A Review and Taxonomy of Interactive Optimization Methods in Operations Research

David Meignan, Universität Osnabrück, Osnabrück, Germany
Sigrid Knust, Universität Osnabrück, Osnabrück, Germany
Jean-Marc Frayret, École Polytechnique de Montréal, Montréal, Canada
Gilles Pesant, École Polytechnique de Montréal, Montréal, Canada
Nicolas Gaud, Université de Technologie de Belfort-Montbéliard, Belfort, France

This is a pre-print version (July 2015) of the article published in *ACM Transactions on Interactive Intelligent Systems*, Volume 5, Issue 3, 2015. The definitive version is available from the DOI: http://dx.doi.org/10.1145/2808234

This work was supported by the Deutsche Forschungsgemeinschaft (DFG), under grant ME 4045/2-1.

A Review and Taxonomy of Interactive Optimization Methods in Operations Research

David Meignan, Universität Osnabrück, Osnabrück, Germany
Sigrid Knust, Universität Osnabrück, Osnabrück, Germany
Jean-Marc Frayret, École Polytechnique de Montréal, Montréal, Canada
Gilles Pesant, École Polytechnique de Montréal, Montréal, Canada
Nicolas Gaud, Université de Technologie de Belfort-Montbéliard, Belfort, France

Abstract

This paper presents a review and a classification of interactive optimization methods. These interactive methods are used for solving optimization problems. The interaction with an end user or decision maker aims at either improving the efficiency of the optimization procedure, enriching the optimization model, or informing the user regarding the solutions proposed by the optimization system. First, we present the challenges of using optimization methods as a tool for supporting decision making, and we justify the integration of the user in the optimization process. This integration is generally achieved via a dynamic interaction between the user and the system. Next, the different classes of interactive optimization approaches are presented. This detailed review includes trial and error, interactive reoptimization, interactive multi-objective optimization, interactive evolutionary algorithms, human-guided search, and other approaches that are less well covered in the research literature. On the basis of this review, we propose a classification that aims to better describe and compare interaction mechanisms. This classification offers two complementary views on interactive optimization methods. The first one focuses on the user's contribution to the optimization process, and the second perspective concerns the components of interactive optimization systems. Finally, on the basis of this review and classification, we identify some open issues and potential perspectives for interactive optimization methods.

1 Introduction

Over the years Operations Research has produced a number of successful computational approaches and software tools for solving complex optimization problems of practical value. Optimization problems are present at all planning levels; from the strategic level for determining long-term orientations of an organization or business, to the operational level to determine the day-to-day operations. If we look at these different levels of decision making, strategic and tactical planning are probably the activities that have most benefited from recent optimization methods. However, many obstacles remain for the integration of advanced optimization methods in operational decision support tools (see, for instance [Maccarthy and Liu, 1993, Kellogg and Walczak, 2007, McCollum, 2006]).

Three major obstacles can be identified for the integration of advanced optimization methods in decision support tools. First, it is generally difficult to obtain a model of an optimization problem that reflects all aspects of the real decision makers' problem. This is particularly true for operational decision making when multiple heterogeneous criteria have to be considered in the optimization model. A second obstacle concerns the possible inadequacy between performances of the optimization method and the real requirements of the users. Finally, mistrust or misunderstanding of automated optimization systems by users also constitute major obstacles to the effective use of advanced optimization methods.

A common research direction to address the first of these three issues is the design of richer, more realistic optimization models. A wide variety of papers proposes to extend

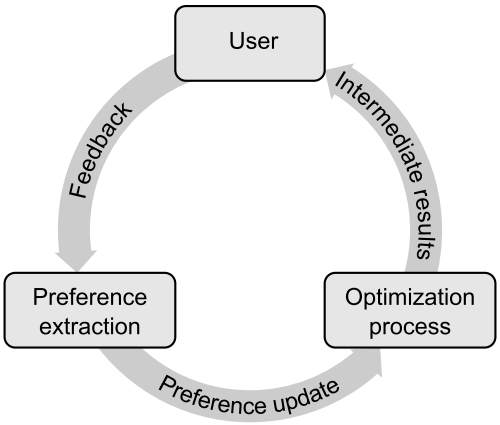

Figure 1: Human-in-the-loop approach for optimization.

general optimization models with additional features that better describe the problem at hand. To give only one example, in [Burke et al., 2004] a literature review on the nurse rostering problem is presented. The authors distinguish at least 40 model features (hard constraints, soft constraints and terms in the objective functions) that contribute to better fit the case studies. If this kind of model improvement can reduce the gap between the optimization model and the real problem, it may also reduce the scope of applications, and could make the implementation of an optimization method more complex.

A second research direction, motivated by the performance issue, consists in designing more efficient optimization methods. Again, a lot of articles in the research literature propose and compare optimization methods that are more efficient than previous ones. An illustration of this race for performance is the vehicle-routing problem for which new algorithms and improved results are still proposed fifty years after the initial problem's formal definition [Laporte, 2009]. However, the gain in performance is often obtained at the expense of the simplicity and flexibility of the methods.

Without questioning the importance of having more realistic optimization models and more efficient optimization methods, neither of these two previous directions seems to be sufficient for ensuring an effective use of advanced optimization methods in decision support tools. An alternative approach to bridge this gap between optimization methods and decision support systems is to consider that the decision maker can actively participate in the optimization process. This aspect is investigated by interactive optimization approaches. With an adequate interaction between an optimization system and its users, the optimization model can be enriched to fit the real problem, the search process can be guided for improving its efficiency, and the user can better understand the system. In short, the main goal of interactive optimization is to turn efficient optimization methods into effective decision tools.

In interactive optimization, the user of an optimization system is involved in the optimization process and can change the result or performance of the optimization. This *human-in-the-loop* approach of the optimization, illustrated in Figure 1, is based on some important assumptions that are often disregarded for automated optimization systems. In particular, the design of an automated optimization system supposes that the optimization problem at hand can be appropriately modeled according to the decision context, and the optimization procedure can be adequately parameterized before using the system under real conditions. In this context, it makes sense to aspire to a fully automated optimization system. In contrast, an interactive approach recognizes some limits to modeling and parameter setting in a real situation, and values the user's expertise in the application domain that can be exploited by the optimization system.

This idea of involving users in the optimization process is not new, and has been investigated since the early 70s in the context of multiobjective optimization [Benayoun et al., 1971, Wallenius, 1975]. Later, the interactive approach was generalized to other optimization problems. In [Fisher, 1985], the author already mentions applications to vehicle routing, location, and scheduling problems. There are now a multitude of interactive optimization methods, ranging from rudimentary trial-and-error to more sophisticated approaches such as interactive multiobjective optimization [Miettinen et al., 2008] and human-guided search [Klau et al., 2010]. Furthermore, it is no longer controversial to consider that man-machine interaction can be valuable for solving complex optimization problems [Barthélemy et al., 2002]. However, it is surprising that relatively little attention is given to the study of the interaction in the optimization field.

This paper attempts to provide a detailed review of interactive optimization methods and to connect studies that until now have been mostly examined separately in the research literature. The existing literature does not yet include a survey on interactive optimization that covers all interactive approaches for solving optimization problems. Previous overviews focus on specific approaches such as human-guided search [Klau et al., 2010], interactive evolutionary computation [Takagi, 2001], and interactive multi-objective optimization [Miettinen et al., 2008]. However, none of those previous works presents a complete survey of interactive optimization approaches.

The objectives of this article are, first, to provide a survey of interactive optimization approaches for solving complex optimization problems and to present the main motivations and limitations of these approaches. Second, a classification of interactive optimization methods is proposed. This classification aims at analyzing existing interactive optimization methods and supporting the development of new interactive optimization methods. The connections we make between the different methods through the classification also aim at avoiding the pitfalls that have already been identified when a new method is designed. Finally, in this article we briefly discuss current issues in the domain of interactive optimization and outline some research directions that, in our opinion, constitute major research perspectives of the field.

The remainder of this paper is organized as follows. In Section 2, the main terminology concerning optimization problems and interactive optimization systems is introduced. Afterwards, in Section 3, obstacles to the usage of optimization methods in decision support tools are described, and based on these obstacles, the main motivations of interactive optimization approaches are identified. Then, in Section 4, a survey of interactive optimization approaches is given. In this survey, besides presenting examples from the literature, limits of the approaches are discussed. In Section 5, a classification of interactive optimization methods is introduced and representative methods are classified according to the proposed criteria. Finally, in Section 6, conclusions and research perspectives can be found.

2 Context and definitions

This paper focuses on interactive optimization methods in the field of Operations Research. The review and the associated classification presented in the remaining sections are thus limited to this domain. In particular, it addresses interactive systems aiming at solving optimization problems by computing candidate solutions. Hence, the studied interactive process is a part of a larger decision-making process. It should be noted that the domain of interactive optimization is closely tied to the domain of interactive machine learning (we refer the reader to [Amershi et al., 2014] and [Fails and Olsen, 2003] for an overview of interactive machine learning). More generally, several interconnections exist between the domains of machine learning and optimization. The boundary between those two domains could be a somewhat blurry because optimization tools can be used for solving machine learning problems, and machine learning techniques can also be applied for

solving optimization problems [Bennett and Parrado-Hernández, 2006, Sra et al., 2012]. In this section, we clarify the boundaries of the domain of interactive optimization by, firstly, defining what an optimization problem is, and then, by stating in which system and during which process the interaction of interest occurs. In addition, the core concepts and the specific terminology used throughout this paper to describe interactive optimization methods are introduced.

2.1 Optimization model definition

In an optimization problem P, we are given a set $\mathcal{X}$ representing the solution space and an *objective function* $f : \mathcal{X} \to \mathbb{R}$. The objective is to find an optimal solution in the set $\mathcal{X}$ according to the objective function f. More precisely, for minimization problems we have to find a solution $x^* \in \mathcal{X}$ with $f(x^*) \leq f(x)$ for all $x \in \mathcal{X}$, for maximization problems we have to find a solution $x^* \in \mathcal{X}$ with $c(x^*) \geq c(x)$ for all x. Any solution x of P is specified by a set of values assigned to the *decision variables*. The possible values are limited by a set of *constraints*, often given in form of equalities, inequalities or logical expressions. In short, a mathematical model of an optimization problem, also called *optimization model*, includes a set of decision variables, an objective function, and a set of constraints. In order to find a solution x with the best possible value $f(x)$, exact or heuristic *optimization procedures* have to be developed which are used to find optimal or near-optimal solutions.

Generally both the objective function and the set of constraints are expressed with generic parameters that need to be set for representing a particular *problem instance*. For example, if the problem involves a graph, the problem can be modeled with "generic" sets of vertices $V = \{v_0, \cdots, v_n\}$ and edges $E = \{(v_i, v_j) | v_i, v_j \in V\}$. The *problem-data*, which are usually known only when the problem must be solved, allows to instantiate such parameters. This process which includes problem-modeling and then problem-instantiation is further detailed in Section 2.3.

Usually, an optimization model captures partially the real problem for which a decision must be made. The differences between the optimization model and the real problem come, for instance, from simplifications that are necessary to make the problem computationally tractable or can be related to the inherent limits of the modeling process (i.e., requiring abstractions and generalizations). Since this divergence between real optimization problems and their models plays an important role in interactive optimization, we adopt specific vocabulary in order to distinguish these two aspects. In this article, a *real problem* or *real decision maker's problem* refers to the actual context of an optimization problem for which a decision must be made. The representation of this problem in the optimization system is referred to as an *optimization model*. Similarly, a *criterion* refers to the measures used to compare different *alternatives* of a real problem, whereas the *objective* designates a mathematical function for evaluating the *solutions* of an optimization model.

For example, the Capacitated Vehicle Routing Problem (CVRP) is a well-known optimization problem that consists of designing the routes of a set of vehicles so that each route starts and ends at the depot, each customer is visited exactly once, and the demand of any route does not exceed the capacity of the vehicles. A solution to this problem is a set of routes defined on a given graph, and the objective is to minimize the total length of the routes. A real context for this optimization problem could be the design of itineraries for a delivery service. The CVRP is a simplified model of this delivery problem since the routes on the CVRP's graph contain less information than the real itineraries, and the objective function is only an approximation of the real criterion to minimize the transportation costs. The latter criterion may combine some aspects that are difficult to model such as the delivery problem, the real staff and vehicle costs, as well as the uncertainty related to the duration of itineraries. This gap between the optimization model and the

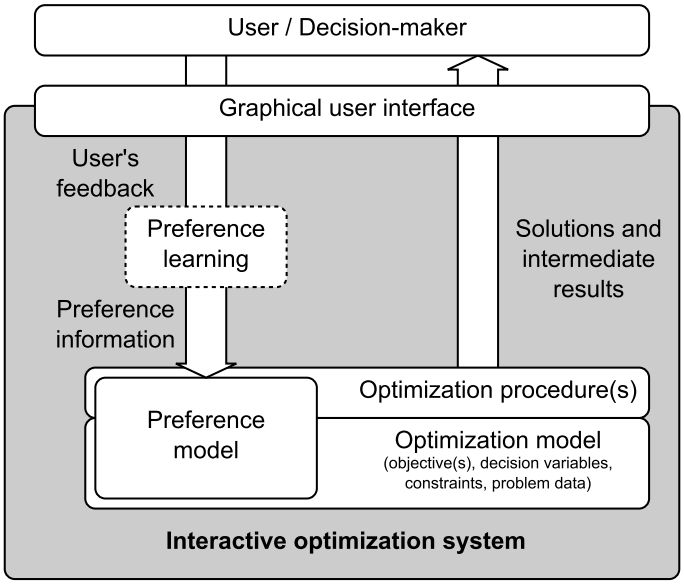

Figure 2: Components of an interactive optimization system.

real problem motivates interactive approaches, which is further discussed in Section 3.1.

2.2 Key components of an interactive optimization system

In any optimization system, the optimization model and the optimization procedures are the core components that provide solutions to problem instances. In interactive optimization systems, both the optimization model and the procedures are used within interaction loops with the user as depicted in Figure 2. In this general architecture, the *optimization model* contains the definition of the decision variables, objectives and constraints of the problem to be solved. When the optimization problem is instantiated, the problem data determine the parameter values of the optimization model.

The *optimization procedures* responsible for solving the problem instances are directly connected to the optimization model. They aim at both providing candidate solutions to the user and producing intermediate results. In the terminology used in this paper, a candidate solution is a solution to an optimization problem that can be considered by the user as a final solution. An intermediate result is information obtained during this optimization process and is not necessarily a solution. An intermediate result could be an incomplete solution or information about the current state of the optimization process. For example, if a branch-and-bound procedure is used for solving a problem instance, the upper and lower bounds can be provided as intermediate results for informing the user on the progress of the solving procedure.

Optimization procedures, thus, produce some solutions and intermediate results that are presented to the user. The user, in turn, can provide a *feedback*. As described in Section 4, the nature of this feedback can take various forms. For example, the user may select, among a set of solutions, the most promising ones as in interactive evolutionary algorithms. The user feedback can also consist of adjustments of parameter values for some interactive multiobjective optimization methods. It can also correspond to a set of modifications of a candidate solution as in interactive reoptimization.

In order to close the loop, the user feedback is used by the system to adjust the optimization procedures or the optimization model through a *preference model* [Miettinen et al., 2008, Wessels and Wierzbicki, 2000]. A preference model corresponds to the information that is derived from the user's feedback. It is used to modify the optimization model or the optimization procedures. This preference model can take many forms. In its most simple form, the preference model can be limited to parameters of the optimization

problem, such as the weight values in an objective function. In a more advanced form, the preference model can be heuristic information that guides the optimization procedure.

In the optimization domain, the term preference model essentially refers to preference information related to the objectives. However, in this article, we propose a broader sense that covers not only the information that captures user's preferences on the optimization model, but also information that impact the optimization procedures. Therefore, in the architecture presented in Figure 2, the preference model covers both the optimization model and the optimization procedures.

In the survey and classification of the literature presented in Sections 4 and 5, a distinction is made between interactive methods where the preference model is intended to modify the optimization model, and methods for which the preference model mainly impacts the optimization procedures. In the first case, feedback loops aim to modify the optimization problem. This is referred to as *problem-oriented interaction* (e.g., adjustment of weight values in an objective function). The main problem-oriented interaction approaches reviewed in this paper are interactive reoptimization (Section 4.2), interactive multiobjective optimization (Section 4.3) and interactive evolutionary algorithms (Section 4.4). In the second case, feedback loops aim to adjust the optimization procedure. This is referred to as *search-oriented interaction* (e.g., adjustments of the parameter setting of an optimization procedure). In this survey, human-guided search approaches (Section 4.5) are introduced as search-oriented interaction.

