Pré-Publicações do Departamento de Matemática
Universidade de Coimbra
Preprint Number 10–18

DIRECT MULTISEARCH FOR MULTIOBJECTIVE OPTIMIZATION

A. L. CUSTÓDIO, J. F. A. MADEIRA, A. I. F. VAZ AND L. N. VICENTE

Abstract: In practical applications of optimization it is common to have several conflicting objective functions to optimize. Frequently, these functions are subject to noise or can be of black-box type, preventing the use of derivative-based techniques.

We propose a novel multiobjective derivative-free methodology, calling it direct multisearch (DMS), which does not aggregate any of the objective functions. Our framework is inspired by the search/poll paradigm of direct-search methods of directional type and uses the concept of Pareto dominance to maintain a list of non-dominated points (from which the new iterates or poll centers are chosen). The aim of our method is to generate as many points in the Pareto front as possible from the polling procedure itself, while keeping the whole framework general enough to accommodate other disseminating strategies, in particular when using the (here also) optional search step. DMS generalizes to multiobjective optimization (MOO) all direct-search methods of directional type.

We prove under the common assumptions used in direct search for single optimization that at least one limit point of the sequence of iterates generated by DMS lies in (a stationary form of) the Pareto front. However, extensive computational experience has shown that our methodology has an impressive capability of generating the whole Pareto front, even without using a search step.

Two by-products of this paper are (i) the development of a collection of test problems for MOO and (ii) the extension of performance and data profiles to MOO, allowing a comparison of several solvers on a large set of test problems, in terms of their efficiency and robustness to determine Pareto fronts.

Keywords: Multiobjective optimization, derivative-free optimization, direct-search methods, positive spanning sets, Pareto dominance, nonsmooth calculus, performance profiles, data profiles.
AMS Subject Classification (2010): 90C29, 90C30, 90C56.

1. Introduction

Many optimization problems involve the simultaneous optimization of different objectives or goals, often conflictual. In this paper, we are interested in the development of *derivative-free methods* (see [9]) for Multiobjective optimization (MOO). Such methods are appropriated when computing the

Date: June 1, 2010.
Support for this work was provided by FCT under the grant PTDC/MAT/098214/2008.

derivatives of the functions involved is expensive, unreliable, or even impossible. Frequently, the term *black-box* is used to describe objective and/or constraints functions for which, given a point, the value of the function is (hopefully) returned and no further information is provided. The significant increase of computational power and software sophistication observed in the last decades opened the possibility of simulating large and complex systems, leading to the optimization of expensive black-box functions. Such type of black-box functions also appear frequently in MOO problems (see, for instance, [22]).

In the classical literature of MOO, solution techniques are typically classified depending on the moment where the decision maker is able to establish preferences relating the different objectives (see [34]). Solution techniques with a *prior articulation of preferences* require an aggregation criterion before starting the optimization, combining the different objective functions into a single one. In the context of derivative-free optimization, this approach has been followed in [3, 32]. Different approaches can be considered when aggregating objectives, among which min-max formulations, weighted sums and nonlinear approaches (see, for instance, [41]), and goal programming [28]. In any case, the decision maker must associate weights or/and goals with each objective function. Since the original MOO problem is then reduced to a single objective problem, a typical output will consist of a single nondominated point. If the preferences of the decision maker change, the whole optimization procedure needs to be reapplied.

Posteriori articulation of preferences solution techniques circumvent these difficulties, by trying to capture the whole Pareto front for the MOO problem. Weighted-sum approaches can also be part of these techniques, considering the weights as parameters and varying them in order to capture the whole Pareto front. However, such methods might be time consuming and might not guarantee an even distribution of points, specially when the Pareto front is nonconvex (see [13]). The normal boundary intersection method [14] was proposed to address these difficulties, but it may provide nondominated points as part of the final output. The class of *posteriori articulation of preferences* techniques also includes heuristics such as genetic algorithms [40] and simulated annealing [42].

The herein proposed algorithmic framework is a member of this latter class of techniques, since it does not aggregate any of the objective functions. Instead, it directly extends, from single to multiobjective optimization, a

popular class of directional derivative-free methods, called direct search [9, Chapter 7]. Each iteration of these methods can be organized around a search step and a poll step. Given a current iterate (a poll center), the poll step in single optimization evaluates the objective function at some neighbor points defined by a positive spanning set and a step size parameter. We do the same for MOO but change the acceptance criterion of new iterates using Pareto dominance, which then requires the updating of a list of (feasible) nondominated points. At each iteration, polling is performed at a point selected from this list and its success is dictated by changes in the list. Our framework encompasses a search step too, whose main purpose is to further disseminate the search process of all the Pareto front.

We coined this new methodology *direct multisearch* (DMS) — as it reduces to direct search when there is only a single objective function. DMS extends to MOO all types of direct-search methods of directional type such as pattern search and generalized pattern search (GPS) [1, 30], generating set search (GSS) [30], and mesh adaptive direct search (MADS) [2].

Our paper is divided as follows. In Section 2, we introduce some of the concepts and terminology used in MOO and in nonsmooth calculus, required for what comes next. Section 3 describes the proposed DMS algorithmic framework. (An example illustrating how DMS works is described in the appendix of the paper.) The convergence analysis can be found in Section 4, where we prove, using Clarke's nonsmooth calculus, that at least a limit point of the sequence of iterates generated by DMS lies in (a stationary form of) the Pareto front.

Section 5 of this paper provides information about how our extensive numerical experiments were performed, in particular we describe the set of test problems, the solvers selected for comparison, the metrics used to assess the ability to compute Pareto fronts, and the use of performance and data profiles in MOO. In Section 6 we report a summary of our computational findings, showing the effectiveness and robustness of DMS to compute a relatively accurate approximated Pareto front (even when the initial list of nondominated points is initialized with a singleton and no search step is used). The paper ends with some final comments and discussion of future work in Section 7.

2. Concepts and definitions

The main theoretical result of this paper states that a limit point of the sequence of iterates generated by a DMS method is Pareto-Clarke stationary.

The goal of this section is to introduce this definition of stationarity as well as other concepts related to Pareto optimality and nonsmooth calculus, required for the presentation and analysis of the DMS framework. For a more complete revision on these subjects, the reader can consult [36] and [7], respectively.

2.1. Multiobjective optimization (MOO). We pose a constrained nonlinear MOO problem in the form:

$$\begin{aligned} \min \quad & F(x) \equiv (f_1(x), f_2(x), \ldots, f_m(x))^\top \\ \text{s.t.} \quad & x \in \Omega \subset \mathbb{R}^n, \end{aligned}$$

where we consider m (≥ 1) real-extended value objective functions $f_i : \mathbb{R}^n \to \mathbb{R} \cup \{+\infty\}, i = 1, \ldots, m$ (forming the objective function $F(x)$), and Ω represents the feasible region.

When several functions are present, given a point, it may be impossible to find another one which simultaneously improves the value of all the objective function components at the given one. The concept of Pareto dominance is crucial for comparing any two points, and to describe it we will make use of the partial order induced by the cone

$$\mathbb{R}^m_+ = \{z \in \mathbb{R}^m : z \geq 0\},$$

defined by

$$F(x) \prec_F F(y) \iff F(y) - F(x) \in \mathbb{R}^m_+ \setminus \{0\}.$$

Given two points x, y in Ω, we say that $x \prec y$ (x dominates y) when $F(x) \prec_F F(y)$. We will also say that a set of points in Ω is nondominated (or indifferent) when no point in the set is dominated by another one in the set. The Pareto front or frontier is the set of points in Ω nondominated by any other one in Ω (see below).

The concept of minimization in single optimization needs to be adapted to MOO. In MOO problems it is common to have several conflicting objective functions. Finding a point which corresponds to a minima for all the objectives considered, meaning an *ideal point*, may be an unrealistic task. Once again, the concept of Pareto dominance is used to characterize global and local optimality.

Definition 2.1. *A point $x_* \in \Omega$ is said to be a global Pareto minimizer of F in Ω if $\nexists y \in \Omega$ such that $y \prec x_*$. If there exists a neighborhood $\mathcal{N}(x_*)$ of x_* such that the previous property holds in $\Omega \cap \mathcal{N}(x_*)$, then x_* is called a local Pareto minimizer of F.*

Rigorously speaking, the Pareto front is the set of global Pareto minimizers. However, our convergence results have a local nature in the sense that deal with necessary conditions for local Pareto minimization.

2.2. Tangent cones and generalized derivatives. To establish first-order optimality conditions for constrained optimization one needs to consider appropriate cones of directions. We start by defining the (Clarke) tangent cone, which we will use to state Pareto first-order stationarity. The definition and notation are taken from [2].

Definition 2.2. *A vector $d \in \mathbb{R}^n$ is said to be a Clarke tangent vector to the set $\Omega \subseteq \mathbb{R}^n$ at the point x in the closure of Ω if for every sequence $\{y_k\}$ of elements of Ω that converges to x and for every sequence of positive real numbers $\{t_k\}$ converging to zero, there exists a sequence of vectors $\{w_k\}$ converging to d such that $y_k + t_k w_k \in \Omega$.*

The set $T_\Omega^{Cl}(x)$ of all Clarke tangent vectors to Ω at x is called the Clarke tangent cone to Ω at x.

We will also need the definition of hypertangent cone since it is strongly related to the type of iterates generated by a direct-search method of directional type. Again we will follow the notation in [2].

Definition 2.3. *A vector $d \in \mathbb{R}^n$ is said to be a hypertangent vector to the set $\Omega \subseteq \mathbb{R}^n$ at the point x in Ω if there exists a scalar $\epsilon > 0$ such that*

$$y + tw \in \Omega, \quad \forall y \in \Omega \cap B(x;\epsilon), \quad w \in B(d;\epsilon), \quad \text{and} \quad 0 < t < \epsilon.$$

The set of all hypertangent vectors to Ω at x is called the hypertangent cone to Ω at x and is denoted by $T_\Omega^H(x)$. Note that the Clarke tangent cone is the closure of the hypertangent one.

If we assume that $F(x)$ is Lipschitz continuous near x (meaning that each $f_i(x)$, $i = 1, \ldots, m$, is Lipschitz continuous in a neighborhood of x), we can define the Clarke-Jahn generalized derivatives of the individual functions along directions d in the hypertangent cone to Ω at x,

$$f_i^\circ(x;d) = \limsup_{\substack{x' \to x, x' \in \Omega \\ t \downarrow 0, x' + td \in \Omega}} \frac{f_i(x' + td) - f_i(x')}{t}, \ i = 1, \ldots, m. \quad (1)$$

These derivatives are essentially the Clarke generalized directional derivatives [7], extended by Jahn [26] to the constrained setting. The Clarke-Jahn

generalized derivatives along directions v in the tangent cone to Ω at x, are computed by taking a limit, i.e., $f_i^\circ(x;v) = \lim_{d \in T_\Omega^H(x), d \to v} f_i^\circ(x;d)$, for $i = 1, \ldots, m$, see [2].

We are now able to introduce the definition of Pareto-Clarke stationarity which will play a key role in our paper.

