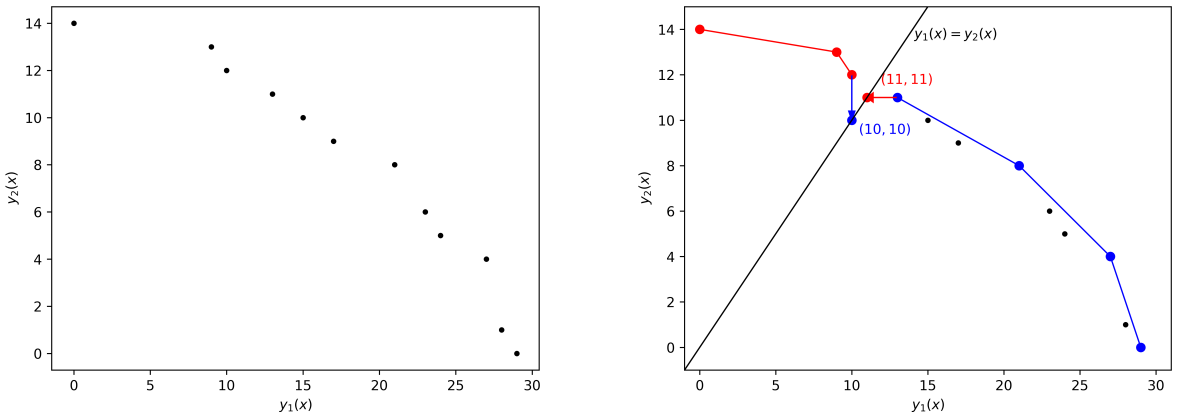


Figure 2: Pareto front and Choquet-optimal solutions of a bi-objective knapsack problem.

Algorithm 4 `Generation of` X_C

IN ↓: a MOCO problem with p objectives to be maximized
OUT ↑: the set X_C
Let σ be a permutation on $\mathcal{P}$, and Σ the set of all permutations
Let $\mathcal{X}_\sigma$ be the set such that $x \in \mathcal{X}_\sigma \iff f_{\sigma_1}(x) \geq f_{\sigma_2}(x) \geq \ldots \geq f_{\sigma_p}(x)$
$X_C \leftarrow \{\}$
for all $\sigma \in \Sigma$ **do**
  --| Determination of the projections:
  Solve the following MOCO problem, called P_σ, in $x \in \mathcal{X}\backslash\mathcal{X}_\sigma$:
  $\max_{x\in\mathcal{X}\backslash\mathcal{X}_\sigma} (f_{\sigma_1}(x), \min(f_{\sigma_1}(x), f_{\sigma_2}(x)), \ldots, \min(f_{\sigma_1}(x), f_{\sigma_2}(x), \ldots, f_{\sigma_p}(x)))$
  Let Y_{P_σ} the Pareto non-dominated points obtained from solving (P_σ)
  Generate the set of all WS-optimal solutions in $\mathcal{X}_\sigma$ (denoted by X_{ws_σ}), while adding the points obtained from $\mathcal{P}_{O_\sigma}(Y_{P_\sigma})$.
  $X_C \leftarrow X_C \cup \mathcal{X}_{ws_\sigma}$
return X_C

Numerical experiments

We have applied the Algorithm 4 to generate the Choquet-optimal solutions of instances of the bi-objective knapsack problem (bKP) and the bi-objective spanning tree problem (bSTP). For the bKP, the CPU time needed to generate all Choquet-optimal solutions was just a little bit higher than the CPU needed to generate all WS-optimal solutions, and much lower than the generation of all Pareto-optimal solutions. For the bSTP, the running time of Algorithm 4 is much higher than the generation of WS-optimal solutions. Indeed, while WS problems can be solved in polynomial time (with the Prim algorithm [206] for example), for the generation of the Choquet-optimal solutions we need to solve the spanning tree problem with an additional constraint ($f_1(x) \geq f_2(x)$ or $f_2(x) \geq f_1(x)$), which is $\mathcal{NP}$-Hard [2]. However, for some instances, it is very interesting to generate the Choquet-optimal solutions instead of the WS-optimal solutions. In Figure 3, we have represented the WS-optimal points (on the left) and the Choquet-optimal points (on the right) of a particular instance of the bi-objective spanning tree problem introduced by Knowles et al. [143]. These type of instances are considered as hard because there are many Pareto non-dominated points located far away from any supported Pareto non-dominated points.

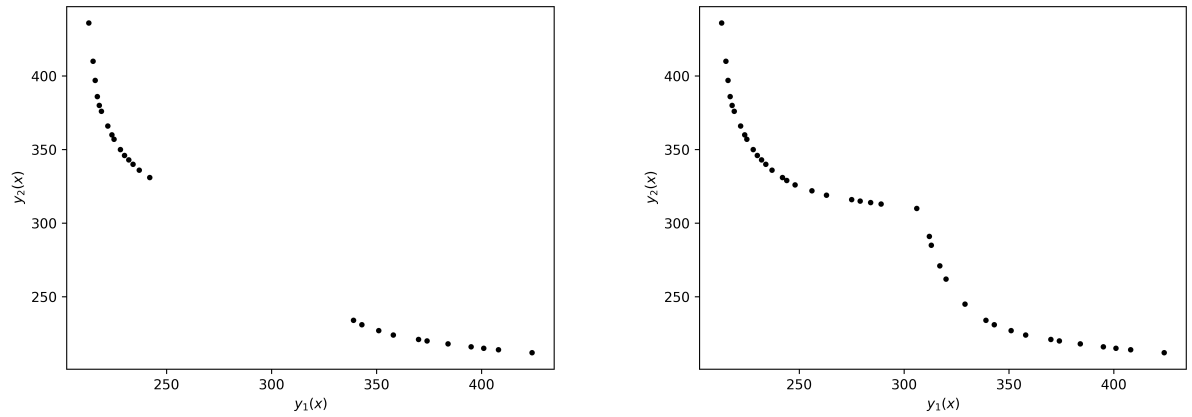


Figure 3: WS-optimal points and Choquet-optimal points for a bi-objective STP instance.

We observe through this figure the interest to generate the Choquet-optimal solutions comparing to the generation of the WS-optimal solutions: with the WS, no solution in the middle of the Pareto front is generated.

Unfortunately, we were not able to generate Choquet-optimal solutions for MOCO problems with more than two objectives. Indeed, e.g., for problems with three objectives, the MOCO problem P_σ that needs to be solved in Algorithm 4 boils down to a bi-objective problem with additional constraints, which is hard to solve. Moreover, all permutations on $\mathcal{P}$ have to be considered, and the number of permutations increase exponentially according to the number of objectives.

For problems with more than two objectives, we have however generated the Choquet-optimal points for explicit sets of alternatives, from size going from 100 to 3000, and for a number of objectives going from 2 to 7, in less running times than a method based on linear programming [173].

4.5 2-additive Choquet integrals

A Choquet integral is a versatile aggregation operator, as it can express preferences to a wider set of solutions than a WS, through the use of a non-additive capacity. However, this model needs also a wider set of parameters to capture this non-additivity ($(2^p - 2)$ parameters to set). For this reason, the concept of k-additivity has been introduced by Grabisch [108] to find a compromise between the expressiveness of the model and the number of needed parameters.

Definition 34. *A capacity v is said to be k-additive if:*

- $\forall \mathcal{A} \subseteq \mathcal{P}, m_v(\mathcal{A}) = 0$ *if* $card(\mathcal{A}) > k$
- $\exists \mathcal{A} \subseteq \mathcal{P}$ *such that* $card(\mathcal{A}) = k$ *and* $m_v(\mathcal{A}) \neq 0$

We have introduced a sufficient condition to produce 2-additive Choquet-optimal solutions of MOCO problems [174]. We have also presented an algorithm to obtain these solutions based on this condition. However, only a subset of the 2-additive Choquet-optimal solutions can be generated (but in our experiments at least 98% of these solutions were generated).

4.6 Additional result

Given the effort required to define the parameters of the Choquet integral in relation to the WS, we have studied the interest of using the Choquet integral instead of the WS [169]. More precisely, for different alternatives evaluated with p criteria, we have evaluated the probability that an alternative optimzing a defined Choquet integral could not have been obtained with a simple WS. This is particularly important in the general context where the alternatives to compare are not explicitly given but are obtained from a MOCO problem. Indeed, in different works [42, 95, 100, 101, 102], the authors define a Choquet integral and then search for an optimal solution according to the defined Choquet integral. Two difficulties are thus introduced: the elicitation of the Choquet integral and the optimization of the Choquet integral for the particular multi-objective problem studied. Given this complexity, it is worth studying the real strength of the Choquet integral and to see whether a simpler method (the WS) could not have been used to obtain the same optimal solution.

To our knowledge, only one group of authors have performed experiments to assess the powerfulness of the Choquet integral. Meyer and Pirlot [184] have compared the ability of related models to represent rankings of alternatives. They compare different aggregators, including the WS, the Choquet integral and additive value functions. To do so, they randomly generate alternatives and define a ranking of the alternatives. Then they check if the models can represent the ranking. They show that the Choquet integral model can represent significantly more orders than the WS, and that the difference becomes quite large when the number of criteria is high. If their work can appear similar to our study, there are two important differences. First, they considered rankings of alternatives, while we checked only the ability of an aggregator to reach one optimal alternative. Second, given a set of alternatives we do not pick up randomly a best alternative, as they did, but we generate randomly a Choquet integral (with a uniform law, thanks to the method developed by Combarro et al. [57]), and check if the alternative optimizing the Choquet integral could not have been obtained with a WS too. Therefore, a probability has been defined, according to the set of possible Choquet integrals, and not according to the set of possible alternatives.

The results showed that the maximal value that this probability can take is close to one, when the number of criteria is higher than four. However, to reach this high probability, particular data sets have been constructed, in favor of the Choquet integral. When the number of WS-optimal alternatives increases in a set, the probability decreases rapidly, and particularly if the number of criteria is higher than four.

5 Interactive methods

5.1 Introduction

We study in this section interactive approaches to solve MOCO problems. In these approaches, repeated interactions with the DM are carried out in order to direct the exploration of the search space. Generally, a parameterized aggregation function is used to take into account the feedback from the DM as the optimization progresses. The parameters of the aggregation function are adapted at each stage to integrate the DM's information.

One of the first interactive method developed is the STEM method [31], developed in 1971, for solving interactively multi-objective linear problems. At each step of the method, a Pareto-optimal solution is selected using an augmented weighted Tchebyshev norm and presented to the DM. If the solution is suitable, the exploration stops. If not, the DM indicates an objective on which she is prepared to degrade the performance, as well as a limit value for this degradation. Following this work, many other interactive methods have been developed, aiming to improve the STEM method. Vincke [253] proposed an interactive method whose aim is to facilitate the dialogue with the DM and to minimize the calculation steps. Zionts and Wallenius [267] presented a method where the DM is requested to provide answers to "yes and no" questions regarding certain trade-offs that she likes or dislikes. Steuer and Choo [234] also used the augmented weighted Tchebyshev norm, but the DM is asked to select her favorite solutions among a subset of solutions. Teghem et al. [243] have extended the STEM method to deal with stochastic programming problems. Korhonen and Laakso [145] proposed a method based on aspiration levels, defining "reference directions" to guide the search for an optimal solution. They use a graphical interface to help the DM throughout the interactive process. A full review of these methods can be found in the survey of Shin and Ravindran [228]. Many of these methods have been then adapted to solve multi-objective mixed-integer, integer and combinatorial optimization problems (see the survey of Alves and Clímaco [6]).

More recently, preference elicitation techniques developed in the AI domain have been integrated into interactive methods for solving MOCO problems. Designing efficient preference elicitation procedures to support decision-making in combinatorial domains is one of the hot topics of algorithmic decision theory. On non-combinatorial domains, various model-based approaches are already available for preference learning. The elicitation process consists in analyzing preference statements provided by the DM to assess the parameters of the decision model and determine an optimal solution (see, e.g., [24, 38, 46, 50, 96, 244]). Within this stream of research, *incremental approaches* are of special interest because they aim to analyze the set of feasible solutions in order to identify the critical preference information needed to find the optimal alternative. By a careful selection of preference queries, they make it possible to determine the optimal choice within large sets, using a reasonably small number of questions [38, 50].

The incremental approach was efficiently used in various decision contexts such as multi-attribute utility theory or multicriteria decision-making [43, 215, 258], decision-making under risk [50, 115, 202, 255] and collective decision-making [168]. However, extending these approaches for decision support on combinatorial domains is more challenging due to the implicit definition of the set of solutions and the huge number of feasible solutions. In order to overcome this problem, several contributions aim in combining standard solution procedures with incremental preference elicitation. Examples can be found in various contexts such as constraint satisfaction [104], committee election [22], matching [75], sequential decision-making under risk [212, 257] and fair multi-agent optimization [37].

In multi-objective optimization, the search procedure combines the generation of Pareto-optimal solutions with preferences queries allowing a progressive reduction of the uncertainty attached to the parameters of the preference aggregation model, in order to progressively focus the exploration on the most attractive solutions [21]. Various attempts to interleave incremental preference elicitation methods and constructive algorithms have been proposed. The basic principle consists in constructing the optimal solution from optimal sub-solutions using the available preference information, and to ask new preference information by comparing partial solutions when necessary. This has been successfully applied by Benabbou and Perny [23] to greedy algorithms, dynamic programming, A^* and branch-and-bound procedures.

In this section, we will develop different interactive methods based on incremental elicitation, i.e., a general exact method, a genetic algorithm and specific adaptions of greedy and local search for matroid optimization. The methods are all based on three major assumptions:

1. An aggregation function with unknown parameters is used to model the preferences of the DM.
2. Questions to the DM are in the form of pairwise comparisons between feasible solutions (or partial solutions in the particular case of the greedy algorithm). The DM is able to answer every preference questions, without ever getting it wrong.

3. Minimax regret approaches are used to determine the most promising solution under the preference imprecision.

5.2 Generalities

We assume here that the DM's preferences over solutions can be represented by a parameterized scalarizing function f_ω that is linear in its parameters ω. A solution $x \in \mathcal{X}$ is preferred to a solution $x' \in \mathcal{X}$ if and only if $f_\omega(y(x)) \leq f_\omega(y(x'))$. To give a few examples, the function f_ω can be a WS, an OWA operator (with an unknwon weight set ω) or a Choquet integral (with an unknown capacity ω). We also assume that the set of parameters ω is not known initially. Instead, we consider a set Θ of pairs (possibly empty) $(a, b) \in \mathbb{R}^p \times \mathbb{R}^p$ such that the evaluation vector a is known to be preferred to the evaluation vector b. This set can be obtained by asking preference queries to the DM. Let Ω_Θ be the set of all parameters ω that are compatible with Θ, i.e., all parameters ω that satisfy the constraints $f_\omega(a) \leq f_\omega(b)$ for all $(a, b) \in \Theta$. Note that since f_ω is linear in ω, Ω_Θ is represented by a convex polyhedron. The problem is now to determine the most promising solution under the preference imprecision (defined by Ω_Θ). To do so, we use the minimax regret approach ([38]), which is commonly used to make robust recommendations under preference imprecision in various decision contexts. It is based on the following definitions:

Definition 35 (Pairwise Max Regret)**.** *The Pairwise Max Regret (PMR) of solution $x \in \mathcal{X}$ with respect to solution $x' \in \mathcal{X}$ is:*

$$PMR(x, x', \Omega_\Theta) = \max_{\omega \in \Omega_\Theta} \{f_\omega(y(x)) - f_\omega(y(x'))\}$$

In other words, $PMR(x, x', \Omega_\Theta)$ is the worst-case loss when recommending solution x instead of solution x' (note that $PMR(x, x', \Omega_\Theta) = \max_{\omega \in \Omega_\Theta} \{f_\omega(y(x')) - f_\omega(y(x))\}$ if f_w needs to be maximized).

Definition 36 (Max Regret)**.** *The Max Regret (MR) of a solution $x \in \mathcal{X}$ is:*

$$MR(x, \mathcal{X}, \Omega_\Theta) = \max_{x' \in \mathcal{X}} PMR(x, x', \Omega_\Theta)$$

Thus $MR(x, \mathcal{X}, \Omega_\Theta)$ is the worst-case loss when selecting the solution x instead of any other feasible solution $x' \in \mathcal{X}$. We can now define the minimax regret:

Definition 37 (MiniMax Regret)**.** *The MiniMax Regret (MMR) of a set $\mathcal{X}$ is:*

$$MMR(\mathcal{X}, \Omega_\Theta) = \min_{x \in \mathcal{X}} MR(x, \mathcal{X}, \Omega_\Theta)$$

According to the minimax regret criterion, an optimal solution is a solution that achieves the minimax regret (i.e., any solution in $\arg\min_{x \in \mathcal{X}} MR(x, \mathcal{X}, \Omega_\Theta)$), allowing to minimize the worst-case loss. Note that if $MMR(\mathcal{X}, \Omega_\Theta) = 0$, then any optimal solution for the minimax regret criterion is necessarily optimal according to the DM's preferences.

Example 19. *We illustrate PMR calculations using Example 16 data (choosing a bike). We consider that the set Θ is initially empty and that the preference model is a WS. We first compute the PMRs for each pair of alternatives. We obtain the following table:*

	Road bike	*Mountain bike*	*Gravel bike*	*City bike*
Road bike	/	8	3	10
Mountain bike	9	/	5	2
Gravel bike	4	5	/	7
City bike	10	2	6	/

Note that as Θ is empty, the set Ω_Θ is composed of all positive weights. If we add the constraint $\sum_{i=1}^{3} \omega_i = 1$, the extreme points of the convex polyhedron representing Ω_Θ are $(1, 0, 0)$, $(0, 1, 0)$ and $(0, 0, 1)$. The PMRs values are thus simply equal to the maximal difference of performance for one of the criteria between two alternatives (i.e., in this simple case there is no need for a linear program to calculate the PMRs). The max regret associated to the road bike is thus equal to 10 (because if we recommend the road bike while the city bike is the favorite bike of the DM, there is a loss of 10 according to the comfort criterion), the max regret of the mountain bike is equal to 9, the max regret of the gravel bike is equal to 7 and the max regret of the city bike is equal to 10. Therefore, the solution of minimax regret is the gravel bike.