Finally, the general architecture presented in Figure 2 may also include some form of *preference learning*. Although it is not systematically implemented, a preference learning procedure aims to generalize the user feedback in order to create a model of the user's preferences. For example, in some interactive evolutionary algorithms, a classifier is trained with the user's feedback in order to capture the preferences of the user (Section 4.4). This classifier is used to enrich the optimization model (i.e., add new relevant information to the current problem instance, such as new constraints). In this case, there is a learning procedure that explicitly generalizes the user's feedback. In contrast, the user feedback can also be integrated into the preference model. For example, a value provided by the user can be directly assigned to a parameter of the optimization model. In this case, there is no learning procedure, although the user preference model still exists. In this article, these alternatives are respectively called *model-based preference integration* and *model-free preference integration* (see Section 5).

2.3 Interactive optimization within the decision process

In this study, we only consider interactive methods that are used in a decision process for solving optimization problems. In Operations Research, the decision process consists of all activities from the identification of a problem to the implementation and control of a decision. However, in the context of an optimization-based decision support process [Shim et al., 2002], two main phases can be distinguished [Forgionne, 2002, Vercellis, 2009], as seen in Figure 3. The first phase is the *design process* that is necessary to implement an optimization system. The second phase is the *decision-making process* per se, during which an instance of an optimization problem is solved. The interaction between the user and the optimization system occurs during this second phase.

In order to clarify in which activities the studied interaction takes place, a brief description of the design process is provided before detailing the activities of the decision-making process. The reader may refer to [Hillier and Lieberman, 2001, Chapter 2] for more details on the design process of an optimization system.

The first step in the design process is the *problem analysis*. It consists in defining the scope as well as the criterion and constraints of the optimization problem. More specifically, this step involves gathering information about the problem, identifying the

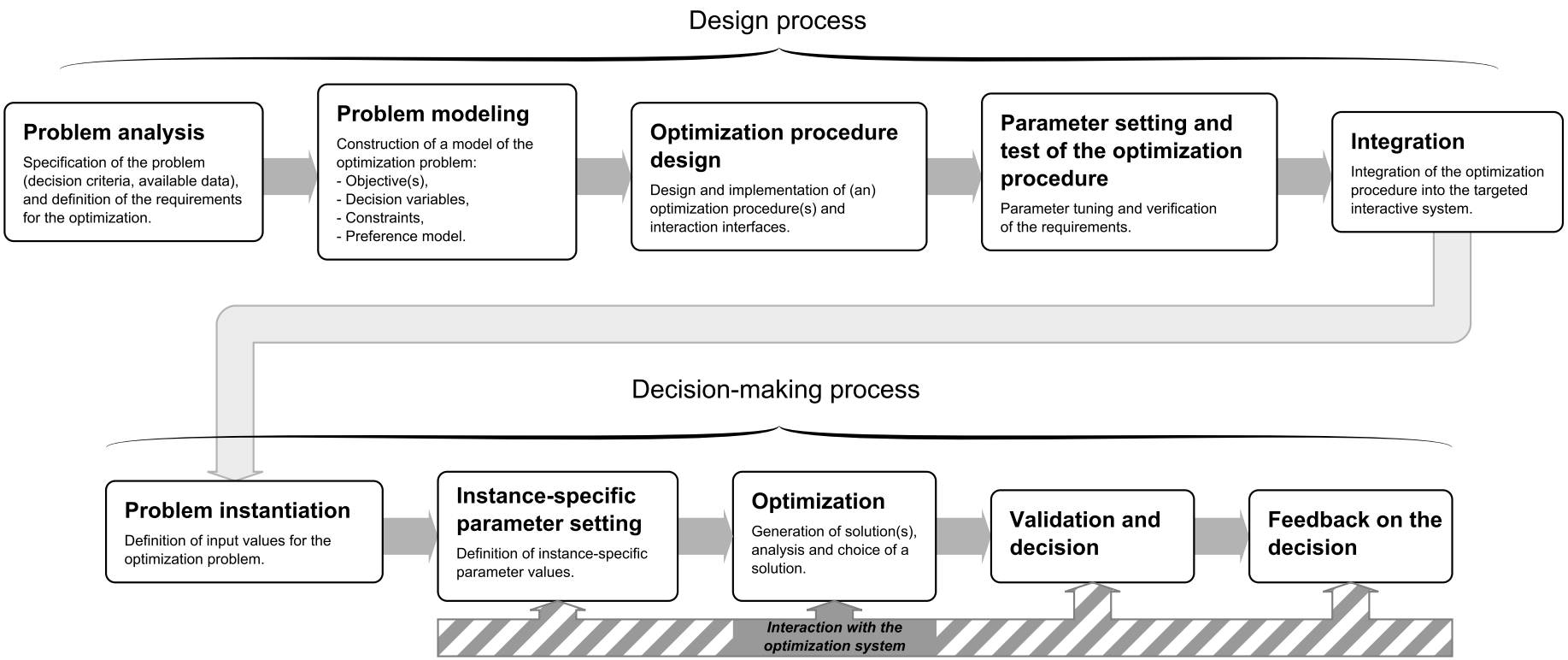

Figure 3: Design and decision-making processes related to an interactive optimization system.

data that can be used for making a decision, and defining the decision criteria. In addition, experts and analysts involved in the design process specify the requirements related to the optimization system. In particular, the expected performance of the optimization system in terms of computation time and quality of results are clarified. The possible interactions between the user and the optimization system are also identified during this initial phase of the design process.

Once the problem is analyzed, a mathematical optimization model is generally developed during the *problem modeling* step. This step aims to explicitly represent the optimization problem. The development of a procedure for solving the problem corresponds to the third step, referred to as the *optimization procedure design* step. Next, this procedure is tuned and tested during the next two steps of the design process, namely the *parameter setting and test of the optimization procedure*. Finally, the optimization procedure is integrated into the system in which it will be used.

The design process results in an optimization system that can be used for making decisions. This second phase, the decision-making process, is summarized in the lower part of Figure 3. In short, the decision-making process considers all steps from the definition of the problem-data for the problem instance to solve, to the implementation of a decision. As in any decision-making process, the steps presented in Figure 3 may overlap, and loop-backs to earlier stages of the process are frequent. We refer the reader to [Shim et al., 2002, Phillips-Wren, 2008] for an overview of the decision-making process in the context of intelligent decision-support systems.

The first step of the decision-making process is the *problem instantiation*. It consists in defining the problem's data that are specific to the decision situation. For instance, if the optimization problem is a vehicle routing problem, the values that characterize the vehicles, the demands and the road network are specified during this phase. The data given to the system at this step do not contain subjective information from the user but are exclusively objective data from a source such as a database.

Next, the user may provide initial preferences during the *instance-specific parameter setting* step. This parameter setting may concern the optimization procedure with the adjustment of strategic parameters such as the stopping criterion of the optimization. The parameters may also be related to the optimization problem. For instance, initial weights for different terms of the objective function may be provided during this step.

Next, during the *optimization* step, also referred to as *solution process* or *solving process*, an optimization procedure is used to generate one or several candidate solutions. For most interactive optimization approaches, the interaction between the user and the interactive optimization system occurs during this step. Generally, the preference model is interactively adjusted until a satisfactory solution is found.

The output of the optimization is a decision alternative (i.e., a solution) that must be validated before its implementation. This is done during the *validation and decision* step. More specifically, the decision alternative is considered and analyzed with respect to the real decision context. In practice, the actual decision context and criteria may be richer than the decision information and criteria used during the optimization step. In addition, the decision-maker may be different from the user of the optimization system. During this step, if a decision alternative is rejected, another solution must be generated.

Finally, when the alternative is validated and implemented, a *feedback* phase may be considered in order to adjust the next decisions with known effects of previous ones.

As mentioned earlier, interactions between the optimization system and the user mainly occur during the optimization step. However, because of feedback loops and the fact that the decision-making process is repeated for multiple problem instances, interactions that impact the optimization problem or procedure can also take place during other steps.

For instance, the instance-specific parameter setting can be done on the basis of previous optimization results and regarded as a part of the interactive process (e.g., trial-and-error approach, Section 4.1). Similarly, both the validation of a decision alternative and the feedback on a tentative decision can also be taken into account in subsequent iterations of the decision process.

3 Limits of optimization methods as decision support tools

The use of optimization methods to support decision-making has been proven successful in the context of a long list of applications. However, optimization methods can have limits with respect to their integration within decision support tools. In particular, this section identifies within the literature three limits that can justify, in some contexts, the use of interactive optimization methods: (1) the inherent limits of optimization models; (2) the inadequate performances of optimization procedures; and (3) the non-acceptance and misunderstanding of optimization systems by their users. This section provides an overview of these limits.

3.1 Inherent limits of optimization models

As previously mentioned, an optimization model is a partial representation of a real problem. An optimization model contains unavoidable inaccuracies that may be problematic for supporting a decision [Roy, 1989, 2005, Bouyssou, 1990]. These inaccuracies are the result of various limitations of the modelling process:

Approximation of complex problem's aspects First, the problem to model may contain criteria or constraints that are difficult to quantify. This situation usually occurs when dealing with rich optimization problems and problems related to human activities such as vehicle routing problems or staff scheduling problems. For these problems, criteria such as risks, tasks' arduousness or perceived duration are particularly difficult to model and are necessarily approximated in an optimization model.

Simplification for model tractability In addition, it may be necessary to simplify the specification of an optimization problem in order to apply a computational optimization approach. Common examples of such a simplification are the combination of several criteria in a single objective function, or the approximation of variables relationships by linear models.

Limited specifications Besides these obstacles inherent to the optimization problem, inaccuracies of an optimization model may also be related to the difficulty in obtaining a complete specification of the problem for designing the model. The experts or analysts who design the optimization model may only have a partial knowledge of the actual context of the optimization problem.

Lack of resources Finally, the time and budget for designing and implementing an optimization model are limited, thus limiting possible refinements of the model.

Overall, an optimization model may underestimate or ignore some aspects of the real problem. This can result in the computation of either unrealistic or unfeasible solutions, or solutions that do not capture domain-related characteristics.

In order to overcome these limitations, a first approach consists in extending general optimization models with additional features that better describe the problems at hand. In Operations Research literature, it is common to see a large number of variants of an initial problem model. For instance, a vehicle routing problem model can be enriched with time-window constraints, transshipments or by considering a heterogeneous fleet of vehicles [Savelsbergh and Sol, 1995]. The additional features such as hard constraints, soft

constraints and objective functions, allow enriching the initial optimization model, and contribute to design more realistic optimization models. However, introducing additional aspects in an optimization model may also reduce the scope of applications, and can potentially increase the difficulty of implementing and using an optimization method.

Approaches such as multiobjective optimization, stochastic optimization and robust optimization, are generic extensions of optimization problems that also aim at improving optimization models. In multiobjective optimization [Branke et al., 2008, Ehrgott and Gandibleux, 2000], the problem to solve is modeled with a set of objective functions to optimize instead of a single objective. This allows the decision maker to express preferences with respect to objectives' priority. Consequently, the inaccuracy induced by aggregating criteria is reduced and the decision support tool is more likely to be adopted by the user. Three approaches to capturing decision-maker's preferences about objectives functions are usually distinguished [Ehrgott and Gandibleux, 2000]. First, when the preferences definition is done *a priori*, the optimization method returns a solution that meets these preferences. Next, when the preferences definition is done *a posteriori*, the optimization method must return a set of non-dominated solutions, which is then analyzed by the decision maker to identify an appropriate solution. Finally, preferences can be progressively introduced or adjusted *interactively*. This interactive approach to capture user preferences is described further in the next section.

Other approaches to deal with model limitations are stochastic and robust optimization. These approaches aim to deal with the inherent uncertainty of a decision context. Instead of searching for a solution to a deterministic problem, the objective, in the presence of uncertainty, is to optimize the expected value of a criterion while minimizing the associated risk. Stochastic programming [Kleywegt and Shapiro, 2001] enables the modeling of the uncertainty by introducing random variables in the optimization model. In robust optimization [Ben-Tal et al., 2009], no specific distribution of the random elements is known, but a set of possible scenarios is introduced. In the most conservative setting, the objective is to find a solution that is feasible for all scenarios (strict robustness). In other words, the worst-case over all scenarios is optimized. However, because a solution may have a poor objective value, other concepts have been proposed. Since not all variables must be fixed before the realized scenario is known, the solution variables may be split into two parts (adjustable robustness, see [Ben-Tal et al., 2009]). While some of them are fixed at the beginning, the remaining variables can be adjusted after the scenario is revealed. Finally, recovery robustness [Liebchen et al., 2009] enables the repairing of parts of the solution using a recovery algorithm.

Despite these improved modeling functionalities, some limitations still persist. In particular, enriching an optimization model may increase the difficulty of finding an efficient optimization method. For instance, introducing uncertainty or robustness aspects in an optimization model increases its complexity, and few efficient computational tools have been developed to deal with these aspects [Beyer and Sendhoff, 2007]. Finally, regardless of the efforts to improve the optimization model during the design, some decision criteria may remain unknown until the decision-making process takes place. In this context, interactive approaches may provide an alternative to overcome such limitations. Indeed, elements of the actual problem that are not initially present in the optimization model may be tackled during the optimization process. Hence, the user's expertise may be exploited in order to enrich or adjust the optimization model.

3.2 Performance inadequacy

The quality and efficiency of the optimization procedure are important conditions for a successful integration of an optimization method into a real-world software application. Several limits may hinder the implementation of such a method from meeting user perfor-

mance requirements. Beside the inherent complexity of some optimization problems, we have identified four major limits which cover different steps of the design process:

Insufficient specifications of performance requirements First, the impediment to meet performance requirements may be related to a lack of precision in end-users' requirements. In particular, during the design process it may be difficult to know what makes an adequate compromise between computation time and solution quality.

Choice of the optimization method Next, the determination of the adequate optimization method remains difficult at the design stage [Rothlauf, 2011] and it turns out to be a critical point for the efficiency of the optimization. This design choice often results from the determination of a trade-off between the tractability of the problem to solve and the validity of the chosen model. In other words, how the model can be simplified so that it can be easily solved while maintaining its validity?

Parameter setting In addition, parameter setting may also have a great impact on the overall efficiency of an optimization procedure. For both exact and approximate optimization approaches, parameter setting still represents a challenge [Hutter et al., 2010b, Adenso-Díaz and Laguna, 2006].

Lack of test data Finally, the data used to test the method at the design stage may substantially differ from the real data used during the decision-making process. In that case, it is difficult to both evaluate the performance of the optimization procedures and to know if performance requirements will be met.

These different factors can directly impact the overall performances of the optimization procedure. A general research direction to improve performance involves the development of new optimization methods. Concerning parameter setting, various automatic approaches have been proposed. Tuning procedures were proven successful in the case of metaheuristics [Birattari, 2005] as well as exact optimization algorithms [Adenso-Díaz and Laguna, 2006, Hutter et al., 2010b]. There is also a growing interest in optimization methods that automatically generate efficient search strategies such as hyper-heuristics [Burke et al., 2013]. However, purely automated approaches are limited when the users' requirements are not clearly defined, or when little or no real data is available for testing the method. In these cases, interactive optimization can be instrumental in capturing, during the optimization process, user expectations and knowledge about the targeted problem instance.

3.3 Non-acceptance and misunderstanding of optimization systems

Another common limit of optimization methods concerns the user acceptance of the optimization system and the user's confidence in the solutions provided by the method. According to [Davis et al., 1989], user acceptance is mainly determined by the *perceived usefulness* and *perceived ease of use* of a system, this is known as the Technology Acceptance Model (TAM). For further details on this model and discussions about its shortcomings we refer the reader to [Legris et al., 2003, Bagozzi, 2008]. For a decision-support context, the *confidence* or *trust* of the user in the system is another key aspect for its acceptance [Lee and See, 2004, Muir, 1987]. The confidence is particularly important for an optimization system as the system must solve complex problems that cannot be fully apprehended by the user. In short, the perceived usefulness, perceived ease of use and confidence in the system are essential for the acceptance of an optimization system by its users.

Concerning the confidence in an optimization system, two problematic situations may occur. A user may have no confidence in an efficient system, or have confidence in an

inefficient one. In the first case, the user may reject solutions that would result in good decisions. In the second case, the user overestimates the effectiveness of the optimization system, and may accept solutions that could be improved by other means. In both situations, we can consider that the user misunderstands the value of the optimization system's results.

The problem of confidence in an inefficient decision support system has been studied in [Davis and Kottemann, 1994]. In an experiment, the authors compare the perceived performance of two decision support approaches. The first one is a 'what-if' analysis approach in which the user tests different alternatives. The second approach is based on decision rules that generate recommendations without involving the user in the recommendation process. The experiment was performed on 52 subjects that used both approaches. Perceived and actual performances were compared. Results indicate that the decision rules approach was not perceived as more efficient than the what-if analysis even though it provided better results. In other words, users had no confidence in the efficient approach and had confidence in the less efficient approach. This illustrates the importance of user perception and confidence as it eventually leads to the system acceptance or not.

