Chapter 22
Reactive Search Optimization: Learning While Optimizing. An Experiment in Interactive Multi-Objective Optimization

Roberto Battiti and Paolo Campigotto

Abstract Reactive Search Optimization advocates the integration of learning techniques into search heuristics for solving complex optimization problems. In this work, the source of learning signals is the decision maker, who is fine-tuning his preferences (formalized as a utility function) based on a learning process triggered by the presentation of tentative solutions. The objective function is not fully defined at the beginning and needs to be refined during the search for a satisfying solution. In practice, this lack of complete knowledge can occur for different reasons: Insufficient or costly knowledge elicitation, soft constraints which are in the mind of the decision maker, revision of preferences after becoming aware of some possible solutions, etc. As a concrete example, the classical case of multi-objective optimization problems (MOOPs) is considered. The focus is on solving MOOPs through interaction with the decision maker, where learning and optimization are deeply interconnected. After a short survey of previously proposed interactive multi-objective optimization methods, an experiment related to learning an appropriate linear combination of objectives is presented.

22.1 Introduction

As a rule of thumb, most of the problem-solving effort in the real world is spent on defining the problem, on specifying in a computable manner the function to be optimized. After this modeling work is completed, optimization becomes in certain cases a commodity. The implication for researchers and developers is that much more effort should be devoted to designing supporting techniques and tools to help the final user, often a decision maker (DM) without expertise in mathematics and in

Roberto Battiti, Paolo Campigotto
DISI - Dipartimento di Ingegneria e Scienza dell'Informazione, Università di Trento, Italy
e-mail: battiti@disi.unitn.it, campigotto@disi.unitn.it

optimization, to define and refine the function to be optimized which corresponds to his real objectives.

Reactive Search Optimization has focused in the last year onto online learning techniques to support the quest for a solution by self-adapting a local search method in a manner depending on the previous history of the search [2]. The word reactive hints at a ready response to events during the search through an internal online feedback loop for the self-tuning of critical parameters. The learning signals consisted of data about the structural characteristics of the instance collected while the algorithm is running. For example, data about sizes of basins of attraction, entrapment of trajectories, repetitions of previously visited configurations. In this context, the algorithm learns by interacting from a previously unknown environment given by an existing (and fixed) problem definition.

This contribution argues that there is a second interesting online learning loop, where learning signals originate from a DM and are intended to *modify and refine the problem definition* itself. This context is potentially very wide, depending on the amount of knowledge about the problem given *a priori*, on the allowed modifications, on the form of the questions posed to the DM. A more complete review of past approaches in this area is out of the scope of this paper. In this short contribution we consider a classic case given by multi-objective optimization problems (MOOPs). The DM is capable of specifying a set of desirable objectives, but the relative importance of the different objectives, the tradeoffs, the proper combination of them into an overall utility function are not made explicit. A MOOP can be stated as:

$$\begin{aligned} &\text{minimize} \quad \mathbf{f}(\mathbf{x}) = \{f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})\} \\ &\text{subject to} \qquad\quad \mathbf{x} \in \Omega \end{aligned}$$

where $\mathbf{x} \in \mathbb{R}^n$ is a vector of n decision variables; $\Omega \subset \mathbb{R}^n$ is the *feasible region* and is typically specified as a set of constraints on the decision variables; $\mathbf{f} : \Omega \to \mathbb{R}^m$ is made of m objective functions which need to be jointly minimized. Objective vectors are images of decision vectors and can be written as $\mathbf{z} = \mathbf{f}(\mathbf{x}) = \{f_1(\mathbf{x}), \ldots, f_m(\mathbf{x})\}$. The above problem is ill-posed whenever objective functions are conflicting, a situation that typically occurs in real-world applications. In these cases, an objective vector is considered optimal if none of its components can be improved without worsening at least one of the others. An objective vector $\mathbf{z}$ is said to *dominate* $\mathbf{z}'$, denoted as $\mathbf{z} \succ \mathbf{z}'$,if $z_k \leq z'_k$ for all k and there exists at least one h such that $z_h < z'_h$. An alternative $\hat{\mathbf{x}}$ is Pareto optimal if there is no other alternative $\mathbf{x} \in \Omega$ such that $\mathbf{f}(\mathbf{x})$ dominates $\mathbf{f}(\hat{\mathbf{x}})$. The set of Pareto optimal alternatives is called *Pareto set* (PS). The corresponding set of Pareto optimal objective vectors is called *Pareto front* (PF).