5.3 Exact method

5.3.1 Presentation

We have proposed a simple exact regret-based interactive method for the determination of a solution according to the DM's preferences [19]. The method is exact in the sense that, if the DM answers correctly to all pairwise comparisons between feasible solutions, the solution recommended at the end of the procedure will have a minimax regret equal to zero and will be thus optimal. Note that the value $MMR(\mathcal{X}, \Omega_\Theta)$ can only decrease when inserting new preference information in Θ, as observed in previous works [24]. Therefore, we have also adapted the method to produce near-optimal solutions, that is we bound the minimax regret of the final solution to a small value (equal to δ), to limit the number of questions asked.

The main principle of the method, called IEEP (for Incremental Elicitation based on Extreme Points), is as follows. At each iteration step, we generate a set of promising solutions using the extreme points of the polyhedron representing Ω_Θ (the set of admissible parameters). We ask then the DM to compare two of these solutions and, we update Ω_Θ according to her answer. The method is stopped whenever a (near-)optimal solution is detected (i.e., a solution $x \in \mathcal{X}$ such that $MR(x, \mathcal{X}, \Omega_\Theta) \leq \delta$ holds). More precisely, taking as input an instance of a MOCO problem P, a tolerance threshold $\delta \geq 0$, a scalarizing function f_ω with unknown parameters ω and an initial set of preference statements Θ, our algorithm iterates as follows:

1. First, the set of all extreme points of the polyhedron Ω_Θ is generated[4]. This set is denoted by EP_Θ and is equal to $\{\omega^1, \omega^2, \ldots, \omega^k, \ldots, \omega^q\}$, with $q = |EP_\Theta|$.

2. Then, for every point $\omega^k \in EP_\Theta$, the instance of P is solved by considering the *precise* scalarizing function f_{ω^k} (ω^k is seen as a weight set). For each scalarizing function, f_{ω^k} the corresponding optimal solution is denoted by x^k.

3. Finally, the $MMR(X_\Theta, \Omega_\Theta)$ value is computed, where $X_\Theta = \{x^k : k \in \{1, \ldots, q\}\}$. If this value is strictly larger than δ, then the DM is asked to compare two solutions $x, x' \in X_\Theta$ and Ω_Θ is updated by imposing the linear constraint $f_\omega(x) \leq f_\omega(x')$ (or $f_\omega(x) \geq f_\omega(x')$ depending on her answer); the algorithm stops otherwise.

The following proposition establishes the validity of IEEP:

Proposition 6. *For any positive tolerance threshold δ, the algorithm IEEP returns a solution $x^* \in \mathcal{X}$ such that the inequality $MR(x^*, \mathcal{X}, \Omega_\Theta) \leq \delta$ holds [19].*

Note that following the addition of preferences information Θ, the set X_Θ can be filtered. Indeed, let us consider a new partial preference relation, called $\succeq_\Theta$, that extends Θ and the Pareto dominance $\succeq_P$.

Definition 38. *Θ-preference dominance relation: we say that a point $u = (u_1, ..., u_p) \in \mathbb{R}^p$ is Θ-preferred to a point $v = (v_1, ..., v_p) \in \mathbb{R}^p$ if, and only if, $g(u) \succ_P g(v)$, where $g(.)$ is a vector function defined as follows: $\mathbb{R}^p \to \mathbb{R}^q; g(y) \to (f_{\omega^1}(y), f_{\omega^2}(y), \ldots, f_{\omega^k}(y), \ldots, f_{\omega^q}(y))$, with f_{ω^k} representing the scalarizing function obtained with the extreme point ω^k (we consider that there are q extreme points associated to the convex polyhedron representing the set of all parameters that are compatible with Θ). We denote this relation by $u \succ_\Theta v$.*

This dominance relations directly results from the linearity of the parameterized scalarizing function f_ω with respect to its parameters [259].

Example 20. *Let's go back to Example 16. Let's consider that the parameterized scalarazing function f_ω is a WS and that the DM tell us that she prefers the mountain bike to the city bike. Therefore, we have $\Theta = \{((11, 9, 15), (9, 8, 17))\}$ and $EP_\Theta = \{(1, 0, 0), (0, 1, 0), (\frac{1}{2}, 0, \frac{1}{2}), (0, \frac{2}{3}, \frac{1}{3})\}$. We can now compute $g(y)$ for each alternative. We obtain $g((17, 18, 7)) = (17, 18, 12, \frac{43}{3})$, $g((11, 9, 15)) = (11, 9, 13, 11)$, $g((14, 14, 10)) = (14, 14, 12, \frac{38}{3})$ and $g((9, 8, 17)) = (8, 9, 13, 11)$. We see thus now that the road bike is preferred to the gravel bike (as $(17, 18, 12, \frac{43}{3})$ Pareto dominates $(14, 14, 12, \frac{38}{3})$), and of course, the mountain bike is preferred to the city bike ($(11, 9, 13, 11)$ Pareto dominates $(8, 9, 13, 11)$). Therefore, it remains only two alternatives: the road bike and the mountain bike.*

We will now illustrate how the IEEP method works, with the help of an example.

Example 21. *Let's consider the multi-objective spanning tree problem (MOSTP) with 5 nodes and 7 edges given in Figure 4. Each edge is evaluated with respect to 3 objectives. Assume that the DM's preferences can be represented by a WS, noted f_ω, with unknown parameters ω. Our goal is to determine an optimal spanning*

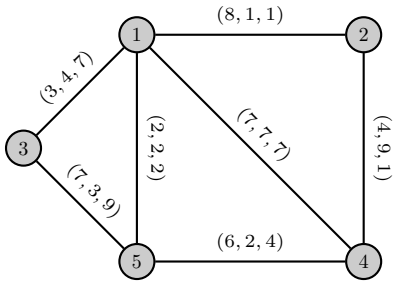


Figure 4: A MOSPT instance with 5 vertices and 3 objectives.

tree for the DM ($\delta = 0$), i.e., a connected acyclic sub-graph with 5 nodes that is f_ω-optimal. We now apply IEEP on this instance, starting with an empty set of preference statements (i.e., $\Theta = \emptyset$).
Initialization: *As $\Theta = \emptyset$, Ω_Θ is initialized to the set of all weighting vectors $\omega = (\omega_1, \omega_2, \omega_3) \in [0,1]^3$ such that $\omega_1 + \omega_2 + \omega_3 = 1$. In Figure 5, Ω_Θ is represented by the triangle ABC in the space (ω_1, ω_2); the value ω_3 is implicitly defined by $\omega_3 = 1 - \omega_1 - \omega_2$. Hence, the initial extreme points are the vectors of the natural basis of the Euclidean space, corresponding to Pareto dominance; in other words, we have $EP_\Theta = \{\omega^1, \omega^2, \omega^3\}$ with $\omega^1 = (1,0,0)$, $\omega^2 = (0,1,0)$ and $\omega^3 = (0,0,1)$. We then optimize according to all weighting vectors in EP_Θ using the Prim algorithm [206], and we obtain the following three solutions: for ω^1, we have a spanning tree x^1 evaluated by $f(x^1) = (15,17,14)$; for ω^2, we obtain a spanning tree x^2 with $f(x^2) = (23,8,16)$; for ω^3, we find a spanning tree x^3 such that $f(x^3) = (17,16,11)$. Hence, we have $X_\Theta = \{x^1, x^2, x^3\}$.*
***Iteration step** 1: Since $MMR(X_\Theta, \Omega_\Theta) = 8 > \delta = 0$, we ask the DM to compare two solutions in X_Θ, say x^1 and x^2. Assume that the DM prefers x^2. In that case, we perform the following updates: $\Theta = \{((23,8,16),(15,17,14))\}$ and $\Omega_\Theta = \{\omega : f_\omega(23,8,16) \leq f_\omega(15,17,14)\}$. We then compute the extreme points of Ω_Θ and we obtain $EP_\Theta = \{\omega^1, \omega^2, \omega^3\}$ with $\omega^1 = (0.53, 0.47, 0)$, $\omega^2 = (0,1,0)$ and $\omega^3 = (0, 0.18, 0.82)$. In Figure 6, Ω_Θ is represented by the triangle BFE. We optimize according to these weights, and we obtain $X_\Theta = \{x^4, x^2, x^3\}$ with $f(x^4) = (19,9,14)$.*
***Iteration step** 2: Here $MMR(X_\Theta, \Omega_\Theta) = 1.18 > \delta = 0$. Therefore, we ask the DM to compare two solutions in X_Θ, say x^3 and x^2. Assume she prefers x^3. We obtain $\Theta = \{((23,8,16),(15,17,14)),((17,16,11),(23,8,16))\}$ and $\Omega_\Theta = \{\omega : f_\omega(23,8,16) \leq f_\omega(15,17,14) \wedge f_\omega(17,16,11) \leq f_\omega(23,8,16)\}$. We compute the corresponding extreme points, which are given by $EP_\Theta = \{(0.43, 0.42, 0.15), (0, 0.18, 0.82), (0, 0.38, 0.62)\}$ (see triangle HGE in Figure 7). We optimize according to these weights, and we obtain $X_\Theta = \{x^3, x^5\}$ (after filtering with the Θ-preference dominance relation) with $f(x^5) = (19,9,14)$.*
***Iteration step** 3: Now $MMR(X_\Theta, \Omega_\Theta) = 1.18 > \delta = 0$. Therefore, we ask the DM to compare x^3 and x^5. Assuming that she prefers x^5, we update Θ by inserting the preference statement $((19,9,14),(17,16,11))$ and we update Ω_Θ by imposing the following additional constraint: $f_\omega(19,9,14) \leq f_\omega(17,16,11)$. The corresponding extreme points are given by $EP_\Theta = \{(0.18, 0.28, 0.54), (0, 0.3, 0.7), (0, 0.38, 0.62), (0.43, 0.42, 0.15)\}$ (see Figure 8). Now, the set X_Θ only includes one spanning tree x^3 (after filtering) and $y(x^3) = (19,9,14)$. Finally, the algorithm stops (since we have $MMR(X_\Theta, \Omega_\Theta) = 0 \leq \delta = 0$) and it returns the solution x^3 (which is guaranteed to be the optimal solution for the DM).*

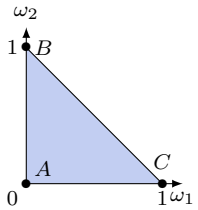


Figure 5: Initial set.

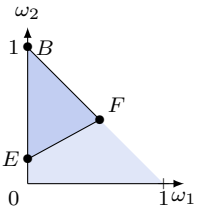


Figure 6: After step 1.

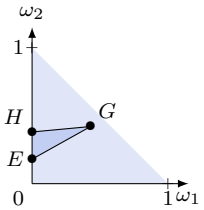


Figure 7: After step 2.

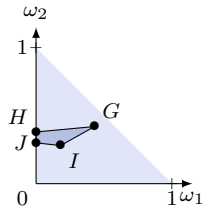


Figure 8: After step 3.

5.3.2 Experimental results

We have applied IEEP to two MOCO problems: the MOSTP and the MOTSP. We have assumed that the DM's preferences can be represented by a WS f_ω, with weights $\omega = (\omega_1, \ldots, \omega_n)$ not known initially. An

[4]This can be obtained by a polyhedral geometry library like polymake (`https://polymake.org`).

important procedure of our approach is the selection of the two solutions to be compared from the set X_Θ (after filtering with the preference dominance relation). Throughout our experiments, we have tested three different selection methods:

- **Random:** The two solutions are randomly chosen in X_Θ.
- **Max-Dist:** We compute the Euclidean distance between all solutions in the objective space, and we choose a pair of solutions maximizing the distance.
- **CSS:** The Current Solution Strategy (CSS) consists in selecting a solution x minimizing the max regret, and one of its best challengers arbitrary chosen in the set $\arg\max_{x'} PMR(x, x', \Omega_\Theta)$ [38]. For instance, in Example 16, with this strategy, the two solutions to be compared would be the gravel bike (the solution of minimax regret) and the city bike (its best challenger).

Note that these three strategies are equivalent when only considering two objectives, since the number of extreme points is always equal to two in this particular case.

Multi-objective spanning tree

We were able to solve instances with up to 100 nodes and 6 objectives in less than 1 minute. We have noticed that Max-Dist is the best strategy for minimizing the number of generated preference queries. More precisely, for all instances, the preferred solution is detected with less than 40 queries and the optimality is established after at most 50 queries. In fact, we can reduce even further the number of preference queries by considering a strictly positive tolerance threshold; to give an example, if we set $\delta = 0.1$ (i.e., 10% of the initial regret), then our algorithm combined with Max-Dist strategy generates at most 20 queries in all considered instances, for an error never exceeding 5%.

We have also compared IEEP to the state-of-the-art method proposed by Benabbou and Perny [20]. The latter consists essentially in integrating incremental elicitation into the Prim algorithm (this method is called IE-Prim hereafter). The main difference between IE-Prim and IEEP is that IE-Prim is constructive: queries are not asked on complete solutions but on partial solutions. We have observed that IEEP outperforms IE-Prim in all settings, allowing the running time and the number of queries to be divided by three in our biggest instances.

Multi-objective traveling salesman problem

Similar results have been obtained for the MOTSP. Instances with up to 100 vertices and 6 objectives have been solved, in less than 10 minutes (the running times are much higher for the MOTSP than for the MOSTP as the TSP is much more difficult to solve exactly with known preferences). Max-Dist remains the best strategy for minimizing the number of generated preference queries.

More details on instances, implementations and results can be found in our related paper [19].

5.4 Genetic algorithm

5.4.1 Presentation

Our exact method is efficient, but limited to relatively small instances. Moreover, it is almost impossible to apply the method with the Choquet integral as a scalarizing function. Indeed, the method requires the determination of the extreme points of the polyhedron representing the preferences, and the number of extreme points for preferences modeled with the Choquet integral quickly becomes too large [57]. Therefore, we present in this section an interactive method based on an evolutionary algorithm.

During these past few years, integrating preferences into evolutionary algorithms has become increasingly popular, see e.g., [40, 260] for some surveys. Due to the high number of different interactive methods that has been developed, Xin et al. [260] have recently established a taxonomy identifying the important factors to differentiate these methods. Four essential design factors are defined: interaction pattern (how the interaction with the DM is scheduled during the run), preference information (how the preference information is obtained from the DM), preference model (utility function, dominance relation or decision rules), and search engine (how the interesting solutions are produced, e.g., mathematical programming techniques or heuristics).

For the new method developed in this section, the interaction with the DM is done during the run, the preference information is retrieved from pairwise comparisons, the preference model is a utility/aggregation function and the search engine can be both mathematical programming techniques or heuristics. To the best

of our knowledge, the existing methods that share the same factors are: the Interactive Evolutionary Metaheuristic (IEM) [203], the Interactive Pareto Memetic Algorithm (IPMA) [130], the Progressively Interactive Evolutionary Multi-Objective approach using Value Functions (PI-EMOVF) [66], the Necessary-preference-enhanced Evolutionary Multi-objective Optimizer (NEMO) [41], the Brain-Computer Evolutionary Multi-objective Optimization Algorithm (BC-EMOA) [15], the Interactive Non-dominated Sorting algorithm with Preference Model called INSPM [198], the Evolutionary Multiple Objective optimization guided by interactive Stochastic Ordinal Regression (EMOSOR) [246] and the Decomposition-Based Interactive Evolutionary Algorithm for Multiple Objective Optimization (DBIEA) [247]. All but the first two of these methods have been adapted for solving continuous multi-objective problems, and they have not been tested in MOCO problems. The first two methods only consider utility functions based on linear or Tchebyshev aggregation. In IEM, linear programming techniques are used to learn the parameters of the utility function, whereas an internal genetic algorithm is employed in IPMA. In [130], IEM and IPMA were directly compared and IPMA achieved better results on the considered MOTSP instances. IPMA follows the classical steps of genetic algorithms while generating pairwise comparison queries periodically to reduce the set of possible utility functions. The frequency of preference queries is controlled by a comparison probability, which is progressively reduced during the search. Instead, our algorithm uses regret-based incremental elicitation techniques to select informative preference queries and generates promising solutions during the search. As a result, our method generates at most 25 queries on existing MOTSP instances with 300 cities and 7 objectives, whereas IPMA needs between 30 and 60 queries on smaller instances (150 cities and 6 objectives).

In the numerical section, we have compared the results obtained by our algorithm to that of a new version of NEMO called NEMO-II [42]. NEMO follows the same scheme as NSGA-II [64] (the reference algorithm for solving continuous multi-objective problems), except that the dominance relation used in the sorting step is replaced by a necessary preference relation which is built from the available preference information (expressed in terms of pairwise comparisons of solutions): a solution a is necessarily preferred to a solution b if, and only if, a is at least as good as b for all value functions compatible with the available preference data. The main difference between NEMO and NEMO-II lies in the computation of the necessary preference relation: NEMO requires solving a quadratic number of linear programs in the worst-case, whereas NEMO-II only performs a linear number of such optimizations. Moreover, NEMO-II is able to handle inconsistencies in the information provided by the DM: if at some point there is no value function compatible with the collected preference information, then the constraints related to the oldest pairwise comparisons are removed until the feasibility is restored. NEMO-II is also able to switch from a simple preference model (a WS) to a more sophisticated one (the 2-additive Choquet integral) in order to capture more complex decision behaviors.