Definition 2.4. *Let F be Lipschitz continuous near a point $x_* \in \Omega$. We say that x_* is a Pareto-Clarke critical point of F in Ω if, for all directions $d \in T_\Omega^{Cl}(x_*)$, there exists a $j = j(d) \in \{1, \ldots, m\}$ such that $f_j^\circ(x_*;d) \geq 0$.*

Definition 2.4 says essentially that there is no direction in the tangent cone that is descent for all the objective functions. If a point is a Pareto minimizer (local or global), then it is necessarily a Pareto-Clarke critical point.

By assuming strict differentiability for each component of the objective function at x_* (meaning that the corresponding Clarke generalized gradient is a singleton), the previous definition of Pareto-Clarke stationarity can be restated using the gradient vectors.

Definition 2.5. *Let F be strictly differentiable at a point $x_* \in \Omega$. We say that x_* is a Pareto-Clarke-KKT critical point of F in Ω if, for all directions $d \in T_\Omega^{Cl}(x_*)$, there exists a $j = j(d) \in \{1, \ldots, m\}$ such that $\nabla f_j(x_*)^\top d \geq 0$.*

3. Direct multisearch for multiobjective optimization

In derivative-free optimization it is common to use an extreme barrier approach to deal with constraints. We adapt the extreme barrier function to multiobjective optimization (MOO) by setting

$$F_\Omega(x) = \begin{cases} F(x) & \text{if } x \in \Omega, \\ (+\infty, \ldots, +\infty)^\top & \text{otherwise.} \end{cases} \tag{2}$$

When a point is infeasible, the components of the objective function F are not evaluated, and the values of F_Ω are set to $+\infty$. This approach allows to deal with black-box type constraints, where only a yes/no type of answer is returned.

We present a general description for direct multisearch (DMS), which encompasses algorithms using different globalization strategies, like those based on rational lattices and only requiring simple decrease of the objective function values for accepting new iterates (see, for example, Generalized Pattern Search [1, 30] and Mesh Adaptive Direct Search [2]), and also algorithms

whose globalization strategy imposes a sufficient decrease condition for accepting new iterates (like Generating Set Search methods [30]).

Following the MOO terminology, introduced in Section 2, the proposed algorithmic framework keeps a list of previously evaluated feasible nondominated points and corresponding step size parameters. This list plays an important role since new iterate points (i.e., poll centers) are chosen from it. Also, as we will see later, success is defined by a change in this list. Thus, we need to introduce the concept of *iterate list* in addition to the concept of *iterate point* (used in direct-search methods of directional type for single optimization).

As in direct-search methods of directional type for single optimization, each iteration is organized around a search step and a poll step, and it is the latter the one determinant to obtain the convergence results. In DMS, the search step is also optional and used to possibly improve algorithmic performance. Each poll step starts by choosing one of the nondominated points stored in the current iterate list as the iterate point (poll center) and by performing a local search around it.

In both the search and the poll steps, a temporary list of points is created first, which stores all the points in the current iterate list and all the points evaluated during the course of the step. This temporary list will then be filtered, removing all the dominated points and keeping only the nondominated ones. Note that from (2), as we will later see in the description of the algorithm, the infeasible points evaluated during the course of the step are trivially removed.

The trial list is then extracted from this filtered list of feasible nondominated points, and must necessarily include (for the purposes of the convergence theory) all the nondominated points which belonged to the iterate list considered at the previous iteration. Different criteria can then be chosen to determine the trial list. A natural possibility is to define the trial list exactly as the filtered one. We will discuss this issue in more detail after the presentation of the algorithmic framework. When the trial list L_{trial} is different from the current iterate list L_k, the new iterate list L_{k+1} is set to L_{trial} (successful search or poll step and iteration). Otherwise, $L_{k+1} = L_k$ (unsuccessful poll step and iteration).

When using sufficient decrease to determine dominancy, one makes use of a forcing function $\rho : (0, +\infty) \to (0, +\infty)$, i.e., a continuous and nondecreasing function satisfying $\rho(t)/t \to 0$ when $t \downarrow 0$ (see [30]). Typical examples

of forcing functions are $\rho(t) = t^{1+a}$, for $a > 0$. To write the algorithm in general terms, we will use $\bar{\rho}(\cdot)$ to either represent an m-uple of forcing functions $(\rho(\cdot), \ldots, \rho(\cdot))^\top$ or the constant, zero vector of dimension m. For the purposes of the search step, we say that the point x dominates y (where α is a step size parameter) if $F(x) \prec_F F(y) - \bar{\rho}(\alpha)$. When considering the poll step, the point $x + \alpha d$ dominates y (where d is a direction used in polling around x) if $F(x + \alpha d) \prec_F F(y) - \bar{\rho}(\alpha\|d\|)$. The criterion for testing dominance can thus be based on simple or sufficient decrease, depending on the algorithmic variant considered.

As we will see later in the convergence analysis, the set of directions to be used for polling is not required to positively span $\mathbb{R}^n$ (although for coherence with the smooth case we will write it so in the algorithm below), and it is not necessarily drawn from a finite set of directions.

Algorithm 3.1 (Direct Multisearch for MOO).

Initialization

Choose $x_0 \in \Omega$ with $f_i(x_0) < +\infty, \forall i \in \{1, \ldots, m\}$, $\alpha_0 > 0$, $0 < \beta_1 \leq \beta_2 < 1$, and $\gamma \geq 1$. Let $\mathcal{D}$ be a (possibly infinite) set of positive spanning sets. Initialize the list of nondominated points and corresponding step size parameters $L_0 = \{(x_0; \alpha_0)\}$.

For $k = 0, 1, 2, \ldots$

(1) **Selection of an iterate point:** Order the list L_k and select a point and the corresponding step size parameter, $(x; \alpha) \in L_k$. Set $(x_k; \alpha_k) = (x; \alpha)$.

(2) **Search step:** Compute a finite set of points $\{z_s\}_{s \in S}$ (in a mesh if $\bar{\rho}(\cdot) = 0$, see Section 4.1) and evaluate F_Ω on each element. Set $L_{add} = \{(z_s; \alpha_k), s \in S\}$.
Call $L_{filtered}$ `= filter(`L_k`,`L_{add}`)` to eliminate dominated points from $L_k \cup L_{add}$. Call L_{trial} `= select(`$L_{filtered}$`)` to determine $L_{trial} \subseteq L_{filtered}$. If $L_{trial} \neq L_k$ declare the iteration (and the search step) successful, set $L_{k+1} = L_{trial}$, and skip the poll step.

(3) **Poll step:** Choose a positive spanning set D_k from the set $\mathcal{D}$. Evaluate F_Ω at the set of poll points $P_k = \{x_k + \alpha_k d : d \in D_k\}$. Set $L_{add} = \{(x_k + \alpha_k d; \alpha_k), d \in D_k\}$.
Call $L_{filtered}$ `= filter(`L_k`,`L_{add}`)` to eliminate dominated points from $L_k \cup L_{add}$. Call L_{trial} `= select(`$L_{filtered}$`)` to determine $L_{trial} \subseteq L_{filtered}$. If $L_{trial} \neq L_k$ declare the iteration (and the poll step) as

successful and set $L_{k+1} = L_{trial}$. Otherwise, declare the iteration (and the poll step) unsuccessful and set $L_{k+1} = L_k$.

(4) **Mesh parameter update:** If the iteration was successful then maintain or increase the corresponding step size parameters: $\alpha_{k,new} \in [\alpha_k, \gamma\alpha_k]$ and replace all the new points $(x_k+\alpha_k d; \alpha_k)$ in L_{k+1} by $(x_k + \alpha_k d; \alpha_{k,new})$; replace also $(x_k; \alpha_k)$, if in L_{k+1}, by $(x_k; \alpha_{k,new})$. Otherwise decrease the step size parameter: $\alpha_{k,new} \in [\beta_1\alpha_k, \beta_2\alpha_k]$ and replace the poll pair $(x_k; \alpha_k)$ in L_{k+1} by $(x_k; \alpha_{k,new})$.

Next we address several issues left open during the discussion and presentation of the DMS framework.

List initialization. For simplicity, the algorithmic description presented initialized the list with a single point, but different strategies, considering several feasible previously evaluated points, can be used in this initialization, with the goal of improving the algorithmic performance. In Section 6.1, we suggest and numerically test three possible ways of initializing the list. Note that a list initialization can also be regarded as a search step in iteration 0.

Ordering the iterate list. The number of elements stored in the list can vary from one to several, depending on the problem characteristics and also on the criteria implemented to determine the trial list. In a practical implementation, when the iterate list stores several points, it may be crucial to order it before selecting a point for polling, as a way of diversifying the search and explore different regions of Ω. A crude order strategy could be, for instance, (i) to always add points to the end of the list and (ii) to move a point already selected as a poll center to the end of the list (doing it at the end of an iteration) for a better dissemination of the search of the Pareto front.

Search step and selection of the iterate point. The search step is optional and, in the case of DMS ($m > 1$), it might act on the iterate list L_k rather than around an individual point. For consistency with single optimization ($m = 1$), we included the selection of the point iterate before the search step. If the search step is skipped or if it fails, this iterate point will then be the poll center. Another reason for this inclusion is to define a step size parameter for the search step.

Polling. As in single optimization, one can have a complete or an opportunistic poll step. In the algorithmic framework presented above, we have used complete polling, which can be a wise choice if the goal is to compute

Algorithm 3.2: $[L_3]$=`filter`(L_1, L_2)

Set $L_3 = L_1 \cup L_2$
for all $x \in L_2$
do
- **for** all $y \in L_3$, $y \neq x$
  - **do**
    - **if** $y \prec x$
      - **then** $\{L_3 = L_3 \backslash \{x\}$
- **if** $x \in L_3$
  - **then**
    - **for** all $y \in L_3$, $y \neq x$
      - **do**
        - **if** $x \prec y$
          - **then**
            - $L_3 = L_3 \backslash \{y\}$
            - **if** $y \in L_2$
              - **then** $\{L_2 = L_2 \backslash \{y\}$

FIGURE 1. Procedure for filtering the dominated points from $L_1 \cup L_2$ (the set union should not allow element repetition), assuming that L_1 is already formed by nondominated points.

the complete Pareto front. Opportunistic polling may be more suitable to deal with functions of considerably expensive evaluation. In this latter case, in order to improve the algorithmic performance, the poll set should be appropriately ordered before polling [10, 12]. Since the convergence results will rely on the analysis of the algorithmic behavior at unsuccessful iterations, which is identical independently of the polling strategy considered (opportunistic or complete), the results hold for both variants without any further modifications.

Filtering dominated points. Note that the filtering process of the dominated points does not require comparisons among all the stored points since the current iterate list L_k is already formed by nondominated points. Instead, only each added point will be compared to the others, and, in particular, (i) if any of the points in the list $L_k \cup L_{add}$ dominates a point in L_{add}, this added point will be discarded; (ii) if an added point dominates any of the remaining points in the list $L_k \cup L_{add}$, all such dominated points will be discarded. An algorithmic description of the procedure used for filtering the list can be found in Figure 1.