Efficient meta-heuristics have been developed for MOOPs, see, e.g., [9] and contained references. As mentioned, asking a user to quantify his utility function *a priori*, before seeing the actual optimization results, is in some cases extremely difficult. If the DM is cooperating with an optimization expert, misunderstanding between the persons may arise. In certain cases, the optimization expert's role can be played by a suitable software tool. This problem can become dramatic when the number of the conflicting objectives in MOOP increases. As a consequence, the DM

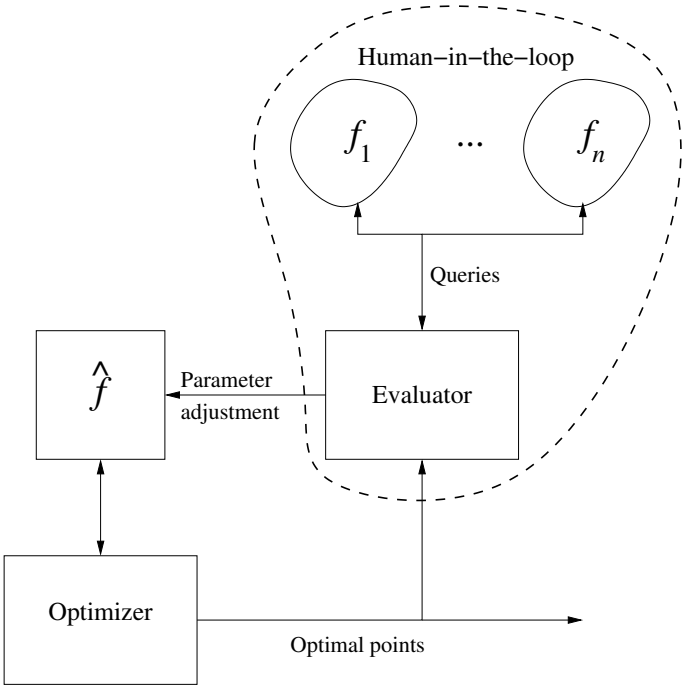

Fig. 22.1: Learning the problem definition from the DM: The case of interactive multi-objective optimization

may be dissatisfied with the solutions presented by the optimization expert. The optimization problem may be solved in an unsatisfactory manner because some of the objectives remain hidden in the mind of the DM. Different drawbacks are present in *a posteriori* approaches, which deliver to the DM a representative set of solutions on the entire Pareto front so that the user can pick his most preferred solution.

Although providing explicit weights and mathematical formulas can be difficult for the DM, for sure he can evaluate the returned solutions. In most cases, the strategy to overcome this situation consists of a joint interactive work between the DM and the optimization expert to change the definition of the problem itself. The optimization tool will then be re-executed over the new version of the problem. This process may be iterated an arbitrary number of times, see Figure 22.1.

The interactive decision making (IDM) approach exploits the user knowledge during the optimization task to guide the search towards better solutions, see [8] for a recent review.

This work proposes a variation of an existing technique to search for an appropriate weighting of the objectives based on user preferences that are elicited during the quest for a satisfactory solution. The proposal aims at improving the robustness

when the constraints derived from the user preferences lead to an infeasible linear problem.

Each run of the method provides a set of non-dominated solutions. The evaluation by the DM is holistic: The user compares complete solutions and picks his favorite one among couples of non-dominated solutions presented to him. The experimental part of this work shows the effectiveness of the proposed technique on some benchmark problems: A small number of questions asked to the DM is sufficient to obtain a satisfactory solution.

This paper is organized as follows. In the next section some representative methods for searching the weight space by pairwise comparison of solutions are summarized. In Section 22.3 our method is presented, followed by the test results to evaluate its performance in Section 22.4. Section 22.5 concludes this chapter.

22.2 Interactive MOOP Solutions by Learning a Weighted Aggregation of Objectives

Modeling real world issues often generates multi-objective optimization problems with a large or infinite number of Pareto optimal solutions. Even when the Pareto-optimal set is finite, the computational effort to obtain the complete PF may be prohibitive. Furthermore, the DM often cannot tackle the overflow of information generated when analyzing the entire PS. In this scenario, the *interactive* decision making (IDM) technique comes to the rescue. It assumes that the optimization expert (or the optimization software) cooperates with the DM. Through the interaction, the search process can be directed towards his favorite Pareto-optimal solutions. In this way, only a part of the PF needs to be generated. At each iteration, the DM can specify his preferences among the solutions provided by the optimization expert. Preferences are expressed in our case by making pairwise comparisons of solutions at each iteration.