5.4.2 Regret-based incremental genetic algorithm

The interactive genetic algorithm that we have proposed uses regret-based incremental elicitation techniques to select individuals from a population. Our algorithm, called RIGA for *Regret-based Incremental Genetic Algorithm*, follows the traditional scheme of genetic algorithms but differs in the following way:

- The population P is composed of pairs (ω, x_ω), where ω is a possible instance of the preference parameters and solution x_ω is near optimal for the scalarazing function f_ω.
- The crossover and mutation operators are applied on parameter instances (not on solutions).

Initial Population To generate the initial population, we have to generate a set of possible preference parameters. Then, for every generated parameter ω, we must determine a solution x_ω that is (near-)optimal for the *precise* scalarizing function f_ω. To do so, we use an existing poly-time solving algorithm (e.g., Prim algorithm for the MOSTP with a WS). These parameters could be generated uniformly at random, leading to a poly-time initialization phase. However, we propose instead to generate the extreme points of the polyhedron Ω_Θ, as it gives better results in practice.

Crossover and Mutation As already mentioned, we perform crossovers and mutations on parameter vectors (not on solutions). For every resulting parameter vector ω, we proceed as follows: we determine a solution x_ω that is f_ω-optimal (or almost) using an existing efficient solving method, and then we add the pair (ω, x_ω) in the population P. To obtain a population of the desired size, we create new parameter vectors by means of convex combinations of vectors in P. This crossover operator is of particular interest in optimization problems with imprecise preference parameters because it only generates new admissible parameters from admissible parameters. In our experiments, we create a new parameter vector ω from two parameter vectors ω', ω'' in P as follows:

$$\omega = \lambda\omega' + (1-\lambda)\omega''$$

where $\lambda \in (0,1)$ is generated uniformly at random. Then, given a mutation rate μ, we perform Gaussian mutations on single objectives. This mutation operator yields very good results in practice, but more sophisticated operators would be worth investigating.

Selection To create the new generation, we select K promising pairs from the population P as follows:

- First, we determine a (near-)optimal solution in population P by means of a regret-based incremental elicitation approach. More precisely, let X_P be the set of all solutions in P. While $MMR(X_P, \Omega_\Theta) > \delta$, the DM is asked to compare two solutions x and x', and state which one is preferred over the other one. The set Ω_Θ of admissible parameters is then updated by inserting the linear constraint $f_\omega(x) \leq f_\omega(x')$ or $f_\omega(x) \geq f_\omega(x')$, depending on her answer. In our experiments, we use the CSS to generate the preference queries. Once the value $MMR(X_P, \Omega_\Theta)$ drops below the threshold δ, we stop asking queries and select a solution x^* that is optimal for the minimax regret criterion, i.e., a solution x^* in $\arg\min_{x \in X_P} MR(x, X_P, \Omega_\Theta)$.
- Then, we compute the distance in objective space between x^* and x for every pair (ω, x) in P and we select the K pairs that minimize the distance to breed the next generation. In our experiments, we use the Euclidean distance, but other distances would be worth investigating.

Termination RIGA stops after M steps and returns a solution arbitrary chosen in $\arg\min_{x \in X_P} MR(x, X_P, \Omega_\Theta)$, where P is the last generated population.

Note that RIGA has different tunable parameters: S the size of the population (generated by crossovers and mutations), $K < S$ the number of pairs selected for the next generation and M the number of generations.

5.4.3 Performance guarantees

We have shown the following result.

Proposition 7. *For any MOCO problem with preferences represented by a WS, an OWA operator or more generally by a Choquet integral, if the problem can be solved (exactly or approximately) in polynomial time (in the problem size) when the preferences are precisely known, then RIGA can be implemented in such way that it runs in polynomial time and asks no more than a polynomial number of queries [27].*

Now we present an execution of RIGA on a small instance of the MOTSP.

Example 22. *We consider an instance of the MOTSP with* 6 *cities and* $p = 3$ *cost functions to be minimized (see Figure 9). We assume here that the DM's preferences over Hamiltonian cycles can be represented by an OWA operator. We now apply RIGA on this instance with* $\delta = 0$, $S = 5$, $K = 2$, $M = 2$.

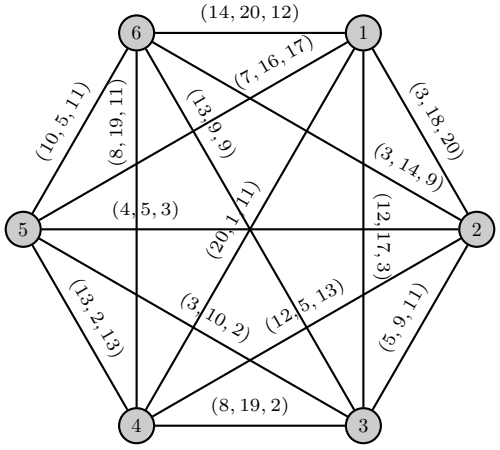


Figure 9: A MOTSP instance with 6 vertices and 3 objectives.

Initialization phase: *The set of admissible weighting vectors is initially defined by* $\Omega_\Theta = \{\omega = (\omega_1, \omega_2, \omega_3) \in [0,1]^3 : \omega_1 + \omega_2 + \omega_3 = 1$ *and* $\omega_1 \leq \omega_2 \leq \omega_3\}$. *Note that we can assume that* Ω_Θ *is represented by a convex polyhedron throughout this subsection, since any constraint of the type* $f_\omega(a) \leq f_\omega(b)$ *is linear in* ω *for any fixed performance vectors* a, b. *In Figure 10, the initial set* Ω_Θ *is represented by the triangle ABC in the space*

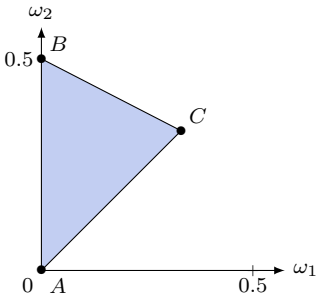


Figure 10: Initial set Ω_Θ.

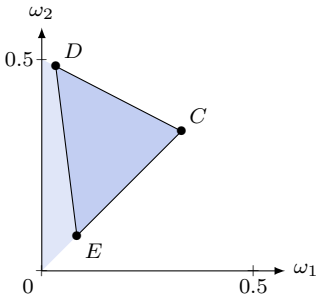


Figure 11: Ω_Θ after 1 query.

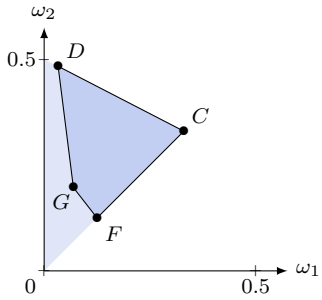


Figure 12: Ω_Θ after 2 queries.

(ω_1, ω_2), ω_3 *being implicitly defined by* $\omega_3 = 1 - \omega_1 - \omega_2$. *The extreme points of* Ω_Θ *are* $(0, 0.5, 0.5)$, $(\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ *and* $(0, 0, 1)$.

Initial population: *In order to generate the initial population, we determine one near-optimal solution for each extreme point of the polyhedron* Ω_Θ *(using a local search procedure, for example). We obtain* $P = \{(\omega^A, x^A), (\omega^B, x^B), (\omega^C, x^C)\}$ *where* $\omega^A = (0, 0.5, 0.5)$, $\omega^B = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$, $\omega^C = (0, 0, 1)$, $y(x^A) = (49, 52, 60)$, $y(x^B) = (39, 50, 66)$, *and* $y(x^C) = (56, 57, 58)$.

First iteration step: *Since* $|P| = 3$ *and* $S = 5$, *we need to generate* 2 *more pairs. Applying the crossover operator on* $\omega^A = (0, 0.5, 0.5)$ *and* $\omega^B = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$, *and then performing a Gaussian mutation on the first objective, we obtain the following vector* $\omega^1 = (0.27, 0.33, 0.40)$. *The function* f_{ω^1} *is then optimized, resulting in the generation of the solution* x^1 *whose cost vector is* $(39, 50, 66)$. *The pair* (ω^1, x^1) *is then inserted in* P. *When applying the crossover operator on* $\omega^B = (\frac{1}{3}, \frac{1}{3}, \frac{1}{3})$ *and* $\omega^C = (0, 0, 1)$, *and after performing a Gaussian mutation on the second objective, we obtain* $\omega^2 = (0.27, 0.33, 0.40)$ *and cycle* x^2 *whose cost vector is* $(56, 57, 58)$. *The pair* (ω^2, x^2) *is then inserted into* P. *Now we have a complete population.*

The selection stage begins. We ask queries to the DM until $MMR(X_P, \Omega_\Theta) \leq \delta = 0$, *where* $X_P = (x^A, x^B, x^C, x^1, x^2)$. *Since* $MMR(X_P, \Omega_\Theta) = 2 > 0$, *we ask the DM to compare two solutions in* X_P, *say* x^A *and* x^B. *Let's suppose that the DM answers: "solution* x^A *is better than solution* x^B*".*

Then Θ *(the set of preference statements) is updated by adding the pair* $(y(x^A), y(x^B))$; *thus we have* $\Theta = \{((49, 52, 60), (39, 50, 66))\}$. *Consequently, the set of admissible parameters* Ω_Θ *is restricted by the linear constraint* $f_\omega(y(x^A)) \leq f_\omega(y(x^B))$, *i.e.,* $\omega_2 \leq \frac{3}{4} - 2\omega_1$. *Now* Ω_Θ *is represented by the triangle DCE in Figure 11, and we have* $MMR(X_P, \Omega_\Theta) = 2 > 0$. *Since the minimax regret is above the threshold, we ask the DM to compare two solutions in* X_P, *say* x^C *and* x^B. *The DM tells us that she prefers solution* x^B *to solution* x^C. *We then update* Θ *by inserting the pair* $((49, 52, 60), (56, 57, 58))$ *and we restrict* Ω_Θ *by imposing the linear constraint* $f_\omega(y(x^A)) \leq f_\omega(y(x^C))$, *i.e.,* $\omega_2 \leq \frac{2}{7} - \frac{9}{7}\omega_1$ *(*Ω_Θ *is now represented by DCFG in Figure 12). Now we have* $MMR(X_P, \Omega_\Theta) = MR(x^A, X_P, \Omega_\Theta) = 0$. *We must select solutions for the next population. Here we have* $x^* = x^A$. *Since* $K = 2$, *we need to select one more pair for the next generation. We choose* x^C *as it is the closest solution to* x^*, *according to the Euclidean distance. Thus, we have* $P = \{(\omega^C, x^C), (\omega^A, x^A)\}$ *for the next iteration step.*

Second iteration step: *Since* $|P| = 2$ *and* $S = 5$, *we have to generate three more pairs. After applying the crossover and mutation operators on* ω^A *and* ω^C, *we obtain* $(0, 0.41, 0.59)$, $(0, 0.21, 0.79)$ *and* $(0, 0.18, 0.82)$. *We then optimize the corresponding OWA functions and, we obtain solutions* x^3, x^4 *and* x^5 *whose cost vectors are* $y(x^3) = (49, 52, 60)$, $y(x^4) = (56, 57, 58)$ *and* $y(x^5) = (56, 57, 58)$ *respectively. Therefore, we have* $P = \{(\omega^A, x^A), (\omega^C, x^C), (\omega^3, x^3), (\omega^4, x^4), (\omega^5, x^5)\}$. *Since* $MMR(X_P, \Omega_\Theta) = MR(x^A, X_P, \Omega_\Theta) = 0 \leq \delta$, *we do not need to ask a new query. At this step, we have* $x^* = x^A$.

Since $M = 2$, *RIGA only performs two iteration steps and then stops by returning the solution* $x^* = x^A$ *(corresponding to the cycle* $2-0-3-1-5-4-2$*), which is here optimal according to the DM's preferences.*

5.4.4 Experimental results

We have experimented the method on two problems: the MOTSP and the multi-objective knapsack problem (MOKP). Three different aggregation functions have been tested: the WS, the OWA operator and the Choquet integral. We have compared RIGA to:

- the Interactive Local Search (ILS): ILS is a regret-based local search that we have developed [26, 27] recently. The method starts from a promising solution generated using a heuristic method. Then, we move from solution to solution using a neighborhood function. In order to select the next solution,

preference queries are asked to discriminate between Pareto non-dominated solutions within neighborhoods. More precisely, at every iteration step, we ask queries until the minimax regret drops below a given threshold δ, and then we move to a neighbor solution that minimizes the max regret. The method stops when no improving solutions are found in the neighborhood.

- the Necessary-preference-enhanced Evolutionary Multi-objective Optimizer (NEMO-II) [42]: in this genetic algorithm, the mutation and crossover operators are applied on solutions (instead of weighting vectors), and a tournament selection method is used to construct populations. At every iteration step, linear programming is used to rank the solutions in the current population, using the collected preference information and the crowding distance. In this method, one preference query is generated every 10 generations: the DM is asked to compare two potentially good solutions selected among those in the current population.

- a two-phase method: this method consists in first constructing the Pareto set (or an approximation of this set), and then applying the CSS strategy on this set until the minimax regret drops below a threshold $\delta \geq 0$.

- the Incremental Elicitation based on Extreme Points (IEEP) (see Section 5.3).

We were able to solve instances of the MOKP with up to 100 items and 6 objectives (in less than 30 seconds), and instances of the MOTSP with up to 300 cities and 6 objectives (in less than 4 minutes). For the MOKP, the error was never more than 0.71% and the number of queries was always less than 15. For the MOTSP, the error was never more than 1.37% and the number of queries was always less than 33 (this high number has been attained when solving the larger instances with 6 objectives and the Choquet integral as scalarazing function). The RIGA method outperforms all the other methods used in the benchmark, in terms of running time, error and number of queries.

From the experiments, it is worth noting that the IEEP method was unable to solve instances with the Choquet integral with more than 3 criteria, in a reasonable time (less than 30 minutes). This can be explained by the fact that the number of extreme points of the polyhedron representing the parameter imprecision increases with the number of criteria. For example, after 25 queries, the number of extreme points is approximately equal to 4500 for problems involving 4 criteria.

The two-phase method also proved to be very expensive in terms of computation time, which underlines the fact that it is very relevant to interleave elicitation and optimization.

Full results can be found in the PhD Thesis of Leroy [157].

5.5 Interactive methods for solving MOCO problems under matroid constraints

In this section, we study the particular case of MOCO problems under matroid constraints. We have proposed two interactive methods, based on greedy search and local search, for solving these problems. Two publications are related to this section. In [29], we studied linear functions, while in [30] we considered submodular functions. As submodular functions are also linear, we will focus here only on submodular functions.

5.5.1 Matroid optimization

We consider the problem of finding an independent set of maximum weight in a matroid. A matroid $\mathcal{M}$ is a pair $(E, \mathcal{I})$ where E is a set of size n (called the *ground set*) and $\mathcal{I} \subseteq 2^E$ is a non-empty collection of sets (called the *independent sets*) such that, for all $X, Y \in 2^E$, the following properties hold:

(A_1): The empty set is independent, i.e., $\emptyset \in \mathcal{I}$

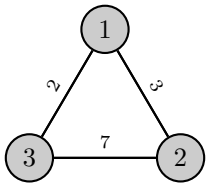
(A_2): $(Y \in \mathcal{I}$ and $X \subseteq Y) \Rightarrow X \in \mathcal{I}$.

(A_3): $(X \in \mathcal{I}$ and $Y \in \mathcal{I}$ and $|Y| > |X|) \Rightarrow \exists e \in Y \setminus X$ such that $X \cup \{e\} \in \mathcal{I}$.

Axiom A_2 is sometimes called the *hereditary property*, whereas A_3 is known as the *augmentation property*. Axiom A_2 implies that all maximal independent sets (w.r.t set inclusion) have the same cardinality. A maximal independent set is called a *basis* of the matroid, and the set of all bases will be denoted by $\mathcal{B}$. The cardinality of a basis is called the *rank* of the matroid (denoted by $r(\mathcal{M})$). A special focus will be given to the *uniform matroid*, which is defined by $\mathcal{I} = \{X \subseteq E : |X| \leq k\}$ for a given positive integer $k \leq n$. In the numerical tests, we will also consider the *partition matroid* which is defined by a collection $\mathcal{D} = \{D_1, \ldots, D_q\}$ of q disjoints subsets of E, a positive integer $d_i \leq |D_i|$ for all $i \in \{1, \ldots, q\}$ and $\mathcal{I} = \{X \subseteq \cup_{i=1}^{q} D_i : \forall i \in \{1, \ldots, q\}, |X \cap D_i| \leq d_i\}$.

The problem of finding a maximum weight independent set in a matroid (simply called "matroid optimization" problem) can be defined as follows: given a matroid $\mathcal{M} = (E, \mathcal{I})$, we want to compute $\max_{X \in \mathcal{I}} z(X)$ where z is a positive set function defined on 2^E measuring the weight (or utility) of any subset of E. Here, we assume that z is monotonic with respect to set inclusion (i.e., $z(X) \leq z(Y)$ for all $X \subset Y \subseteq E$). Note that the latter assumption implies that we can focus on the bases of the matroid when searching for an optimal independent subset. The optimization of a set function under a matroid constraint has received much attention since the seminal work of Edmonds [79]. This problem has for example multiple applications in various contexts such as recruitment, committee election, combinatorial auctions, scheduling, resource allocation, facility location and sensor placement. Various algorithms are now available to solve this problem, either to optimality or approximately, for specific classes of set functions [47, 186, 190, 191, 217, 230, 254].