Algorithm 3.3: $[L_{trial}]$=`select`$(L_{filtered})$

Set $L_{trial} = L_{filtered}$

Algorithm 3.4: $[L_{trial}]$=`select`$(L_{filtered})$

if $L_k = \{x_k\} \not\subseteq L_{filtered}$
then { Choose $x \in L_{filtered}$; Set $L_{trial} = \{x\}$
else { Set $L_{trial} = L_k$

FIGURE 2. Two procedures for selecting the trial list L_{trial} from the list of filtered nondominated points $L_{filtered}$. The list L_k represents the iterate list considered at the current iteration. Note that in both algorithms all the nondominated points in L_k are included in L_{trial}, as required for the convergence theory.

Selecting the trial list. As we have pointed out before, a natural candidate for the new iterate list is $L_{trial} = L_{filtered}$, in particular if our goal is to determine as many points in the Pareto front as possible. However, other choices $L_{trial} \subset L_{filtered}$ can be considered. A more restrictive strategy, for instance, is to always consider an iterate list formed by a single point. In such a case, success is achieved if the new iterate point dominates the current one. This type of algorithm fits in our framework since it suffices to initialize the list as a singleton and to only consider in L_{trial} the point that dominates the one in L_k when it exists, or the point already in L_k, otherwise. An algorithmic description of these two procedures can be found in Figure 2.

In the appendix of this paper we describe an example illustrating how DMS (Algorithm 3.1) works.

4. Convergence analysis

One of the key ingredients to state global convergence properties for direct-search methods of directional type is to establish the existence of a subsequence of step size parameters converging to zero. There are two main strategies which can be used to enforce this property in this class of methods: (i)

to ensure that all new iterates lie in a rational lattice or (ii) to impose a sufficient decrease condition in the objective function values when accepting new iterates. To derive this result for direct multisearch (DMS), we need the iterates to lie in a compact set in the former case, and the objective functions must be bounded below in the latter situation.

Assumption 4.1. *The level set* $L(x_0) = \bigcup_{i=1}^{m} L_i(x_0)$ *is compact, where* $L_i(x_0) = \{x \in \Omega : f_i(x) \leq f_i(x_0)\}, i = 1, \ldots, m$. *The objective function components of* F *are bounded below in* $L(x_0)$.

4.1. Globalization using rational lattices. When considering continuously differentiable functions, a finite set of directions which satisfies appropriate integrality requirements is enough to ensure convergence in single optimization. Generalized Pattern Search (GPS) [1, 30] makes use of such a set of directions by setting $D = \mathcal{D}$.

Assumption 4.2. *The set* D *of positive spanning sets is finite and the elements of* D *are of the form* $G\bar{z}_j$, $j = 1, \ldots, |D|$, *where* $G \in \mathbb{R}^{n \times n}$ *is a nonsingular matrix and each* $\bar{z}_j$ *is a vector in* $\mathbb{Z}^n$.

To deal with the presence of nondifferentiability, it is desirable to consider an infinite set of directions $\mathcal{D}$, which should be dense (after normalization) in the unit sphere. However, if globalization is to be ensured by rational lattices, then some care must be taken when generating the set $\mathcal{D}$, as it is the case in Mesh Adaptive Direct Search (MADS) [2], where generating iterates in integer lattices is guaranteed by the first requirement of the next assumption.

Assumption 4.3. *Let* D *represent a finite set of positive spanning sets satisfying Assumption 4.2.*

The set $\mathcal{D}$ *is so that the elements* $d_k \in D_k \subseteq \mathcal{D}$ *satisfy the following conditions:*

(1) d_k *is a nonnegative integer combination of the columns of* D.
(2) The distance between x_k *and the point* $x_k + \alpha_k d_k$ *tends to zero if and only if* α_k *does:*

$$\lim_{k \in K} \alpha_k \|d_k\| = 0 \iff \lim_{k \in K} \alpha_k = 0,$$

for any infinite subsequence K.

(3) The limits of all convergent subsequences of $\bar{D}_k = \{d_k/\|d_k\| : \ d_k \in D_k\}$ *are positive spanning sets for* $\mathbb{R}^n$.

The third requirement above is not used in the convergence theory when applied to nonsmooth objective functions, but is included for consistency with the smooth case and because it is part of the MADS original presentation [2].

Also, the strategy for updating the step size parameter must conform to some form of rationality.

Assumption 4.4. *The step size parameter is updated as follows: Choose a rational number* $\tau > 1$, *a nonnegative integer* $m^{max} \geq 0$, *and a negative integer* $m^{min} \leq -1$. *If the iteration is successful, the step size parameter is maintained or increased by taking* $\alpha_{k,new} = \tau^{m^+}\alpha_k$, *with* $m^+ \in \{0, \ldots, m^{max}\}$. *Otherwise, the step size parameter is decreased by setting* $\alpha_{k,new} = \tau^{m^-}\alpha_k$, *with* $m^- \in \{m^{min}, \ldots, -1\}$.

By setting $\beta_1 = \tau^{m^-}$, $\beta_2 = \tau^{-1}$, and $\gamma = \tau^{m^+}$, the updating strategy described in Assumption 4.4 conforms with those of Algorithm 3.1.

An additional condition imposes that the search step will be conducted in a previously (implicitly defined) mesh or grid (see Assumption 4.5 below). We point out that poll points must also lie on the mesh (i.e., $P_k \subset M_k$), but such a requirement is trivially satisfied from the definition of the mesh M_k given below.

Assumption 4.5. *The search step in Algorithm 3.1 only evaluates points in*

$$M_k \ = \ \bigcup_{x \in E_k} \{x + \alpha_k D z : z \in \mathbb{N}_0^{|D|}\},$$

where E_k *is the set of all the points evaluated by the algorithm previously to iteration* k.

As a result of the previous assumptions, we can state the desired convergence result for the sequence of step size parameters, which was originally established by Torczon [43] in the context of pattern search and generalized by Audet and Dennis to GPS [1] and MADS [2] for single optimization.

Theorem 4.1. *Let Assumption 4.1 hold. Algorithm 3.1 under one of the Assumptions 4.2 or 4.3 combined with Assumptions 4.4–4.5 and* $\bar{\rho}(\cdot) = 0$ *generates a sequence of iterates satisfying*

$$\liminf_{k \to +\infty} \alpha_k \ = \ 0.$$

Proof: In order to arrive to a contradiction, let us assume that there is a strictly positive lower bound for the step size parameter. Classical arguments, similar to the ones used by Torczon [43] and Audet and Dennis [1] for single optimization, allow us to conclude that all the iterates and poll points (i.e., points of the form $x_k+\alpha_k d$, for $d \in D_k$) generated by DMS (Algorithm 3.1) lie in a rational lattice. The intersection of a compact set with a rational lattice is finite and thus the number of points which can be added to the iterate list is finite. It remains to show that the algorithm cannot cycle among this finite set of points.

If a point is removed from the iterate list, then it is because it is dominated by another point in the new iterate list. Thus, by transitivity, it can never be added again to the iterate list. At each successful iteration, at least one new point is added to the iterate list. Since the number of points which can be added is finite, the number of successful iterations must also be finite, which, according to the step size updating rules, contradicts the fact that there is a lower bound on the step size parameter. ■

4.2. Globalization by imposing sufficient decrease. A different globalization strategy consists in using a forcing function, by considering $\bar{\rho}(\cdot) = \rho(\cdot)$ in Algorithm 3.1, imposing sufficient rather than simple decrease when accepting new iterates. The following result is relatively classic in nonlinear (single objective) optimization. Kolda, Lewis and Torczon [30] (see also [9, Section 7.7]) derive it in the context of direct-search methods of directional type, when applied to single objective optimization. We will need the following assumption (which, note, was already part of Assumption 4.3).

Assumption 4.6. *The distance between x_k and the point $x_k + \alpha_k d_k$ tends to zero if and only if α_k does:*

$$\lim_{k\in K} \alpha_k \|d_k\| \;=\; 0 \iff \lim_{k\in K} \alpha_k \;=\; 0,$$

for all $d_k \in D_k$ and for any infinite subsequence K.

Note that Assumption 4.6 is a weak condition on the set of directions $\mathcal{D}$. A normalized set of directions $\mathcal{D}$ dense in the unit sphere meets such a requirement.

Theorem 4.2. *Let Assumption 4.1 hold. Algorithm 3.1, when $\bar{\rho}(\cdot)$ is a forcing function and Assumption 4.6 holds, generates a sequence of iterates*

satisfying

$$\liminf_{k\to+\infty} \alpha_k \;=\; 0.$$

Proof: Let us assume that $\liminf_{k\to+\infty} \alpha_k \;\neq\; 0$, meaning that there is $\alpha_* > 0$ such that $\alpha_k > \alpha_*$, for all k. From Assumption 4.6, we then know that there is $\alpha_*^d > 0$ such that $\alpha_k \|d_k\| > \alpha_*^d$, for all k and $d_k \in D_k$. At each unsuccessful iteration k, the corresponding step size parameter is reduced by at least $\beta_2 \in (0, 1)$, and thus the number of successful iterations must be infinite. Since $\rho(\cdot)$ is a non decreasing function, which satisfies $\rho(t) > 0$, for $t > 0$, there exists $\rho_* > 0$ such that $\rho(\alpha_k) \geq \rho(\alpha_*) \geq \rho_*$ and $\rho(\alpha_k \|d_k\|) \geq \rho(\alpha_*^d) \geq \rho_*$, for all k and $d_k \in D_k$, with $\rho_* = \min(\rho(\alpha_*), \rho(\alpha_*^d))$, taking into account what can happen in both the search and the poll steps.

At each successful iteration, at least one point is added to the iterate list. Let $\{l_k\}$ denote the set of indices of successful iterations and consider the sequence $\{w_{l_k}\}$, where each w_{l_k} is chosen as one of the points which was added to the iterate list at iteration l_k. For any of these points, at least one of the components of the objective function F decreased, which then implies

$$\forall k,\ \exists i_{l_{k+1}} \in \{1, \ldots, m\}, \quad f_{i_{l_{k+1}}}(w_{l_{k+1}}) - f_{i_{l_{k+1}}}(w_{l_k}) \;<\; -\rho_*.$$

Since the number of components of the objective function is finite, there exists an $i \in \{1, \ldots, m\}$ such that, passing to a subsequence $\{l'_k\} \subset \{l_k\}$ if necessary,

$$\forall k, \quad f_i(w_{l'_{k+1}}) - f_i(w_{l'_k}) \;<\; -\rho_*.$$

Thus, $\liminf_{k\to+\infty} f_i(w_{l'_k}) = -\infty$, which contradicts Assumption 4.1. ∎

4.3. Refining subsequences and directions. The convergence analysis of direct-search methods of directional type for single optimization relies on the analysis of the behavior of the algorithm at limit points of sequences of unsuccessful iterates, denoted by refining subsequences (a concept formalized in [1]). The same will happen with DMS.

Definition 4.1. *A subsequence $\{x_k\}_{k\in K}$ of iterates corresponding to unsuccessful poll steps is said to be a refining subsequence if $\{\alpha_k\}_{k\in K}$ converges to zero.*

Assumption 4.1, Theorems 4.1 or 4.2, and the updating strategy of the step size parameter allow us to establish the existence of at least a convergent refining subsequence (see, e.g., [9, Section 7.3]).