IDM techniques typically convert the multi-objective formulation of the problem into a single-objective formulation aiming at modeling the user's utility function. This process is often referred as "scalarization," Under mild conditions, a Pareto-optimal solution of MOOP can be an optimal solution of a scalar optimization problem in which the single objective is an aggregation of the objectives, see, e.g., [12]. The problem of generating the PF is therefore decomposed into a number of scalar objective optimization sub-problems. Weights-based IDM techniques build an aggregation function by weighting the individual objectives with m weights. A single-objective optimization run can then be called to deliver optimal solutions of the scalar problem, or suitable approximations thereof. Therefore, the preference of the DM can be encoded by tuning the weights. Constraints on the weights are derived from the partial preferences, and the admissible region in weight space is progressively reduced until a satisfying solution is generated.

A seminal paper for the weight space reduction approach is [13], introducing a technique for multi-objective linear programming problems with an unknown under-

lying linear value function. In [14] this technique is generalized to handle pseudo-concave value functions. The pairwise comparison of solutions is carried out in two ways: Comparison of the values of the objectives of the solutions and comparison of the tradeoffs of the objectives of the solutions. In particular, the DM is required to compare pairwise adjacent optimal extreme solutions on the vertices of the polyhedron in the objective space to a trial optimal solution, so long as an adjacent optimal extreme solution is sufficiently distinct from the current trial solution and the comparison is not trivial. The preference information provided by the DM is translated into linear constraints for the weights. Then, a feasible solution for the set of constraints in the weight space is obtained by using linear programming. The novel linear combination of the objectives is then optimized to obtain a new trial solution. The process terminates when the weight space has been reduced enough to identify a final solution or all the tradeoffs for the current trial solution are undesirable.

Another example of the weight space reduction approach is [4]. The characteristics of the algorithm are the pairwise comparison of solutions required to the DM, the translation of the preference expressed by the DM into linear constraints in the weight space, the generation of a weight vector consistent with the constraints created so far and the generation of a trial optimal solution optimizing the aggregation function. A Tchebycheff function is used to locally approximate the value function of the DM (see [11] for details). As a consequence, the constraints on the weights generate multiple disjoints convex regions in the weight space, while in [14] a single consistent region is obtained. The process terminates when the weight space has been reduced enough that two significantly different solutions cannot be generated or when a specified maximum number of questions has been asked to the DM.

While in [14] and [4] the search in the weight space is organized in the form of weight cuts, in [11] the search is performed by "zooming" in the weight space. Randomly generated feasible weight vectors are used to identify a representative set of optimal solutions. In particular, the initialization phase of the algorithm randomly generates $2P$ weight vectors. At each subsequent iteration, for each weight vector, the corresponding aggregation function is solved to obtain $2P$ optimal solutions, which are then filtered to obtain a set of P maximally dispersed solutions. The DM is then required to select his most preferred solution among the set of solutions. If the DM is satisfied, the algorithm stops. Otherwise, $2P$ weight vectors are selected in a neighborhood of the weight vector corresponding to the most preferred solution and a new iteration of the algorithm is performed. In this way, the weight space is contracted around the weights that produced the most preferred solution and the search process focuses onto a more concentrated area of the objective space corresponding to the DM preferences.

The method described in this paper shares with the approaches [14], [4] and [11] the interaction with the DM realized via pairwise comparison of solutions. However, our approach tackles a broader class of multi-objective optimization problems: MOOP problems that are not necessarily linear but with convex Pareto fronts, so that a weighted sum is always guaranteed to identify all Pareto-optimal objective vectors, for appropriate values of the weights. The preference of the DM is locally approximated by a linear function. When the linear approximation of the value func-

tion generates infeasible problems in the weight space, the constraints are relaxed by introducing penalty-violation variables whose values are minimized.

22.3 Learning User Preferences in MOOP: The Algorithm

Our goal consists of learning the non-dominated solution preferred by the DM. The original MOOP is scalarized by means of the *weighted sum* method (see, e.g., [7]). In particular, we assume that the user provides the different objectives in the MOOP, but he cannot quantify the weights of the different objectives before seeing the actual optimization results.

Our system learns the weight vector $\mathbf{w} = (w_1, w_2, \ldots, w_m)$, with $\sum_{i=1}^{m} w_i = 1$, by optimizing the linear combinations g of the objectives:

$$g(\mathbf{x}, \mathbf{w}) = w_1 f_1(\mathbf{x}) + w_2 f_2(\mathbf{x}) + \cdots + w_m f_m(\mathbf{x})$$

In a more compact form:

$$g(x_1, x_2, \ldots, x_n, \mathbf{w}) = \mathbf{f}(\mathbf{x})^T \cdot \mathbf{w}$$

Without loss of generality, we assume that g must be minimized. Furthermore, we assume that the DM indicates his favorite solution among the couple of solutions presented at each iteration. The preferences are translated into constraints that the weights must satisfy. Assume $\mathbf{a} = (a_1, a_2, \ldots, a_n)$ and $\mathbf{b} = (b_1, b_2, \ldots, b_n)$ the two solutions provided by the system. The preference $\mathbf{a} \succ \mathbf{b}$ of the DM for the solution $\mathbf{a}$ w.r.t. the solution $\mathbf{b}$ is represented by the following constraint:

$$g(\mathbf{a}, \mathbf{w}) < g(\mathbf{b}, \mathbf{w})$$

Our approach, inspired by the work in [1], consists of finding a solution $\mathbf{w}$ for the set of constraints on the weights generated by interacting with the DM.