Interactions appear to have a positive impact on user's acceptance and confidence. Interactions, and thus interactive optimization, seem to help the user learn about the optimization problem [Belton et al., 2008]. It also helps the user better understand the optimization procedures [Klau et al., 2002]. With a better perception of the system, the user is more inclined to accept it. In addition, adequate interactions may also prevent situations in which a user over-estimates the effectiveness of an optimization system.

The interactive methods reviewed in the next section have been primarily designed to address the inherent limits of optimization models and the difficulty to obtain adequate performances. However, the understanding and acceptance of the system by users are often mentioned in the literature as a second argument to justify an interactive approach. The fact that interactions help the user learn about the optimization problem and its associated procedures is a characteristic shared by most interactive optimization methods.

4 Survey of interactive optimization approaches

This section presents a survey of existing interactive optimization approaches. These approaches can be defined as optimization methods with which an end-user or decision-maker can interact. This interaction takes place during a decision-making process and allows the user to significantly modify the results or performances of the optimization system. Considering this definition, this overview focuses on interaction between human users and optimization systems. Interaction with a non-human third party such as in optimization-simulation coupling (e.g., [Fu, 2002]) is out of the scope of this survey. In addition, the interaction implies that several information exchanges can be performed, and that the user's feedback depends on intermediate optimization results (i.e. the interaction process is an interaction loop). The post-analysis of optimization results is not considered as an interaction, unless it is used to modify the optimization system for solving other problem instances. Finally, dynamic optimization (e.g., [Nguyen et al., 2012]) is not considered in this overview since the dynamic nature of data is not a user's feedback on the optimization system, even if a human operator provides these data.

Overall, five classes of interactive optimization approaches are presented in this section. The first two, trial-and-error and interactive reoptimization, correspond to basic interactive methods. They are commonly used in practice, however, they have a limited coverage in the research literature due to the trivial nature of the interaction. The three remaining classes, namely interactive multiobjective optimization, interactive evolutionary computation and human-guided search, are more sophisticated approaches for which extensive literature exists. In addition to these five classes, other interactive optimization

methods, which do not correspond to any of these classes, are briefly presented in Section 4.6.

The presentation of these classes of interactive optimization approaches follows the same outline. First, the general principle of the interaction is explained. Next, illustrative examples drawn from the literature are presented. Finally, their main limitations are discussed.

4.1 Trial and error

Trial-and-error is one of the simplest approaches for involving a user in the optimization process. In the context of interactive optimization, a trial-and-error method can be defined as an iterative and direct adjustment of the optimization system by the user. More precisely, the user feedback is not generalized and corresponds to preference information used by the optimization system. In addition, each iteration of the optimization process is independent of the previous one. A common example is when the user modifies optimization parameters and restarts the optimization procedure.

Trial-and-error approaches can be used for either modifying elements of the optimization model or adjusting the optimization procedure. In the first case, the approach can be seen as a what-if analysis. The user simply adjusts data, constraints, or objectives of the optimization problem, and the optimization system provides a solution that evaluates these modifications. In the second, and more advanced case, the user knows the basic principles of the optimization process, and more specifically, the role of some parameters. Therefore, the user can adjust the value of these parameters in order to adapt, in a rather limited manner, the optimization process. For instance, this can be useful to adjust convergence speed or to increase the diversity of a heuristic search procedure.

An early successful application of trial-and-error interactive optimization is described in [Braklow et al., 1992]. The authors developed an optimization system for the design of delivery routes of a freight service. The main interaction is based on a trial-and-error approach. The user can modify some aspects of the problem in order to reflect features that are not modeled by the optimization problem. Subsequently, a new solution is generated by the system. The optimization system, called SYSNET, has been used by the freight company YRC (formerly Yellow Transportation) resulting in substantial cost savings.

A more recent approach, based on an interactive trial-and-error approach, is presented in [Cesta et al., 2003]. This approach deals with the determination of schedules for data transmission from a space probe. The optimization system called Mexar was developed for the Mars-Express mission started in 2003 and extended until 2014. The optimization problem consists in determining sequences of data packet transmissions within a set of time frames. It is a variant of the bin packing problem. The problem is modeled as a constraint satisfaction problem and solved using heuristics and metaheuristic search methods. The user can interact with the optimization system, using trial-and-error, for both adjusting the search strategy and modifying constraints. More precisely, the user monitors the optimization procedure in order to identify potential problems such as stagnation of the search around a local-optimum. When the user identifies a problem, he can adjust parameters and rerun the optimization procedure. Similarly for the optimization model, the user can iteratively refine the problem in order to integrate problem-domain knowledge, by relaxing or adding constraints. An updated version of this tool, called Mexar-2, is used at the European Space Agency since 2005. It is based on the same interactive framework [Cesta et al., 2007].

Another trial-and-error approach, presented in [Halim and Lau, 2007], is used for tuning trajectory metaheuristics, i.e. metaheuristics based on local-search. The authors propose a framework for visualizing the behavior of the search algorithm and identifying problematic behavior such as stagnation of the search. The visualizations help the user to

adjust parameters or to select new components for the strategy. This approach is applied to the problem of tuning a tabu search method. For this case study, the user is able to add some strategic rules that modify the search behavior when specific conditions are met.

In these different trial-and-error methods, the user constructs a mental model of the optimization problem and optimization method. He learns about the relationships between the parameters or components he changes and the response of the system. This implicit knowledge allows the user to progressively adjust the system in order to meet his preferences. Although it is quite intuitive and simple to implement, this form of interaction has several weaknesses. First, a trial-and-error approach does not use a memory or computational model of the user's preferences. The interactive process relies on the memory of the user for controlling the exploration of different configurations and for comparing the different alternatives. Therefore, some cognitive biases can affect the exploration and, more importantly, the cognitive load may limit the user in its exploration. The direct consequence of the important cognitive load is that only a limited number of configurations are considered at the expense of the user's preferences. A second major limit of trial-and-error approaches is that the user requires some knowledge about the optimization procedure or optimization model for which he changes the parameters or components. Usually, we assume that users of decision-support tools have an expertise in the application domain, but it is risky to consider that users properly understand the optimization procedure or the model of the optimization problem [Barthélemy et al., 2002]. In order to mitigate these aspects, appropriate visualizations of results are necessary to support the trial-and-error process. Some visualization tools and graphical representations have been proposed in the context of optimization systems. In [Jones, 1994], the author reviews a wide range of representations of optimization results. In [Miettinen, 2014], the author gives a survey of visualization methods for multiobjective optimization. However, considering the previous limitations and even with appropriate visualizations, trial-and-error seems rather limited and the other interactive approaches presented in the following sections are probably more advisable ways to take into account the user's preferences.

4.2 Interactive reoptimization

Contrary to trial-and-error that can be used for either adjusting the optimization problem or tuning the optimization procedure, interactive reoptimization only aims at refining the optimization problem. In interactive reoptimization, the refinement is done by progressively adjusting a candidate solution using a reoptimization procedure. In this article, we introduce the term *interactive reoptimization* to designate reoptimization methods that are used in an interactive context. Other reoptimization methods have been proposed for dynamic optimization problems without any user interaction, for instance in [Ausiello et al., 2007] and [Zych, 2012]. In dynamic optimization problems, the problem instance changes over time, and the solution has to evolve with the updated problem instance. However, the reoptimization in a dynamic context does not correspond to an interactive optimization approach as the perturbations do not come from the user.

As mentioned previously, an optimization model may contain some simplifications or inaccuracies that require some adjustments by the user during the decision-making process. To correct these inaccuracies, the user can directly modify a candidate solution provided by the optimization procedure. However, manually modifying a solution has major drawbacks if it is not assisted by an optimization procedure. First, it may be difficult for the user to apprehend all constraints and objectives of the optimization problem when a solution is manually edited. In addition, due to the complexity of considered optimization problems, it is generally difficult or impossible to reflect the modification to the whole solution. The fact that a local modification in a solution implies adjusting other parts of the solution is referred to as the cascading or propagation effect [Pinedo, 2012]. Interactive reoptimization

aims at overcoming these obstacles. In this approach, solutions modified by the user are reoptimized. The reoptimization procedure globally optimizes a solution to take into account local modifications applied by the user.

Starting from an initial candidate solution, the interactive reoptimization process alternates between two phases. First, the user specifies changes to be made on the current solution. Second, a reoptimization procedure is applied to perform the changes and to optimize the rest of the solution accordingly. The process is iterated until a satisfactory solution is obtained.

For the reoptimization, some specific objectives or constraints are added to the problem model. First, a constraint or objective ensures that the changes requested or performed by the user on the previous candidate solution are present in the reoptimized solution. In addition, an objective can be added to minimize the distance between the reoptimized solution and the previous candidate solution. This minimization of the impact of the reoptimization is required to maintain coherence in the interactive process. If possible, the reoptimized solution should be similar to the previous solution because the changes requested by the user have been expressed on the basis of the previous solution. With a completely different reoptimized solution, these changes may lose their meaning. Overall, minimizing the distance between the reoptimized solution and the previous solution favors the convergence toward a satisfactory solution for the user. This property of the reoptimization to minimize the perturbations it induced is referred to as the *stability* [Hamel et al., 2012].

Basic interactive reoptimization approaches are presented, for example, in [van Vliet et al., 1992] for a variant of vehicle routing problems with time windows, and in [Pinedo, 2012] for different flow shop problems. In both cases, the user can manually modify a solution, and then apply a reoptimization procedure. The reoptimization procedure is applied on a portion of the solution while modifications made by the user are frozen, i.e. the parts modified by the user remain unchanged after the reoptimization.

In [Hamel et al., 2012], a more advanced interactive reoptimization approach is presented for linear optimization. The proposed method is used to select an adequate solution from the set of optimal solutions. The interactive process is as follows. A first "average" optimal solution is computed and presented to the user. Next, the user can modify the values of some decision variables within the given ranges of optimality. A new optimal solution is then computed by a reoptimization procedure to satisfy the desired changes. Finally, the user can refine his preferences and reoptimize the solution until a satisfactory solution is obtained. In [Hamel et al., 2012], the authors propose four variants of their reoptimization method and study two properties of reoptimization they call *stability* and *responsiveness*. The stability, as previously mentioned, is the ability of a reoptimization procedure to minimize the changes it induced on a solution. The responsiveness is the property of the reoptimization procedure to produce a solution in a short time. They propose different procedures to optimize these two properties, including different objectives for reducing the reoptimization distance and the reuse of previous results to improve the computation time. The experimental results indicate that the computation time for reoptimizing a solution is substantially shorter than the time for obtaining an initial optimal solution. The authors report a maximum reoptimization time of 600 milliseconds for problems that necessitate up to 45 minutes for the initial optimization. In addition, the introduction of an objective to ensure stability significantly reduces the distance between candidate solutions and the corresponding reoptimized solutions, thus favoring the interactive process.

Another interactive reoptimization approach that makes use of an explicit stability objective is proposed in [Meignan, 2014]. The reoptimization approach is applied to a

shift scheduling problem. The user can request some changes to be made on a candidate solution and then apply a reoptimization procedure. The objective function for the reoptimization combines the initial optimization objectives, an objective for applying the changes requested by the user, and another objective to minimize the modifications induced by the reoptimization (the stability objective). In [Meignan, 2014], a computational study of the proposed reoptimization procedure is performed. For the considered problem it is shown that a global reoptimization procedure is required to adjust candidate solutions even when requested changes concern a very limited part of the solutions. This latter study also illustrates the difficulty of adjusting a solution.

A shortcoming of interactive reoptimization is that the changes requested by a user may impair the quality of the solution by over-constraining the problem in comparison to an exact expression of the missing feature. This limit of interactive reoptimization has been observed in [Meignan, 2015] for an experiment on interactive reoptimization conducted with 16 test subjects. As explained previously, interactive reoptimization allows a user to introduce some constraints or objectives that are not initially designed, or allows him to adjust inaccurate constraints or objectives. The "missing features" (i.e. constraint or objective not modelled) are expressed by the user with local changes of a solution. However, these changes may not represent the exact missing features, and therefore the user may need to request more changes than it would be necessary to correct optimally the solution. This excess of change requests can potentially impairs the quality of the solution obtained after the reoptimization.

4.3 Interactive multiobjective optimization

Interactive multiobjective optimization is probably the most prominent class of interactive methods in the optimization literature. These interactive methods, as well as non-interactive multiobjective optimization methods, address the problem of determining an adequate compromise between different objectives [Branke et al., 2008].

A large majority of optimization problems solved by decision support tools contains multiple conflicting optimization criteria. When the optimization problem is designed, it may be difficult to combine these criteria in one single objective function. The aggregation of different objectives is generally difficult because the relation between the objectives can be complex, and an adequate trade-off may only be known at the time of the decision-making. Therefore, instead of aggregating the objectives which could result in inaccuracies in the problem model, in multiobjective optimization they are kept separated, and the appropriate trade-off is determined during the decision-making process.

For multiobjective optimization problems, the Pareto-front represents the set of solutions for which no improvement in one objective is possible without deteriorating the value of at least one other objective. Each solution of the Pareto-front corresponds to a particular trade-off between the objectives. The preferences provided by the user during the decision-making process should allow the determination of an adequate solution within this set. Three modes for defining preferences are usually distinguished [Ehrgott and Gandibleux, 2000]. The preferences can be provided by the user before the optimization step during the parameter-setting. In this *a priori* mode, the optimization procedure generates a solution that matches, as best as possible, the preferences of the user. In an *a posteriori* mode, the user integrates his preferences after the optimization step. In this case, the optimization method determines a set of solutions that approximates the Pareto-front. Then, this set is analyzed by the user to identify an appropriate solution. Finally, the preferences can be defined *interactively* during the solving process.

The basic principle of the interactive process is the same for most interactive multiobjective optimization methods [Miettinen et al., 2008]. An initial solution or set of solutions is generated. Then, two steps are repeated until a candidate solution is accepted by the

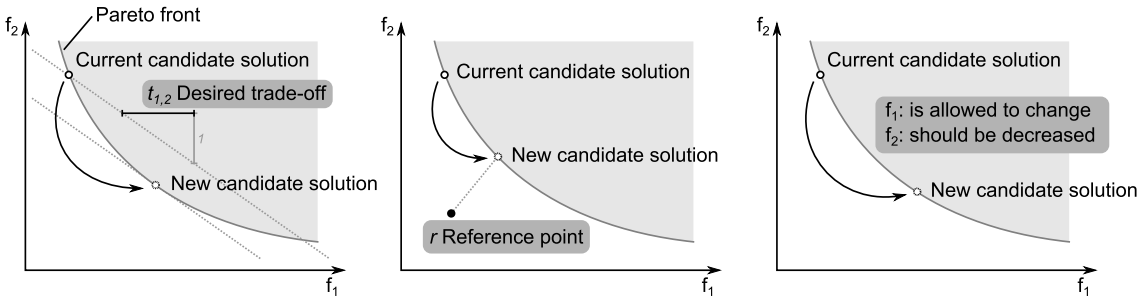

Figure 4: Illustration of the three types of preference information in interactive multiobjective optimization. On the left, the preference is expressed as a trade-off value. In the middle, the preference is given by a reference-point. On the right, the preference is specified with a classification of the objectives.

user. In the first step, the user evaluates some proposed candidate solutions, or provides other preference information with regard to the proposed solutions. This information is used to generate or update the preference model. In the second step, new candidate solutions are generated based on the updated preference model and then proposed to the user. Note that an a-priori method can easily be considered in an interactive process where preferences are adjusted by trial-and-error.

The interactive approach has several advantages over a-priori and a-posteriori approaches. First, in an a-priori approach, it is generally difficult for the user to provide adequate preference information without knowing the possible solutions. This difficulty is not present in an interactive approach where the user can refine his preferences based on the proposed solutions. For a-posteriori approach a different problem arises. It can be difficult to generate a good approximation of the Pareto-front. The set of solutions may be too large, and limiting its size using some parameters may hinder finding interesting solutions. This is not an issue for interactive approaches because Pareto solutions can be progressively generated. Besides, in interactive approaches the user may guide the search toward interesting regions of the Pareto-front. This guidance may contribute to avoid the generation of irrelevant solutions and thus reduces the computational time [Branke, 2008].

In [Miettinen et al., 2008], the authors give an overview of interactive methods for solving multiobjective optimization problems. They identify three categories of interactive methods according to the type of preference information extracted from the user's feedback: trade-off based methods, reference-point approaches, and classification-based methods. These three types of preference information are illustrated in Figure 4.