Theorem 4.3. *Let Assumption 4.1 hold. Consider a sequence of iterates generated by Algorithm 3.1 under the scenarios of either Subsection 4.1 (rational lattices) or Subsection 4.2 (sufficient decrease). Then there is at least one convergent refining subsequence* $\{x_k\}_{k\in K}$.

The first stationarity result in our paper will establish appropriate nonnegativity of generalized directional derivatives (see Definition 2.4) computed along certain limit directions, designated as refining directions (a notion formalized in [2]).

Definition 4.2. *Let* x_* *be the limit point of a convergent refining subsequence. If the limit* $\lim_{k\in K'} d_k/\|d_k\|$ *exists, where* $K' \subseteq K$ *and* $d_k \in D_k$*, and if* $x_k + \alpha_k d_k \in \Omega$*, for sufficiently large* $k \in K'$*, then this limit is said to be a refining direction for* x_*.

Note that refining directions exist trivially in the unconstrained case $\Omega = \mathbb{R}^n$.

4.4. Convergence results. We are now in a position to state the main convergence result of our paper.

Theorem 4.4. *Consider a refining subsequence* $\{x_k\}_{k\in K}$ *converging to* $x_* \in \Omega$ *and a refining direction* d *for* x_* *in* $T_\Omega^H(x_*)$*. Assume that* F *is Lipschitz continuous near* x_**. Then, there exists a* $j = j(d) \in \{1,\ldots,m\}$ *such that* $f_j^\circ(x_*;d) \geq 0$.

Proof: Let $\{x_k\}_{k\in K}$ be a refining subsequence converging to $x_* \in \Omega$ and $d = \lim_{k\in K''} d_k/\|d_k\| \in T_\Omega^H(x_*)$ a refining direction for x_*, with $d_k \in D_k$ and $x_k + \alpha_k d_k \in \Omega$ for all $k \in K'' \subseteq K$.

For $j \in \{1,\ldots,m\}$ we have

$$\begin{aligned}
f_j^\circ(x_*;d) &= \limsup_{\substack{x'\to x_*,\, x'\in\Omega \\ t\downarrow 0,\, x'+td\in\Omega}} \frac{f_j(x'+td)-f_j(x')}{t} \\
&\geq \limsup_{k\in K''} \frac{f_j(x_k+\alpha_k\|d_k\|(d_k/\|d_k\|))-f_j(x_k)}{\alpha_k\|d_k\|} - r_k \\
&= \limsup_{k\in K''} \frac{f_j(x_k+\alpha_k d_k)-f_j(x_k)+\bar\rho(\alpha_k\|d_k\|)}{\alpha_k\|d_k\|} - \frac{\bar\rho(\alpha_k\|d_k\|)}{\alpha_k\|d_k\|} - r_k \\
&\geq \limsup_{k\in K''} \frac{f_j(x_k+\alpha_k d_k)-f_j(x_k)+\bar\rho(\alpha_k\|d_k\|)}{\alpha_k\|d_k\|}.
\end{aligned}$$

The first inequality follows from $\{x_k\}_{k\in K''}$ being a feasible refining subsequence and the fact that $x_k + \alpha_k d_k$ is feasible for $k \in K''$. The term r_k is bounded above by $\nu||d - d_k/\|d_k\|\|$, where ν is the Lipschitz constant of F near x_*. Note, also, that the limit $\lim_{k\in K''} \bar{\rho}(\alpha_k\|d_k\|)/(\alpha_k\|d_k\|)$ is 0 for both globalization strategies (Subsections 4.1 and 4.2). In the case of using rational lattices (Subsection 4.1), one uses $\bar{\rho}(\cdot) = 0$. When imposing sufficient decrease (Subsection 4.2), this limit follows from the properties of the forcing function and Assumption 4.6.

Since $\{x_k\}_{k\in K}$ is a refining subsequence, for each $k \in K''$, $x_k+\alpha_k d_k$ does not dominate x_k. Thus, for each $k \in K''$ it is possible to find $j(k) \in \{1, \ldots, m\}$ such that $f_{j(k)}(x_k + \alpha_k d_k) - f_{j(k)}(x_k) + \bar{\rho}(\alpha_k\|d_k\|) > 0$. Since the number of objective functions components is finite, there must exists one, say $j = j(d)$, for which there is an infinite set of indices $K''' \subseteq K''$ such that

$$f^\circ_{j(d)}(x_*; d) \;\geq\; \limsup_{k\in K'''} \frac{f_{j(d)}(x_k + \alpha_k d_k) - f_{j(d)}(x_k) + \bar{\rho}(\alpha_k\|d_k\|)}{\alpha_k\|d_k\|} \;\geq\; 0.$$

■

If we assume strict differentiability of F at the point x_*, the conclusion of the above result will be $\nabla f_j(x_*)^\top d \geq 0$.

Convergence for a Pareto-Clarke critical point (see Definition 2.4) or a Pareto-Clarke-KKT critical point (see Definition 2.5) can be established by imposing density in the unit sphere of the set of refining directions associated with x_*. We note that this assumption is stronger than just considering that the normalized set of directions $\mathcal{D}$ is dense in the unit sphere.

Theorem 4.5. *Consider a refining subsequence $\{x_k\}_{k\in K}$ converging to $x_* \in \Omega$. Assume that F is Lipschitz continuous near x_*. If the set of refining directions for x_* is dense in $T^{Cl}_\Omega(x_*)$, then x_* is a Pareto-Clarke critical point.*

If, in addition, F is strictly differentiable at x_*, then this point is a Pareto-Clarke-KKT critical point.

Proof: Given any direction v in the Clarke tangent cone, one has that

$$f^\circ_j(x_*; v) \;=\; \lim_{\substack{d \to v \\ d \in T^H_\Omega(x_*)}} f^\circ_j(x_*; d),$$

for all $j \in \{1, \ldots, m\}$ (see [2]).

Since the number of objective functions is finite, and from the previous theorem, there must exist a sequence of directions $\{d_w\}_{w\in W}$ in $T_\Omega^H(x_*)$, converging to v such that $f_j^\circ(x_*;d_w) \geq 0$ for all directions d_w in that sequence and for some $j = j(v) \in \{1,\ldots,m\}$. The first statement of the theorem follows by taking limits of the Clarke generalized derivatives in this sequence (and the second one results trivially). ■

Note that the assumption of density of the set of refining directions in the unit sphere is not required only because of the presence of constraints. In fact, it is also necessary even without constraints because one can easily present examples where the cone of directions simultaneously descent for all objective functions can be as narrow as one would like.

In the following corollary, we state the previous results for the particular case of single objective optimization, where the number of the objective function components equals one.

Corollary 4.1. *Let $m = 1$ and $F = (f_1) = f$.*

Under the conditions of Theorem 4.4, if $d \in T_\Omega^H(x_*)$ is a refining direction for x_*, then $f^\circ(x_*;d) \geq 0$.

Under the conditions of Theorem 4.5, the point x_* is a Clarke critical point, i.e., $f^\circ(x_*;v) \geq 0, \forall v \in T_\Omega^{Cl}(x_*)$.

If, additionally, we require the inclusion of all the nondominated points in the iterate list, and if it is finite the number of iterations for which the cardinality of the iterate list exceeds one, we can establish first-order convergence for an ideal point.

Corollary 4.2. *Consider the algorithmic variant where $L_{trial} = L_{filtered}$ in all iterations (Algorithm 3.3). Assume that is finite the number of iterations for which the cardinality of $\{L_k\}_{k\in K}$ exceeds one.*

Under the conditions of Theorem 4.4, if $d \in T_\Omega^H(x_*)$ is a refining direction for x_*, we have, for all $j \in \{1,\ldots,m\}$, $f_j^\circ(x_*;d) \geq 0$.

Under the conditions of Theorem 4.5, the point x_* is an ideal point, i.e.,

$$f_j^\circ(x_*;v) \;\geq\; 0, \quad \forall j \in \{1,\ldots,m\},\ \forall v \in T_\Omega^{Cl}(x_*).$$

Proof: Let us recall the proof of Theorem 4.4 until its last paragraph. Now, by assumption, it is possible to consider an infinite subset of indices $K''' \subseteq K''$ such that $|L_k| = 1$, for each $k \in K'''$. The selection criterion for the iterate list ensures that for each $k \in K'''$, $x_k + \alpha_k d_k$ is dominated by x_k and it follows

trivially that $f_j^\circ(x_*; d) \geq 0$ for all $j \in \{1, \ldots, m\}$. The proof of the second assertion follows the same type of arguments of the proof of Theorem 4.5. ■

5. Test problems, solvers, metrics, and profiles

5.1. Test problems. We have collected 100 multiobjective optimization (MOO) problems reported in the literature involving only simple bounds constraints, i.e., problems for which $\Omega = [\ell, u]$ with $\ell, u \in \mathbb{R}^n$ and $\ell < u$. All test problems were modeled by us in AMPL [21] and are available for public testing at `http://www.mat.uc.pt/dms`.

The problems and their dimensions are given in Table 1. To avoid a long presentation we do not describe their mathematical formulations, which can be found in the AMPL model files. We also provide in Table 1 the original references for these problems — noting, however, that in some cases the formulation coded differed from the literature due to errors, mismatches or lack of information found in the corresponding papers.

5.2. Solvers tested. We have considered in our numerical studies the following publicly available solvers for MOO without derivatives:

- AMOSA (Archived MultiObjective Simulated Annealing) [5] — `www.isical.ac.in/~sriparna_r/software.html`;
- BIMADS (BI-Objective Mesh Adaptive Direct Search) [3] tested only for problems with two objective functions — `www.gerad.ca/nomad/Project/Home.html`;
- Epsilon-MOEA (Epsilon MultiObjective Evolutionary Algorithm) [16] — `www.iitk.ac.in/kangal/codes.shtml`;
- GAMULTI (Genetic Algorithms for Multiobjective, MATLAB toolbox) — `www.mathworks.com`;
- MOPSO (MultiObjective Particle Swarm Optimization) [8] — `delta.cs.cinvestav.mx/~ccoello/EMOO/EMOOsoftware.html`;
- NSGA-II (Nondominated Sorting Genetic Algorithm II, C version) [17] — `www.iitk.ac.in/kangal/codes.shtml`;
- NSGA-II (MATLAB implementation by A. Seshadri) — `www.mathworks.com/matlabcentral/fileexchange/10429-nsga-ii-a-multi-objective-optimization-algorithm`;
- PAES (Pareto Archived Evolution Strategy) [29] — `dbkgroup.org/knowles/multi`.