All the weights are initialized with random values in the interval (0,1), and then normalized so that their sum is 1. At each iteration, two non-dominated solutions $\mathbf{a}$ and $\mathbf{b}$ are compared by the DM. Both solutions are obtained by minimizing a linear combination of the objective functions of the input problem:

$\text{minimize}_{\mathbf{x}}\ g(\mathbf{x}, \mathbf{w})$
subject to
$\quad \mathbf{x} \in \Omega$

In particular, the first solution $\mathbf{a}$ is obtained by using the current weight vector $\mathbf{w}^{curr}$ found by applying the *middlemost weights* technique [10], solving the following linear programming problem:

$\text{maximize}_{\mathbf{w}}\ \gamma$
subject to
$\quad \sum_{i=1}^{m} w_i = 1$

$$g(\mathbf{a},\mathbf{w}) \leq g(\mathbf{b},\mathbf{w}) - \gamma, \forall\ \mathbf{a} \succ \mathbf{b}$$
$$w_i \geq \gamma,\ i = 1,\ldots,m$$
$$\gamma \geq 0$$

The motivation for the above technique is to increase the robustness of the weight vector by keeping it as far as possible from all constraints.

The second solution $\mathbf{b}$ is obtained by using the weight vector $\mathbf{w}^{pert}$, generated by perturbing $\mathbf{w}^{curr}$. In particular, $\mathbf{w}^{pert}$ is the feasible point in the weight space maximizing the distance from the constraints and from the middlemost point (i.e., $\mathbf{w}^{curr}$). The point $\mathbf{w}^{pert}$ is obtained by solving the following non-linear programming problem:

$$\text{maximize}_{\mathbf{w}}\ \gamma + d(\mathbf{w},\mathbf{w}^{curr})$$
subject to
$$\sum_{i=1}^{m} w_i = 1$$
$$g(\mathbf{a},\mathbf{w}) \leq g(\mathbf{b},\mathbf{w}) - \gamma, \forall\ \mathbf{a} \succ \mathbf{b}$$
$$w_i \geq \gamma,\ i = 1,\ldots,m$$
$$d(\mathbf{w},\mathbf{w}^{curr}) \geq d^{min}$$
$$\gamma \geq 0$$

where $d(\cdot,\cdot)$ is the Euclidean distance function and the constraint

$$d(\mathbf{w},\mathbf{w}^{curr}) \geq d^{min}$$

ensures that the minimum distance d^{min} value is not violated. This constraint defines the diversity of the solutions $\mathbf{a}$ and $\mathbf{b}$, avoiding the embarrassment of the DM when he is required to compare too similar solutions.

An infeasible set of linear constraints on the weights may be generated when the DM's preference information cannot be represented by a linear combination of the different objectives, i.e., a solution w that satisfies all constraints simultaneously may not exist. As a consequence, the linear programming problem generated by the middlemost weights technique and by the perturbation strategy is infeasible. In this case, a linear local approximation of the preference information is performed, by introducing a vector $\varepsilon \in \mathbb{R}^k$ of slack variables (k is the number of constraints on the weights generated), whose norm is then minimized. This leads to relaxed constraints of the form:

$$g(\mathbf{a}_j,\mathbf{w}) < g(\mathbf{b}_j,\mathbf{w}) + \varepsilon_j,\ j = 1,\ldots,k$$

where $\mathbf{a}_j$ and $\mathbf{b}_j$ are two solutions provided by the system. Taking into account the definition of g, the above equation can be rewritten as:

$$(\mathbf{f}(\mathbf{a}_j) - \mathbf{f}(\mathbf{b}_j))^T \cdot \mathbf{w} - \varepsilon_j < 0,\ j = 1,\ldots,k$$

The set of k constraints on the weights can be represented by a compact matrix form:

$$M_C \cdot \mathbf{w} - \varepsilon < 0$$

where each row of the $k \times m$ matrix M_C is $(\mathbf{f}(\mathbf{a}_j) - \mathbf{f}(\mathbf{b}_j))^T$, $j = 1, \ldots, k$. To update the weight vector $\mathbf{w}^{curr}$, the following quadratic programming problem is solved:

$$\begin{aligned}
&\text{minimize}_{\mathbf{w}} \parallel \varepsilon \parallel^2 \\
&\text{subject to} \\
&\quad M_C \cdot \mathbf{w} - \varepsilon < 0 \\
&\quad \varepsilon \in \mathbb{R}^k
\end{aligned}$$

where $\parallel \varepsilon \parallel$ is the L2-norm of the vector ε.