In trade-off based methods, the preference information is expressed as relative variations of objective values between solutions. An example of trade-off information is given on the left-hand side of the Figure 4. In this example, the desired trade-off rate noted $t_{1,2}$ is provided. The trade-off rate corresponds to the gain on f_1 which is expected for a deterioration of f_2 by one unit. On the figure, this ratio is verified for the new candidate solution. Such a trade-off information is used for instance in the Guided Multi-Objective Evolutionary Algorithm (G-MOEA) proposed in [Branke et al., 2001]. In G-MOEA, the user provides the desired minimum and maximum values of the trade-off rate. Then, a set of candidate solutions between the defined bounds are generated using an evolutionary algorithm. Based on this set of solutions, the user can select a satisfying solution, or adjust the trade-off rates to obtain new solutions. In this approach, the trade-off information is referred to as *subjective trade-off* [Miettinen et al., 2008]. The values are directly set by the user and do not necessarily correspond to achievable trade-offs. Trade-off information can also be obtained from measurable properties of the problem, for instance by comparing feasible solutions. In this case, the preference information is an *objective trade-off*

and the feedback of the user consists in an evaluation of these objective properties of the problem. A review of trade-off based interactive methods is given in [Eskelinen, 2008] with a particular focus on objective trade-off based methods.

For reference-point approaches, the user specifies his preferences with desired values, or range of values, for each objective. Then, one or several candidate solutions on the Pareto-front are generated as close as possible to the desired reference points. This approach is illustrated at the center of Figure 4 with one reference-point. In comparison to trade-off based methods, reference-point approaches are more suitable for the discrete case, generally indifferent towards the shape of the Pareto frontier, and the preference information is usually more intuitive [Deb and Sundar, 2006]. The Light Beam Search approach [Jaszkiewicz and Słowiński, 1999] is a reference-point approach that provides two ways of exploring the Pareto-front. First, the user defines a reference point and a candidate solution is obtained by projection on the Pareto-front. This reference point can be modified to produce new candidate solutions. Second, in addition to the candidate solution obtained by projection, a set of neighboring solutions is provided to the user. These neighbors allow a local exploration of the Pareto-front. In [Jaszkiewicz and Słowiński, 1999], this approach is illustrated on a chemical engineering problem which consists in selecting parameters for the production of a plastic compound. Four conflicting objectives are considered for the optimization of the production process. The authors observe that, thanks to the two methods for changing the preferences, a satisfactory solution can be obtained in a limited number of iterations.

Finally, in classification-based methods, the user identifies iteratively the objective function that should be improved and the ones that could be deteriorated with respect to a given intermediate solution. Therefore, this preference information is expressed by classifying objective functions according to preference categories. The user may also indicate desired values, or a range of variation, for some objective functions. An illustration of classification-based methods is given on the right-hand side of Figure 4. The preference information given by the user is a classification of the objectives that directs the search for new candidate solutions. Such a classification approach is used, for instance, in the NIMBUS method [Miettinen and Mäkelä, 2000]. In this interactive method, when a candidate solution is presented to the user, he can express his preference by assigning a class to each objective functions. Five classes are available: a) objective functions whose values should be improved from the current values, b) objective functions whose values should be improved until they reach the desired values given by the user, c) objective functions whose current values are acceptable, d) objective functions whose values can be impaired until they reach the bounds given by the user, e) objective functions whose values are allowed to change freely. Then, a new solution is generated according to this preference information. The process is repeated until a solution with a satisfying trade-off between objectives is found by the user. The NIMBUS method has been applied to diverse application domains such as the process of continuous casting of steel [Miettinen, 2007], dose-plan for radiotherapy [Ruotsalainen et al., 2010], wastewater treatment [Hakanen et al., 2011], power plant process [Tveit et al., 2012], and heat exchanger network [Laukkanen et al., 2012].

Each of these three groups of interactive methods has its benefits and limitations. A first issue concerns the difficulty for a user to provide preference information, in particular when numeric values have to be set. On this regard, trade-off based methods appear to be less intuitive for the user than reference-point and classification-based methods [Deb and Sundar, 2006, Shin and Ravindran, 1991]. However, the advantage of trade-off based information is the potential to reuse the same information for solving different problem instances. For instance, a trade-off rate that has been defined for a first problem instance

may remain meaningful for a second problem for which the magnitudes of objective values are significantly different. In this case, reference-point or classification information defined for the first problem instance cannot be reused for the second one.

In [Greco et al., 2008], the authors propose a method to overcome the difficulty of providing numerical values. In the proposed method, called Dominance-Based Rough Set Approach (DRSA), the user has to classify candidate solutions into *good solutions* or *inadequate solutions* categories. Then, the system infers some rules that restrict the region of the Pareto-front in which new candidate solutions are sampled. For this method, no numerical values have to be provided by the user. In addition, the rules presented in form of *[if..., then...]* decision rules are intelligible and can be easily revised. The DRSA method facilitates the definition of preference information. However, the reusability of this preference information is limited because the premises of inferred rules use reference values. As previously indicated, such values generally cannot be reused for solving different problem instances.

Finally, beside these difficulties in defining preference information and reusing previous preferences, a general limit of interactive multiobjective optimization methods relies on the process of exploring the Pareto-front. During this interactive process, the user may tend to limit the number of explored trade-offs due to the systematic presence of a loss in some objective values at each step of the exploration. By definition, moving from one Pareto optimal solution to another causes a deterioration in at least one objective and a gain in at least another objective. This exploration of different trade-offs by the user may be subject to two cognitive biases, namely *anchoring effect* and *loss-aversion* [Arnott, 2006, Smith et al., 2008]. In the context of Pareto-front exploration, the anchoring effect can be viewed as the tendency of a user to stay focused on the first provided solutions and to pay less attention to alternative trade-offs that are presented later. The loss-aversion would be the fact that when a user compares two Pareto solutions he will consider the losses in objective values more important than equivalent gains. As a result, the user will tend to avoid losses even when there is an equivalent gain. A consequence of these two cognitive biases is that users may limit their exploration of the Pareto-front. The NAUTILUS method proposed in [Miettinen et al., 2010] aims at minimizing the impact of these possible cognitive biases and in particular the loss-aversion bias. Instead of starting the interactive process from a Pareto optimal solution, a non-optimal solution is provided. Subsequently, the user indicates at different steps which objective value to improve until a Pareto optimal solution is attained. During this interactive process, the user progressively focusses on a satisfactory trade-off and the intermediate solutions systematically improve at least one objective value without any loss in other objectives. Other interactive approaches that provide intermediate non-Pareto optimal solutions may also prevent the loss-aversion bias. For instance, in interactive evolutionary approaches for multiobjective optimization problems [Jaszkiewicz and Branke, 2008] the loss-aversion bias may be reduced by the fact that candidate solutions are progressively improved while the user guides the search toward an interesting region of the Pareto front.

4.4 Interactive evolutionary algorithms

Interactive evolutionary algorithms form another class of interactive optimization methods. These approaches concern optimization problems for which objectives are difficult to quantify, or for which mathematical models are inappropriate. In other words, it addresses problems that require the user's subjective evaluation of solutions.

Interactive evolutionary algorithms are evolutionary algorithms in which the evaluation of solutions is provided entirely, or partially, by the user [Takagi, 2001, Banzhaf, 1997]. An outline of a simple interactive evolutionary procedure is presented in Algorithm 1. The search starts by generating an initial population of solutions and then evolutionary search

operators are applied. Within the evolutionary loop (i.e. recombination, mutation and selection), the user interacts with the system to provide the evaluation of the solutions according to his own perception of the quality of the solutions. The evaluation provided by the user at each generation is used to select the solutions at the origin of the next generation. This interactive process is repeated until the user identifies a satisfactory solution.

Algorithm 1: Outline of a simple interactive evolutionary procedure.

```
1 P ← initializePopulation()
2 NewSolutionsRequested ← TRUE
3 while NewSolutionsRequested do
4 |   P' ← recombine(P)
5 |   P' ← mutate(P')
6 |   Evaluation ← requestUserEvaluation(P')
7 |   P ← select(P', Evaluation)
8 |   NewSolutionsRequested ← requestUserToContinue()
9 end
```

The interaction between the user and the system allows the introduction of subjective optimization criteria that could not be formulated explicitly or identified before the decision-making process. The subjective evaluation of solutions can rely, for instance, on the aesthetic judgment of the user. For instance, in [Kim and Cho, 2000] an interactive genetic algorithm is applied for a problem of fashion design. The objective is to find an appropriate combination of clothes' colors and shapes. A solution encodes the styles and colors of the different pieces of clothing. Then, during the evolutionary search process, the solutions of the current population are displayed in a form of 3D models and the user evaluates these solutions using rating scales. The solutions that are the most rated are retained for creating the next generation and the solution with the best score is kept unaltered for the next generation. This elitism strategy simplifies the user's evaluation by providing a reference to the previous generation. It also facilitates the identification of a final solution without backtracking to previous generations. An extensive review of applications of interactive evolutionary computation is given in [Takagi, 2001].

The main issue of interactive evolutionary algorithms is the cognitive load of the evaluation process for the user. An evolutionary process normally requires a large number of evaluations in order to converge to interesting regions of the search space. Due to user fatigue, it is not conceivable to ask the user to evaluate too many solutions. The evaluations may not be reliable, nor the interaction appropriate, if the number of solutions to evaluate is too large.

Different variants of interactive evolutionary algorithms have been proposed to reduce the cognitive load of solutions' evaluation. A first approach consists of using objective criteria to limit the number of solutions the user has to evaluate in a population. In [Banzhaf, 1997], the author suggests that, before the evaluation, a representative subset of solutions is selected and evaluation is done by the user only on these selected solutions. The fitness values are then generalized to the entire population of solutions. This approach has been implemented, for instance, in [Lee and Cho, 1999] for a problem of image retrieval. The authors use a clustering method to select some representative solutions that are evaluated by the user. Then, the evaluation is generalized to the whole population of solutions. For this, the fitness of a non-evaluated solution is calculated according to its distance to evaluated solutions.

Another direction for reducing the number of evaluations made by the user consists in

learning a model of the user fitness, hereinafter referred to as fitness model. This model generalizes the subjective evaluations made by the user and represents the correlation or response between some attributes of solutions and the respective user's evaluations. In this model-based approach, the evaluations of solutions are used as training instances to progressively adjust the fitness model. When the model has accumulated enough data, it can be used to automatically assign a fitness value to generated solutions. User intervention can then be limited to the adjustment of this fitness model. Different methods have been investigated to learn such a user fitness, e.g. case-based reasoning [Babbar-Sebens and Minsker, 2010], artificial neural networks [Biles et al., 1996], as well as support vector machines [Llorà et al., 2005]. However, interactive evolutionary algorithms remain limited for solving complex optimization problems. They are only suitable when the user can, and indeed must, provide a reliable evaluation of the solutions.

4.5 Human-guided search

In interactive evolutionary algorithms and interactive multiobjective optimization, the contribution of the user aims at enriching the optimization problem during the optimization process by, respectively, defining the evaluation of the solutions, and adjusting the trade-off between objective functions. Contrary to these classes of approaches that focus on the enrichment of the optimization model, human-guided search procedures propose to improve the efficiency of the optimization procedure. In the human-guided search approach, the user provides information that guides the optimization process [Klau et al., 2010] and that has no direct impact on the optimization model. Such a combination between human and computer solving strategies has been investigated in early works [Krolak et al., 1971]. The initial motivation was to make the best use of the limited computational capabilities of computers by taking advantage of human's problem-solving abilities. However, the development of computers has changed the targeted issues, and human-guided search now intends to improve the efficiency and possibly robustness of optimization procedures by exploiting users' heuristic information for guiding the search procedures.

Optimization methods considered in human-guided search are local-search procedures such as hill-climbing [Anderson et al., 2000] and tabu-search [Klau et al., 2002]. The interactive process alternates between the application of this local-search procedure and a feedback session in which the user can express some preferences on how to improve the current solution. The main preference mechanism allows the user to restrain the search space by selecting the parts of the current solution where the search should focus. More precisely, an initial solution is proposed to the user so that he can define different degrees of *mobility* or assign *penalties* to solution' components. This is done in order to specify which parts are satisfactory, and which ones need to be improved. The next optimization step will try to improve the solution using a guided local-search procedure. The guidance consists in focusing the local-search on elements of the solution with high mobility or strong penalty while keeping low mobility or not penalized elements in place. This procedure can be viewed as an interactive variant of the guided local search metaheuristic [Voudouris and Tsang, 2003] for which the user provides the heuristic information to reduce the size of the neighborhood.

An illustration of the usage of mobilities is given in Figure 5. This example is inspired by the human-guided search approach applied to the Capacitated Vehicle Routing Problem with Time-Windows (CVRP-TW) presented in [Anderson et al., 2000] and [Scott et al., 2002]. In the upper-left side of the figure, a candidate solution is represented. On this solution, the user can assign different mobilities to customers in order to guide the next local-search step. A high mobility on a customer indicates that the route has to be improved, and a low mobility indicates that the route should remain unchanged. In this simplistic example, high mobilities are assigned to the customers above the depot since

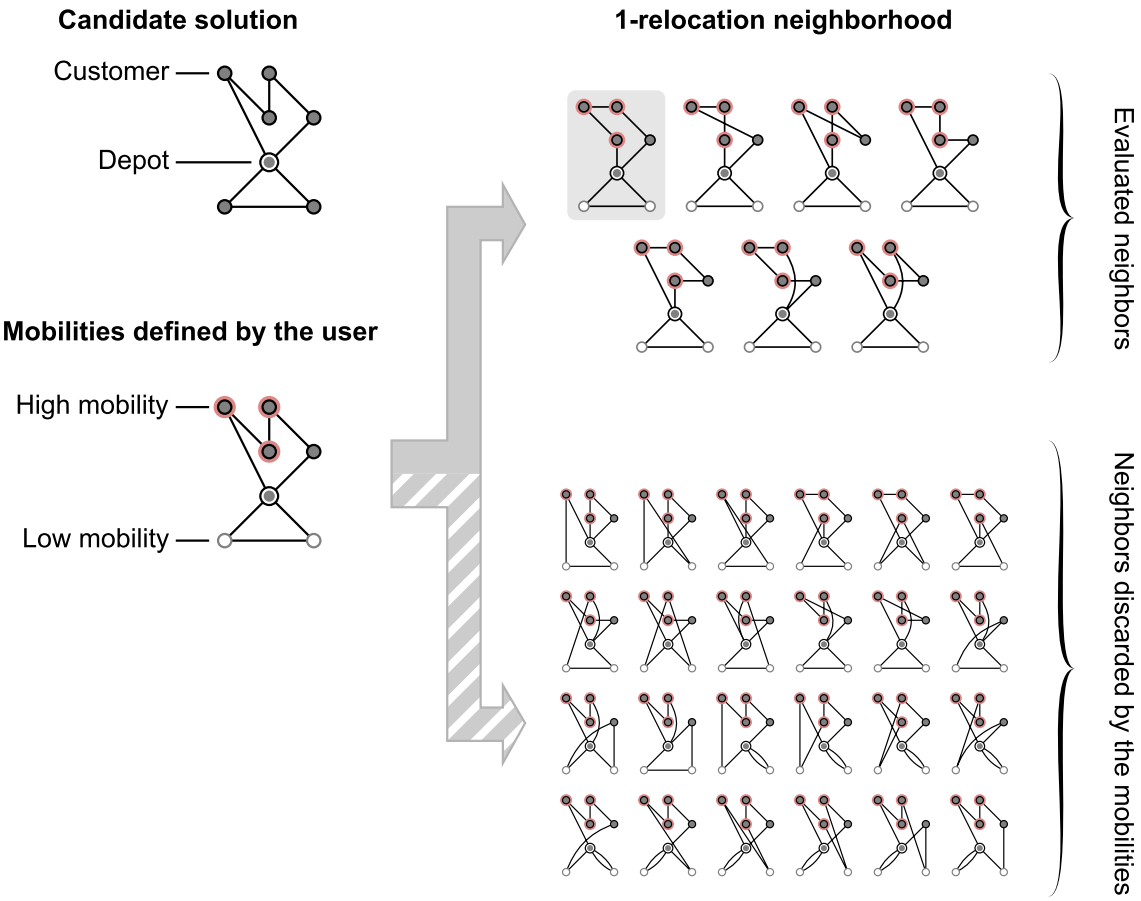

Figure 5: Example on a VRP of the restriction of a neighborhood by mobilities. The solution on a gray background in the *1-relocation* neighborhood is the best neighbor with a better cost than the initial solution.

the shape of the route suggests a possible improvement on this part of the solution. These mobilities are then used to guide the local-search procedure applied to the candidate solution. On the right-hand side of the figure, the *1-relocation* neighborhood of the initial solution is given. To obtain this neighborhood, a move corresponds to the relocation of one customer [Gendreau and Tarantilis, 2010] as in the human-guided search method proposed in [Anderson et al., 2000]. Thanks to the mobilities, the exploration of the neighborhood is limited to the seven solutions represented in the upper right of the figure, and allows to obtain a better solution represented in gray. As illustrated, if the user defines appropriate mobilities, the performance of the local-search procedure can be improved by reducing the number of neighbors that are evaluated. It should be noted that, in Figure 5, the solution is improved with only one move. However, for real implementations of the human-guided search approach, several moves are performed during the local search.