Example 23. *A famous matroid optimization problem is the minimum spanning tree problem. Let's consider the following simple complete graph composed of 3 nodes. The set E of the corresponding matroid contains all edges of the graph, i.e., $E = \{(1,2),(2,3),(1,3)\}$ and the set $\mathcal{I}$ of independent sets is composed of all subsets of edges that do not form any cycle, that is: $\mathcal{I} = \{\emptyset, \{(1,2)\}, \{(1,3)\}, \{(2,3)\}, \{(1,2),(1,3)\}, \{(1,2),(2,3)\}, \{(1,3),(2,3)\}\}$ (this matroid is sometimes called the graphic matroid). Let's consider the following positive set function z measuring the weight of any independent set X, equal to $z(X) = \max_{e \in E} c(e) - \sum_{x \in \mathcal{X}} c(x)$, where $c(e)$ represents the cost associated to an edge e. Then the matroid optimization problem consists in searching for a base of maximum weight which is equivalent to the minimum spanning tree problem (the optimal basis is here equal to $\{(1,2),(1,3)\}$).*



When the set function is additive (i.e., the value of any set is defined as the sum of the values of its elements), it is well known, after Edmonds [79], that the problem can be efficiently solved by a greedy algorithm. However, preferences are not always representable by additive functions due to possible interactions among elements. In decision theory, the additivity of utilities is often relaxed and submodular utility functions are frequently used to guarantee a principle of diminishing returns [3, 155, 254]. This principle states that adding an element to a smaller set has more value than adding it to a larger set, formally the set function z should satisfy the following property: $z(X \cup \{i\}) - z(X) \geq z(Y \cup \{i\}) - z(Y)$ whenever $X \subseteq Y$ and $i \notin Y$. This is known to be equivalent to submodularity of function z defined by: $z(X \cup Y) + z(X \cap Y) \leq z(X) + z(Y)$ for all X, Y.

The maximization of a submodular function is NP-hard in general because it includes the max-cut problem as special case [47]. Approximate greedy and local search algorithms have been proposed for the maximization problem, and some interesting worst case bounds on the quality of the approximations returned are known [47, 191, 230, 254].

We assume here that z is a set function representing the subjective preferences of a DM: for any two sets $X, Y \in 2^E$, X is preferred to Y if and only if $z(X) \geq z(Y)$. Hence, finding a maximum weight basis amounts to determining an optimal basis according to the DM's preferences. Moreover, we assume that z is initially not known. Instead, we are given a (possibly empty) set Θ of pairs $(X, Y) \in \mathcal{I} \times \mathcal{I}$ such that X is known to be preferred to Y by the DM. Such preference data can be obtained by asking comparison queries to the DM (i.e., by asking the DM to compare two subsets and state which one is preferred). Let Z be the *uncertainty set* implicitly defined as the set of all functions z that are compatible with Θ, i.e., such that $z(X) \geq z(Y)$ for all $(X, Y) \in \Theta$. The problem is now to determine the most promising basis under preference imprecision. To this end, we consider the minimax regret decision criterion (as done previously). We give below the definitions of pairwise max regrets (PMR), max regret (MR) and minimax regret (MMR) in the context of matroid optimization.

Definition 39. *For any collection of sets $\mathcal{S} \subseteq 2^E$ and for any two sets $X, Y \in \mathcal{S}$:*
$PMR(X, Y, Z) = \max_{z \in Z} \{z(Y) - z(X)\}$
$MR(X, \mathcal{S}, Z) = \max_{Y \in \mathcal{S}} PMR(X, Y, Z)$
$MMR(\mathcal{S}, Z) = \min_{X \in \mathcal{S}} MR(X, \mathcal{S}, Z)$

For matroid optimization problems, computing the MMR value at every step of the elicitation procedure may induce prohibitive computation times, as it may require to compute the pairwise max regrets for all

pairs of distinct bases in $\mathcal{B}$. Therefore, as done in the preceding sections, we propose to combine search and regret-based incremental elicitation to reduce both computation times and number of queries. More precisely, preference queries are generated during the search to progressively reduce the set Z until being able to determine a (near-)optimal basis.

5.5.2 An interactive greedy algorithm

For problems where z is exactly observable, good approximate solutions can be constructed using the following simple greedy algorithm: starting from $X = \emptyset$, the idea is to select an element $e \in E \backslash X$ that maximizes the marginal contribution to X, i.e.,

$$\Delta(e|X) = z(X \cup \{e\}) - z(X) \tag{4}$$

without loosing the independence property. The algorithm stops when no more element can be added to X (the set X is a basis at the end of the procedure). For monotonic submodular set functions, this greedy algorithm has an approximation ratio of $(1 - \frac{1}{e}) \approx 0.63$ for the uniform matroid and an approximation ratio of $\frac{1}{2}$ in the general case [92, 191].

For problems where the set function w is imprecisely known, we propose an interactive version of the greedy algorithm that generates preference queries only when it is necessary to discriminate between some elements. More precisely, queries are generated only when the available preference data is not sufficient to identify an element that could be added to the set X, to ensure that the returned basis is a good approximate solution with provable guarantees. We implement this idea by computing minimax regrets on sets $\mathcal{S} = \{X \cup \{e\} : e \in E \backslash X \text{ s.t. } X \cup \{e\} \in \mathcal{I}\}$, asking preference queries at step i until $MMR(\mathcal{S}, W)$ drops below a given threshold $\delta_i \geq 0$, where δ_i is a fraction of the tolerance threshold δ such that $\sum_{i=1}^{r(\mathcal{M})} \delta_i = \delta$ (see Algorithm 5).

Algorithm 5 `Interactive Greedy Algorithm`

IN ↓: $\mathcal{M}=(E,\mathcal{I})$: a matroid, δ: a tolerance threshold, Z: a set of set functions.
OUT ↑: a (near-)optimal basis X.
$X \leftarrow \emptyset$
$E_c \leftarrow E$
for $i \in \{1, \ldots, r(\mathcal{M})\}$ **do**
  $\mathcal{S} \leftarrow \{X \cup \{e\} : e \in E_c\}$
  while $\text{MMR}(\mathcal{S}, Z) > \delta_i$ **do**
    Ask the DM to compare two elements of $\mathcal{S}$
    Update Z according to the DM's answer
  Select $e \in E_c$ such that $\text{MR}(X \cup \{e\}, \mathcal{S}, Z) \leq \delta_i$ and move e from E_c to X
  Remove from E_c all elements e such that $X \cup \{e\} \notin \mathcal{I}$
return X

We now present an execution of our algorithm on a small example.

Example 24. *We consider in this example the maximum coverage problem over a uniform matroid. In this problem, we have a set of q elements: $V = \{v_1, \ldots, v_q\}$ and a family of n subsets $E = \{S_1, \ldots, S_n\}$. Each subset covers some elements. As each element has a utility, the goal is to select a limited number k of subsets such that the sum of the utilities of the covered elements are maximized.*

We study the following instances: $q = 10$ and $n = 8$. The 8 subsets are defined as follows: $S_1 = \{v_3, v_4, v_5\}, S_2 = \{v_1, v_3\}, S_3 = \{v_6, v_{10}\}, S_4 = \{v_2, v_8\}, S_5 = \{v_7, v_9\}, S_6 = \{v_6, v_7, v_{10}\}, S_7 = \{v_2, v_8\}, S_8 = \{v_1, v_3, v_5\}$. A feasible solution is a collection of subsets $X \subseteq E$ such that $|X| \leq k$ (here we set $k = 2$), and the goal is to identify a feasible solution X maximizing the following set function $z(X)$ defined on 2^E:

$$z(X) = \sum_{v \in \bigcup_{S \in X} S} u(v) \tag{5}$$

where $u(v) \geq 0$ is the utility of an element $v \in V$. In that case, it can be proved that w is monotone and submodular [124]. We further assume that all elements $v \in V$ are evaluated with respect to 3 criteria (denoted by u_1, u_2, and u_3), and their evaluations are given in Table 2. Then, the utility of any element $v \in V$ is:

$$u(v) = \sum_{i=1}^{3} \lambda_i u_i(v) \tag{6}$$

where $\lambda = (\lambda_1, \lambda_2, \lambda_3) \in \mathbb{R}_+$ represents the value system of the DM.

	v_1	v_2	v_3	v_4	v_5	v_6	v_7	v_8	v_9	v_{10}
u_1	4	2	2	3	7	6	8	7	7	1
u_2	5	7	1	2	3	1	5	1	9	1
u_3	4	5	3	7	2	5	3	8	4	4

Table 2: Performance vectors attached to elements in Example 1.

We start the execution with no preference data, and therefore we have to consider all weighting vectors ω in the set $\Omega_\Theta = \{\omega \in [0,1]^3 : \sum_{i=1}^3 \omega_i = 1\}$, which implicitly defines the uncertainty set Z using Equations (5-6). In Figure 13, Ω_Θ is represented by the triangle ABC in the space (ω_1, ω_2), ω_3 being implicitly defined by $\omega_3 = 1 - \omega_1 - \omega_2$. Now, let's execute the Algorithm 5 with $\delta = 0$. Note that only two iteration steps will be needed, as the rank of the uniform matroid is equal to $k = 2$.

First iteration step: *We have $X = \emptyset$ and $E_c = E$, and therefore $\mathcal{S} = E$. Since $MMR(\mathcal{S}, Z) = 6 > 0$, the DM is asked to compare two elements of $\mathcal{S}$, say S_5 and S_7. Let's consider that she prefers S_5. Then Z is updated by imposing the constraint $z(\{S_5\}) \geq z(\{S_7\})$ which amounts to restricting Ω_Θ by imposing $\omega_2 \geq \frac{1}{2} - \omega_1$. Now Ω_Θ is represented by the polyhedron $BCDE$ in Figure 14. Since $MMR(\mathcal{S}, Z) = 2.5$, the DM is asked to compare two subsets, say S_5 and S_6. Let's consider that she preferred S_5. Then, Z is updated by imposing the constraint $z(\{S_5\}) \geq z(\{S_6\})$, which amounts to further restricting Ω_Θ by imposing $\omega_2 \geq \frac{5}{12} - \frac{5}{12}\omega_1$. Now Ω_Θ is represented by the polyhedron $BCFE$ in Figure 15. We have $MMR(\mathcal{S}, W) = MR(\{S_5\}, \mathcal{S}, Z) = 0$, and therefore S_5 is added to X.*

Second iteration step: *We have $X = \{S_5\}$ and $E_c = E \backslash \{S_5\}$, and therefore $\mathcal{S} = \{\{S_5\} \cup \{S\} : S \in E_c\}$. Since $MMR(\mathcal{S}, Z) = 1.5$, we ask the DM to compare two elements of $\mathcal{S}$, say $\{S_5, S_8\}$ and $\{S_5, S_7\}$. Let's consider that the DM prefers $\{S_5, S_8\}$. The uncertainty set Z is therefore updated by imposing $z(\{S_5, S_8\}) \geq z(\{S_5, S_7\})$, i.e., $\lambda_2 \geq 1 - \frac{8}{3}\lambda_1$. Now Ω_Θ is represented by the triangle BGC in Figure 16. Since we have $MMR(\mathcal{S}, Z) = MR(\{S_5, S_8\}, \mathcal{S}, Z) = 0$, then the subset S_8 is added to X.*

As $|X| = k = 2$, the algorithm stops and returns $X = \{S_5, S_8\}$ which is the optimal solution for this instance. This shows that we are able to make good recommendations without knowing λ precisely (here only 3 queries are needed).

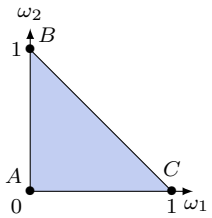


Figure 13: Initial set Ω_Θ.

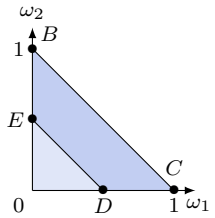


Figure 14: 1 query.

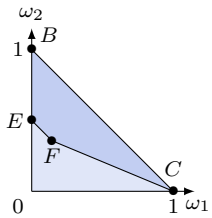


Figure 15: 2 queries.

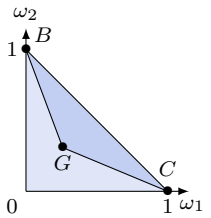


Figure 16: 3 queries.

We have provided theoretical guarantees on the quality of the returned solution [30].

Proposition 8. *Let Z_f be the final set Z when Algorithm 5 stops. For the uniform matroid, Algorithm 5 is guaranteed to return a basis X such that:*

$$\forall w \in Z_f,\ z(X) \geq \big(1 - \frac{1}{e}\big) z(X^*) - \delta, \quad \textit{where } X^* \in \arg\max_{Y \in \mathcal{I}} z(Y).$$

And a more general result (for any type of matroids):

Proposition 9. *Let Z_f be the final set Z when Algorithm 5 stops. Algorithm 5 is guaranteed to return a basis X such that:*

$$\forall z \in Z_f,\ z(X) \geq \frac{1}{2}\big(z(X^*) - \delta\big), \quad \textit{where } X^* \in \arg\max_{Y \in \mathcal{I}} z(Y).$$

5.5.3 An interactive local search

We consider now another efficient way of constructing a good approximate solution to matroid optimization problems with monotonic submodular functions. More precisely, we focus on the following simple local search approach: starting from an arbitrary basis X, the idea is to replace one element $e \in X$ by an element

$e' \in E \backslash X$ such that $X \cup \{e'\} \backslash \{e\}$ belongs to $\mathcal{I}$ and is better than X. This simple exchange principle can be iterated until reaching a local optimum. When z is exactly observable, the local search algorithm has an approximation ratio of 1/2, even in the special case of the uniform matroid [92]. When z is not known, the local search algorithm can be combined with a preference elicitation method, which collects preference data only when it is necessary to identify improving swaps. To implement this idea, we have proposed the Algorithm 6 where $\mathcal{N}(X)$ is the neighborhood of the basis X (i.e., the set of bases that differ from X by exactly one element). The procedure `ComputeInitialBasis` called in line 1 can be any heuristic providing a good starting solution.

Algorithm 6 `Interactive Local Search Algorithm`

IN ↓: $\mathcal{M}=(E,\mathcal{I})$: a matroid, δ: a tolerance threshold, Z: a set of set functions.
OUT ↑: a (near-)optimal basis X.
$X \leftarrow$ `ComputeInitialBasis`$(\mathcal{M})$
improve $\leftarrow$ **true**
while improve **do**
  $\mathcal{N}(X) \leftarrow \{X' \in \mathcal{B} : |\{X \setminus X'\} \cup \{X' \setminus X\}| = 2\}$
  $\mathcal{S} \leftarrow \mathcal{N}(X) \cup \{X\}$
  while $\mathrm{MMR}(\mathcal{S}, Z) > \delta / r(\mathcal{M})$ **do**
    Ask the DM to compare two elements of $\mathcal{S}$
    Update Z according to the DM's answer
  if $\mathrm{MR}(X, \mathcal{S}, Z) \leq \delta / r(\mathcal{M})$ **then**
    improve $\leftarrow$ **false**
  else
    $X \leftarrow$`RandomSelect`$(\arg\min_{X' \in \mathcal{N}(X)} MR(X', \mathcal{S}, Z))$
return X

The following proposition shows that the basis returned by Algorithm 6 is a good approximate solution [30].

Proposition 10. *Let Z_f be the final set z when Algorithm 6 stops. Algorithm 6 is guaranteed to return a basis X such that:*

$$\forall z \in Z_f,\ z(X) \geq \frac{1}{2}\big(z(X^*) - \delta\big), \quad \textit{where } X^* \in \arg\max_{Y \in \mathcal{I}} z(Y).$$

5.5.4 Experimental results

We have tested the two algorithms on two problems: the maximum coverage problem and the collective subset selection problem [231]. Two matroid constraints have been considered for each problem: the uniform matroid and the partition matroid. The algorithms were evaluated through three performance indicators: number of queries, computation times and empirical error, expressed as a percentage from the optimal solution. Two tolerance thresholds have been used: $\delta = 0$ and $\delta = 20\%$ of the initial maximum regret (to reduce the number of preference queries). We have considered instances of the maximum coverage problem with a set $V = \{v_1, \ldots, v_q\}$ of $q = 100$ elements, and a family E of $n = 80$ subsets of V. The family of subsets is generated as suggested by Resende [213]. The utility of an element $v \in V$ is defined by a WS $u^\lambda(v) = \sum_{i=1}^{p} \lambda_i u_i(v)$ where u_i is the evaluation of v on criterion $i \in \mathcal{P}$. Utilities are randomly generated within $[1, 10]$ and three values of p are considered: $p = 4, 6$, and 8. The DM's preferences are then represented by a submodular monotone set function w defined by:

$$z(X) = \sum_{v \in \bigcup_{S \in X} S} u^\lambda(v)$$

for any $X \subseteq E$. For the uniform matroid, we focus on subsets of size at most $k = 16$, i.e., $\mathcal{I} = \{X \subset E : |X| \leq 16\}$. For the partition matroid, set E is randomly partitioned into $q = 4$ sets $\mathcal{D} = \{D_1, \ldots, D_q\}$, and at most $d_i = 4$ elements can be selected for all $i \in \{1, \ldots, q\}$, i.e., $\mathcal{I} = \{X \subseteq E : \forall i \in \{1, \ldots, 4\}, |X \cap D_i| \leq 4\}$. To generate preference queries during the execution of our algorithms, we use the CSS.

For $\delta = 0$, we have observed that the interactive greedy algorithm was outperformed by the interactive local search procedures: the interactive greedy algorithm was about 10 times slower on average and asked more preference queries. We also observed that using $\delta = 0.2$ allows to significantly reduce the number of queries, without increasing the error too much (max 2.3%). Finally, we observed that our algorithms perform better on the uniform matroid than on the partition matroid which is a little more complex.