In the case of the perturbation strategy, the *relaxed* formulation consists of the following programming problem:

$$\begin{aligned}
&\text{minimize}_{\mathbf{w}} \parallel \varepsilon \parallel^2 - d(\mathbf{w}, \mathbf{w}^{curr}) \\
&\text{subject to} \\
&\quad M_C \cdot \mathbf{w} - \varepsilon < 0 \\
&\quad d(\mathbf{w}, \mathbf{w}^{curr}) \geq d^{min} \\
&\quad \varepsilon \in \mathbb{R}^k
\end{aligned}$$

The termination condition of our algorithm is given by a limit on the number of iterations performed. The pseudocode for our approach is in Figure 22.2.

22.4 Experimental Results

The experiments performed involve the formalization of the DM preference information into a value function. In particular, the linear value function:

$$\mathrm{VF}_l = \sum_{i=1}^{m} \lambda_i f_i \tag{22.1}$$

the quadratic value function:

$$\mathrm{VF}_q = K - (\sum_{i=1}^{m} (\lambda_i (f_i^{id} - f_i))^2)^{1/2} \tag{22.2}$$

the L_4-metric value function:

$$\mathrm{VF}_{l4} = K - (\sum_{i=1}^{m} (\lambda_i (f_i^{id} - f_i))^4)^{1/4} \tag{22.3}$$

and the Tchebycheff value function:

$$\mathrm{VF}_t = K - \max_{1 \leq i \leq m} (\lambda_i |f_i^{id} - f_i|) \tag{22.4}$$

are used, where $\mathbf{f}^{id}$ is the *ideal* objective vector, $\mathbf{f}$ is the objective vector of the current solution, K is a positive number ensuring that the corresponding values of

```
1.  procedure learningWeights
2.      input: {f_1, f_2, ..., f_m}
3.      output: w
4.
5.      initialize randomly w^curr
6.      initialize the set constraints on the weights to the empty set
7.      do
8.                  /* perturbation */
9.          generate w^pert
10.                 /* generate the new constraint */
11.         generate a couple of solutions (a, b)
12.         using weight vectors w^curr and w^pert
13.         by solving:
14.             minimize_x g(x, w)
15.             subject to
16.                 x ∈ Ω
17.
18.         generate a new constraint on the weights based on
19.         the user evaluation of the couple of solutions
20.         add the new constraint to the set of constraints
21.                 /* update the weights */
22.         obtain new w^curr by solving:
23.             maximize_w γ
24.             subject to
25.                 Σ_{i=1}^{m} w_i = 1
26.                 g(a, w) ≤ g(b, w) − γ, ∀ a ≻ b
27.                 w_i ≥ γ, i = 1, ..., m
28.                 γ ≥ 0
29.         if the problem above is infeasible, obtain new w^curr by solving:
30.             minimize_w ‖ ε ‖^2
31.             subject to
32.                 M_C · w − ε < 0
33.                 ε ∈ R^k
34.     until TERMINATION_CONDITION is met
35.     return w
```

Fig. 22.2: Outline of the algorithm learning the weight vector by interacting with the DM

VF_q, VF_{l4} and VF_t are positive, and $\lambda \in \mathbb{R}^m$ is a weight vector with $\lambda_i > 0$ and $\sum_{i=1}^{m} \lambda_i = 1$. The *ideal* vector in the objective space is obtained by minimizing each of the objective functions independently subject to the constraints set Ω.

The experimental results show the quality of the solution generated by our algorithm. Let $\mathbf{f}^{fin}$ be the objective vector for the final Pareto-optimal alternative $\mathbf{x}^{fin}$ discovered by the algorithm during a single run. Let $\mathbf{f}^{worst}$ the objective vector for the worst alternative $\mathbf{x}^{worst}$, i.e., the Pareto optimal solution with lowest value function value. Similarly, $\mathbf{f}^{best}$ is the objective vector for the best alternative $\mathbf{x}^{best}$, i.e., the Pareto optimal solution with highest value function value. The values $\mathbf{f}^{worst}$ and $\mathbf{f}^{best}$ are identified by exhaustive search over the Pareto frontier.