This human-guided search method with mobilities is studied in [Anderson et al., 2000] and the results of an experiment with users is reported. Four test subjects are asked to solve eight different instances of the CVRP-TW using a human-guided search approach. During the interactive optimization process, the user defines the mobilities to guide the next search iteration. He also perturbs the current solution in order to diversify the search. The local-search procedure invoked by the users is a hill-climbing search. The results are compared to an automated procedure that consists in a multi-start hill-climbing method. The results indicate that on average the costs obtained after 1.5 hours of human-guided search are attained with 5.0 hours of computation by the automated procedure. In addition, for the considered problem instances, the results of the human-guided search approach

are comparable to that of state-of-the-art automated approaches. These results suggest that the interaction has transformed a basic local-search procedure into a competitive optimization procedure.

In [Klau et al., 2002], the authors present a human-guided search platform and report some results on four applications: a graph layout problem, a variation of the traveling salesman problem, a protein-folding problem, and a jobshop problem. The interaction process proposed by the platform is similar to the one used for the CVRP-TW application presented in [4]. The user can assign three degrees of mobility to the elements of a solution: low, medium and high. In addition, the user can manually modify solutions, backtrack to previous solutions, change some parameters of the search procedures, and also invoke a search procedure to improve the current solution. The available procedures are a hill-climbing procedure and a tabu search method. Both procedures use the mobilities to reduce the size of the explored neighborhoods. For the tabu search, the low-mobility elements are considered as tabu elements for the whole improvement step. The high-mobility elements are initially considered for the moves, and medium-mobility elements can be impacted by the moves of high-mobility elements. Contrary to the hill-climbing procedure, the tabu search is able to diversify the search and does not require that the user perturbs the solution to escape from a local optimum. The results reviewed in [Klau et al., 2002] tend to indicate that a human-guided search approach can improve the performance of simple local-search procedures in terms of solution quality when the computational time is limited. Hence, one particularly interesting aspect of human-guided search is the possibility for the user to control how much effort the optimization procedure should spend on particular sub-problems [Scott et al., 2002].

In [Chimani et al., 2005], the authors use the same platform as in [Klau et al., 2002] for solving a scheduling problem in the domain of air freight. However, the human-guided search method is combined with an interactive reoptimization approach. In addition to the mechanism of mobilities, the user can add new constraints to reflect features that are not modelled in the optimization problem. When the optimization procedure is invoked, an additional term in the objective function minimizes the modifications of the solution in order to maintain the consistency of the new constraints.

A similar combination of human-guided search and interactive reoptimization is considered in [Meignan et al., 2011] for solving a network design problem in the field of forestry. The objective of the optimization problem is to locate a set of tree-shaped paths in a graph [Meignan et al., 2012]. For this problem the problem-domain expertise of the user is required to generate realistic solutions. In addition, user interaction is also required for generating new candidate solutions in a short time because the optimization is computationally very expensive. Unlike the previous methods, the proposed guidance mechanism does not use different degrees of mobility (low, high, and medium). Instead, the user selects within the current solution the parts on which the search should focus. Then, during local-search the diversification (i.e. perturbation moves applied on the current solution) is focused on the selected parts, and the intensification (i.e. the application of an hill-climbing procedure in this case) considers the whole solution. In this study, an experiment has been performed with an expert planner. The solutions obtained with the interactive approach have been compared to solutions that have been previously obtained by a manual planning method. Using the interactive approach, the expert obtained on average a gain of 9% compared to the manually planned routes. In addition, the interactive solution design took between 30 minutes and 1 hour. This time greatly improves the duration of a manual design that normally lasts several hours.

In these different applications of the human-guided search approach, the human guidance aims at improving the efficiency of the optimization procedure with heuristic infor-

mation provided by the user. This interaction is based on the assumption that the user may identify in the problem or solutions some features that are not fully exploited by the optimization algorithm. This assumption may constitute a limit of the approach because the user requires some knowledge about the optimization method and has to understand the impact of his interaction on the optimization procedure. For instance, in the human-guided search method for solving the CVRP-TW presented in [Anderson et al., 2000], the perturbation of a candidate solution performed by the user requires some knowledge about local search procedures. The operation of altering a solution may appear counterintuitive for a user who seeks to improve the solution and is not familiar with the concept of search diversification. In short, a major issue of human-guided search is that users require some knowledge about the optimization procedure while in a decision-making process the user of an optimization tool is generally sought for his expertise in the problem domain [Barthélemy et al., 2002].

4.6 Additional interactive optimization approaches

In addition to the five classes of interactive approaches detailed in previous sections, other recent albeit less investigated interactive optimization approaches have been proposed. This section briefly presents some of these interactive approaches that introduce original ways to interact with an optimization system.

4.6.1 Asynchronous-teams

In the human-guided search approach and trial-and-error approaches for parameter tuning, the user supports the optimization procedure for obtaining a more efficient optimization. Another way of supporting the optimization procedure is investigated in the Asynchronous-teams (A-Teams) approach [Talukdar et al., 1998, Rachlin et al., 1999]. A-Teams is an architecture in which computational agents share a population of solutions and cooperate in order to optimize this set of solutions. Each agent contributes to the evolution of a population of solutions by generating, improving, or removing solutions from the set. In [Rachlin et al., 1999], the authors suggest that, in an A-Teams architecture, humans can participate as agents in the optimization process. A human agent can generate, edit, and remove solutions. Contrary to human-guided search or trial-and-error, the role of the user in an A-Teams would be the same as an optimization procedure. This approach would confer a high degree of autonomy for the user. This autonomy can be interesting in order to fully exploit user's search heuristics in the optimization procedure. However, no concrete implementation of this approach in which the user has the same role as the artificial agents is reported in the scientific literature.

4.6.2 Crowdsolving (Crowdsourcing for solving complex optimization problems)

In the same line of A-team which allows users to act as a search procedure, collaborative solving tools such as FoldIt [Eiben et al., 2012] or Open-Phylo [Kwak et al., 2013] propose to exploit large groups of users for solving complex optimization problems [Schrope, 2013]. The principle is that users manually solve the proposed problem instances, compete for obtaining the best solution, and may collaborate by sharing solutions and strategies. In this approach, the optimization is, in fact, performed by users. For instance, FoldIt is a collaborative puzzle game for optimizing protein structure. The user edits a 3-dimensional representation of a protein for optimizing some properties such as its stability or its capacity to react with specific molecules [Eiben et al., 2012]. The results obtained by the community of players may contribute to the design of new proteins for curing diseases.

However, different aspects limit the application of the crowdsolving approach for solving complex optimization problems. First, crowsolving requires a large community of

active users. Second, it is necessary to disclose problem data to this community of users which is not possible in multiple application fields (e.g. optimization problems dealing with strategic or personal data). In addition, because the optimization is performed by users, the solving process may not be reliable in terms of quality of the solutions and can also be inconsistent from one instance to another. Finally, the solving process may take time, in the order of several days or weeks, before good solutions are obtained by users. For instance, in [Eiben et al., 2012] the authors analyze the results obtained after one and two weeks of collaborative solving. These different constraints make collective solving at the boundary of the scope of interactive optimization defined in Section 2. In particular, it is difficult to consider such an approach in the context of a decision-making process.

From an optimization perspective, in the A-team and the crowdsolving approaches, it is questionable to rely on users for performing the optimization per se (i.e. for generating and improving solutions manually). In fact, it seems reasonable to assume that a computational method would be able to explore much more solutions that a user with the same operators for editing solutions. In addition, these operators could be used to develop heuristic search strategies, such as metaheuristics, that are similar to user strategies. However, there are two interesting perspectives to manual optimization performed by users. First, the users can gain insight into the optimization problem by solving it manually. Thus, manual optimization can be useful for learning purpose. Second, it may be possible to analyze user strategies for designing or improving optimization procedures. This second perspective is explored in [Khatib et al., 2011] with an extension of the FoldIt tool were users can create and improve algorithms. In [Khatib et al., 2011], the authors mentioned that the algorithms that have been developed collaboratively by FoldIt users have comparable performances to state of the art optimization algorithms.

4.6.3 Interactive parameter tuning

In some situations, the user of an optimization-based decision support system is able to evaluate the behavior of a search process. For instance, he can determine if the search space has been properly explored or if it is necessary to continue the current optimization or not. In these cases, some parameters of the optimization procedure can be regulated by the user. More generally, instead of only using automated mechanisms to tune the optimization procedure, relying on human judgment to adjust some parameters can be an effective alternative. This interaction is exploited in some trial-and-error approaches for adjusting the values of parameters. However, a trial-and-error approach is rarely a viable option for parameter setting due to the complexity of the problem of tuning parameters [Adenso-Díaz and Laguna, 2006, Hutter et al., 2010b] and the fact that it requires some knowledge about the optimization procedure. An intermediate approach between fully automated procedures and trial-and-error for tuning parameters is proposed in [Hutter et al., 2010a]. The authors propose to combine classical regression techniques and human judgment in order to identify reasonable parameter settings. In the proposed interactive parameter tuning approach, the user can guide the process that builds and refines a response model of parameters. The response model is based on classical regression models which can be easily represented and interpreted. The interactive process relies on human judgment to identify important parameters and to adjust ranges for parameter values. Guiding the parameter tuning process is particularly useful when the computational time budget is reduced and only a small number of configurations can be evaluated. In addition, it can help the user to learn about influence and interaction of parameters.

4.6.4 Hyper-interactive evolutionary computation

Hyper-interactive Evolutionary Computation (HIEC) [Bush and Sayama, 2011] introduces an original way of guiding a search process which differs from the human-guided

search approach presented in Section 4.5. In HIEC, the user controls an evolutionary process by choosing when, and how, each operator is applied to the current population of solutions. The user can thus decide how the search space is explored. In addition, the user is also responsible for the selection process. He selects the solutions of each generation according to the perception he has of the quality of the solutions. This subjective evaluation of the solutions is similar to the evaluation process in interactive evolutionary algorithms. In sum, HIEC combines the principle of guiding an optimization algorithm with the interactive definition of the objective function. In [Bush and Sayama, 2011], the authors present different experiments with subjects in which HIEC is used as a teaching aid and a tool for creative design. The HIEC approach introduces an original interaction mechanism and an interesting approach for understanding artificial evolutionary processes. However, as an optimization tool the method appears to combine the shortcomings of both the human-guided search approach and interactive evolutionary algorithms.

4.6.5 Long-term preference inference

In [Meignan and Knust, 2013], the authors propose an interactive optimization method that addresses shortcomings of interactive reoptimization. As mentioned in Section 4.2, in interactive reoptimization, the local modifications applied by the user on a candidate solution are specific to a problem instance and cannot be reused. Therefore, when multiple problem instances are solved with the same inaccurate optimization model, the user has to intervene for each instance in order to adjust the solutions, which can become a laborious process. A second limit of interactive reoptimization is that the reoptimization can be suboptimal because it corrects locally a missing feature (constraint or objective) that could be expressed for the whole solution. The interactive method presented in [Meignan and Knust, 2013] aims at overcoming these two limitations by generalizing the preferences that are expressed by the user during an interactive reoptimization process. In the proposed method, the process of reoptimization remains the same as other interactive reoptimization methods. First, the user indicates the desired changes on a candidate solution. The candidate solution is then reoptimized, and the user can express additional changes until a satisfactory solution is found. However, the feedback accumulated during this iterative process is dynamically analyzed to identify possible patterns in the parts of the solutions changed by the user. These patterns are expressed as conjunctive rule sets that generalize the preferences of the user. When some relevant patterns are identified, the respective conjunctive rule sets are proposed to the user who has the possibility to add them as additional objectives. The main advantages of this approach are that inferred objectives can be reused for solving other problem instances and that inferred objective are expressed globally on solutions. In addition, the inferred objectives that are expressed as rules are easily understandable by the user. However, the method may suffer from typical weaknesses of classification rule learning methods such as the overfitting-avoidance-bias [Fürnkranz, 1999] which can be problematic if the user feedback is noisy.

5 Classification of interactive optimization methods

In this section, we propose a classification of interactive optimization methods which consists of two complementary views. The first one focuses on the user and groups interactive methods according to the user's contribution to the optimization process. This first part of the classification gives an overall picture of existing interactive optimization approaches which is completed by a second perspective that addresses the optimization system and characterizes its different components. We propose three entries for the second part of the classification, namely, the type of feedback integration, the lifetime of preference information, and the type of optimization procedure.

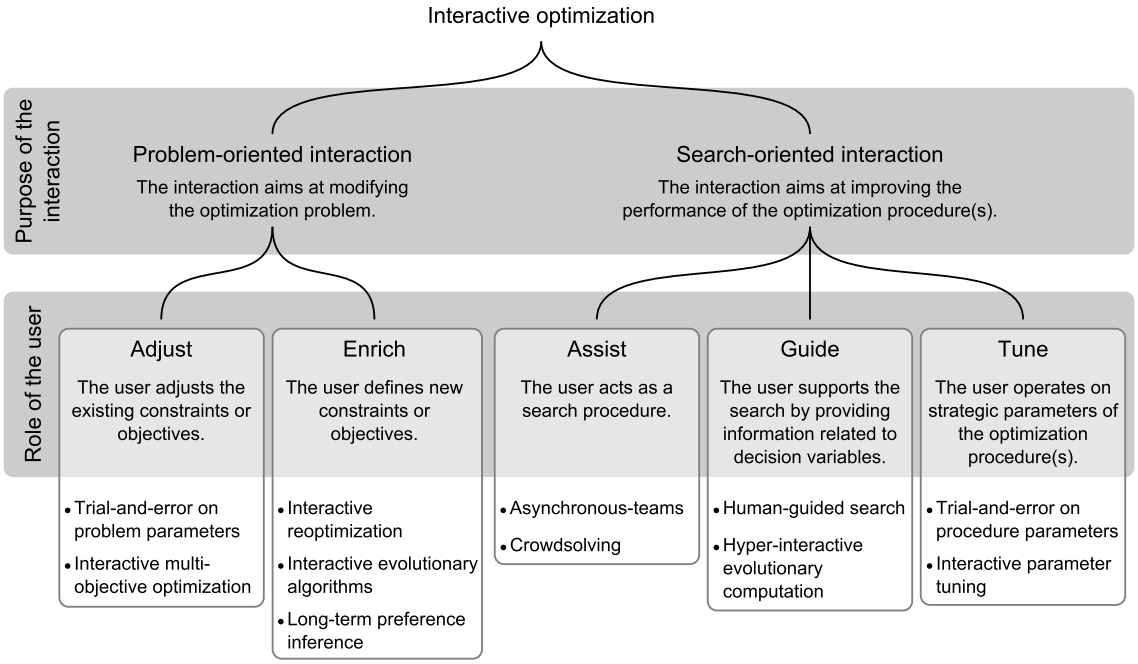

Figure 6: Classification of interactive optimization methods according to the purpose of the interaction and the role of the user in the optimization process.

The objective of this classification is, first, to introduce a common terminology for describing and analyzing interactive optimization methods. Second, the proposed entries of the classification aim at supporting the development of new interactive optimization methods by identifying the main design choices that have to be made for the development of an interactive optimization system. Finally, the characterization of interactive methods with the proposed classification may help to identify promising or unexplored interaction mechanisms.

5.1 Purpose of the interaction and role of the user

The first part of the classification concerns the contribution of the user in the optimization process. Two types of contribution were already identified through the review of interactive methods. First, the user can provide information for completing the optimization model. Second, the user can be involved in the optimization process for improving the efficiency of the optimization procedure. We introduced in Section 2 the terms *problem-oriented interaction* and *search-oriented interaction* for these two types of interaction. In the proposed classification, we refine this partition between problem-oriented and search-oriented interaction, and we identify five different roles that characterize the contribution of the user. For a problem-oriented interaction the user can either *adjust* or *enrich* the optimization problem. In a search-oriented interaction the possible roles of the user within the optimization process are to *assist*, *guide*, or *tune* the search procedures. A definition of each of these roles is given below.

Roles of the user in a problem-oriented interaction

A problem-oriented interaction primarily aims at modifying the optimization model in order to better fit the decision maker's problem. For this type of interaction, the user's preferences complete the definition of the optimization problem. This type of interaction assumes that the user has a valuable knowledge of the problem domain. For instance, the interaction in interactive reoptimization is problem-oriented. In interactive reoptimization, the user adds new constraints to the problem based on his knowledge of the real

optimization problem. These new constraints enrich the problem and make the solutions more suitable to the real context.