For the collective subset selection problem, we observed that the interactive greedy algorithm outperformed the interactive local search procedure.

6 Conclusion and future work

Exact and heuristic methods for solving MOCO problems

In Section 2 we have presented exact and heuristic methods with the aim of generating Pareto-optimal sets of MOCO problems. While the first exact methods developed were only able to solve small size instances of MOCO problems, the recent developments have allowed to solve bigger size instances. For example, with the method of Tamby and Vanderpooten [241] based on the ϵ-constraint method and a division of the search space into zones, instances of the MO knapsack problem presenting more than 8 000 Pareto non-dominated points (50 items, 5 objectives) could be solved in reasonable CPU time (about 1 hour). For the MO assignment problem, instances with more than 24 000 Pareto non-dominated could be solved ($n = 20$, 4 objectives). With heuristic methods, more than 250 000 Pareto non-dominated points for instances of the MOTSP could be generated with the method of Cornu et al. [61] (300 cities, 3 objectives). Dealing with such high size set of points in heuristic methods was made possible mainly by the development of new data structures and methods for archiving a non-dominated set. However, generating large sets, while theoretically interesting, does not seem to be very useful in practice. Indeed, it becomes very difficult for the DM to interpret such sets, who is often more interested in smaller sets.

Good representation

If the use of Pareto dominance is still necessary, it is crucial to find out a compact representation of the Pareto-optimal set in order to present DMs with a manageable collection of points that accurately describe the various options. The main challenge of methods aiming to generate a compact representation is to decide which solutions to keep and which to discard as new ones are generated. Three properties are often used to measure the quality of a representation: coverage (each Pareto-dominated point is either represented or covered by at least one point in the representation), spacing (ensures that there is adequate spacing between any two points within the representation) and cardinality (to minimize, in order to keep the representation as manageable as possible) [88, 216]. To achieve these properties, the ϵ-dominance relation (see Definition 15 in Section 2.3) is often employed.

For heuristic methods, as the coverage property is more difficult to achieve, it is more common to use indicator-based methods which rely on a quality indicator (e.g., the hypervolume [44] or the ϵ-indicator [45]), or even decomposition-based [265] methods, which are based on a division of the MOCO problem into several single-objective problems (see the recent survey of Li et al. [158]).

In Section 2.5, we have presented a method for updating a Pareto archive, based on the ND-Tree data structure, which is an unbounded archive. It would be interesting to study how to adapt the method for bounded archives. As in ND-Tree a division of the objective space is performed using a clustering approach, the idea would be to limit the number of points in each cluster. The challenge is to select the best representative points in order to keep high-quality approximations.

Other dominance relations

In Section 5, we have defined the Θ-preference dominance relation, i.e., a dominance relation where some preference information Θ is taken into account (see Definition 38 in Section 5.3.1). Kaddani et al. [136] have provided exact methods to generate the non-dominated set according to this preference dominance relation (they assume that the DM's preferences can be represented by a WS). We have also studied the integration of this preference relation into PLS [170]. Nonetheless, adapting PLS proved to be a challenging task. Indeed, when little or no information is available, the dominance relation provides enough diversity to guarantee a good exploration of the search space. When more information is integrated into the preference relation, the method converges much faster, but the search space is less explored, which can lead to the generation of solutions of poorer quality. It would therefore be interesting to develop an adaptive PLS method, i.e., one capable of adapting to the dominance relation. Indeed, the more restricted the dominance relationship, the more necessary it is to add search diversification mechanisms (e.g., adding dominated solutions to the population). This topic is also closely linked to the automatic algorithm configuration domain [195], would deserve to be studied in the context of PLS with preference information. A step further will be to dynamically updating the parameters of the algorithm according to the instances or to the behavior of the algorithm during the search [34], and to attain an "any-time" behavior, i.e., whatever the time allowed to the algorithm, the best set of parameters is used in order to get the best approximation that is possible to reach for the time considered [71, 77].

Machine learning based methods

In recent years, interest in machine learning (ML) techniques to address CO problems has grown considerably [32]. One of the reasons for this increased attention is the considerable number of practical problems that have to be solved repeatedly with slight variations in the input parameters. Consequently, it is possible to improve performance with ML by targeting specific problem distributions, as traditional solvers lack the ability to identify similarities with previously encountered instances. Note that CO techniques (e.g., metaheuristics) have also been used to address ML problems, in particular for the general problem of extracting information from large data sets [69].

As Bengio et al. [32] points out, there are two main paradigms for using ML in CO: the first involves approximating expert knowledge through imitation (supervised learning [63]), while the second aims to go beyond this knowledge by learning new policies through experience (reinforcement learning [238]). The authors also classify the different strategies employed to achieve these goals into three distinct groups. The first category involves the direct replacement of conventional methods by ML models. The second category uses ML to refine algorithm parameters, including initializing algorithms with candidate solutions. The third approach, perhaps the most extensively studied, revolves around a hybridization of techniques. For example, there is a considerable body of literature on exploiting ML to improve exact solvers by replacing repetitive decision processes in traditional algorithms, e.g., best-cut prediction or node and variable selection in branch and bound methods.

Concerning the direct application of ML techniques to solve MO problems, to our knowledge, there are only very few results. Even for single-objective optimization, there are still no completely convincing results, and existing classical solvers still outperform approaches derived directly from ML [135]. Instead, methods are used to reduce the size of problems. For example, Sun [237] tries to predict the probability of an edge belonging to an optimal solution of the TSP. Using prediction techniques for the MOTSP problem seems an ambitious challenge. Indeed, an edge usually belongs to several Pareto-optimal solutions. The idea would therefore be to predict the frequency of occurrence of an edge in the Pareto-optimal solutions. We had already carried out similar work in the context of the bi-objective TSP, where the aim was to reduce the set of candidate edges for 2-opt moves in a local search [171]. The approach was instead based on generating some supported Pareto-optimal solutions and observing the frequency of occurrence of edges in these solutions. It would therefore be interesting to compare this technique with a pure ML approach. This could be done by using specific ML techniques for graph learning, namely graph neural networks (GNNs) [48, 165]. GNN is a promising deep learning approach for graph-based learning. The fundamental design principle behind GNNs is the use of pairwise message passing, enabling graph nodes to continuously refine their representations through iterative information exchange with their neighboring nodes. To our knowledge, GNNs has only been applied to solve a MO facility location problem [164] and it would be exciting to adapt it for other MOCO problems.

We have not found in the literature any ML method combined with exact approaches to solve MOCO problems. However, there is an extensive literature on the application of ML methods, particularly deep reinforcement learning (DRL) methods, to heuristically solve MOCO problems [140, 160, 163, 263, 266]. DRL, is a branch of machine learning that unites RL with deep learning. RL focuses on the task of a computational agent acquiring decision-making abilities through iterative experimentation. DRL extends this approach by integrating deep learning techniques, enabling agents to make decisions based on unprocessed input data, using artificial neural networks. In the literature, learning-based techniques for addressing combinatorial optimization problems can be broadly categorized into two groups: end-to-end DRL approaches and heuristic search methods that are guided by DRL. Most of the DRL techniques to solve MOCO problems are end-to-end approaches: the MOCO problem is decomposed into several single-objective CO problems, and then a single model or multiple models are used to solve these sub-problems [140, 266]. One prospect would therefore be to integrate DRL techniques to guide the local search methods presented in Section 2.

Lorenz dominance

In Section 3, we have presented the Lorenz dominance and its use in multi-agent combinatorial optimization problems for generating fair solutions. An efficient algorithm for generating all Lorenz-optimal solutions of bi-objective CO problems has been presented. Its extension to problems with more than two objectives is a natural perspective. Another point that would be worth to be studied in the following: Lorenz dominance allows generating fair solutions for multi-agent problems. However, in some applications, it can happen that an auxiliary cost function must be taken into account. Let's consider for example an allocation problem where each agent has some preferences for the items to be allocated, and the allocation of an item to an agent has a certain cost. In this case, the goal is to find a fair allocation between the agents, while minimizing

the total allocation cost. Therefore, the problem goes back to the minimization of a linear function (the cost function) under the constraint that the solutions are fair (i.e., Lorenz-optimal). This topic has been widely studied in the context of Pareto dominance (finding a solution that optimizes a linear function under the constraint that the solution is Pareto-optimal) [1, 36, 166, 242]. For solving this problem, a simple method would be to compute solutions in non-decreasing order of the cost function using a k-best algorithm, until a Lorenz-optimal solution has been generated (a Lorenz efficiency test will be thus needed, to be able to decide if a solution is Lorenz-optimal or not). However, if the solution minimizing the cost function is far to be Lorenz-optimal, the convergence of the method could be really slow. It would be more relevant to investigate the recent ideas developed for Pareto dominance, based on a decomposition of the search region into a union of search zones (i.e., subset of the objective space where new non-dominated points can be found) [242].

Choquet integral

In Section 4 we have presented a method for generating all Choquet-optimal solutions of MOCO problems. The method is however only effective for bi-objective problems. We have tried to develop a method for generating only the 2-additive Choquet-optimal, but only a subset of the 2-additive Choquet-optimal points could be generated. More experiments will be needed to show the differences between WS optimal solutions, Choquet-optimal solutions and 2-additive Choquet optimal solutions of MOCO problems. It will also be interesting to study what brings exactly and concretely (for a DM) the 2-additive Choquet optimal solutions that are not WS optimal solutions, given that they are harder to compute.

Interactive methods

In Section 5 we have presented different interactive methods for solving MOCO problems: an exact method (IEEP), a genetic algorithm (RIGA) and a specific greedy/local search for problems under matroid constraints. All the methods are based on the same principle: a function linear with respect to its (unknown) parameters is used to represent the preferences of the DM, the preference information is retrieved from pairwise comparisons and min-max regret approaches are used to ensure the convergence of the methods. A primary limitation of these methods is that the preference model must be chosen at the start of the method, with no option to change the model during the resolution process. More flexibility will be thus needed, to be able to change the model during the resolution, according to the preferences indicated by the DM, as in the case of the evolutionary algorithm developed by Branke et al. [42]. Starting with the Choquet integral could also be an option, as this operator is general and can model many different ones, but our experiments showed that it was much more difficult to optimize. Moreover, in our study comparing WS-optimal points and Choquet-optimal point [169], it has been demonstrated that the probability to generate a Choquet optimal point which is not WS-optimal is very low for MOCO problems.

We would also like to take a closer look at the choice of the two solutions to be compared. Indeed, in most of our experiments, we used the CSS strategy, which consists in selecting a solution minimizing the max regret, and one of its best challengers [38]. However, in our experiments with IEEP, we have seen that it is not always the best strategy for minimizing the number of questions put to the DM. Indeed, if the solution minimizing the max regret is the preferred solution of the DM (but some more iterations are needed before the method converges), this solution will correspond to the solution of minimax regret, and the pairwise comparison put to the DM will always contain this solution, which does not bring much diversity. Ciomek et al. [53] have proposed different heuristic strategies for selecting the best pairwise comparisons in the context of MCDA problems. It will be interesting to see if these strategies could be integrated in our methods for solving MOCO problems.

Finally, the matroid problems we have dealt with only concern the maximization of a monotone submodular function. However, it would be also possible to study the *minimization* of submodular functions [182]. Surprisingly, the properties of algorithms for the minimization case differ strongly from algorithms developed for maximization [129]. For example, the problem of finding a minimum-cost cut in a graph can be solved with the Stoer-Wagner algorithm [235] in polynomial time, while finding a maximum-value cut in a graph is NP-hard [58]. On the other hand, the minimization of submodular functions under constraints becomes more difficult. For example, applying the greedy algorithm to the minimization of a submodular function under the uniform matroid constraint has a performance guarantee depending on the "curvature" of the submodular function [8]. The closer the curvature is to 0, the closer the function is to being modular (i.e., linear). So, for a totally linear function, the greedy algorithm is guaranteed to find the optimal solution, whereas for a highly "curved" submodular function, there is no longer any guarantee (whereas there is a guarantee of $(1-\frac{1}{e})$ whatever the curvature of the curve for maximization). For these problems, it would be thus advisable to study new approaches in order to obtain performance guarantees.

References

[1] M. Abbas and D. Chaabane. Optimizing a linear function over an integer efficient set. *European Journal of Operational Research*, 174(2):1140–1161, 2006.

[2] V. Aggarwal, Y.P. Aneja, and K.P.K. Nair. Minimal spanning tree subject to a side constraint. *Computer & Operations Research*, 4:287–296, 1982.

[3] S. Ahmed and A. Atamtürk. Maximizing a class of submodular utility functions. *Mathematical programming*, 128(1):149–169, 2011.

[4] R.K. Ahuja, Ö. Ergun, J.B. Orlin, and A.P. Punnen. A survey of very large-scale neighborhood search techniques. *Discrete Appl. Math.*, 123(1-3):75–102, 2002.

[5] N. Altwaijry and M. Bachir Menai. Data structures in multi-objective evolutionary algorithms. *Journal of Computer Science and Technology*, 27(6):1197–1210, 2012.

[6] M.J. Alves and J. Clímaco. A review of interactive methods for multiobjective integer and mixed-integer programming. *European Journal of Operational Research*, 180(1):99–115, 2007.

[7] K. A. Andersen, K. Jørnsten, and M. Lind. On bicriterion minimal spanning trees: an approximation. *Computers & Operations Research*, 23(12):1171–1182, 1996.

[8] W. Bai andf J.A. Bilmes. Greed is still good: Maximizing monotone submodular+supermodular (BP) functions. In J.G. Dy and A. Krause, editors, *Proceedings of the 35th International Conference on Machine Learning, ICML*, volume 80 of *Proceedings of Machine Learning Research*, pages 314–323. PMLR, 2018.

[9] Y.P. Aneja and K.P.K. Nair. Bicriteria transportation problem. *Management Science*, 25:73–78, 1979.

[10] E. Angel, E. Bampis, and L. Gourvès. A dynasearch neighborhood for the bicriteria traveling salesman problem. In X. Gandibleux, M. Sevaux, K. Sörensen, and V. T'kindt, editors, *Metaheuristics for Multiobjective Optimisation*, pages 153–176. Springer. Lecture Notes in Economics and Mathematical Systems Vol. 535, Berlin, 2004.

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

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

[13] M. Barth and K. Boriboonsomsin. Real-world carbon dioxide impacts of traffic congestion. *Transportation Research Record*, 1(2058):163–171, 2008.

[14] M. Barth, T. Younglove, and G. Scora. Development of a heavy-duty diesel modal emissions and fuel consumption model. Technical report, California Partners for Advanced Transit and Highways (PATH), UC Berkeley, 2005.

[15] R. Battiti and A. Passerini. Brain-computer evolutionary multiobjective optimization: a genetic algorithm adapting to the decision maker. *IEEE Transactions on Evolutionary Computation*, 14(5):671–687, 2010.

[16] C. Bazgan, H. Hugot, and D. Vanderpooten. Solving efficiently the 0-1 multi-objective knapsack problem. *Computers & Operations Research*, 36(1):260–279, 2009.

[17] T. Bektaş and G. Laporte. The pollution-routing problem. *Transportation Research Part B: Methodological*, 45 (8):1232–1250, 2011.

[18] N. Benabbou and T. Lust. A general interactive approach for solving multi-objective combinatorial optimization problems with imprecise preferences. In *Proceedings of the Twelfth International Symposium on Combinatorial Search, SOCS*, pages 164–165. AAAI Press, 2019.

[19] N. Benabbou and T. Lust. An interactive polyhedral approach for multi-objective combinatorial optimization with incomplete preference information. In *Scalable Uncertainty Management - 13th International Conference, SUM*, volume 11940 of *Lecture Notes in Computer Science*, pages 221–235. Springer, 2019.

[20] N. Benabbou and P. Perny. On possibly optimal tradeoffs in multicriteria spanning tree problems. In T. Walsh, editor, *Algorithmic Decision Theory - 4th International Conference, ADT*, volume 9346 of *Lecture Notes in Computer Science*, pages 322–337. Springer, 2015.

[21] N. Benabbou and P. Perny. Incremental weight elicitation for multiobjective state space search. In *The Twenty-Ninth AAAI Conference on Artificial Intelligence*, pages 1093–1099. AAAI Press, 2015.

[22] N. Benabbou and P. Perny. Solving multi-agent knapsack problems using incremental approval voting. In *22nd European Conference on Artificial Intelligence, ECAI*, pages 1318–1326, 2016.

[23] N. Benabbou and P. Perny. Interactive resolution of multiobjective combinatorial optimization problems by incremental elicitation of criteria weights. *EURO Journal on Decision Processes*, 6(3):283–319, 2018.

[24] N. Benabbou, P. Perny, and P. Viappiani. Incremental elicitation of Choquet capacities for multicriteria choice, ranking and sorting problems. *Artificial Intelligence*, 246:152–180, 2017.

[25] N. Benabbou, L. Galand, and T. Lust. Experimental analysis of greedy strategies to minimize pairwise comparisons in multicriteria decision aiding. In *90th meeting of the Euro working group on Multi-Criteria Decision Aiding (EWG-MCDA 90)*, 2019.

[26] N. Benabbou, C. Leroy, T. Lust, and P. Perny. Combining local search and elicitation for multi-objective combinatorial optimization. In *Algorithmic Decision Theory - 6th International Conference, ADT*, pages 1–16, 2019.

[27] N. Benabbou, C. Leroy, and T. Lust. Regret-based elicitation for solving multi-objective knapsack problems with rank-dependent aggregators. In *24th European Conference on Artificial Intelligence, ECAI*, pages 419–426, 2020.

[28] N. Benabbou, C. Leroy, and T. Lust. An interactive regret-based genetic algorithm for solving multi-objective combinatorial optimization problems. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 2335–2342. AAAI Press, 2020.