The quality of the final solution is measured by the scaled improvement percentage with respect to the worst case:

$$\mathrm{QM}(\mathbf{f}^{fin}) = \frac{\mathrm{VF}(\mathbf{f}^{fin}) - \mathrm{VF}(\mathbf{f}^{worst})}{\mathrm{VF}(\mathbf{f}^{best}) - \mathrm{VF}(\mathbf{f}^{worst})} \cdot 100 \tag{22.5}$$

The performance of our algorithm is successful if a high value for $\mathrm{QM}(\mathbf{f}^{fin})$ is observed within few interactions with the DM. Considering the pairwise comparison asked to the DM as a single question, the number of questions asked is equal to the number of iterations of our algorithm. The algorithm is implemented by using the Matlab 7.6 framework and the optimization functions provided by the Matlab Optimization Toolbox. The benchmark selected consists of 2-objective and 3-objective optimization problems with convex PF.

For the perturbation strategy, the minimum distance value d^{min} between $\mathbf{w}^{pert}$ and $\mathbf{w}^{curr}$ is set to 0.05 and to 0.15 in the case of the 2-objective and the 3-objective optimization problems considered, respectively.

The 2-objective benchmark problems are the *Binh* [3] problem:

minimize
$$f_1(x_1,x_2) = (x_1)^2 + (x_2)^2$$
$$f_2(x_1,x_2) = (x_1-5)^2 + (x_2-5)^2$$
subject to
$$-5 \leq x_1, x_2 \leq 10$$

and the *Generalized Schaffer problem* [5]:

minimize
$$f_1(\mathbf{x}) = \frac{2}{\phi}\left(\sum_{i=1}^{n}(x_i)^2\right)^{\frac{\phi}{2}}$$
$$f_2(\mathbf{x}) = \frac{2}{\phi}\left(\sum_{i=1}^{n}(1-x_i)^2\right)^{\frac{\phi}{2}}$$
subject to
$$0 \leq x_i \leq 10, \forall\, i = 1,\ldots,n$$

where n represents the number of decision variables. The curvature of the PF is defined by means of the parameter ϕ: $\phi < 1$ results in concave PFs, $\phi = 1$ in a linear PF and $\phi > 1$ in convex PFs. For our experiments, different convex Pareto fronts are generated.

Tables 22.1 and 22.2 show the results obtained by our algorithm over the *Binh* and the *Schaffer* problems, respectively. The values in the tables are the mean and the standard deviation values of the quality measure given by Eq. (22.5) obtained over 50 runs of the algorithm. Different values for the weight vector λ of the value function are considered.

Consider Table 22.1. For all the value functions considered, after 10 iterations of our algorithm the value of $\mathrm{QM}(\mathbf{f}^{fin})$ is greater than 99.5% of the value of $\mathrm{VF}(\mathbf{f}^{best})$ and the standard deviation of the distribution of the values from which $\mathrm{QM}(\mathbf{f}^{fin})$ is obtained is smaller than 1%. Table 22.2 shows that after 10 iterations of our algorithm, the divergence of the quality of the solution is less than 0.2% from the quality of the best solution over the Pareto front.

Table 22.1: Performance of the algorithm over the *Binh* problem

	linear VF		*quadratic VF*		L_4*-metric VF*		*Tchebycheff VF*	
iteration	mean	std	mean	std	mean	std	mean	std
10	99.886	0.255	99.935	0.166	99.718	0.974	99.550	0.447
50	99.972	0.058	99.971	0.046	99.918	0.305	99.563	0.453

Table 22.2: Performance of the algorithm over the *Schaffer* problem

	linear VF		*quadratic VF*		L_4*-metric VF*		*Tchebycheff VF*	
iteration	mean	std	mean	std	mean	std	mean	std
10	99.971	0.090	99.976	0.077	99.821	0.698	99.916	0.090
50	99.977	0.079	99.981	0.070	99.855	0.650	99.920	0.086

Table 22.3 contains the results of our algorithm over the following three-objectives optimization problem, taken from [6]:

minimize
$$f_1(x_1,x_2) = (x_1)^2 + (x_2-1)^2$$
$$f_2(x_1,x_2) = (x_1)^2 + (x_2+1)^2 + 1$$
$$f_3(x_1,x_2) = (x_1-1)^2 + x_2^2 + 2$$
subject to
$$-2 \leq x_1,x_2 \leq 2$$

while Table 22.4 shows the performance of our algorithm on the following three-objectives optimization problem:

minimize
$$f_1(x_1,x_2,x_3) = (x_1-1)^2 + (x_2)^2 + (x_3)^2$$
$$f_2(x_1,x_2,x_3) = (x_1)^2 + (x_2-1)^2 + (x_3)^2$$
$$f_3(x_1,x_2,x_3) = (x_1)^2 + (x_2)^2 + (x_3-1)^2$$
subject to
$$-2 \leq x_1,x_2,x_3 \leq 2$$

The values in the tables represent the mean and the standard deviation values of the quality measure given by Eq. (22.5) obtained over 50 runs of the algorithm. Again, different values for the weight vector λ of the value function are considered.