In a problem-oriented interaction, the user can play two different roles: *Adjuster* and *Enricher*.

- The user is an *adjuster* when he adjusts constraints or objectives. In this case, the type and extent of the modifications made by a user are defined during the design of the optimization system. When a problem instance is solved, it is assumed that the problem model is complete and the user knowledge is only required to adjust some parameters of constraints or objectives. In other words, it is assumed that each candidate solution is a possible alternative for the decision-maker's problem, and the interaction helps to identify the most satisfying solution. The user plays this role of adjusting the optimization problem, for instance, in interactive multiobjective optimization where the interaction allows the definition of a satisfactory trade-off among the attainable ones.

- The second possible role in a problem-oriented interaction is *Enricher*. The user plays this role when he modifies the initial definition of the optimization problem by adding or removing some constraints or objectives. In this case, it is assumed that the initial problem model may be incomplete, and a candidate solution may not be a valid alternative. The role of the user is therefore to enrich the optimization problem in order to obtain useful solutions. Interactive reoptimization belongs to this class of interactive methods for which the user enriches the optimization problem with new constraints.

Roles of the user in a search-oriented interaction

A search-oriented interaction aims at improving the efficiency of the optimization procedure. Contrary to a problem-oriented interaction, a search-oriented interaction does not modify the optimization model. The feedback provided by the user impacts the optimization procedure without changing the constraints and objectives of the optimization model. For instance, in the human-guided search approach, the mobilities defined by the user modify the exploration of the search space, but the constraints and objectives that apply on solutions remain unchanged. In human-guided search, the interaction allows the user to direct search efforts in order to obtain a good solution more quickly.

Generally, search-oriented interaction requires some knowledge about the optimization procedure. In human-guided search, for instance, the mobilities set by a user modify the behavior of a local-search procedure. In this case, the interaction is possible only if the user understands the process of local-search.

In the proposed classification, the user can play three different roles during a search-oriented interaction: *Assistant*, *Guide* and *Tuner*.

- The *Assistant* role is played when the user assists the search process by acting himself as a search procedure that generates, selects or modifies solutions. In the asynchronous-teams framework, users play this role and can directly modify the solutions managed by the team of agents.

- The *Guide* role is characterized by the fact that the user modifies the behavior of the search procedures and controls the exploration of the search space. In the human-guided search approach, for example, the user directs the search without, himself, acting as a search procedure (i.e. the user does not modify directly the solutions). However, the user still controls the actions of the optimization procedure and determines how the optimization procedure explores the search space.

- The last role for search-oriented interactions is the *Tuner* role played by the user when he tunes the optimization procedure. In this case, the user has no direct control on the operations of the optimization procedure, but the interaction allows the user to adjust some strategic parameters of the search procedure. The user plays this role, for instance, in the interactive parameter setting approach.

Summary and discussion

Figure 6 illustrates the first part of the classification which concerns user's contribution. At the top level, a distinction is made between problem-oriented and search-oriented interactions. Then, the second level presents the different roles a user can play during the interaction. This classification provides an overview of the different modes of interaction, putting aside the means by which the interactive optimization is achieved. This perspective is completed in the next section with a characterization of the main components involved in the interaction.

In the classification proposed in Figure 6, the purpose of the interaction (at the top level) may appear to be ambiguous for some interactive approaches. In fact, it is possible that an interaction results in both a modification of the optimization model and an improvement of the optimization efficiency. For instance, in interactive multiobjective optimization the user's feedback is intended to determine an adequate trade-off between objectives. This is a problem-oriented interaction. However, the preference information may also guide the search process toward an adequate solution. Therefore, it may improve the performance of the optimization procedure [Miettinen et al., 2008, Branke, 2008]. This additional outcome could be viewed as a search-oriented interaction. Similarly, for human-guided search, the definition of mobilities that primarily aims at guiding the search process may also enable the user to introduce new features in candidate solutions [Klau et al., 2010]. In this case too, the interaction impacts both the optimization procedure and the definition of the problem. However, most of the ambiguities between problem-oriented and search-oriented interaction can be solved by examining two aspects of interactive methods. First, the distinction between problem-oriented and search-oriented interaction is made on the main purpose of the interaction. In the proposed classification, presented in Figure 6, we analyzed the initial motivations of interactive methods and disregarded possible side-effects of the interaction. Second, the type of information in the preference model should clarify the purpose of the interaction. In a problem-oriented interaction, preference information concerns the problem model, such as trade-off information in interactive multiobjective optimization. For a search-oriented interaction, the preference information relates to the optimization procedure. For instance, in the human-guided search approach, the mobilities which form the preference model concern the optimization procedure, and in particular modify the behavior of the local search.

Concerning the role of the user, it should be noted that some implementations of interactive methods combine multiple types of interaction for which the user may have different roles. For instance, in the human-guided search approach, the initial role of the user is to guide the search process by assigning mobilities (i.e. he has the *Guide* role in the classification). However, some implementations of the human-guided search approach also require, or allow, the user to perturb candidate solutions in order to diversify the search. For this second interaction, the user explores the search space which is referred to as the *Assistant* role, in addition to his initial role of *Guide*. In the classification of interactive approaches, given in the lower part of Figure 6, we only considered the main interaction of each approach for determining the role of the user. However, for some implementations of these approaches, the user can play multiple roles in the optimization process. This is illustrated in the detailed taxonomy given in Figures 7 to 9. In this taxonomy, the gray cells specify the different roles a user can play. For some interactive methods, presented in this

taxonomy, multiple roles are assigned to the user which indicates that the implementation of the method provides multiple ways for the user to interact.

In conclusion, this first part of the classification provides basic guidelines for deciding which interactive approach could be adequate in a given context. The distinction between problem-oriented and search-oriented interactions clarifies the objective of the interaction. If the optimization model needs to be improved by the users of an optimization system, a problem-oriented interaction must be considered. If, on the other hand, the optimization procedure requires some adjustments when problem instances are solved, then a search-oriented interaction should be examined. The roles identified in the classification further clarify how users can support the optimization system. This objective of the classification to provide a methodological basis for the design and integration of interactive optimization methods is also pursued in the second part of the classification.

5.2 Characterization of interactive optimization systems

The first part of the classification presented in the previous section focuses on the user's contribution to the optimization process. This perspective is completed in this section by a characterization of the key elements of an interactive optimization system. To this end, three criteria are used to characterize an interactive optimization system, namely the type of feedback integration, the preference information lifetime, and the type of optimization procedure. These additional classification criteria describe the interaction process from a system perspective. This allows the identification of the differences and similarities between different implementations of an interactive approach. A description of the three additional classification criteria is provided below. For each criterion, we also briefly discuss the interest of the different classes.

A taxonomy of the different interactive optimization methods presented throughout the review is proposed in Figures 7 to 9. It should be noted that the table does not present an exhaustive list of existing implementations of interactive methods. Rather, it provides a representative list of interactive methods. In the table, the references are grouped by the corresponding interactive approach. The gray columns correspond to the first part of the classification that focuses on the user. Then, the next three groups of columns correspond to the criteria concerning the optimization system. Finally, the last two columns summarize, for each interactive method, the content of the user feedback and the elements that compose the preference model.

Type of feedback integration (model-free/model-based)

In this second part of the classification, the first element that is characterized is the integration of the user's feedback into the preference model. This integration can be done in two different ways. The user's feedback can be either directly used to compute preference information, or the feedback can be generalized through the preference model. We introduced the terms *model-free* and *model-based* integration for these two ways of integrating the feedback of the user. In a model-free approach, the values of the preference model are directly updated with the feedback of the user. In a model-based approach, a model of the feedback (or model of the user's preferences) is learned, and this model is used to derive the preference information. For instance, in most interactive methods for multiobjective optimization, the values provided by the user are directly used in the objective function. There is no generalization of the feedback. This corresponds to a model-free integration. As can be seen from the taxonomy, the majority of interactive optimization methods uses a model-free integration of the feedback. However, some interactive methods propose a model-based integration of the feedback. For instance, the Dominance-based Rough Set Approach (DRSA) [Greco et al., 2008] is an exception among interactive multiobjective optimization methods in which the feedback of the user is generalized. In this interactive

Reference	Interaction purpose and role of the user					Feedback integration		Preference info. lifetime			Type of optim. procedure			User feedback	Preference model
	Problem-oriented interaction		Search-oriented interaction												
	Adjust	Enrich	Assist	Guide	Tune	Model-free	Model-based	Step-based	Short-term	Long-term	Exact approach	Heuristic	Metaheuristic		
Trial-and-error on problem parameters															
[Braklow et al. 1992]	●					●			●			●		Problem parameter values	Problem parameters
[Cesta et al. 2003, Cesta et al. 2007]	●				●	●			●			●	●	Addition and suppression of constraints - Parameter values of the optimization procedure	Status (enabled or disabled) of constraints - Optimization procedure parameters
Interactive multiobjective optimization															
[Branke et al. 2001]	●					●			●				●	Minimum and maximum trade-off rates of the objectives	Constraints reducing the Pareto front
[Greco et al. 2008]	●						●		●		●			Selection of preferred solutions - Selection of decision rules	Constraints reducing the Pareto front (decision rules)
[Hakanen et al. 2011]	●					●		●					●	Classification of objective functions - Selection of most preferred solution - Aspiration levels - Upper bounds	Parameters of the objective function - Constraints reducing the Pareto front
[Jaszkiewicz and Slowinski 1999]	●					●			●		●			Reference points (aspiration and reservation points) - Parameter values for defining neighborhoods (indifference, preference and veto thresholds) - Selection of a neighboring solution	Parameters of the objective function (reference point of a achievement scalarizing function) - Parameters for neighborhood
[Laukkanen et al. 2012]	●					●		●			●			Classification of objective functions - Selection of most preferred solution - Aspiration levels - Upper bounds	Parameters of the objective function - Constraints reducing the Pareto front
[Miettinen 2007]	●					●		●					●	Classification of objective functions - Aspiration levels - Upper bounds	Parameters of the objective function - Constraints reducing the Pareto front
[Miettinen et al. 2010]	●					●		●			●			Ranking of objectives - Improvement factors	Parameters of the objective function
[Ruotsalainen et al. 2010]	●					●		●			●			Classification of objective functions - Selection of most preferred solution - Aspiration levels - Upper bounds	Parameters of the objective function - Constraints reducing the Pareto front
[Tveit et al. 2012]	●					●		●			●			Classification of objective functions - Aspiration levels - Upper bounds	Parameters of the objective function - Constraints reducing the Pareto front

Figure 7: Classification of representative examples of interactive optimization methods.

Reference	Interaction purpose and role of the user					Feedback integration		Preference info. lifetime			Type of optim. procedure			User feedback	Preference model
	Problem-oriented interaction		Search-oriented interaction												
	Adjust	Enrich	Assist	Guide	Tune	Model-free	Model-based	Step-based	Short-term	Long-term	Exact approach	Heuristic	Metaheuristic		
Interactive reoptimization															
[Hamel et al. 2012]		●				●			●		●			Desired values of decision variables	Constraint fixing the desired values of decision variable - Objective to minimize the deviation from the modified solution
[Meignan 2014]		●				●			●				●	Desired modifications of the current solution	Objective to maximize the satisfaction of desired modifications - Objective to minimize the deviation from the previous solution
[Pinedo 2012]		●				●			●			●		Modifications of the current solution	Constraint fixing the modifications
[van Vliet et al. 1992]		●				●			●			●		Modifications of the current solution	Constraint fixing the modifications
Interactive evolutionary algorithms															
[Babbar-Sebens and Minsker 2010]		●					●		●				●	Rating of solutions	Case-based memory for evaluating solutions
[Biles et al. 1996]		●					●		●				●	Rating of solutions	Artificial neural network for evaluating solutions
[Kim and Cho 2000]		●				●		●					●	Rating of solutions	Evaluation of solutions
[Lee and Cho 1999]		●				●		●					●	Rating of solutions	Evaluation of solutions
[Llora et al. 2005]		●					●		●				●	Selection of preferred solutions	Support vector machine for the selection of solutions
Long-term preference inference															
[Meignan and Knust 2013]		●				●	●		●	●			●	Desired modifications of the current solution - Selection of inferred constraints	Objective to maximize the satisfaction of desired modifications - Inferred objectives
Asynchronous-teams															
[Rachlin et al. 1999]			●			●		●					●	Modification, addition and suppression of solutions in the population	Solutions modified, added and removed from the population
Crowdsolving															
[Khatib et al. 2011]			●		●	●		●		●		●		Modification of the current solution, edition of heuristic strategies	Solutions modified, pool of heuristic strategies

Figure 8: Classification of representative examples of interactive optimization methods. (Continued)

Reference	Interaction purpose and role of the user					Feedback integration		Preference info. lifetime			Type of optim. procedure			User feedback	Preference model
	Problem-oriented interaction		Search-oriented interaction												
	Adjust	Enrich	Assist	Guide	Tune	Model-free	Model-based	Step-based	Short-term	Long-term	Exact approach	Heuristic	Metaheuristic		
Human-guided search															
[Anderson et al. 2000]			●	●		●		●					●	Assignment of mobilities - Perturbation of the current solution - Parameter values of the optimization procedure	Restrictions on local-moves - Starting solution of the local-search - Parameters of the optimization procedure
[Chimani et al. 2005]		●		●		●		●	●				●	Assignment of mobilities - Modification of the current solution	Restrictions on local-moves - Starting solution of the local-search - Objective to minimize the deviation from the previous solution
[Klau et al. 2002]			●	●		●		●					●	Assignment of mobilities - Modification of the current solution - Parameter values of the optimization procedure	Restrictions on local-moves - Starting solution of the local-search - Parameters of the optimization procedure
[Lesh et al. 2003]			●	●		●		●					●	Assignment of mobilities - Modification of the current solution - Parameter values of the optimization procedure	Restrictions on local-moves - Starting solution of the local-search - Parameters of the optimization procedure
[Meignan et al. 2011]		●		●		●		●	●				●	Selection of the part of the solution to optimize - Definition of mandatory and undesired elements	Restriction on perturbation moves - Constraint to add mandatory elements and avoid undesired ones
[Scott et al. 2002]			●	●		●		●					●	Assignment of mobilities - Perturbation of the current solution	Restrictions on local-moves - Starting solution of the local-search
Hyper-interactive evolutionary algorithm															
[Bush and Sayama 2011]		●		●		●		●					●	Selection of evolutionary operators - Selection of preferred solutions	Sequence of evolutionary operators - Selection of solutions
Trial-and-error on optimization procedure parameters															
[Halim and Lau 2007]					●	●				●			●	Addition of strategic rules for the optimization procedure	Strategic rules of the optimization procedure
Interactive parameter tuning															
[Hutter et al. 2010a]					●	●				●			●	Ranges for parameter values - Selection of the regression model	Parameters of the tuning procedure - Model selection

Figure 9: Classification of representative examples of interactive optimization methods. (Continued)

method, a set of decision rules is inferred from the feedback of the user and then used as preference model. Several interactive evolutionary algorithms also propose a model-based integration of the feedback using, for instance, case-based reasoning [Babbar-Sebens and Minsker, 2010], artificial neural networks [Biles et al., 1996] or support vector machines [Llorà et al., 2005].

A major interest of a model-based approach is the possibility to derive potentially complex preference information from a simple feedback of the user. An artificial neural network can, for instance, capture complex relations between different solutions using a simple feedback such as the selection or rejection of solutions. In contrast, it is generally more difficult for the user to express his preferences for model-free approaches. For instance, in interactive multiobjective optimization, most of the model-free approaches require the user to set preference values that are difficult to determine, such as trade-off rates [Branke et al., 2001], or values for reference points [Jaszkiewicz and Słowiński, 1999]. In short, the user's feedback is generally simpler for a model-based approach than for a model-free approach.

There are, however, several drawbacks to model-based approaches. From a design perspective, the development of a model-based approach requires additional efforts for developing the learning procedure. In addition, the learning procedure that generalizes the user's feedback may result in additional parameters that have to be adjusted. Finally, considering the interaction, a model-based approach is likely requiring more information from the user in order to learn an adequate model. In conclusion, a model-based approach is potentially easier to use than a model-free one. In turn, a model-based approach requires more effort in its design and may also necessitate more feedback information from the user.