[29] N. Benabbou, C. Leroy, T. Lust, and P. Perny. Combining preference elicitation with local search and greedy search for matroid optimization. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence*, pages 12233–12240. AAAI Press, 2021.

[30] N. Benabbou, C. Leroy, T. Lust, and P. Perny. Interactive optimization of submodular functions under matroid constraints. In *Algorithmic Decision Theory - 7th International Conference, ADT*, Lecture Notes in Computer Science, pages 1–15. Springer, 2021.

[31] R. Benayoun, J. de Montgolfier, J. Tergny, and O. Laritchev. Linear programming with multiple objective functions: Step method (stem). *Mathematical Programming*, 1(1):366–375, 1971.

[32] Y. Bengio, A. Lodi, and A. Prouvost. Machine learning for combinatorial optimization: A methodological tour d'horizon. *European Journal of Operational Research*, 290(2):405–421, 2021.

[33] J.-F. Bérubé, M. Gendreau, and J-Y. Potvin. An exact epsilon-constraint method for bi-objective combinatorial optimization problems: application to the traveling salesman problem with profits. *European Journal of Operational Research*, 194:39–50, 2009.

[34] A. Blot, M.-E. Marmion, L. Jourdan, and H.H. Hoos. Automatic configuration of multi-objective local search algorithms for permutation problems. *Evolutionary Compututation*, 27(1):147–171, 2019.

[35] F. Bökler and P. Mutzel. Output-sensitive algorithms for enumerating the extreme nondominated points of multiobjective combinatorial optimization problems. In N. Bansal and I. Finocchi, editors, *Algorithms - ESA*, pages 288–299. Springer Berlin Heidelberg, 2015.

[36] N. Boland, H. Charkhgard, and M. Savelsbergh. A new method for optimizing a linear function over the efficient set of a multiobjective integer program. *European Journal of Operational Research*, 260(3):904–919, 2017.

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

[38] C. Boutilier, R. Patrascu, P. Poupart, and D. Schuurmans. Constraint-based optimization and utility elicitation using the minimax decision criterion. *Artifical Intelligence*, 170(8–9):686–713, 2006.

[39] V.J. Bowman. On the relationship of the Tchebycheff norm and the efficient frontier of multiple-criteria objectives. In H. Thiriez and S. Zionts, editors, *Multiple criteria decision making*, pages 76–86. Springer. Lecture Notes in Economics and Mathematical Systems, 1976.

[40] J. Branke, K. Deb, K. Miettinen, and R. Slowiński, editors. *Multiobjective Optimization: Interactive and Evolutionary Approaches*. Springer-Verlag, 2008.

[41] J. Branke, S. Greco, R. Słowiński, and P. Zielniewicz. Learning value functions in interactive evolutionary multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 19(1):88–102, 2015.

[42] J. Branke, S. Corrente, S. Greco, R. Slowinski, and P. Zielniewicz. Using Choquet integral as preference model in interactive evolutionary multiobjective optimization. *European Journal of Operational Research*, 250(3): 884–901, 2016.

[43] D. Braziunas and C. Boutilier. Minimax regret based elicitation of generalized additive utilities. In *Twenty-Third Conference on Uncertainty in Artificial Intelligence, UAI*, pages 25–32, 2007.

[44] K. Bringmann and T. Friedrich. Convergence of hypervolume-based archiving algorithms. *IEEE Transactions on Evolutionary Computation*, 18(5):643–657, 2014.

[45] K. Bringmann, T. Friedrich, and P. Klitzke. Two-dimensional subset selection for hypervolume and epsilon-indicator. In *Annual Conference on Genetic and Evolutionary Computation, GECCO*, page 589–596, 2014.

[46] E. Brochu, F. Nando D, and G. Abhijeet. Active preference learning with discrete choice data. In *Advances in neural information processing systems*, pages 409–416, 2008.

[47] G. Calinescu, C. Chekuri, M. Pal, and J. Vondrák. Maximizing a monotone submodular function subject to a matroid constraint. *SIAM Journal on Computing*, 40(6):1740–1766, 2011.

[48] Q. Cappart, D. Chételat, E.B. Khalil, A. Lodi, C. Morris, and P. Velickovic. Combinatorial optimization and reasoning with graph neural networks. *Journal of Machine Learning Research*, 24:130:1–130:61, 2023.

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

[50] U. Chajewska, D. Koller, and R. Parr. Making rational decisions using adaptive utility elicitation. In *Seventeenth National Conference on Artificial Intelligence*, pages 363–369, 2000.

[51] A. Chateauneuf and J-Y. Jaffray. Some characterizations of lower probabilities and other monotone capacities through the use of Möbius inversion. *Mathematical Social Sciences*, 17(3):263–283, 1989.

[52] G. Choquet. Theory of capacities. *Annales de l'Institut Fourier*, 5:131–295, 1953.

[53] K. Ciomek, M. Kadzinski, and T. Tervonen. Heuristics for selecting pair-wise elicitation questions in multiple criteria choice problems. *European Journal of Operational Research*, 262:693–707, 2017.

[54] V. N. Coelho, T. A. Oliveira, I. M. Coelho, B. N. Coelho, P. J. Fleming, F. G. Guimarães, H. R. D. Lourenço, M.J.F. Souza, E.-G. Talbi, and T. Lust. Generic Pareto local search metaheuristic for optimization of targeted offers in a bi-objective direct marketing campaign. *Computers & Operations Research*, 78:578–587, 2017.

[55] V.N. Coelho, M.J.F. Souza, I.M. Coelho, F.G. Guimarães, T. Lust, and R.C. Cruz. Multi-objective approaches for the open-pit mining operational planning problem. *Electronic Notes in Discrete Mathematics*, 39:233–240, 2012.

[56] J.L. Cohon. *Multiobjective Programming and Planning*. Academic Press, New York, 1978.

[57] E.F. Combarro, I. Díaz, and P. Miranda. On random generation of fuzzy measures. *Fuzzy Sets and Systems*, 228:64–77, 2013.

[58] C.W. Commander. *Maximum cut problem, MAX-CUTMaximum Cut Problem, MAX-CUT*, pages 1991–1999. Springer US, Boston, MA, 2009.

[59] H. W. Corley. Efficient spanning trees. *Journal of Optimization Theory and Applications*, 45(3):481–485, 1985.

[60] M. Cornu. *Local search, data structures and Monte Carlo search for multi-objective combinatorial optimization problems*. PhD thesis, Université Paris-Dauphine, Paris, 2017.

[61] M. Cornu, T. Cazenave, and D. Vanderpooten. Perturbed decomposition algorithm applied to the multi-objective traveling salesman problem. *Computers & Operations Research*, 79:314 – 330, 2017.

[62] L. Costa, T. Lust, R. Kramer, and A. Subramanian. A two-phase Pareto local search heuristic for the bi-objective pollution-routing problem. *Networks*, 72(3):311–336, 2018.

[63] P. Cunningham, M. Cord, and S.J. Delany. *Supervised Learning*, pages 21–49. Springer, 2008.

[64] K. Deb, A. Pratap, S. Agarwal, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[65] K. Deb, M. Mohan, and S. Mishra. Evaluating the ϵ-domination based multi-objective evolutionary algorithm for a quick computation of Pareto-optimal solutions. *IEEE Transactions on Evolutionary Computation*, 13(4): 501–525, 2005.

[66] K. Deb, A. Sinha, P. J. Korhonen, and J. Wallenius. An interactive evolutionary multiobjective optimization method based on progressively approximated value functions. *IEEE Transactions on Evolutionary Computation*, 14(5):723–739, 2010.

[67] E. Demir, T. Bektaş, and G. Laporte. A comparative analysis of several vehicle emission models for road freight transportation. *Transportation Research Part D: Transport and Environment*, 16(5):347–357, 2011.

[68] E. Demir, T. Bektaş, and G. Laporte. An adaptive large neighborhood search heuristic for the pollution-routing problem. *European Journal of Operational Research*, 223(2):346–359, 2012.

[69] C. Dhaenens and L. Jourdan. Metaheuristics for data mining: survey and opportunities for big data. *Annals of Operations Research*, 314(1):117–140, 2022.

[70] Y. Dodge. *Gini Index*, pages 231–233. Springer New York, 2008.

[71] M.Á. Domínguez-Ríos, F. Chicano, and E. Alba. Effective anytime algorithm for multiobjective combinatorial optimization problems. *Information Sciences*, 565:210–228, 2021.

[72] M. Drozdík, Y. Akimoto, H. Aguirre, and K. Tanaka. Computational cost reduction of nondominated sorting using the M-front. *IEEE Transactions on Evolutionary Computation*, 19(5):659–678, 2015.

[73] M.M. Drugan and D. Thierens. Path-guided mutation for stochastic Pareto local search algorithms. In R. Schaefer, C. Cotta, J. Kolodziej, and G. Rudolph, editors, *Parallel Problem Solving from Nature - PPSN XI*, volume 6238 of *Lecture Notes in Computer Science*, pages 485–495. Springer, 2010.

[74] M.M. Drugan and D. Thierens. Stochastic pareto local search: Pareto neighbourhood exploration and perturbation strategies. *Journal of Heuristics*, 18(5):727–766, 2012.

[75] J. Drummond and C. Boutilier. Preference elicitation and interview minimization in stable matchings. In *Proceedings of AAAI'14*, pages 645–653, 2014.

[76] J. Dubois-Lacoste, M. López-Ibáñez, and T. Stützle. A hybrid TP+PLS algorithm for bi-objective flow-shop scheduling problems. *Computers & Operations Research*, 38(8):1219–1236, 2011.

[77] J. Dubois-Lacoste, M. López-Ibáñez, and T. Stützle. Anytime Pareto local search. *European Journal of Operational Research*, 243(2):369–385, 2015.

[78] K. Dächert, J. Gorski, and K. Klamroth. An augmented weighted Tchebycheff method with adaptively chosen parameters for discrete bicriteria optimization problems. *Computers & Operations Research*, 39:2929–2943, 12 2012.

[79] J. Edmonds. Matroids and the greedy algorithm. *Mathematical programming*, 1(1):127–136, 1971.

[80] B. Egon. A class of location, distribution and scheduling problems: modeling and solution methods. *ill. Carnegie Mellon University, Pittsburgh, PA: Design Research Center*, pages 1–21, 1982.

[81] M. Ehrgott. *Multicriteria Optimization. Second edition.* Springer, Berlin, 2005.

[82] M. Ehrgott and X. Gandibleux. Approximative solution methods for multiobjective combinatorial optimization. *Top 12*, pages 1–63, 2004.

[83] M. Ehrgott and K. Klamroth. Connectedness of efficient solutions in multiple criteria combinatorial optimization. *European Journal of Operational Research*, 97(1):159–166, 1997.

[84] M. Ehrgott, C. Guler, H. W. Hamacher, and L. Shao. Mathematical optimization in intensity modulated radiation therapy. *4OR: a Quarterly Journal of Operations Research*, 6(3):199–262, 2008.

[85] M. Ehrgott, X. Gandibleux, and A. Przybylski. *Exact methods for multi-objective combinatorial optimisation*, pages 817–850. Springer New York, 2016.

[86] D. Eppstein. k-best enumeration, 2014.

[87] A.E. Ezugwu, A.M. Ikotun, O.O. Oyelade, L. Abualigah, J.O. Agushaka, C.I. Eke, and A.A. Akinyelu. A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects. *Engineering Applications of Artificial Intelligence*, 110:104743, 2022.

[88] S. Faulkenberg and M.M. Wiecek. On the quality of discrete representations in multiple objective programming. *Optimization and Engineering*, 11(3):423–440, 2010.

[89] J. E. Fieldsend, R. M. Everson, and S. Singh. Using unconstrained elite archives for multiobjective optimization. *IEEE Transactions on Evolutionary Computation*, 7(3):305–323, 2003.

[90] J.E. Fieldsend. Data structures for non-dominated sets: implementations and empirical assessment of two decades of advances. In Carlos Artemio Coello Coello, editor, *GECCO '20: Genetic and Evolutionary Computation Conference*, pages 489–497. ACM, 2020.

[91] J.E. Fieldsend. On the active use of an ND-Tree-based archive for multi-objective optimisation. *Emma Hart*, page 23, 2022.

[92] M.L. Fisher, G.L. Nemhauser, and L.A. Wolsey. *An analysis of approximations for maximizing submodular set functions-II*, volume 8, pages 73–87. Springer, 1978.

[93] K. Florios and G. Mavrotas. Generation of the exact Pareto set in multi-objective traveling salesman and set covering problems. *Applied Mathematics and Computation*, 237:1–19, 2014.

[94] N. Forget, S.L. Gadegaard, K. Klamroth, L.R. Nielsen, and A. Przybylski. Branch-and-bound and objective branching with three or more objectives. *Computers & Operations Research*, 148:106–118, 2022.

[95] H. Fouchal, X. Gandibleux, and F. Lehuédé. Preferred solutions computed with a label setting algorithm based on Choquet integral for multi-objective shortest paths. In *IEEE Symposium on Computational Intelligence in Multicriteria Decision-Making (MDCM)*, pages 143 – 150. IEEE, 2011.

[96] J. Fürnkranz and E. Hüllermeier. *Preference learning*. Springer, 2010.

[97] H.N. Gabow, J.L. Bentley, and R.E. Tarjan. Scaling and related techniques for geometry problems. In *Proceedings of the Sixteenth Annual ACM Symposium on Theory of Computing*, STOC '84, pages 135–143. ACM, 1984.

[98] L. Galand and T. Lust. Exact methods for computing all Lorenz optimal solutions to biobjective problems. In *Algorithmic Decision Theory, 4th International Conference, ADT*, volume 9346 of *Lecture Notes in Computer Science*, pages 305–321. Springer, 2015.

[99] L. Galand and T. Lust. Multiagent fair optimization with Lorenz dominance. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS*, pages 1895–1896. ACM, 2015.

[100] L. Galand, P. Perny, and O. Spanjaard. Choquet-based optimisation in multiobjective shortest path and spanning tree problems. *European Journal of Operational Research*, 204(2):303–315, 2010.

[101] L. Galand, P. Perny, and O. Spanjaard. A branch and bound algorithm for Choquet optimization in multicriteria problems. In *Proceedings Lecture Notes in Economics and Mathematical Systems*, volume 634, pages 355–365, 2011.

[102] L. Galand, J. Lesca, and P. Perny. Dominance rules for the Choquet integral in multiobjective dynamic programming. In *Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence IJCAI*, pages 538–544. AAAI Press, 2013.

[103] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness (Series of Books in the Mathematical Sciences)*. W. H. Freeman, first edition edition, 1979.

[104] M. Gelain, M.S. Pini, F. Rossi, K. B. Venable, and T. Walsh. Elicitation strategies for soft constraint problems with missing preferences: Properties, algorithms and experimental studies. *Artificial Intelligence*, 174(3):270–294, 2010.

[105] F. Glover and G. Kochenberger. *Handbook of Metaheuristics*. Kluwer, Boston, 2003.

[106] J. Gorski, K. Klamroth, and S. Ruzika. Connectedness of efficient solutions in multiple objective combinatorial optimization. *Journal of Optimization Theory and Applications*, 150(3):475–497, 2011.

[107] M. Grabisch. The application of fuzzy integrals in multicriteria decision making. *European Journal of Operational Research*, 89:445–456, 1996.

[108] M. Grabisch. k-order additive discrete fuzzy measures and their representation. *Fuzzy Sets and Systems*, 92: 167–189, 1997.

[109] M. Grabisch and C. Labreuche. A decade of application of the Choquet and Sugeno integrals in multi-criteria decision aid. *4OR*, 6(1):1–44, 2008.

[110] M. Grabisch and M. Roubens. Application of the Choquet integral in multicriteria decision making. *Computer Science*, pages 1–27, 2008.

[111] M. Grabisch, I. Kojadinovic, and P. Meyer. A review of methods for capacity identification in Choquet integral based multi-attribute utility theory: applications of the kappalab R package. *European Journal of Operational Research*, 186(2):766–785, 2008.

[112] M. Grabisch, J-C. Marichal, R. Mesiar, and E. Pap. *Aggregation functions*, volume 127 of *Encyclopedia of Mathematics and its Applications*. Cambridge University Press, Cambridge, UK, 2009.

[113] A.P. Guerreiro, C.M. Fonseca, and L. Paquete. The hypervolume indicator: computational problems and algorithms. *ACM Computing Surveys*, 54(6), 2021.

[114] P. Gupta, R. Janardan, M. Smid, and B. Dasgupta. The rectangle enclosure and point-dominance problems revisited. *International Journal of Computational Geometry & Applications*, 7(5):437–455, 1997.

[115] V. Ha and P. Haddawy. Problem-focused incremental elicitation of multi-attribute utility models. In *Proceedings of the Thirteenth Conference on Uncertainty in Artificial Intelligence*, pages 215–222. Morgan Kaufmann, 1997.

[116] W. Habenicht. *Essays and surveys on multiple criteria decision making: proceedings of the fifth international conference on multiple criteria mecision making, MCDM*, chapter Quad trees, a datastructure for discrete vector optimization problems, pages 136–145. Springer Berlin Heidelberg, 1983.

[117] Y. Haimes, L. Lasdon, and D. Wismer. On a bicriterion formulation of the problems of integrated system identification and system optimization. *IEEE Transactions on Systems, Man, and Cybernetics*, 1:296–297, 1971.

[118] P. Halffmann, L.E. Schäfer, K. Dächert, K. Klamroth, and S. Ruzika. Exact algorithms for multiobjective linear optimization problems with integer variables: A state of the art survey. *Journal of Multi-Criteria Decision Analysis*, 29(5-6):341–363, 2022.

[119] H.W. Hamacher and G. Ruhe. On spanning tree problems with multiple objectives. *Annals of Operations Research*, 52:209–230, 1994.