Table 22.3: Performance of the algorithm over the first 3-objective optimization problem considered

	linear VF		*quadratic VF*		L_4*-metric VF*		*Tchebycheff VF*	
iteration	mean	std	mean	std	mean	std	mean	std
10	99.750	0.354	99.506	0.554	99.254	0.840	97.566	1.816
50	99.892	0.130	99.817	0.173	99.724	0.255	98.644	1.235

Table 22.4: Performance of the algorithm over the second 3-objective optimization problem considered

	linear VF		*quadratic VF*		*L_4-metric VF*		*Tchebycheff VF*	
iteration	mean	std	mean	std	mean	std	mean	std
10	99.617	0.440	99.498	0.479	99.277	0.684	96.120	3.482
50	99.842	0.145	99.738	0.247	99.620	0.359	97.896	1.915

For both problems, after 50 iterations of our algorithm the quality of the solution is at most 2.2% less than the quality of the best solution over the Pareto front. While a run of 10 iterations of the algorithm returns a solution whose average quality differs less than 4% from the value of $\mathrm{QM}(\mathbf{f}^{fin})$, in the worst case.

At each iteration of our algorithm, if the linear programming problem generated by the *middlemost weights* technique is infeasible, a linear local approximation of the value function is performed. For instance, assuming a Tchebycheff value function, this case has been observed on average for 18.94% and 6.48% of the iterations performed while solving the first and the second 3-objective optimization problem considered, respectively.

Figures 22.3 and 22.4 depict the evolution of the solution quality during the algorithm execution. The error bars represent the standard deviation of the measurements. In particular, Figure 22.3 shows the performance over the *Binh* problem in the case of the Tchebycheff value function, while Figure 22.4 refers to the second 3-objective optimization problem considered in the case of the L_4-metric value function. Within 5 iterations the quality of the solution becomes greater than 98.4% and 93.1% of the value of the best Pareto-optimal solution in the case of the *Binh* and the second 3-objective optimization problem considered, respectively.

Even if the most important metric to measure the performance of an interactive approach is given by the number of interactions with the DM, note that the time required by the optimization stage alternating the decision making phases is negligible. For example, in the case of the DM simulated by a quadratic value function, the mean time to perform 50 iterations of our algorithm to solve the *Binh* problem is 7 seconds on an Intel Xeon 2.8 GHz computer with 32 GB RAM.

22.5 Conclusion

In this work, a weight space reduction method solving nonlinear multi-objective optimization problems is introduced. The method is suitable for MOOPs with convex Pareto front. We develop an iterative approach by asking just one question at each iteration to the decision maker. The question consists of the comparison of a couple of non-dominated solutions. The response of the decision maker is translated into a linear constraint, reducing the feasible weight space. The decision maker's preference is approximated by a linear model. When the constraints in the weight space

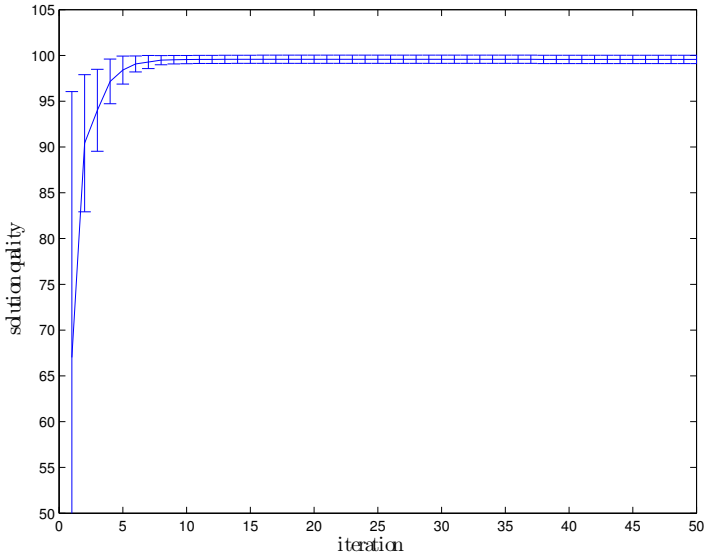

Fig. 22.3: Evolution of the quality of the final solution while solving the *Binh* problem in the case of the Tchebycheff value function. The values are averaged over 50 runs

make the problem infeasible, they are relaxed by introducing penalties for violations that are to be minimized, leading to a quadratic optimization problem. Preliminary results show promising performance on a set of benchmark problems.