Preference information lifetime (step-based/short-term/long-term)

The preference information lifetime is a characteristic of the preference model. It is the period of validity of the information contained in the preference model. We distinguish three types of preference information according to their lifetime, namely step-based, short-term (for a single problem instance) and long-term preference information (for multiple problem instances). A step-based preference information is specific to a part of the optimization process. The information may not be valid for the whole optimization process and it must be redefined at different stages. For instance, in the human-guided search approach, the mobilities that guide the search depend on the progress of the optimization process. The mobilities defined at the early stage of the optimization are likely to be inappropriate at the end of the optimization. Consequently, this preference information in human-guided search corresponds to a step-based information. A short-term preference information is potentially valid for the whole optimization process, but cannot be reused for solving another problem instance. For instance, in interactive reoptimization, the modifications applied to a candidate solution are potentially valid until the end of the optimization process. However, it generally makes no sense to reuse the same changes on another problem instance. In this case, the preference model consists of short-term preference information. Finally, a long-term preference information can possibly be reused for solving different problem instances. A simple example of such a preference information is given by the strategic parameters of an optimization procedure that are defined by interaction with the user. Generally, after having determined such a parameter, the value can be reused for solving multiple instances of the optimization problem.

The dynamics of the interaction between the user and the optimization system is strongly related to the period of validity of the preference information stored in the preference model. A short lifetime necessitates frequent updates of the preference information, and therefore, requires a regular feedback from the user. A longer lifetime potentially implies less interaction with the user when the right preference information has been de-

termined. Hence, it is interesting to exploit long-term preference information to reduce the burden of interaction on the user. This is especially true when different instances of an optimization problem have to be solved regularly.

In the taxonomy presented in Figures 7 to 9, very few methods use long-term preference information. In fact, the preference model of most of the considered methods relies on values that are specific to problem instances. For instance, in interactive multiobjective optimization, reference points or aspiration levels correspond to values that are significant only for the instance at hand. These values generally cannot be reused for another problem instance. As we mentioned in Section 4.3, for some trade-off based methods the preference information may have the potential to be reused but, to the best of our knowledge, this opportunity has not yet been studied. The development of interactive optimization methods that exploit long-term preference information represents, for us, an important perspective for the domain of interactive optimization. This point is further discussed in the conclusion of the article.

Type of optimization procedure (exact/heuristic/metaheuristic)

The type of optimization procedure is another aspect that characterizes an interactive optimization system. In the proposed classification, optimization procedures are classified as exact methods, heuristics or metaheuristics. Although it is possible to refine this classification of optimization procedures (e.g. [Blum and Roli, 2003] for metaheuristics), a finer partition would not be necessarily relevant for comparing the existing interactive methods.

The three types of optimization procedures can briefly be defined as follows. An exact optimization procedure is a procedure that guarantees to find an optimal solution. Examples of such exact approaches are the simplex method for linear programming and branch-and-bound algorithms. A heuristic is an approximate optimization procedure, usually dependent on the optimization problem, which generates a solution by applying a limited set of rules. Finally, a metaheuristic is a problem-independent heuristic search strategy that is based on the exploration of the solution space. In order to be applied on an optimization problem, a metaheuristic has to be specialized, for instance, by defining neighborhood structures that are specific to the problem. Metaheuristics include trajectory-based metaheuristics (such as tabu search), population-based metaheuristics (such as evolutionary algorithms), and model-based metaheuristics (such as estimation of distribution algorithms).

In the presented taxonomy, most of the implemented optimization procedures are non-exact algorithms (heuristics or metaheuristics). More precisely, only 7 out of the 32 analyzed methods are implemented with an exact solving procedure. The majority of the remaining interactive methods uses a metaheuristic. In fact, metaheuristics are particularly suitable for providing good solutions in a short time, which is often necessary in an interactive context. In addition, a key assumption in interactive optimization is that the optimization model may be inaccurate, or the performance of the optimization procedure may be inappropriate. This assumption justifies an interaction with the user. It also reduces the interest of computing an optimal solution. In this context, computing a satisfactory solution for the user can be more important than obtaining an optimal solution with respect to the optimization model. Therefore, the benefit of using metaheuristics to provide good solutions in reasonable time is particularly attractive for interactive optimization.

The classification according to the type of optimization procedure also reveals two groups of interactive methods. On the one hand, some interactive methods are independent of the type of optimization procedure, such as most interactive multiobjective or reoptimization methods. In this case, during the design of the optimization system the

main aspects that will guide the choice of an optimization procedure are the characteristics of the optimization model and the requirements in terms of quality and computation time. On the other hand, interactive methods such as interactive evolutionary algorithms or human-guided search, are specific to a type of optimization procedure. For these methods, the possible contribution of the user in the optimization process is decisive for the choice of an optimization procedure. Overall, this second group of interactive methods shows that the performance is not the only criterion for choosing an optimization procedure. Some optimization procedures can provide special opportunities to integrate user preferences.

User feedback and preference model

The two last columns of the taxonomy presented in Figures 7 to 9 correspond to the feedback information provided by the user and the elements which form the preference model. This information is given to illustrate the classification, and to recall, for each interactive method, the implementation that has been previously described in the review.

The feedback of the user is the data provided through the graphical interface of the optimization system. For instance, in an interactive evolutionary algorithm the feedback generally consists of the rating of solutions, or the selection of preferred solutions. The preference model, on the other hand, is the representation of the user's preference in the system. This preference model possibly generalizes the feedback of the user, in which case we referred the integration of the feedback to as model-based.

The last column of the taxonomy, describing the preference model of each method, is a good illustration of the difference between problem-oriented and search-oriented interaction. For problem-oriented interactions, the preference models are related to the optimization problem. The models correspond to objectives, constraints and parameters of the optimization model that are determined by the feedback of the user. For search-oriented interactions, the preference models have an impact on the optimization procedure. Also, these models generally correspond to parameters of the optimization procedure or heuristic information obtained through the interaction with the user.

The two last columns of the taxonomy also highlight the difference between model-free and model-based interactive methods. For model-free interactive methods, which represent the majority of the considered methods, the preference model simply contains, regroups, or combines the feedback provided by the user. In contrast, for model-based methods, the user feedback is the input data for learning the preference model.

6 Conclusion

In this work, we have presented a review of interactive optimization methods. The considered interactive methods are used during a decision process for solving optimization problems. Hence, the interaction occurs between an end-user and an optimization system and it allows the user to significantly modify the results or the performance of the optimization system. This review presents different alternatives of integrating a user in an optimization process and covers interactive methods that until now have been mostly studied separately in the research literature.

In addition, we proposed a classification of interactive optimization methods with two complementary points of view. The first part of the classification is user-oriented and identifies the different roles a user can play in an optimization process. The second part of the classification describes the interaction from a system perspective and characterizes the main elements of an interactive optimization system. This classification constitutes a first step toward a methodological approach for designing interactive optimization systems.

The review and classification revealed some challenges and research issues in the domain of interactive optimization. We discuss below some aspects that, in our opinion,

represent promising perspectives for interactive optimization.

The taxonomy of interactive approaches presented in Figures 7 to 9 shows that most of the reviewed approaches have been designed for short-term preferences. Reusing previous preference information for solving new problem instances is rarely considered. However, this aspect seems to be interesting for a wide range of optimization-based decision support tools. Timetabling, shift scheduling, and rostering are a few examples of optimization problems that could possibly need to be solved on a weekly or daily basis by the same decision-maker. For these applications, the ability of the system to learn the user's preferences over several problem instances could lead to a better efficiency of the interaction between the system and the user. Learning long-term preferences can allow the user to avoid having to repeatedly provide similar information on different problem instances. Instead, he can focus on features specific to each problem instance. However, the integration of long-term preferences may raise new issues. First, by solving multiple problem instances, the user gains experience and his understanding of the system evolves. This learning process of the user could cause inconsistencies in the user's feedback over time. In addition, long-term preference information has a larger impact than short-term preferences that are redefined for each new problem instance. Therefore, it is critical that the user is able to validate, understand, and adjust long-term preference information [Kulesza et al., 2009]. Finally, the integration of long-term preferences requires the development of appropriate computational methods that generalize the user's feedback.

Another interesting aspect concerns the development of more natural, flexible and intelligent interaction mechanisms. Mixed-initiative interaction and interface agents seem to be promising paradigms which can be used to improve interaction in the context of optimization-based decision support tools. This research direction is already investigated, for instance, in [Babbar-Sebens and Minsker, 2012] and [Chéné et al., 2014] in the context of interactive evolutionary algorithms and interactive reoptimization respectively. Mixed initiative refers to a flexible interaction strategy, where agents can negotiate in order to determine their role in the accomplishment of tasks, and collaborate to dynamically adjust their contributions [Allen, 1999]. In the context of interactive optimization, a mixed-initiative interaction implies that the user can be involved at different levels of the optimization process. The system should provide interaction mechanisms ranging from assisting a manual optimization, to an automated optimization process monitored by the user. In addition, with such a mixed-initiative interaction, the contribution of the user in the optimization task can change over time. Finally, the system should dynamically adapt the interaction with the user according to the problem at hand and the experience gained. More generally, we believe that the development of more natural and intuitive forms of interaction with optimization system is essential for the integration of advanced optimization methods in decision support tools. This requires taking into account user requirements, decision context, and human factors in the early stage of the development of an optimization system.

References

B. Adenso-Díaz and M. Laguna. Fine-tuning of algorithms using fractional experimental designs and local search. *Operations Research*, 54(1):99–114, 2006.

J. F. Allen. Mixed-initiative interaction. *Intelligent Systems and their Applications, IEEE*, 14(5):14–23, 1999. doi: 10.1109/5254.796083.

S. Amershi, M. Cakmak, W. B. Knox, and T. Kulesza. Power to the people: The role of humans in interactive machine learning. *AI Magazine*, 35(4):105–120, 2014.

D. Anderson, E. Anderson, N. Lesh, J. Marks, B. Mirtich, D. Ratajczak, and K. Ryall. Human-guided simple search. In *Seventeenth National Conference on Artificial Intelligence*, pages 209–216, 2000.

D. Arnott. Cognitive biases and decision support systems development: a design science approach. *Information Systems Journal*, 16(1):55–78, 2006. doi: 10.1111/j.1365-2575.2006.00208.x.

G. Ausiello, V. Bonifaci, and B. Escoffier. *Computability in Context: Computation and Logic in the Real World*, chapter Complexity and Approximation in Reoptimization, pages 101–129. Imperial College Press, 2007. ISBN: 978-1-84816-245-7.

M. Babbar-Sebens and B. Minsker. A case-based micro interactive genetic algorithm (cbmiga) for interactive learning and search: Methodology and application to groundwater monitoring design. *Environmental Modelling & Software*, 25:1176–1187, 2010. doi: 10.1016/j.envsoft.2010.03.027.

M. Babbar-Sebens and B. Minsker. Interactive genetic algorithm with mixed initiative interaction for multi-criteria ground water monitoring design. *Applied Soft Computing*, 12:182–195, 2012. doi: 10.1016/j.asoc.2011.08.054.

R. P. Bagozzi. The legacy of the technology acceptance model and a proposal for a paradigm shift. *Journal of the Association for Information Systems*, 8(4):244–254, 2008.

W. Banzhaf. *Handbook of Evolutionary Computation*, chapter Interactive Evolution, pages 1–6. IOP Publishing Ltd and Oxford University Press, 1997.

J.-P. Barthélemy, R. Bisdorff, and G. Coppin. Human centered processes and decision support systems. *European Journal of Operational Research*, 136(2):233–252, 2002. doi: 10.1016/S0377-2217(01)00112-6.

V. Belton, J. Branke, P. Eskelinen, S. Greco, J. Molina, F. Ruiz, and R. Słowiński. *Multiobjective Optimization*, chapter Interactive multiobjective optimization from a learning perspective, pages 405–433. Springer, 2008.

A. Ben-Tal, L. E. Ghaoui, and A. Nemirovski. *Robust Optimization*. Princeton Series in Applied Mathematics. Princeton University Press, 2009.

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. doi: 10.1007/BF01584098.

K. P. Bennett and E. Parrado-Hernández. The interplay of optimization and machine learning research. *Journal of Machine Learning Research*, 7:1265–1281, 2006.

H.-G. Beyer and B. Sendhoff. Robust optimization - a comprehensive survey. *Computer Methods in Applied Mechanics and Engineering*, 196(33–34):3190–3218, 2007. doi: 10.1016/j.cma.2007.03.003.

J. A. Biles, P. G. Anderson, and L. W. Loggi. Neural network fitness functions for a musical iga. In *Proceedings of the International Symposium on Intelligent Industrial Automation*, 1996.

M. Birattari. *The Problem of Tuning Metaheuristics*. PhD thesis, Université Libre de Bruxelles, 2005.

C. Blum and A. Roli. Metaheuristics in combinatorial optimization: Overview and conceptual comparison. *ACM Computing Surveys*, 35(3):268–308, 2003.

D. Bouyssou. *Readings in Multiple Criteria Decision-Aid*, chapter Building criteria: a prerequisite for MCDA, pages 58–80. Springer Berlin Heidelberg, 1990. doi: 10.1007/978-3-642-75935-2_4.

J. W. Braklow, W. W. Graham, S. M. Hassler, K. E. Peck, and W. B. Powell. Interactive optimization improves service and performance for yellow freight system. *Interfaces*, 22 (1):147–172, 1992. doi: 10.1287/inte.22.1.147.

J. Branke. *Multiobjective Optimization*, chapter Consideration of Partial User Preferences in Evolutionary Multiobjective Optimization, pages 157–178. Springer, 2008. doi: 10.1007/978-3-540-88908-3_6.

J. Branke, T. Kaußler, and H. Schmeck. Guidance in evolutionary multi-objective optimization. *Advances in Engineering Software*, 32(6):499–507, 2001. doi: 10.1016/S0965-9978(00)00110-1.

J. Branke, K. Deb, K. Miettinen, and R. Słowiński, editors. *Multiobjective Optimization, Interactive and Evolutionary Approaches*, volume 5252 of *Lecture Notes in Computer Science*. Springer, 2008. ISBN: 978-3-540-88907-6.

E. K. Burke, P. D. Causmaecker, G. V. Berghe, and H. V. Landeghem. The state of the art of nurse rostering. *Journal of Scheduling*, 7(6):441–499, 2004. doi: 10.1023/B:JOSH.0000046076.75950.0b.

E. K. Burke, M. Gendreau, M. Hyde, G. Kendall, G. Ochoa, E. Özcan, and R. Qu. Hyper-heuristics: a survey of the state of the art. *Journal of the Operational Research Society*, 64:1695–1724, 2013. doi: 10.1057/jors.2013.71.

B. J. Bush and H. Sayama. Hyperinteractive evolutionary computation. *IEEE Transactions on Evolutionary Computation*, 15(3):424–433, 2011. doi: 10.1109/TEVC.2010.2096539.

A. Cesta, G. Cortellessa, A. Oddi, and N. Policella. A csp-based interactive decision aid for space mission planning. In *Advances in Artificial Intelligence, 8th Congress of the Italian Association for Artificial Intelligence*, volume 2829 of *Lecture Notes in Computer Science*, pages 511–522, 2003. doi: 10.1007/978-3-540-39853-0_42.

A. Cesta, G. Cortellessa, S. Fratini, A. Oddi, M. Denis, A. Donati, N. Policella, E. Rabenau, and J. Schulster. Mexar2: AI solves mission planner problems. *IEEE Intelligent Systems*, 22(4):12–19, 2007. doi: 10.1109/MIS.2007.75.

F. Chéné, J. Gaudreault, and C.-G. Quimper. A mixed-initiative system for interactive tactical supply chain optimization. In *International Conference of Modeling and Simulation*, 2014.

M. Chimani, N. Lesh, M. Mitzenmacher, C. Sidner, and H. Tanaka. A case study in large-scale interactive optimization. In *International Conference on Artificial Intelligence and Applications*, 2005.

F. D. Davis and J. E. Kottemann. User perceptions of decision support effectiveness: Two production planning experiments. *Decision Sciences*, 25(1):57–76, 1994. doi: 10.1111/j.1540-5915.1994.tb00516.x.

F. D. Davis, R. P. Bagozzi, and P. R. Warshaw. User acceptance of computer technology: A comparison of two theoretical models. *Management Science*, 35(8):982–1003, 1989. doi: 10.1287/mnsc.35.8.982.

K. Deb and J. Sundar. Reference point based multi-objective optimization using evolutionary algorithms. In *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*, pages 635–642, 2006. doi: 10.1145/1143997.1144112.

M. Ehrgott and X. Gandibleux. A survey and annotated bibliography of multiobjective combinatorial optimization. *OR-Spektrum*, 22(4):425–460, 2000. doi: 10.1007/s002910000046.

C. B. Eiben, J. B. Siegel, J. B. Bale, S. Cooper, F. Khatib, B. W. Shen, Foldit Players, B. L. Stoddard, Z. Popovic, and D. Baker. Increased diels-alderase activity through backbone remodeling guided by foldit players. *Nature Biotechnology*, 30:190–192, 2012. doi: 10.1038/nbt.2109.

P. Eskelinen. Objective trade-off rate information in interactive multiobjective optimization methods: a survey of theory and applications. Technical Report 445, Helsinki School of Economics, 2008. ISBN: 9789524882200.