[120] P. Hansen and N. Mladenovic. Variable neighborhood search: principles and applications. *European Journal of Operational Research*, 130(3):449 – 467, 2001.

[121] G.H. Hardy, J.E. Littlewood, G. Pólya, G. Pólya, et al. *Inequalities*. Cambridge university press, 1952.

[122] J. A. Hartigan and M. A. Wong. A k-means clustering algorithm. *JSTOR: Applied Statistics*, 28(1):100–108, 1979.

[123] A. Herzel, S. Ruzika, and C. Thielen. Approximation methods for multiobjective optimization problems: a survey. *INFORMS Journal on Computing*, 33(4):1284–1299, 2021.

[124] D.S. Hochbaum and A. Pathria. Analysis of the greedy approach in problems of maximum k-coverage. *Naval Research Logistics (NRL)*, 45(6):615–627, 1998.

[125] T. Holzmann and J.C. Smith. Solving discrete multi-objective optimization problems using modified augmented weighted Tchebychev scalarizations. *European Journal of Operational Research*, 271(2):436–449, 2018.

[126] H. Hoos and T. Stützle. *Stochastic Local Search: Foundations and Applications*. Elsevier, 2004.

[127] H.H. Hoos and T. Stützle. Stochastic local search algorithms: an overview. In *Springer Handbook of Computational Intelligence*, pages 1085–1105. Springer, 2015.

[128] M. Inja, C. Kooijman, M. De Waard, D. Roijers, and S. Whiteson. Queued Pareto local search for multi-objective optimization. In *Proceedings of the Thirteenth International Conference on Parallel Problem Solving from Nature, PPSN*, pages 589–599, 2014.

[129] R.K. Iyer, S. Jegelka, and J.A. Bilmes. Fast semidifferential-based submodular function optimization. In *Proceedings of the 30th International Conference on Machine Learning, ICML*, volume 28 of *JMLR Workshop and Conference Proceedings*, pages 855–863. JMLR.org, 2013.

[130] A. Jaszkiewicz. Interactive multiobjective optimization with the Pareto memetic algorithm. *Foundations of Computing and Decision Sciences*, 32:15–32, 2004.

[131] A. Jaszkiewicz. Improved quick hypervolume algorithm. *Computers & Operations Research*, 90:72–83, 2018.

[132] A. Jaszkiewicz. Many-objective Pareto local search. *European Journal of Operational Research*, 271(3):1001–1013, 2018.

[133] A. Jaszkiewicz and T. Lust. Proper balance between search towards and along Pareto front: biobjective TSP case study. *Annals of Operations Research*, 254(1-2):111–130, 2017.

[134] A. Jaszkiewicz and T. Lust. ND-tree-based update: A fast algorithm for the dynamic nondominance problem. *IEEE Transactions on Evolutionary Computation*, 22(5):778–791, 2018.

[135] C. Joshi, Q. Cappart, L-M. Rousseau, and T. Laurent. Learning the travelling salesperson problem requires rethinking generalization. *Constraints*, 27:1–29, 04 2022.

[136] S. Kaddani, D. Vanderpooten, J.M. Vanpeperstraete, and H. Aissi. Weighted sum model with partial preference information: application to multi-objective optimization. *European Journal of Operational Research*, 260(2): 665–679, 2017.

[137] L. Ke, Q. Zhang, and R. Battiti. Hybridization of decomposition and local search for multiobjective optimization. *IEEE Transactions on Cybernetics*, 44(10):1808–1820, 2014.

[138] Y. Kergosien, A. Giret, E. Neron, and G. Sauvanet. An efficient label-correcting algorithm for the multi-objective shortest path problem. *INFORMS Journal on Computing*, 2021.

[139] G. Kirlik and S. Sayın. A new algorithm for generating all nondominated solutions of multiobjective discrete optimization problems. *European Journal of Operational Research*, 232(3):479–488, 2014.

[140] K.Li, T. Zhang, and R. Wang. Deep reinforcement learning for multiobjective optimization. *IEEE Transactions on Cybernetics*, 51(6):3103–3114, 2021.

[141] J. Knowles and D. Corne. The Pareto archived evolution strategy: a new baseline algorithm for multiobjective optimisation. In *1999 Congress on Evolutionary Computation*, pages 98–105. IEEE Service Center, 1999.

[142] J.D. Knowles and D.W. Corne. Enumeration of Pareto optimal multi-criteria spanning trees – a proof of the incorrectness of Zhou and Gen's proposed algorithm. *European Journal of Operational Research*, 143(3):543–547, 2002.

[143] J.D. Knowles, R.A. Watson, and D.W. Corne. Reducing Local Optima in Single-Objective Problems by Multi-objectivization. In *Proceedings of the First International Conference on Evolutionary Multi-Criterion Optimization, EM0*, pages 268–282. Springer-Verlag. Lecture Notes in Computer Science No. 1993, 2001.

[144] C. Kooijman, M. De Waard, M. Inja, D. Roijers, and S. Whiteson. Pareto local policy search for MOMDP planning. In *ESANN 2015: Proceedings of the 23rd European Symposium on Artificial Neural Networks, Special Session on Emerging Techniques and Applications in Multi-Objective Reinforcement Learning*, pages 53–58, 2015.

[145] P.J. Korhonen and J. Laakso. A visual interactive method for solving the multiple criteria problem. *European Journal of Operational Research*, 24(2):277 – 287, 1986. Mathematical Programming Multiple Criteria Decision Making.

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

[147] M.M. Kostreva, W. Ogryczak, and A. Wierzbicki. Equitable aggregations and multiple criteria analysis. *European Journal of Operational Research*, 158(2):362–377, 2004. Methodological Foundations of Multi-Criteria Decision Making.

[148] R. Kramer, A. Subramanian, T. Vidal, and L.A.F. Cabral. A matheuristic approach for the pollution-routing problem. *European Journal of Operational Research*, 243(2):523–539, 2015.

[149] J.B. Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical society*, 7(1):48–50, 1956.

[150] H. T. Kung, F. Luccio, and F. P. Preparata. On finding the maxima of a set of vectors. *Journal of the ACM*, 22(4):469–476, 1975.

[151] R. Lacour. *Exact and approximate solving approaches in multi-objective combinatorial optimization, application to the minimum weight spanning tree problem.* PhD thesis, Université Paris-Dauphine, Paris, 2014.

[152] Ailsa H. Land and Alison G. Doig. An automatic method of solving discrete programming problems. *Econometrica*, 28:497, 1960.

[153] B. Lang. Space-partitioned nd-trees for the dynamic nondominance problem. *IEEE Transactions on Evolutionary Computation*, 26(5):1004–1014, 2022.

[154] M. Laumanns, L. Thiele, and E. Zitzler. An adaptative scheme to generate the Pareto front based on the epsilon-constraint method. Technical Report 199, Technischer Bericht, Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH), 2004.

[155] B. Lehmann, D. Lehmann, and N. Nisan. Combinatorial auctions with decreasing marginal utilities. *Games and Economic Behavior*, 55(2):270–296, 2006.

[156] M.S. Leitner, I. Ljubić, and M. Sinnl. A computational study of exact approaches for the bi-objective prize-collecting steiner tree problem. *INFORMS Journal on Computing*, 27(1):118–134, 2015.

[157] C. Leroy. *Incremental Elicitation combined with heuristic methods for solving multi-objective combinatorial optimization problems*. PhD thesis, Sorbonne Université, Paris, France, 2022.

[158] M. Li, M. López-Ibáñez, and X. Yao. Multi-objective archiving, 2023.

[159] X. Li, H. Chehade, F. Yalaoui, and L. Amodeo. 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)*, pages 1–6, 2011.

[160] Y. Li, J. Wang, and Z. Liu. An adaptive multi-objective evolutionary algorithm with two-stage local search for flexible job-shop scheduling. *International Journal of Computational Intelligence Systems*, 14(1):54–66, 2021.

[161] A. Liefooghe, J. Humeau, S. Mesmoudi, L. Jourdan, and E-G. Talbi. On dominance-based multiobjective local search: design, implementation and experimental analysis on scheduling and traveling salesman problems. *Journal of Heuristics*, 18(2):317–352, 2012.

[162] S. Lin and B.W. Kernighan. An effective heuristic algorithm for the traveling-salesman problem. *Operations Research*, 21:498–516, 1973.

[163] X. Lin, Z. Yang, and Q. Zhang. Pareto set learning for neural multi-objective combinatorial optimization. In *The Tenth International Conference on Learning Representations, ICLR*. OpenReview.net, 2022.

[164] S. Liu, X. Yan, and Y. Jin. End-to-end Pareto set prediction with graph neural networks for multi-objective facility location. In *Proceedings of the Twelfth International Conference on Evolutionary Multi-criterion Optimization, EMO*, pages 147–161. Springer Nature Switzerland, 2023.

[165] Z. Liu and J. Zhou. *Introduction to graph neural networks*, volume 45 of *Synthesis lectures on artificial intelligence and machine learning*. Morgan & Claypool Publishers, 2020.

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

[167] H.R. Lourenço, O.C. Martin, and T. Stützle. Iterated local search: framework and applications. In M. Gendreau and J.-Y. Potvin, editors, *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*, pages 363–397. Springer US, 2010.

[168] T. Lu and C. Boutilier. Robust approximation and incremental elicitation in voting protocols. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence, IJCAI*, page 287–293. AAAI Press, 2011.

[169] T. Lust. Choquet integral versus weighted sum in multicriteria decision contexts. In *Algorithmic Decision Theory - 6th International Conference, ADT*, volume 9346 of *Lecture Notes in Computer Science*, pages 288–304. Springer, 2015.

[170] T. Lust. Interactive Pareto local search with imprecise trade-offs. In *Proceedings of DA2PL: from Multiple Criteria Decision Aid to Preference Learning*, pages 1–8, 2018.

[171] T. Lust and A. Jaszkiewicz. Speed-up techniques for solving large-scale biobjective TSP. *Computers & Operations Research*, 37:521–533, 2010.

[172] T. Lust and A. Rolland. Choquet optimal set in biobjective combinatorial optimization. *Computers & Operations Research*, 40(10):2260–2269, 2013.

[173] T. Lust and A. Rolland. On the computation of Choquet optimal solutions in multicriteria decision contexts. In *Proceedings of the 7th Multi-Disciplinary International Workshop on Artificial Intelligence (MIWAI)*, pages 131–142, 2013.

[174] T. Lust and A. Rolland. 2-additive Choquet optimal solutions in multiobjective optimization problems. In *Proceedings of the 15th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems IPMU*, volume 442 of *Communications in Computer and Information Science*, pages 256–265. Springer, 2014.

[175] T. Lust and J. Teghem. Two-phase Pareto local search for the biobjective traveling salesman problem. *Journal of Heuristics*, 16(3):475–510, 2010.

[176] T. Lust and J. Teghem. The multiobjective traveling salesman problem: a survey and a new approach. In C. Coello Coello, C. Dhaenens, and L. Jourdan, editors, *Advances in Multi-Objective Nature Inspired Computing*, volume 272 of *Studies in Computational Intelligence*, pages 119–141. Springer Berlin Heidelberg, 2010.

[177] T. Lust and J. Teghem. The multiobjective multidimensional knapsack problem: a survey and a new approach. *International Transactions in Operational Research*, 19(4):495–520, 2012.

[178] T. Lust and D. Tuyttens. Variable and large neighborhood search to solve the multiobjective set covering problem. *Journal of Heuristics*, 20(2):165–188, 2014.

[179] T. Lust, N. Meskens, and T. Monteiro. Ordonnancement multiobjectif du bloc opératoire avec une prise en compte d'une affectation équilibrée des compétences des infirmières. In *MOSIM International Conference on Modeling, Optimization and Simulation*, pages 1–10, 2012.

[180] E.Q.V. Martins. On a multicriteria shortest path problem. *European Journal of Operational Research*, 16(2): 236–245, 1984.

[181] G. Mavrotas. Effective implementation of the ϵ-constraint method in multi-objective mathematical programming problems. *Applied Mathematics and Computation*, 213(2):455–465, 2009.

[182] S.T. McCormick. Submodular function minimization. In K. Aardal, G.L. Nemhauser, and R. Weismantel, editors, *Discrete Optimization*, volume 12 of *Handbooks in Operations Research and Management Science*, pages 321–391. Elsevier, 2005.

[183] M. Mesquita-Cunha, J.R. Figueira, and A.P. Barbosa-Póvoa. New ϵ-constraint methods for multi-objective integer linear programming: A Pareto front representation approach. *European Journal of Operational Research*, 306(1):286–307, 2023.

[184] P. Meyer and M. Pirlot. On the expressiveness of the additive value function and the Choquet integral models. In *Proceedings of DA2PL: from Multiple Criteria Decision Aid to Preference Learning*, pages 48–56, 2012.

[185] K. Miettinen, F. Ruiz, and A.P. Wierzbicki. *Introduction to multiobjective mptimization: interactive approaches*, pages 27–57. Springer Berlin/Heidelberg, 2008.

[186] M. Minoux. Accelerated greedy algorithms for maximizing submodular set functions. In *Optimization techniques*, pages 234–243. Springer, 1978.

[187] S. Mostaghim and J. Teich. Quad-trees: a data structure for storing Pareto sets in multiobjective evolutionary algorithms with elitism. In *Evolutionary Multiobjective Optimization, Book Series: Advanced Information and Knowledge*, pages 81–104. Springer-Verlag London, 2004.

[188] S. Mostaghim, J. Teich, and A. Tyagi. Comparison of data structures for storing Pareto-sets in MOEAs. In *Proceedings of the Congress on Evolutionary Computation (CEC)*, volume 1, pages 843–848, 2002.

[189] Y. Nan, K. Shang, H. Ishibuchi, and L. He. Reverse strategy for non-dominated archiving. *IEEE Access*, 8: 119458–119469, 2020.

[190] G.L. Nemhauser and L.A. Wolsey. Maximizing submodular set functions: formulations and analysis of algorithms. In *North-Holland Mathematics Studies*, volume 59, pages 279–301. Elsevier, 1981.

[191] G.L. Nemhauser, L.A. Wolsey, and M.L. Fisher. An analysis of approximations for maximizing submodular set functions-I. *Mathematical programming*, 14(1):265–294, 1978.

[192] W. Ogryczak. Inequality measures and equitable approaches to location problems. *European Journal of Operational Research*, 122(2):374–391, 2000.

[193] V. Ojalehto, K. Miettinen, and M.M. Mäkelä. Interactive software for multiobjective optimization: IND-NIMBUS. *WSEAS Transactions on Computers*, 6:87–94, 2007.

[194] Ö. Özpeynirci and M. Köksalan. An exact algorithm for finding extreme supported nondominated points of multiobjective mixed integer programs. *Management Sciences*, 56(12):2302–2315, 2010.

[195] G. Papa and C. Doerr. Dynamic control parameter choices in evolutionary computation: GECCO. In *Proceedings of the Genetic and Evolutionary Computation Conference GECCO*, pages 927–956. ACM, 2020.

[196] C.H. Papadimitriou and M. Yannakakis. On the approximability of trade-offs and optimal access of web sources. *Proceedings 41st Annual Symposium on Foundations of Computer Science*, pages 86–92, 2000.

[197] L. Paquete, M. Chiarandini, and T. Stützle. *Pareto local optimum sets in the biobjective traveling salesman problem: an experimental study*, pages 177–199. Springer. Lecture Notes in Economics and Mathematical Systems Vol. 535, 2004.

[198] L.R. Pedro and R.H.C. Takahashi. INSPM: An interactive evolutionary multi-objective algorithm with preference model. *Information Sciences*, 268:202 – 219, 2014.

[199] P. Perny and O. Spanjaard. An axiomatic approach to aobustness in search problems with multiple scenarios. In *Proceedings of the 19th conference on Uncertainty in Artificial Intelligence, UAI*, pages 469–476, 2003.

[200] P. Perny, O. Spanjaard, and L-X. Storme. A decision-theoretic approach to robust optimization in multivalued graphs. *Annals of Operations Research*, 147(1):317–341, 2006.

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

[202] P. Perny, P. Viappiani, and A. Boukhatem. Incremental preference elicitation for decision making under risk with the rank-dependent utility model. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence, UAI*, page 597–606. AUAI Press, 2016.

[203] S.P. Phelps and M. Köksalan. An interactive evolutionary metaheuristic for multiobjective combinatorial optimization. *Management Science*, 49(12):1726–1738, 2003.

[204] D. Pisinger and S. Ropke. Large neighborhood search. In *Handbook of Metaheuristics*, volume 146 of *International Series in Operations Research & Management Science*, pages 399–419. Springer US, 2010.

[205] F.P. Preparata and M.I. Shamos. *Computational geometry: an introduction*. Springer-Verlag New York, Inc., New York, NY, USA, 1985.

[206] R. C. Prim. Shortest connection networks and some generalizations. *Bell System Technology Journal*, 36: 1389–1401, 1957.

[207] A. Przybylski and X. Gandibleux. Multi-objective branch and bound. *European Journal of Operational Research*, 260(3):856–872, 2017.

[208] A. Przybylski, X. Gandibleux, and M. Ehrgott. A two phase method for multi-objective integer programming and its application to the assignment problem with three objectives. *Discrete Optimization*, 7(3):149–165, 2010.

[209] A. Przybylski, X. Gandibleux, and M. Ehrgott. A recursive algorithm for finding all nondominated extreme points in the outcome set of a multiobjective integer programme. *INFORMS Journal on Computing*, 22(3): 371–386, 2010.

[210] A. Przybylski, K. Klamroth, and R. Lacour. A simple and efficient dichotomic search algorithm for multiobjective mixed integer linear programs, 2019.