Future directions of research will study how well the algorithm deals with inaccurate and moderately inconsistent answers obtained from the decision maker. In the experiments performed during this work, the decision maker is assumed to provide perfect preference information according to the value function considered. The introduction of penalties allows also to deal with inaccuracies in the evaluation of the solutions by the decision maker (i.e., evaluation noise). Additional work in progress, which will be presented in a different and more extended paper, is related to more complex and possibly nonlinear scalarization methods, different learning techniques obtained from the machine learning community, and the consideration of a wider set of benchmark problems.

References

1. Andronescu, M., Condon, A., Hoos, H., Mathews, D. H., Murphy, K. P.: Efficient parameter estimation for RNA secondary structure prediction. Bioinformatics 23(13), 19–28 (2007)
2. Battiti, R., Brunato, M., Mascia, F.: Reactive Search and Intelligent Optimization. Vol. 45 of Operations research/Computer Science Interfaces. Springer Verlag (2008)

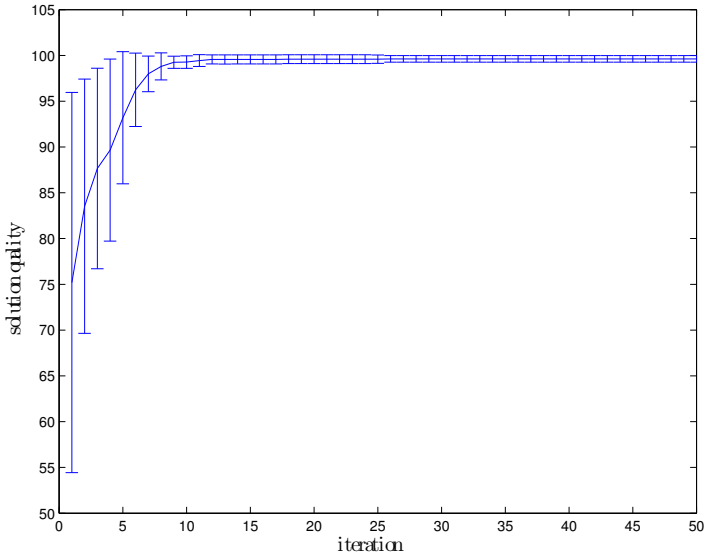

Fig. 22.4: Evolution of the quality of the final solution while solving the second 3-objective optimization problem considered in the case of the L_4-metric value function. The values are averaged over 50 runs

3. Binh, T. T., Korn, U.: An evolution strategy for the multiobjective optimization. In: Proceedings of the Second International Conference Mendel, pp. 23–28 (1996)
4. Dell, R., Karwan, M.: An interactive MCDM weight space reduction method utilizing a Tchebycheff utility function. Naval Research Logistics 37(2), 403–418 (1990)
5. Emmerich, M.: A rigorous analysis of two bi-criteria problem families with scalable curvature of the pareto fronts. Technical Report LIACS TR 2005-05, Leiden Institute of Advanced Computer Science, Leiden, The Netherlands (2005)
6. Lamont, G. B., Veldhuizen, D. A. V.: Evolutionary Algorithms for Solving Multi-Objective Problems. Kluwer, Norwell, MA (2002)
7. Miettinen, K.: Introduction to multiobjective optimization: Noninteractive approaches. Lecture Notes in Computer Science 5252, 1–26 (2008)
8. Miettinen, K., Ruiz, F., Wierzbicki, A.: Introduction to multiobjective optimization: Interactive approaches. Lecture Notes In Computer Science 5252, 27–57 (2008)
9. Molina, J., Laguna, M., Martí, R., Caballero, R.: SSPMO: A scatter tabu search procedure for non-linear multiobjective optimization. INFORMS Journal on Computing 19(1), 91–100 (2007)
10. Phelps, S. P., Köksalan, M.: An interactive evolutionary metaheuristic for multiobjective combinatorial optimization. Management Science 49(12), 1726–1738 (2003)
11. Steuer, R., Choo, E.: An interactive weighted Tchebycheff procedure for multiple objective programming. Mathematical Programming 26(1), 326–344 (1983)
12. Zhang, Q., Li, H.: MOEA/D: A multiobjective evolutionary algorithm based on decomposition. IEEE Transactions on Evolutionary Computation 11(6), 712–731 (2007)
13. Zionts, S., Wallenius, J.: An interactive programming method for solving the multi-criteria problem. Management Science 22(6), 652–663 (1976)
14. Zionts, S., Wallenius, J.: An interactive multiple objective linear programming method for a class of underlying nonlinear utility functions. Management Science 29(5), 519–529 (1983)