Problem	n	m	Problem	n	m	Problem	n	m
BK1 [24]	2	2	I5 [23]	8	3	MOP3 [24]	2	2
CL1 [6]	4	2	IKK1 [24]	2	3	MOP4 [24]	3	2
Deb41 [15]	2	2	IM1 [24]	2	2	MOP5 [24]	2	3
Deb512a [15]	2	2	Jin1 [27]	2	2	MOP6 [24]	2	2
Deb512b [15]	2	2	Jin2 [27]	2	2	MOP7 [24]	2	3
Deb512c [15]	2	2	Jin3 [27]	2	2	OKA1 [38]	2	2
Deb513 [15]	2	2	Jin4 [27]	2	2	OKA2 [38]	3	2
Deb521a [15]	2	2	Kursawe [31]	3	2	QV1 [24]	10	2
Deb521b [15]	2	2	L1ZDT4 [18]	10	2	Sch1 [24]	1	2
Deb53 [15]	2	2	L2ZDT1 [18]	30	2	SK1 [24]	1	2
DG01 [24]	1	2	L2ZDT2 [18]	30	2	SK2 [24]	4	2
DPAM1 [24]	10	2	L2ZDT3 [18]	30	2	SP1 [24]	2	2
DTLZ1 [17]	7	3	L2ZDT4 [18]	30	2	SSFYY1 [24]	2	2
DTLZ1n2 [17]	2	2	L2ZDT6 [18]	10	2	SSFYY2 [24]	1	2
DTLZ2 [17]	12	3	L3ZDT1 [18]	30	2	TKLY1 [24]	4	2
DTLZ2n2 [17]	2	2	L3ZDT2 [18]	30	2	VFM1 [24]	2	3
DTLZ3 [17]	12	3	L3ZDT3 [18]	30	2	VU1 [24]	2	2
DTLZ3n2 [17]	2	2	L3ZDT4 [18]	30	2	VU2 [24]	2	2
DTLZ4 [17]	12	3	L3ZDT6 [18]	10	2	WFG1 [24]	8	3
DTLZ4n2 [17]	2	2	LE1 [24]	2	2	WFG2 [24]	8	3
DTLZ5 [17]	12	3	lovison1 [33]	2	2	WFG3 [24]	8	3
DTLZ5n2 [17]	2	2	lovison2 [33]	2	2	WFG4 [24]	8	3
DTLZ6 [17]	22	3	lovison3 [33]	2	2	WFG5 [24]	8	3
DTLZ6n2 [17]	2	2	lovison4 [33]	2	2	WFG6 [24]	8	3
ex005 [25]	2	2	lovison5 [33]	3	3	WFG7 [24]	8	3
Far1 [24]	2	2	lovison6 [33]	3	3	WFG8 [24]	8	3
FES1 [24]	10	2	LRS1 [24]	2	2	WFG9 [24]	8	3
FES2 [24]	10	3	MHHM1 [24]	1	3	ZDT1 [47]	30	2
FES3 [24]	10	4	MHHM2 [24]	2	3	ZDT2 [47]	30	2
Fonseca [20]	2	2	MLF1 [24]	1	2	ZDT3 [47]	30	2
I1 [23]	8	3	MLF2 [24]	2	2	ZDT4 [47]	10	2
I2 [23]	8	3	MOP1 [24]	1	2	ZDT6 [47]	10	2
I3 [23]	8	3	MOP2 [24]	4	2	ZLT1 [24]	10	3
I4 [23]	8	3						

TABLE 1. A description of our test set. Recall that n is the number of variables and m is the number of objective functions.

However, in order to keep the paper to a reasonable size and not to confuse the reader with excessive information, we are only reporting later (see Section 6.2) a part of the numerical tests that were performed. Besides four versions of our DMS, the selected solvers were AMOSA, BIMADS, and NSGA-II (C version), since these were the ones who exhibited the best performance in the above mentioned test set. The numerical results regarding the remaining codes can be found in `http://www.mat.uc.pt/dms`.

5.3. Metrics and profiles used for solver comparison. In the multiobjective case, one is interested in assessing the ability of a solver to obtain points which are Pareto optimal and to compute a highly diversified subset of the whole Pareto front. With these two goals in mind, we present in the next subsections the metrics used to assess the performance of the tested solvers. While there are other metrics in the literature, we have selected the ones presented herein due to its applicability to a large set of test problems. In particular, using a metric that considers the distance from the obtained Pareto front to the true Pareto one implies the knowledge of the latter for all the problems in the test set. In addition, presenting results for a metric that only considers a small number of test problems is meaningless. Despite not including a metric that requires the true Pareto front, we present later, and for illustrative purposes, a few plots depicting the computed Pareto front for some selected solvers on a small subset of problems where such information is available.

5.3.1. *Performance profiles.* In order to present values of the different metrics for all problems and all solvers considered, we have used the so-called performance profiles, as suggested in [19] (see also [44] and the references therein for the use of performance profiles in global derivative-free optimization). Performance profiles are depicted by the plot of a cumulative distribution function $\rho(\tau)$ representing a performance ratio for the different solvers. Let $\mathcal{S}$ be the set of solvers and $\mathcal{P}$ be the set of problems. Let $t_{p,s}$ denote the performance of the solver $s \in \mathcal{S}$ on the problem $p \in \mathcal{P}$ — lower values of $t_{p,s}$ indicate better performance. This performance ratio is defined by first setting $r_{p,s} = t_{p,s}/\min\{t_{p,s} : s \in \mathcal{S}\}$, for $p \in \mathcal{P}$ and $s \in \mathcal{S}$. Then, one defines $\rho_s(\tau) = (1/n_p)|\{p \in \mathcal{P} : r_{p,s} \leq \tau\}|$, where n_p is the number of test problems. Thus, the value of $\rho_s(1)$ is the probability of the solver s winning over the remaining ones. If we are only interested in determining which solver is the best (in the sense of winning the most), we compare the values of $\rho_s(1)$ for

all the solvers. At the other end, solvers with the largest probabilities $\rho_s(\tau)$ for large values of τ are the most robust ones (meaning the ones who solved the largest number of problems in $\mathcal{P}$).

5.3.2. *Purity metric.* The first metric considered by us is called *Purity* [4] and is used to compare the Pareto fronts obtained by different solvers. Again, let $\mathcal{S}$ be the set of solvers and $\mathcal{P}$ be the set of problems. Let $F_{p,s}$ denote the approximated Pareto front determined by the solver $s \in \mathcal{S}$ for problem $p \in \mathcal{P}$. Let also F_p denote an approximation to the true Pareto front of problem p, calculated by first forming $\cup_{s\in\mathcal{S}} F_{p,s}$ and then removing from this set any dominated points. The Purity metric consists then in computing, for solver $s \in \mathcal{S}$ and problem $p \in \mathcal{P}$, the ratio $c_{p,s}^{F_p}/c_{p,s}$, where $c_{p,s}^{F_p} = |F_{p,s} \cap F_p|$ and $c_{p,s} = |F_{p,s}|$. This metric is thus represented by a number $\bar{t}_{p,s} = c_{p,s}^{F_p}/c_{p,s}$ between zero and one. Higher values for $\bar{t}_{p,s}$ indicate a better Pareto front in terms of the percentage of nondominated points.

When using performance profiles to analyze the performance of the solvers measured by the Purity metric, we need to set $t_{p,s} = 1/\bar{t}_{p,s}$ (then, again, lower values of $t_{p,s}$ indicate better performance). Note that when a solver s is not able to obtain a single nondominated point in F_p, we obtain $\bar{t}_{p,s} = 0$, and thus $t_{p,s} = +\infty$, meaning that solver s was 'unable' to solve problem p.

The Purity metric has shown to be sensitive to the number and type of solvers considered in a comparison. In fact, when two 'similar' solvers produce similar approximated Pareto fronts, their performance under the Purity metric deteriorates significantly since many of these points will dominate each other. This effect will then let a third solver easily win among the three. Thus, we decided to only compare solvers in pairs when using the Purity metric. Still, since we have two solvers and a large number of problems, we present the results using performance profiles.

An additional difficulty is the inclusion of stochastic solvers in numerical comparisons. Since two different runs of such solvers may produce different solutions, we decided to make 10 runs for each stochastic solver on each single problem. From these 10 runs, we then selected the best and the worst run. The best run simply consists of the run that has the higher percentage of nondominated solutions when compared to the remaining ones. In a similar way, the worst run is selected as the one with the lowest percentage of nondominated points.

5.3.3. *Spread metric.* The second type of metric used by us tries to measure the extent of the spread achieved in the obtained Pareto front. Since we are interested in obtaining a set of solutions that spans the entire Pareto front, the proposed metric has to consider 'extreme' solutions. However, the true Pareto front is not known for the majority of the problems in the test set, and such extreme points must be determined from the obtained Pareto fronts. For the computation of these extreme points, in an attempt to have information as good as possible, we considered all runs of all solvers (including the ones for which the results are not reported in this paper).

We considered essentially two formulae for the spread metric, and let us start by the case $m = 2$. The first one consists of taking the maximum distance between points

$$\Gamma \;=\; \Gamma_{p,s} \;=\; \max_{i\in\{0,\ldots,N\}}\{d_i\}, \tag{3}$$

where d_i, for $i = 1, \ldots, N-1$, is the Euclidian distance between two consecutive points in the approximated Pareto front (of cardinal N). The quantities d_0 and d_N are the distances to the extreme points (see Figure 3 for an illustration). The second spread metric is the one proposed in [17], defined as

$$\Delta \;=\; \Delta_{p,s} \;=\; \frac{d_0 + d_N + \sum_{i=1}^{N-1} |d_i - \bar{d}|}{d_0 + d_N + (N-1)\bar{d}}, \tag{4}$$

where $\bar{d}$ is the average of the distances d_i, $i = 1, \ldots, N-1$.

The metric Γ measures the maximum distance between points in the approximated Pareto fronts, while the metric Δ indicates how well the points are distributed in the fronts. The value of Γ is positive. The value of Δ is nonnegative and may be zero when all the distances are equal to the average of the distances and the extreme points are included in the obtained Pareto front (i.e., when we have $d_0 = d_N = 0$).

We also need to use performance profiles when analyzing the results measured in terms of the Γ and Δ metrics since, again, one has the issue of having several solvers on many problems. In these cases, we have set $t_{p,s} = \Gamma_{p,s}$ or $t_{p,s} = \Delta_{p,s}$ depending on the metric considered.