[211] A. Raith and M. Ehrgott. A two-phase algorithm for the biobjective integer minimum cost flow problem. *Computers & Operations Research*, 36(6):1945–1954, 2009.

[212] K. Regan and C. Boutilier. Eliciting additive reward functions for Markov decision processes. In *Proceedings of the Twenty-Second International Joint Conference on Artificial Intelligence, IJCAI*, pages 2159–2164. AAAI Press, 2011.

[213] M. Resende. Computing approximate solutions of the maximum covering problem with GRASP. *Journal of Heuristics*, 4(2):161–177, 1998.

[214] G-C. Rota. On the foundations of combinatorial theory I. Theory of Möbius Functions. *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete*, 2(2):340 – 368, 1964.

[215] A. A. Salo and R. P. Hamalainen. Preference ratios in multiattribute evaluation (PRIME)-elicitation and decision procedures under incomplete information. *IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans*, 31(6):533–545, 2001.

[216] S. Sayın. Measuring the quality of discrete representations of efficient sets in multiple objective mathematical programming. *Mathematical Programming*, 87:543–560, 2000.

[217] A. Schrijver. A combinatorial algorithm minimizing submodular functions in strongly polynomial time. *Journal of Combinatorial Theory, Series B*, 80(2):346–355, 2000.

[218] O. Schütze. A new data structure for the nondominance problem in multi-objective optimization. In *Proceedings of the 2nd International Conference on Evolutionary Multi-criterion Optimization, EMO*, pages 509–518. Springer-Verlag, 2003.

[219] O. Schütze and C. Hernández. *Archiving Strategies for Evolutionary Multi-objective Optimization Algorithms*. Springer, 2021.

[220] O. Schütze, S. Mostaghim, M. Dellnitz, and J. Teich. Covering Pareto sets by multi-level evolutionary subdivision techniques. In *Proceedings of the 2nd International Conference on Evolutionary Multi-Criterion Optimization, EMO*, pages 118–132. Springer, 2003.

[221] A. Sen. Utilitarianism and welfarism. *Journal of Philosophy*, 76(9):463–489, 1979.

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

[223] P. Serafini. Some considerations about computational complexity for multi objective combinatorial problems. In J. Jahn and W. Krabs, editors, *Recent Advances and Historical Development of Vector Optimization*, pages 222–232, Berlin, Heidelberg, 1987. Springer Berlin Heidelberg.

[224] P. Serafini. Simulated annealing for multi-objective optimization problems. In *Proceedings of the Tenth International Conference on Multiple Criteria Decision Making, MCDM*, volume 1, pages 87–92, 1992.

[225] P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In *Proceedings of the 4th International Conference on Principles and Practice of Constraint Programming, CP*, pages 417–431, London, UK, 1998. Springer-Verlag.

[226] J. Shi, Q. Zhang, B. Derbel, A. Liefooghe, and J. Sun. Parallel Pareto local search revisited: first experimental results on bi-objective UBQP. In *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO*, pages 753–760. ACM, 2018.

[227] J. Shi, Q. Zhang, and J. Sun. PPLS/D: Parallel Pareto Local Search based on Decomposition. *IEEE Transactions on Cybernetics*, 50(3):1060–1071, 2020.

[228] W.S. Shin and A. Ravindran. Interactive multiple objective optimization: survey i—continuous case. *Computers & Operations Research*, 18(1):97–114, 1991.

[229] A.F Shorrocks. Ranking income distributions. *Economica*, 50(197):3–17, 1983.

[230] P. Skowron. FPT approximation schemes for maximizing submodular functions. *Information and Computation*, 257:65–78, 2017.

[231] P. Skowron, P. Faliszewski, and J. Lang. Finding a collective set of items: from proportional multirepresentation to group recommendation. *Artificial Intelligence*, 241:191–216, 2016.

[232] F. Sourd and O. Spanjaard. A multi-objective branch-and-bound framework. Application to the bi-objective spanning tree problem. *INFORMS Journal of Computing*, 20(3):472–484, 2008.

[233] S. Steiner and T. Radzik. Computing all efficient solutions of the biobjective minimum spanning tree problem. *Computers & Operations Research*, 35(1):198–211, 2008. Part Special Issue: Applications of OR in Finance.

[234] R. Steuer and E. Choo. An interactive weighted Tchebycheff procedure for multiple objective programming. *Mathematical Programming*, 26:326–344, 10 1983.

[235] M. Stoer and F. Wagner. A simple min-cut algorithm. *Journal of the ACM*, 44(4):585–591, 1997.

[236] M. Sun and R.E. Steuer. Quad trees and linear list for identifying nondominated criterion vectors. *INFORM Journal on Computing*, 8(4):367–375, 1996.

[237] Y. Sun, A. Ernst, X. Li, and J. Weiner. Generalization of machine learning for problem reduction: a case study on travelling salesman problems. *OR Spectrum: Quantitative Approaches in Management*, 43(3):607–633, 2021.

[238] R.S. Sutton and A.G. Barto. *Reinforcement Learning: An Introduction*. The MIT Press, second edition, 2018.

[239] E. Talbi, M. Rahoual, M.H. Mabed, and C. Dhaenens. A hybrid evolutionary approach for multi-criteria optimization problems: application to the flow shop. In *Proceedings of the 1st Int. Conf. on Multi-Criterion Optimization*, Berlin, 2001. Springer.

[240] E-G. Talbi. *Metaheuristics: From Design to Implementation*. Wiley-Blackwell, 2009.

[241] S. Tamby and D. Vanderpooten. Enumeration of the nondominated set of multiobjective discrete optimization problems. *INFORMS Journal on Computing*, 33(1):72–85, 2021.

[242] S. Tamby and D. Vanderpooten. Optimizing over the efficient set of a multi-objective discrete optimization problem. In *21st International Symposium on Experimental Algorithms, SEA*, volume 265, pages 9:1–9:13. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 2023.

[243] J. Teghem, D. Dufrane, M. Thauvoye, and P. Kunsch. Strange: An interactive method for multi-objective linear programming under uncertainty. *European Journal of Operational Research*, 26(1):65–82, 1986.

[244] A.F. Tehrani, W. Cheng, K. Dembczyński, and E. Hüllermeier. Learning monotone nonlinear models using the Choquet integral. *Machine Learning*, 89(1-2):183–211, 2012.

[245] C. Teixeira, J. Covas, T. Stützle, and A. Gaspar-Cunha. Application of Pareto local search and multi-objective ant colony algorithms to the optimization of co-rotating twin screw extruders. In *Proceedings of the EU/Meeting 2009: Debating the future: new areas of application and innovative approaches*, pages 115–120, 2009.

[246] M.K. Tomczyk and M. Kadziński. EMOSOR: Evolutionary multiple objective optimization guided by interactive stochastic ordinal regression. *Computers & Operations Research*, 108:134–154, 2019.

[247] M.K. Tomczyk and M. Kadziński. Decomposition-based interactive evolutionary algorithm for multiple objective optimization. *IEEE Transactions on Evolutionary Computation*, 24(2):320–334, 2020.

[248] V. Torra. The weighted OWA operator. *International Journal of Intelligent Systems*, 12(2):153–166, 1997.

[249] B. Ulungu and J. Teghem. The two phase-method: an efficient procedure to solve bi-objective combinatorial optimization problems. *Foundations of computing and decision sciences*, 20(2):149–165, 1995.

[250] B. Ulungu and J. Teghem. Solving multi-objective knapsack problem by a branch-and-bound procedure. In *Multicriteria Analysis*, pages 269–278. Springer Berlin Heidelberg, 1997.

[251] E.L. Ulungu, J. Teghem, Ph. Fortemps, and D. Tuyttens. MOSA method: a tool for solving multiobjective combinatorial optimization problems. *Journal of Multi-Criteria Decision Analysis*, 8(4):221–236, 1999.

[252] E.S Van der Poort, M. Libura, G. Sierksma, and J.A Van der Veen. Solving the k-best traveling salesman problem. *Computers & Operations Research*, 26(4):409–425, 1999.

[253] P. Vincke. Une méthode interactive en programmation linéaire à plusieurs fonctions économiques. *Revue française d'automatique, informatique, recherche opérationnelle. Recherche opérationnelle*, 10:5–20, 1976.

[254] J. Vondrák. Optimal approximation for the submodular welfare problem in the value oracle model. In *Proceedings of the fortieth annual ACM symposium on Theory of computing*, pages 67–74, 2008.

[255] T. Wang and C. Boutilier. Incremental utility elicitation with the minimax regret decision criterion. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence, IJCAI*, pages 309–316. AAAI Press, 2003.

[256] L. Weerasena, M.M. Wiecek, and B. Soylu. An algorithm for approximating the Pareto set of the multiobjective set covering problem. *Annals of Operations Research*, 248(1):493–514, 2017.

[257] P. Weng and B. Zanuttini. Interactive value iteration for Markov decision processes with unknown rewards. In *Proceedings of the 23rd International Joint Conference on Artificial Intelligence, IJCAI*, pages 2415–2421. AAAI Press, 2013.

[258] C. C. White, A. P. Sage, and S. Dozono. A model of multiattribute decisionmaking and trade-off weight determination under uncertainty. *IEEE Transactions on Systems, Man, and Cybernetics*, 14(2):223–229, 1984.

[259] M. M. Wiecek. Advances in cone-based preference modeling for decision making with multiple criteria. *Decision Making in Manufacturing and Services*, 1:153–173, 2007.

[260] B. Xin, L. Chen, J. Chen, H. Ishibuchi, K. Hirota, and B. Liu. Interactive multiobjective optimization: a review of the state-of-the-art. *IEEE Access*, 6:41256–41279, 2018.

[261] R. R. Yager. On ordered weighted averaging aggregation operators in multicriteria decision making. *IEEE Transactions on Systems, Man, and Cybernetics*, 18(1):183–190, 1988.

[262] R.R. Yager. On using the Shapley value to approximate the Choquet integral in cases of uncertain arguments. *IEEE Transactions on Fuzzy Systems*, 26(3):1303–1310, 2018.

[263] T. Ye, Z. Zhang, J. Chen, and J. Wang. Weight-specific-decoder attention model to solve multiobjective combinatorial optimization problems. In *IEEE International Conference on Systems, Man, and Cybernetics, SMC*, pages 2839–2844. IEEE, 2022.

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

[265] Q. Zhang and H. Li. MOEA/D: A multiobjective evolutionary algorithm based on decomposition. *IEEE Transactions on Evolutionary Computation*, 11(6):712–731, 2007.

[266] Y. Zhang, J. Wang, Z. Zhang, and Y. Zhou. MODRL/D-EL: multiobjective deep reinforcement learning with evolutionary learning for multiobjective optimization. In *International Joint Conference on Neural Networks, IJCNN*, pages 1–8. IEEE, 2021.

[267] S. Zionts and J. Wallenius. An interactive programming method for solving the multiple criteria problem. *Management Science*, 22(6):652–663, 1976.

[268] E. Zitzler. *Evolutionary algorithms for multiobjective optimization: methods and applications*. PhD thesis, Swiss Federal Institute of Technology (ETH), Zurich, Switzerland, 1999.

[269] E. Zitzler, L. Thiele, M. Laumanns, C.M. Fonseca, and V. Grunert da Fonseca. Performance assessment of multiobjective optimizers: an analysis and review. *IEEE Transactions on Evolutionary Computation*, 7(2): 117–132, 2003.

[270] M. Özlen and M. Azizoğlu. Multi-objective integer programming: A general approach for generating all non-dominated solutions. *European Journal of Operational Research*, 199(1):25–35, 2009.

Major personal publications

Exact and heuristic methods for solving MOCO problems

1. T. Lust and A. Jaszkiewicz. Speed-up techniques for solving large-scale biobjective TSP. *Computers & Operations Research*, 37:521–533, 2010.
2. T. Lust and J. Teghem. Two-phase Pareto local search for the biobjective traveling salesman problem. *Journal of Heuristics*, 16(3):475–510, 2010.
3. T. Lust and J. Teghem. The multiobjective traveling salesman problem: a survey and a new approach. In C. Coello Coello, C. Dhaenens, and L. Jourdan, editors, *Advances in Multi-Objective Nature Inspired Computing*, volume 272 of *Studies in Computational Intelligence*, pages 119–141. Springer Berlin Heidelberg, 2010.
4. V.N. Coelho, M.J.F. Souza, I.M. Coelho, F.G. Guimarães, T. Lust, and R.C. Cruz. Multi-objective approaches for the open-pit mining operational planning problem. *Electronic Notes in Discrete Mathematics*, 39:233–240, 2012.
5. T. Lust and J. Teghem. The multiobjective multidimensional knapsack problem: a survey and a new approach. *International Transactions in Operational Research*, 19(4):495–520, 2012.
6. T. Lust, N. Meskens, and T. Monteiro. Ordonnancement multiobjectif du bloc opératoire avec une prise en compte d'une affectation équilibrée des compétences des infirmières. In *MOSIM International Conference on Modeling, Optimization and Simulation*, pages 1–10, 2012.
7. T. Lust and D. Tuyttens. Variable and large neighborhood search to solve the multiobjective set covering problem. *Journal of Heuristics*, 20(2):165–188, 2014.
8. A. Jaszkiewicz and T. Lust. Proper balance between search towards and along Pareto front: biobjective TSP case study. *Annals of Operations Research*, 254(1-2):111–130, 2017.
9. V. N. Coelho, T. A. Oliveira, I. M. Coelho, B. N. Coelho, P. J. Fleming, F. G. Guimarães, H. R. D. Lourenço, M.J.F. Souza, E.-G. Talbi, and T. Lust. Generic Pareto local search metaheuristic for optimization of targeted offers in a bi-objective direct marketing campaign. *Computers & Operations Research*, 78:578–587, 2017.
10. A. Jaszkiewicz and T. Lust. ND-tree-based update: A fast algorithm for the dynamic nondominance problem. *IEEE Transactions on Evolutionary Computation*, 22(5):778–791, 2018.
11. T. Lust. Interactive Pareto local search with imprecise trade-offs. In *Proceedings of DA2PL: from Multiple Criteria Decision Aid to Preference Learning*, pages 1–8, 2018.
12. L. Costa, T. Lust, R. Kramer, and A. Subramanian. A two-phase Pareto local search heuristic for the bi-objective pollution-routing problem. *Networks*, 72(3):311–336, 2018.

Lorenz dominance

1. L. Galand and T. Lust. Multiagent fair optimization with Lorenz dominance. In *Proceedings of the 2015 International Conference on Autonomous Agents and Multiagent Systems, AAMAS*, pages 1895–1896. ACM, 2015.
2. L. Galand and T. Lust. Exact methods for computing all Lorenz optimal solutions to biobjective problems. In *Algorithmic Decision Theory, 4th International Conference, ADT*, volume 9346 of *Lecture Notes in Computer Science*, pages 305–321. Springer, 2015.

Choquet integral

1. T. Lust and A. Rolland. On the computation of Choquet optimal solutions in multicriteria decision contexts. In *Proceedings of the 7th Multi-Disciplinary International Workshop on Artificial Intelligence (MIWAI)*, pages 131–142, 2013.
2. T. Lust and A. Rolland. Choquet optimal set in biobjective combinatorial optimization. *Computers & Operations Research*, 40(10):2260–2269, 2013.
3. T. Lust and A. Rolland. 2-additive Choquet optimal solutions in multiobjective optimization problems. In *Proceedings of the 15th International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems IPMU*, volume 442 of *Communications in Computer and Information Science*, pages 256–265. Springer, 2014.
4. T. Lust. Choquet integral versus weighted sum in multicriteria decision contexts. In *Algorithmic Decision Theory - 6th International Conference, ADT*, volume 9346 of *Lecture Notes in Computer Science*, pages 288–304. Springer, 2015.

Interactive Methods

1. N. Benabbou and T. Lust. An interactive polyhedral approach for multi-objective combinatorial optimization with incomplete preference information. In *Scalable Uncertainty Management - 13th International Conference, SUM*, volume 11940 of *Lecture Notes in Computer Science*, pages 221–235. Springer, 2019.
2. N. Benabbou and T. Lust. A general interactive approach for solving multi-objective combinatorial optimization problems with imprecise preferences. In *Proceedings of the Twelfth International Symposium on Combinatorial Search, SOCS*, pages 164–165. AAAI Press, 2019.
3. N. Benabbou, C. Leroy, T. Lust, and P. Perny. Combining local search and elicitation for multi-objective combinatorial optimization. In *Algorithmic Decision Theory - 6th International Conference, ADT*, pages 1–16, 2019.
4. N. Benabbou, L. Galand, and T. Lust. Experimental analysis of greedy strategies to minimize pairwise comparisons in multicriteria decision aiding. In *90th meeting of the Euro working group on Multi-Criteria Decision Aiding (EWG-MCDA 90)*, 2019.
5. N. Benabbou, C. Leroy, and T. Lust. An interactive regret-based genetic algorithm for solving multi-objective combinatorial optimization problems. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 2335–2342. AAAI Press, 2020.
6. N. Benabbou, C. Leroy, and T. Lust. Regret-based elicitation for solving multi-objective knapsack problems with rank-dependent aggregators. In *24th European Conference on Artificial Intelligence, ECAI*, pages 419–426, 2020.
7. N. Benabbou, C. Leroy, T. Lust, and P. Perny. Combining preference elicitation with local search and greedy search for matroid optimization. In *The Thirty-Fifth AAAI Conference on Artificial Intelligence*, pages 12233–12240. AAAI Press, 2021.
8. N. Benabbou, C. Leroy, T. Lust, and P. Perny. Interactive optimization of submodular functions under matroid constraints. In *Algorithmic Decision Theory - 7th International Conference, ADT*, Lecture Notes in Computer Science, pages 1–15. Springer, 2021.