J. A. Fails and D. R. Olsen, Jr. Interactive machine learning. In *Proceedings of the 8th International Conference on Intelligent User Interfaces*, pages 39–45, 2003. doi: 10.1145/604045.604056.

M. L. Fisher. Interactive optimization. *Annals of Operations Research*, 5(3):539–556, 1985. doi: 10.1007/BF02023610.

G. A. Forgionne. *Decision Making Support Systems: Achievements and Challenges for the New Decade*, chapter An Architecture for the Integration of Decision Making Support Functionalities, pages 1–19. Idea Group Publishing, 2002. doi: 10.4018/978-1-59140-045-5.ch001.

M. C. Fu. Optimization for simulation: Theory vs. practice. *INFORMS Journal on Computing*, 14(3):192–215, 2002. doi: 10.1287/ijoc.14.3.192.113.

J. Fürnkranz. Separate-and-conquer rule learning. *Artificial Intelligence Review*, 13(1):3–54, 1999. doi: 10.1023/A:1006524209794.

M. Gendreau and C. D. Tarantilis. Solving large-scale vehicle routing problems with time windows: The state-of-the-art. Technical Report CIRRELT-2010-04, Interuniversity Research Center on Enterprise Networks, Logistics and Transportation, 2010.

S. Greco, B. Matarazzo, and R. Słowiński. *Multiobjective Optimization*, chapter Dominance-Based Rough Set Approach to Interactive Multiobjective Optimization, pages 121–155. Springer, 2008. doi: 10.1007/978-3-540-88908-3_5.

J. Hakanen, K. Miettinen, and K. Sahlstedt. Wastewater treatment: New insight provided by interactive multiobjective optimization. *Decision Support Systems*, 51(2):328–337, 2011. doi: 10.1016/j.dss.2010.11.026.

S. Halim and H. C. Lau. *Metaheuristics, Progress in Complex Systems Optimization*, volume 39 of *Operations Research/Computer Science Interfaces Series*, chapter Tuning Tabu Search Strategies Via Visual Diagnosis, pages 365–388. Springer US, 2007. doi: 10.1007/978-0-387-71921-4_19.

S. Hamel, J. Gaudreault, C.-G. Quimper, M. Bouchard, and P. Marier. Human-machine interaction for real-time linear optimization. In *IEEE International Conference on Systems, Man, and Cybernetics*, pages 673–680, 2012. doi: 10.1109/ICSMC.2012.6377804.

F. S. Hillier and G. J. Lieberman. *Introduction to operations research*. McGraw-Hill, seventh edition, 2001.

F. Hutter, T. Bartz-Beielstein, H. H. Hoos, K. Leyton-Brown, and K. P. Murphy. Sequential model-based parameter optimization: an experimental investigation of automated and interactive approaches. In T. Bartz-Beielstein, M. Chiarandini, L. Paquete, and M. Preuss, editors, *Empirical Methods for the Analysis of Optimization Algorithms*, chapter 15, pages 361–411. Springer, 2010a. doi: 10.1007/978-3-642-02538-9_15.

F. Hutter, H. H. Hoos, and K. Leyton-Brown. Automated configuration of mixed integer programming solvers. In *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*, volume 6140 of *Lecture Notes in Computer Science*, pages 186–202. Springer, 2010b. doi: 10.1007/978-3-642-13520-0_23.

A. Jaszkiewicz and J. Branke. *Multiobjective Optimization*, chapter Interactive multiobjective evolutionary algorithms, pages 179–193. Springer, 2008.

A. Jaszkiewicz and R. Słowiński. The 'light beam search' approach - an overview of methodology applications. *European Journal of Operational Research*, 113(2):300–314, 1999. doi: 10.1016/S0377-2217(98)00218-5.

C. V. Jones. Visualization and optimization. *INFORMS Journal on Computing*, 6(3): 221–257, 1994. doi: 10.1287/ijoc.6.3.221.

D. L. Kellogg and S. Walczak. Nurse scheduling: From academia to implementation or not? *Interfaces*, 37(4):355–369, 2007. doi: 10.1287/inte.1070.0291.

F. Khatib, S. Cooper, M. D. Tyka, K. Xu, I. Makedon, Z. Popović, D. Baker, and Foldit Players. Algorithm discovery by protein folding game players. *Proceedings of the National Academy of Sciences of the United States of America*, 108(47):18949–18953, 2011. doi: 10.1073/pnas.1115898108.

H.-S. Kim and S.-B. Cho. Application of interactive genetic algorithm to fashion design. *Engineering Applications of Artificial Intelligence*, 13(6):635–644, 2000. doi: 10.1016/S0952-1976(00)00045-2.

G. W. Klau, N. Lesh, J. Marks, and M. Mitzenmacher. Human-guided tabu search. In *Eighteenth National Conference on Artificial Intelligence*, pages 41–47. The AAAI Press, 2002.

G. W. Klau, N. Lesh, J. Marks, and M. Mitzenmacher. Human-guided search. *Journal of Heuristics*, 16(3):289–310, 2010. doi: 10.1007/s10732-009-9107-5.

A. J. Kleywegt and A. Shapiro. *Handbook of Industrial Engineering*, chapter Stochastic Optimization, pages 2625–2649. John Wiley & Sons, New York, 3rd edition, 2001. doi: 10.1002/9780470172339.ch102.

P. Krolak, W. Felts, and G. Marble. A man-machine approach toward solving the traveling salesman problem. *Communications of the ACM*, 14(5):327–334, 1971. doi: 10.1145/362588.362593.

T. Kulesza, W.-K. Wong, S. Stumpf, S. Perona, R. White, M. M. Burnett, I. Oberst, and A. J. Ko. Fixing the program my computer learned: Barriers for end users, challenges for the machine. In *Proceedings of the 14th International Conference on Intelligent User Interfaces*, pages 187–196, 2009. doi: 10.1145/1502650.1502678.

D. Kwak, A. Kam, D. Becerra, Q. Zhou, A. Hops, E. Zarour, A. Kam, L. Sarmenta, M. Blanchette, and J. Waldispühl. Open-phylo: a customizable crowd-computing platform for multiple sequence alignment. *Genome Biology*, 14(10), 2013. doi: 10.1186/gb-2013-14-10-r116.

G. Laporte. Fifty years of vehicle routing. *Transportation Science*, 43(4):408–416, 2009. doi: 10.1287/trsc.1090.0301.

T. Laukkanen, T.-M. Tveit, V. Ojalehto, K. Miettinen, and C.-J. Fogelholm. Bilevel heat exchanger network synthesis with an interactive multi-objective optimization method. *Applied Thermal Engineering*, 48:301–316, 2012. doi: 10.1016/j.applthermaleng.2012.04.058.

J. D. Lee and K. A. See. Trust in automation: Designing for appropriate reliance. *Human Factors*, 46(1):50–80, 2004. doi: 10.1518/hfes.46.1.50_30392.

J.-Y. Lee and S.-B. Cho. Sparse fitness evaluation for reducing user burden in interactive genetic algorithm. In *IEEE International Fuzzy Systems Conference Proceedings*, volume 2, pages 998–1003, 1999. doi: 10.1109/FUZZY.1999.793088.

P. Legris, J. Ingham, and P. Collerette. Why do people use information technology? a critical review of the technology acceptance model. *Information & Management*, 40(3):191–204, 2003. doi: 10.1016/S0378-7206(01)00143-4.

C. Liebchen, M. Lübbecke, R. Möhring, and S. Stiller. *Robust and Online Large-Scale Optimization*, volume 5868 of *Lecture Notes in Computer Science*, chapter The Concept of Recoverable Robustness, Linear Programming Recovery, and Railway Applications, pages 1–27. Springer-Verlag Berlin Heidelberg, 2009. doi: 10.1007/978-3-642-05465-5_1.

X. Llorà, K. Sastry, D. E. Goldberg, A. Gupta, and L. Lakshmi. Combating user fatigue in iGAs: partial ordering, support vector machines, and synthetic fitness. In *Proceedings of the 7th Annual Conference on Genetic and Evolutionary Computation*, pages 1363–1370, 2005. doi: 10.1145/1068009.1068228.

B. L. Maccarthy and J. Liu. Addressing the gap in scheduling research: a review of optimization and heuristic methods in production scheduling. *International Journal of Production Research*, 31(1):59–79, 1993. doi: 10.1080/00207549308956713.

B. McCollum. A perspective on bridging the gap between theory and practice in university timetabling. In E. K. Burke and H. Rudová, editors, *Practice and Theory of Automated Timetabling VI*, volume 3867 of *Lecture Notes in Computer Science*, pages 3–23. Springer-Verlag Berlin Heidelberg, 2006. doi: 10.1007/978-3-540-77345-0_1.

D. Meignan. A heuristic approach to schedule reoptimization in the context of interactive optimization. In *Proceedings of the 2014 Conference on Genetic and Evolutionary Computation*, pages 461–468. ACM, 2014. doi: 10.1145/2576768.2598213.

D. Meignan. An experimental investigation of reoptimization for shift scheduling. In *Proceedings of the 11th Metaheuristics International Conference*, 2015.

D. Meignan and S. Knust. Interactive optimization with long-term preferences inference on a shift scheduling problem. In A. Fink and J. Geiger, editors, *Proceedings of the 14th European Metaheuristics Workshop*, pages 1–6, 2013.

D. Meignan, J.-M. Frayret, and G. Pesant. An interactive heuristic approach for the P-forest problem. In *2011 IEEE International Conference on Systems Man and Cybernetics*, pages 1009–1013, 2011. doi: 10.1109/ICSMC.2011.6083801.

D. Meignan, J.-M. Frayret, G. Pesant, and M. Blouin. A heuristic approach to automated forest road location. *Canadian Journal of Forest Research*, 42(12):2130–2141, 2012. doi: 10.1139/x2012-140.

K. Miettinen. Using interactive multiobjective optimization in continuous casting of steel. *Materials and Manufacturing Processes*, 22(5):585–593, 2007. doi: 10.1080/10426910701322468.

K. Miettinen. Survey of methods to visualize alternatives in multiple criteria decision making problems. *OR Spectrum*, 36(1):3–37, 2014. doi: 10.1007/s00291-012-0297-0.

K. Miettinen and M. M. Mäkelä. Interactive multiobjective optimization system WWW-NIMBUS on the internet. *Computers & Operations Research*, 27(7–8):709–723, 2000. doi: 10.1016/S0305-0548(99)00115-X.

K. Miettinen, F. Ruiz, and A. P. Wierzbicki. *Multiobjective Optimization*, chapter Introduction to Multiobjective Optimization: Interactive Approaches, pages 27–58. Springer, 2008. doi: 10.1007/978-3-540-88908-3_2.

K. Miettinen, P. Eskelinen, F. Ruiz, and M. Luque. NAUTILUS method: An interactive technique in multiobjective optimization based on the nadir point. *European Journal of Operational Research*, 206(2):426–434, 2010. doi: 10.1016/j.ejor.2010.02.041.

B. M. Muir. Trust between humans and machines, and the design of decision aids. *International Journal of Man-Machine Studies*, 27(5–6):527–539, 1987. doi: 10.1016/S0020-7373(87)80013-5.

T. T. Nguyen, S. Yang, and J. Branke. Evolutionary dynamic optimization: A survey of the state of the art. *Swarm and Evolutionary Computation*, 6:1–24, 2012. doi: 10.1016/j.swevo.2012.05.001.

G. Phillips-Wren. Assisting human decision making with intelligent technologies. In *Knowledge-Based Intelligent Information and Engineering Systems*, volume 5177 of *Lecture Notes in Computer Science*, pages 1–10, 2008. doi: 10.1007/978-3-540-85563-7_1.

M. L. Pinedo. *Scheduling Theory, Algorithms, and Systems*, chapter Design and Implementation of Scheduling Systems: Basic Concepts, pages 459–483. Springer, fourth edition, 2012.

J. Rachlin, R. Goodwin, S. Murthy, R. Akkiraju, F. Wu, S. Kumaran, and R. Das. A-teams: An agent architecture for optimization and decision-support. In *Intelligent Agents V: Agents Theories, Architectures, and Languages*, volume 1555 of *Lecture Notes in Computer Science*, pages 261–276, 1999. doi: 10.1007/3-540-49057-4_17.

F. Rothlauf. *Design of Modern Heuristics*, chapter Optimization Problems, pages 7–44. Natural Computing Series. Springer, 2011. doi: 10.1007/978-3-540-72962-4_2.

B. Roy. Main sources of inaccurate determination, uncertainty and imprecision in decision models. *Mathematical and Computer Modelling*, 12(10–11):1245–1254, 1989. doi: 10.1016/0895-7177(89)90366-X.

B. Roy. Paradigms and challenges. In *Multiple Criteria Decision Analysis: State of the Art Surveys*, volume 78 of *International Series in Operations Research & Management Science*, pages 3–24. Springer New York, 2005. doi: 10.1007/0-387-23081-5_1.

H. Ruotsalainen, K. Miettinen, and J.-E. Palmgren. Interactive multiobjective optimization for 3D HDR brachytherapy applying IND-NIMBUS. In *New Developments in Multiple Objective and Goal Programming*, volume 638 of *Lecture Notes in Economics and Mathematical Systems*, pages 117–131. Springer Berlin Heidelberg, 2010. doi: 10.1007/978-3-642-10354-4_8.

M. W. P. Savelsbergh and M. Sol. The general pickup and delivery problem. *Transportation Science*, 29(1):17–29, 1995. doi: 10.1287/trsc.29.1.17.

M. Schrope. Solving tough problems with games. *Proceedings of the National Academy of Sciences of the United States of America*, 110(18):7104–7106, 2013. doi: 10.1073/pnas.1306643110.

S. D. Scott, N. Lesh, and G. W. Klau. Investigating human-computer optimization. In *Proceedings of the Conference on Human Factors in Computing Systems*, pages 155–162, 2002. doi: 10.1145/503376.503405.

J. Shim, M. Warkentin, J. F. Courtney, D. J. Power, R. Sharda, and C. Carlsson. Past, present, and future of decision support technology. *Decision Support Systems*, 33(2):111–126, 2002. doi: 10.1016/S0167-9236(01)00139-7.

W. S. Shin and A. Ravindran. Interactive multiple objective optimization: Survey I - continuous case. *Computers & Operations Research*, 18(1):97–114, 1991. doi: 10.1016/0305-0548(91)90046-T.

E. D. Smith, M. Piatelli-Palmarini, and T. Bahill. *Decision Modeling and Behavior in Complex and Uncertain Environments*, volume 21 of *Springer Optimization and Its Applications*, chapter Cognitive Biases Affect the Acceptance of Tradeoff Studies, pages 227–249. Springer New York, 2008. doi: 10.1007/978-0-387-77131-1_10.

S. Sra, S. Nowozin, and S. J. Wright. *Optimization for machine learning*, chapter Introduction: Optimization and Machine Learning, pages 1–17. MIT Press, 2012.

H. Takagi. Interactive evolutionary computation: fusion of the capabilities of EC optimization and human evaluation. *Proceedings of the IEEE*, 89(9):1275–1296, 2001. doi: 10.1109/5.949485.

S. Talukdar, L. Baerentzen, A. Gove, and P. De Souza. Asynchronous teams: Cooperation schemes for autonomous agents. *Journal of Heuristics*, 4(4):295–321, 1998. doi: 10.1023/A:1009669824615.

T.-M. Tveit, T. Laukkanen, V. Ojalehto, K. Miettinen, C.-J. F. T.-M. Tveit, T. Laukkanen, V. Ojalehto, K. Miettinen, and C.-J. Fogelholm. Interactive multi-objective optimisation of configurations for an oxyfuel power plant process for co2 capture. *Chemical Engineering Transactions*, 29:433–438, 2012. doi: 10.3303/CET1229073.

A. van Vliet, C. G. E. Boender, and A. H. G. Rinnooy Kan. Interactive optimization of bulk sugar deliveries. *Interfaces*, 22(3):4–14, 1992. doi: 10.1287/inte.22.3.4.

C. Vercellis. *Business Intelligence: Data Mining and Optimization for Decision Making*. Wiley, 2009.

C. Voudouris and E. P. K. Tsang. *Handbook of Metaheuristics*, chapter Guided local search, pages 185–218. Kluwer Academic, 2003.

J. Wallenius. Comparative evaluation of some interactive approaches to multicriterion optimization. *Management Science*, 21(12):1387–1396, 1975. doi: 10.1287/mnsc.21.12.1387.

J. Wessels and A. P. Wierzbicki. *Model-Based Decision Support Methodology with Environmental Applications*, volume 9 of *Mathematical Modelling: Theory and Applications*, chapter Model-Based Decision Support, pages 9–28. Springer, 2000.

A. Zych. *Reoptimization of NP-hard problems*. PhD thesis, Eidgenössische Technische Hochschule, ETH Zürich, 2012. Nr. 20257.