The major drawback of the measures described in equations (3) and (4) is that they cannot be easily extended to problems with more than two objective functions. In fact, it is not possible to define without ambiguity what are two neighbor points in the approximated Pareto front when $m > 2$, which then

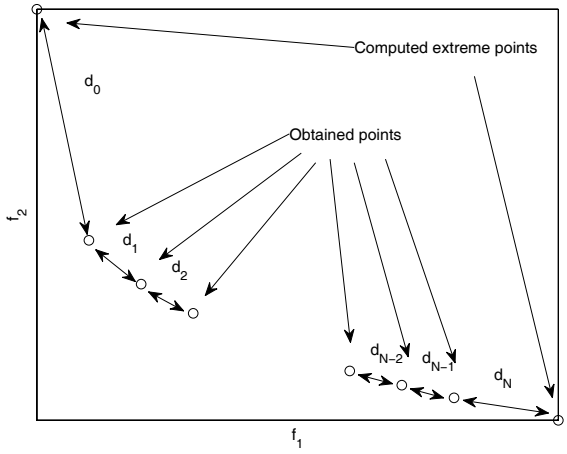

FIGURE 3. Distances between points in an approximated Pareto front, for $m = 2$, to be used by the metrics Γ and Δ.

poses difficulties to the computation of the distances d_i, $i = 0, \ldots, N$. The following formulae extend the concepts of (3) and (4) for higher dimensional objective spaces ($m > 2$)

$$\Xi \;=\; \Xi_{p,s} \;=\; \max_{j\in\{1,\ldots,m\}} \left(\max_{i\in\{0,\ldots,N\}} \{\delta_{i,j}\} \right), \tag{5}$$

where $\delta_{i,j} = (f_{i+1,j} - f_{i,j})$ (and we assume that the objective function values have been sorted by increasing order for each j), and

$$\Theta \;=\; \Theta_{p,s} \;=\; \max_{j\in\{1,\ldots,m\}} \left(\frac{\delta_{0,j} + \delta_{N,j} + \sum_{i=1}^{N-1} |\delta_{i,j} - \bar{\delta}_j|}{\delta_{0,j} + \delta_{N,j} + (N-1)\bar{\delta}_j} \right), \tag{6}$$

where $\bar{\delta}_j$, for $j = 1, \ldots, m$, is the average of the distances $\delta_{i,j}$, $i = 1, \ldots, N-1$. The quantities $\delta_{i,j}$, $i = 0, \ldots, N$, $j = 1, \ldots, m$, are depicted in Figure 4 for $m = 2$. Note that the metrics Ξ and Θ reduce, respectively, to Γ and Δ when $m = 2$ and one uses the infinity norm in these latter ones.

5.3.4. *Data profiles.* One possible way of assessing how well derivative-free solvers perform in terms of the number of evaluations is given by the so-called data profiles proposed in [37] for single optimization. Suppose there is only one objective function $f(x)$. For each solver, a data profile consists of a plot of the percentage of problems that are solved for a given budget of function evaluations. Let $h_{p,s}$ be the number of function evaluations required

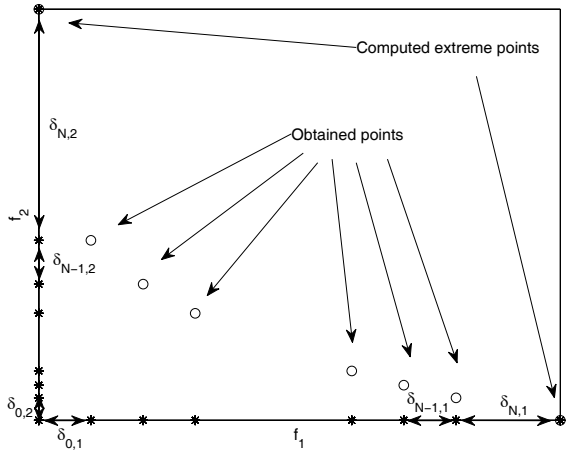

FIGURE 4. Distances between points in an approximated Pareto front to be used by the metrics Ξ and Θ when $m > 2$. For simplicity, we depict the case for $m = 2$.

for solver $s \in \mathcal{S}$ to solve problem $p \in \mathcal{P}$ (up to a certain accuracy). The data profile cumulative function is then defined by

$$d_s(\sigma) \;=\; \frac{1}{|\mathcal{P}|}|\{p \in \mathcal{P} : h_{p,s} \leq \sigma\}|. \tag{7}$$

A critical issue related to data profiles is when a problem is considered as being solved. The authors in [37] suggested that a problem is solved (up to some level ε of accuracy) when

$$f(x_0) - f(x) \;\geq\; (1-\varepsilon)(f(x_0) - f_L), \tag{8}$$

where x_0 is the initial guess and f_L is the best obtained objective function value among all solvers.

In the multiobjective case we need to consider instead a reference Pareto front F_p in order to determine whether a problem $p \in \mathcal{P}$ has been solved or not. Then, a solver s is said to solve problem p, up to an accuracy of ε, if the percentage of points obtained in the reference Pareto front F_p is equal to or greater than $1-\varepsilon$, i.e., if

$$\frac{|F_{p,s} \cap F_p|}{|F_p|/|\mathcal{S}|} \;\geq\; 1-\varepsilon, \tag{9}$$

where $F_{p,s}$ is the approximated Pareto front obtained by solver s on problem p. Note that in (9) the number of points in F_p is divided by the number

of solvers in $\mathcal{S}$ in an attempt to consider that all solvers are expected to contribute equally to the reference Pareto front.

The reference Pareto front can be computed in a number of possible ways depending on the choice of solvers (and on how long we let them run). To have meaningful results for our data profiles (in other words, a significant number of points in the numerator of (9)), we considered only the solvers in the set $\mathcal{S}$ chosen for comparison and a maximum number of 5000 function evaluations. The reference Pareto front is then computed by forming the union of the output fronts of the solvers and eliminating from there all the dominated points.

Following [37], we also divided σ in (7) by $n+1$ (the number of points needed to build a *simplex gradient*). Finally, note also that we did not consider any spread metric for data profiles since such metrics might not increase monotonically with the budget σ of function evaluations (a consequence of this fact would be that a problem could be considered unsolved after had been considered solved earlier in the running sequence).

6. Numerical experience

6.1. Comparing different DMS variants. The simplest possible version of direct multisearch (DMS), Algorithm 3.1, initializes the list of nondominated points with a singleton ($L_0 = \{(x_0; \alpha_0)\}$) and considers an empty search step in all iterations. This version is referred to as DMS(1). Since no initial guess has been provided along with the majority of the problems in our test set, it was our responsibility to define a default value for the initial point x_0 to be used in DMS(1). A reasonable (perhaps the most neutral) choice is $x_0 = (u + \ell)/2$.

Since DMS is competing against population based algorithms, it is desirable to equip it with the possibility of starting from an initial list different from a singleton. Such a list can be computed by first generating a set S_0 of points and then eliminating from those the dominated ones. Let S_0^{nd} denote the resulting set. The initial list is then given by $L_0 = \{(x; \alpha_0), x \in S_0^{nd}\}$. We considered the three following ways of generating S_0 (taking $|S_0| = n$ and $S_0 \subseteq \Omega = [\ell, u]$ in all of them):

- DMS(n,line), where S_0 is formed by equally spaced points on the line connecting ℓ and u, i.e., $S_0 = \{\ell + i/(n-1)(u-\ell), i = 0, \ldots, n-1\}$;

- DMS(n,lhs), where S_0 is generated using the Latin Hypercube Sampling strategy (see [35]). In this strategy, a multi-interval in $\mathbb{R}^n$ is partitioned into n multi-subintervals of equal dimension and points are uniformly randomly generated in each one of these multi-subintervals. The Latin Hypercube Sampling strategy generates random points by randomly permuting these points among the multi-subintervals. Our numerical implementation uses the MATLAB function `lhsdesign` from the Statistics Toolbox, followed by a shifting and scaling of the generated points in $[0,1]^n$ to the multi-interval $[\ell, u]$;
- DMS(n,rand), where the n elements of S_0 are uniformly randomly generated in the multi-interval $[\ell, u]$ (see, for instance, [39]). In this case, our numerical implementation uses the MATLAB function `rand`, followed by a shifting and scaling of the generated points in $[0,1]^n$ to the multi-interval $[\ell, u]$.

Algorithm 3.1 allows for a variety of ways of selecting the trial list from the filtered list. We chose to work with Algorithm 3.3, meaning that $L_{trial} = L_{filtered}$. The strategy chosen to manage the list consisted of always add points to the end of the list and move a point already selected as a poll center to the end of the list (at the end of an iteration).

For all the variants tested (DMS(1), DMS(n,line), DMS(n,lhs), and DMS(n,rand)), we chose[1] $D_k = [I_n \ -I_n]$, where I_n is the identity matrix of order n. Also, for all variants, we picked $\alpha_0 = 1$ and adopted a stopping criterion consisting of the step size α_k being lower than a predefined threshold $\alpha_\epsilon = 10^{-3}$ for all points in the list or a maximum of 20000 objective functions evaluations. The step size parameter was halved in unsuccessful iterations and maintained in successful ones.

Figures 5–7 depict performance profiles of the Purity metric for the four above mentioned variants of DMS. When a stochastic variant is involved (DMS(n,lhs) or DMS(n,rand)), the figures show the best and worst run comparisons as explained in Subsection 5.3.2. We can easily see that DMS(n,line) is the best variant, either in terms of efficiency or robustness, although the gains when comparing to DMS(1) are not overwhelming. In fact, reading the values of the curves of Figure 5 for $\tau = 1$, we can observe that both

[1]It is important to note that the result of Theorem 4.5 was derived under the assumption that the set of refining directions was dense in the unit sphere. We also tried in our numerical setting to use a poll set D_k equal to $[Q_k \ -Q_k]$ (where Q_k is an orthogonal matrix computed by randomly generating the first column) but the results were worse.

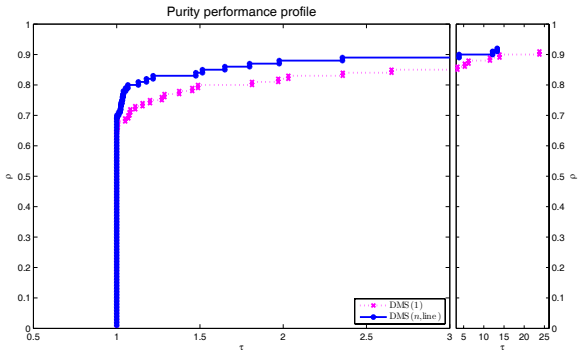

FIGURE 5. Comparing DMS(n,line) and DMS(1) based on performance profiles of the Purity metric.

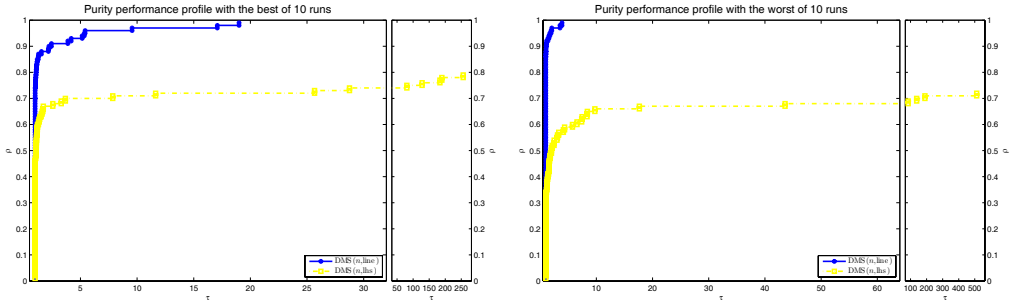

FIGURE 6. Comparing DMS(n,line) and DMS(n,lhs) based on performance profiles of the Purity metric.

DMS(n,line) and DMS(1) are able to attain the best metric value for close to 70% of the problems. In terms of robustness, and reading the same curves but now for large values of τ, we observe that both DMS(n,line) and DMS(1) are able to provide at least one nondominated point for slightly more than 90% of the problems. However, DMS(n,line) is significantly better than DMS(n,lhs) (see Figure 6) and DMS(n,rand) (see Figure 7), in terms of both efficiency and robustness, even when considering the best Pareto front obtained for 10 runs. For the sake of brevity, we do not provide pairwise comparisons among DMS(1), DMS(n,lhs), and DMS(n,rand).

The performance profiles of the spread metrics Ξ and Θ are given in Figure 8 for average values of the stochastic variants (the minimum and maximum values were also analyzed and do not change the conclusions stated next). In general, we can say that DMS(1) and DMS(n,line) exhibit a similar performance in terms of both metrics, better than the remaining ones regarding efficiency.

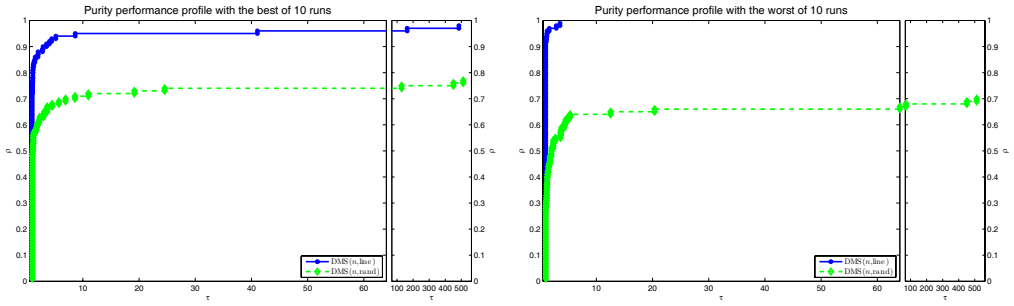

FIGURE 7. Comparing DMS(n,line) and DMS(n,rand) based on performance profiles of the Purity metric.

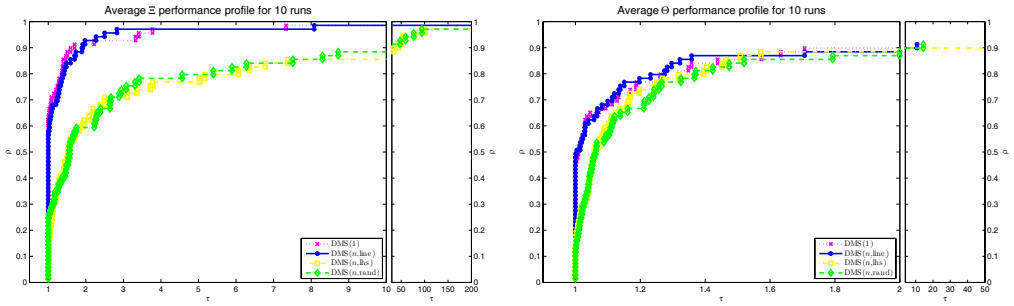

FIGURE 8. Comparing DMS(1), DMS(n,line), DMS(n,lhs), and DMS(n,rand) based on performance profiles of the Ξ (left) and Θ (right) metrics (taking average values for stochastic variants).

6.2. Comparing DMS to other solvers. In this section we present a comparison of the DMS(n,line) variant against the selected solvers AMOSA, BIMADS, and NSGA-II (C version). Because BIMADS can only deal with MOO problems with two objectives ($m = 2$), we will report results using the metrics Γ and Δ.

The selected solvers have been tested using their default parameters values except for the population size and number of iterations (generations). For AMOSA, we considered an initial temperature of 100, a final temperature of 2.5×10^{-6}, and a cooling factor of 0.6, yielding a total of 20650 objective functions evaluations. For NSGA-II (C version), we set a population of 100 points for 200 generations, yielding a total of 20000 objective functions evaluations. As mentioned before, for the DMS(n,line) solver, we imposed a stopping criterion consisting of $\alpha_k < \alpha_\epsilon = 10^{-3}$ for all points in the list or a maximum of 20000 objective functions evaluations. While AMOSA and NSGA-II (C version) always use the objective functions evaluations budget,

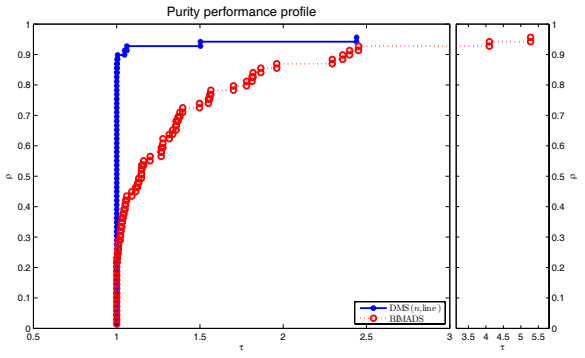

FIGURE 9. Comparing DMS(n,line) and BIMADS based on performance profiles of the Purity metric (only problems with two objective functions were considered).

the DMS(n,line) may stop earlier due to the convergence of all the points in the list to the requested step size accuracy. For BIMADS, a limit of 20000 objective function evaluations is also imposed. The BIMADS delta criteria was set to true meaning that the runs are also stopped when the step or mesh size parameter falls below a threshold (which is set in some problem dependent way).

From the performance profile of Figure 9, we can observe that, when using the Purity metric as a comparison measure, DMS(n,line) performs better than BIMADS in terms of efficiency, being about the same with respect to robustness. Figure 10 compares DMS(n,line) to AMOSA, also in terms of the Purity metric, being the former better for both the best and worst Pareto fronts obtained by AMOSA. Considering the performance profiles plotted in Figure 11 for the Purity metric as well, we can conclude that DMS(n,line) performs better than NSGA-II (C version) in terms of efficiency. Regarding robustness, DMS(n,line) slightly outperforms NSGA-II (C version) when considering its worst Pareto front, and slightly looses compared to its best Pareto front.

Figure 12 depicts the performance profiles using the spread metrics Γ and Δ (see (3) and (4)) for problems where $m = 2$ (again we only show the results for average values of the stochastic variants as the ones for minimum and maximum values do not affect our conclusions). One can observe that DMS(n,line) exhibits the best overall performance, although NSGA-II (C version) is slightly more efficient in terms of the Δ metric. Such conclusions are true mainly in terms of efficiency, since the four solvers seem to be equally

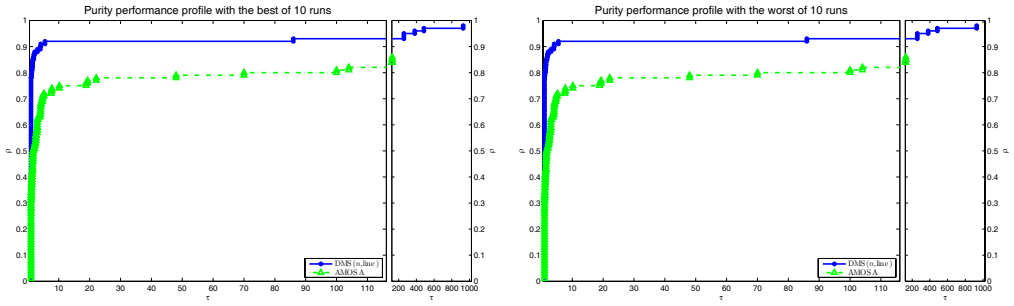

FIGURE 10. Comparing DMS(n,line) and AMOSA based on performance profiles of the Purity metric.

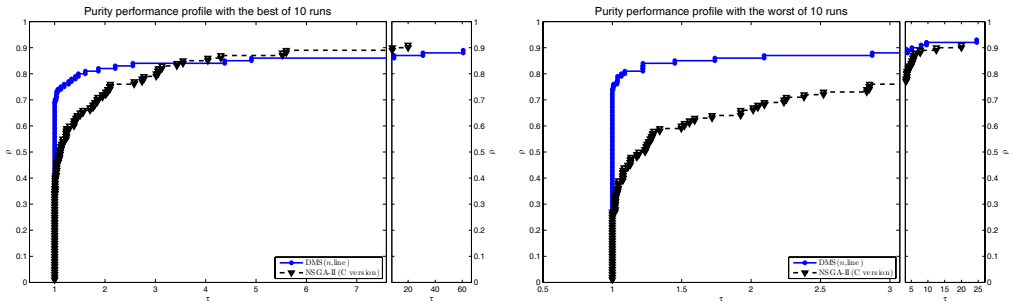

FIGURE 11. Comparing DMS(n,line) and NSGA-II (C version) based on performance profiles of the Purity metric.

robust under both metrics. These conclusions are also supported from the performance profiles of Figure 13 using the spread metrics Ξ and Θ (see (5) and (6)) and all problems ($m \geq 2$).

As previously mentioned, we did not use any metric which requires the knowledge of the true Pareto front. This set is known, however, for some of the problems, such as Problems ZDT1–ZDT4 and ZDT6. In Figures 14–16, we present plots depicting the approximated Pareto fronts for the selected solvers as well as the true ones, for Problems ZDT1, ZDT3, and ZDT6 (we omit here the cases of ZDT2 and ZDT4 since the corresponding plots are little informative).

When the true Pareto front is known, which is the case for these five problems (see `http://www.tik.ee.ethz.ch/sop/download/supplementary/testproblems`), one can also use the Purity metric to compare the approximated Pareto fronts to the true one. Table 2 presents such results for the 5 problems under consideration. The true Pareto front was computed using analytical formulas for $f_2(f_1)$ and an equally spaced grid of step 10^{-5} for f_1.

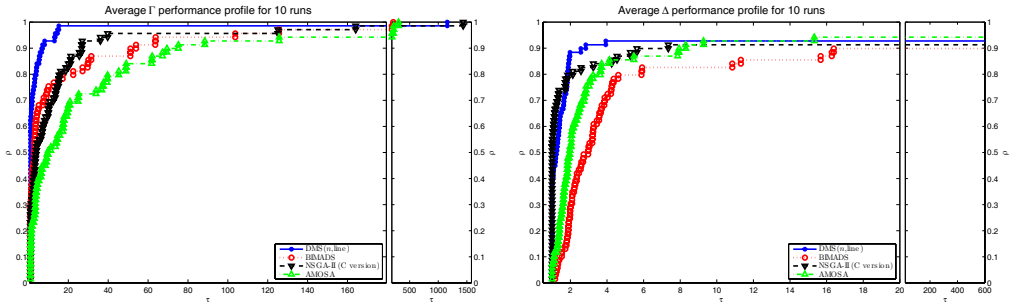

FIGURE 12. Comparing AMOSA, BIMADS, DMS(n,line), and NSGA-II (C version) based on performance profiles of the Γ (left) and Δ (right) metrics (taking average values for stochastic variants); only problems with two objective functions were considered.

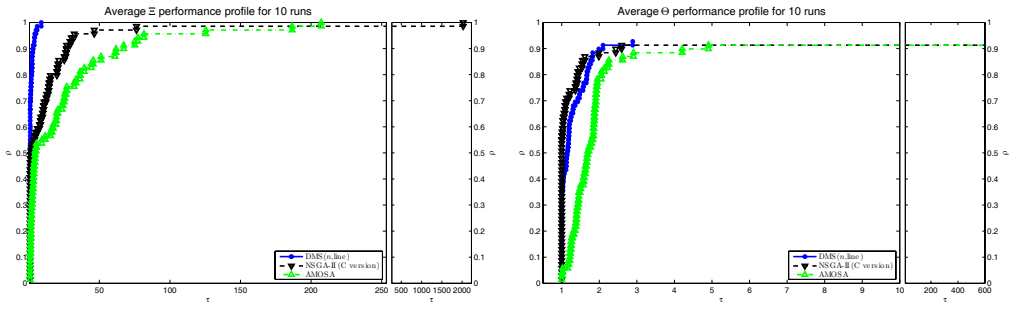

FIGURE 13. Comparing AMOSA, DMS(n,line), and NSGA-II (C version) based on performance profiles of the Ξ (left) and Θ (right) metrics (taking average values for stochastic variants).

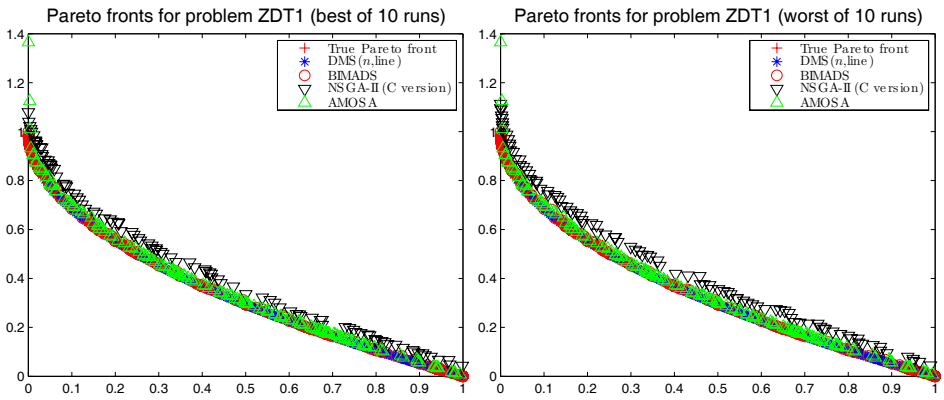

FIGURE 14. True and approximated Pareto fronts for ZDT1.

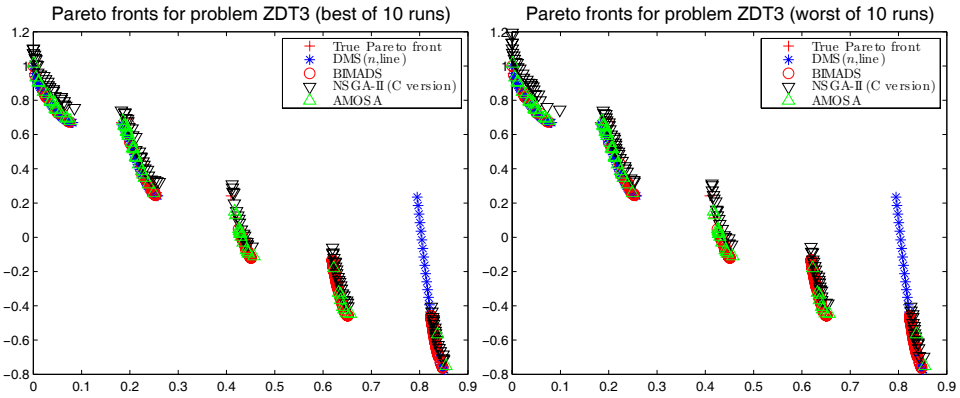

FIGURE 15. True and approximated Pareto fronts for ZDT3.

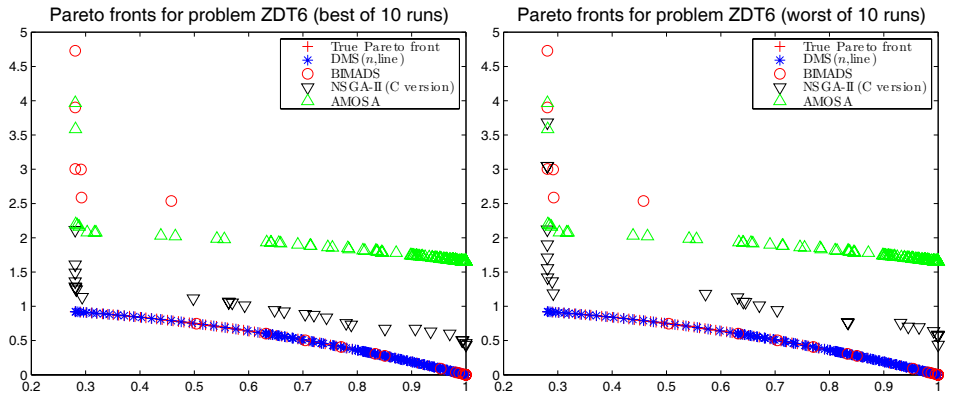

FIGURE 16. True and approximated Pareto fronts for ZDT6.

One can see, for problems ZDT1 and ZDT2, that at least 95% of the points in the approximated Pareto front computed by DMS(n,line) are not dominated by the true ones (up to a certain precision). BIMADS performed clearly the best for ZDT4. NSGA-II (C version) and AMOSA, on the other hand, are unable to obtain a single nondominated point for all problems. Finally, in Table 3 we provide the values of the spread metrics for the selected 4 solvers on these 5 problems.

So far we have only reported numerical results about the quality of the approximated Pareto fronts, giving no indication on the number of evaluations of the objective functions made by the different solvers. While NSGA-II (C version) and AMOSA took all the available budget (20000 overall evaluations) for all the problems in the test set, BIMADS and the different versions

Problem	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
DMS(n,line)	0.974	0.950	0.804	0.029	0.992
BIMADS	0.126	0.176	0.083	0.915	0.682
NSGA-II (C version, best)	0.000	0.000	0.000	0.000	0.000
NSGA-II (C version, worst)	0.000	0.000	0.000	0.000	0.000
AMOSA (best)	0.000	0.000	0.000	0.000	0.000
AMOSA (worst)	0.000	0.000	0.000	0.000	0.000

TABLE 2. The Purity metric values ($\bar{t}_{p,s}$, see Section 5.3.2) for true Pareto front *versus* selected solvers.

Problem	ZDT1	ZDT2	ZDT3	ZDT4	ZDT6
			Γ		
DMS(n,line)	0.044	0.013	0.537	0.143	3.808
BIMADS	0.043	0.035	0.262	0.151	1.791
NSGA-II (C version)	0.106	0.162	0.176	11.807	3.123
AMOSA	0.365	0.146	0.216	0.094	1.652
			Δ		
DMS(n,line)	0.337	0.277	0.864	0.645	1.027
BIMADS	1.205	1.209	1.128	1.917	1.122
NSGA-II (C version)	0.454	0.590	0.631	0.785	0.963
AMOSA	0.762	0.666	0.843	0.859	1.558

TABLE 3. The Γ and Δ metrics values for the selected solvers. (Only average values are provided for stochastic solvers.)

of DMS managed to solve a number of problems without exhausting the budget. In Figures 17 and 18 we provide data profiles for the four solvers under consideration, AMOSA, BIMADS, DMS(n,line), and NSGA-II (C version), on the biobjective subset of our test set, corresponding to four values of accuracy $\varepsilon = 0.5, 0.25, 0.1, 0.05$. We chose to report only results for the best versions of the stochastic solvers AMOSA and NSGA-II (C version). So, for instance, in Figure 17 (left), we can conclude that if a budget of 1000 objective functions evaluations is imposed, then both BIMADS and DMS(n,line) were able to solve around 54% of the problems in the sense of (9). These

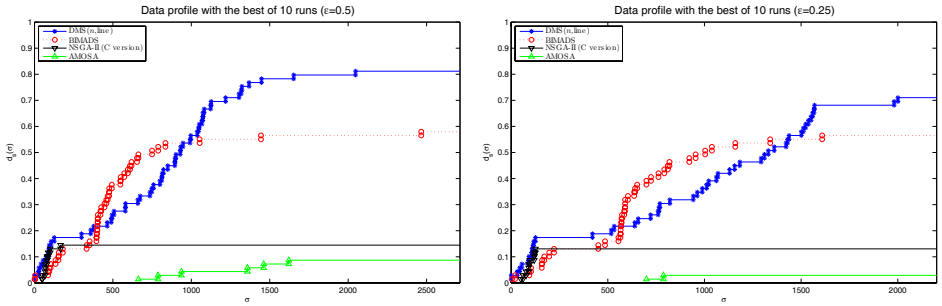

FIGURE 17. Data profiles for AMOSA, BIMADS, DMS(n,line), and the NSGA-II (C version) solvers ($\varepsilon = 0.5$ on the left and $\varepsilon = 0.25$ on the right).

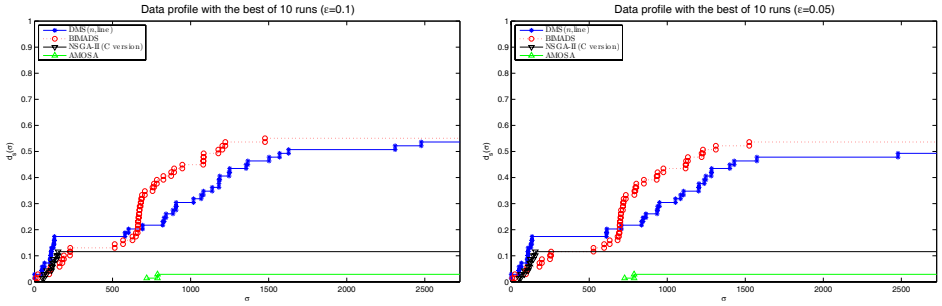

FIGURE 18. Data profiles for AMOSA, BIMADS, DMS(n,line), and the NSGA-II (C version) solvers ($\varepsilon = 0.1$ on the left and $\varepsilon = 0.05$ on the right).

two solvers seem clearly the most efficient ones for budgets up to 2500 evaluations, being BIMADS better for more accurate solutions and DMS(n,line) better for less accurate ones.

7. Conclusions

In this paper we introduced, analyzed, and tested a new algorithmic approach for multiobjective optimization (MOO) without derivatives. This approach has been called *direct multisearch* (DMS) since it naturally generalizes direct search (of directional type) from single to multiobjective optimization. The principles of DMS are extremely simple. Instead of updating a single point per iteration, it updates an iterate list of nondominated points. Iteration success is measured by changes in the iterate list. Each iteration of DMS includes provision for an optional search step. Polling is also applied, as in single optimization, at a selected point of the iterate list. Both steps can

add points to the iterate list, forming a filtered intermediate list, and there is significant flexibility in the way a trial list is formed from this filtered list.

The goal of DMS is to approximate the true (global, if possible) Pareto front, although theoretically one is only able to prove that there is a limit point in a stationary form of this front, as no aggregation or scalarization technique is incorporated in DMS. For this purpose, and to be able to state results for nonsmooth objective functions, we introduced in this paper the notion of a Clarke-Pareto stationary or critical point. Our results can be further generalized for discontinuous objective functions following the steps in [46].

Our numerical experience has shown that DMS is a highly competitive technique for derivative-free MOO. Although we tested a few variants of DMS, in particular in what the initial list of nondominated points is concerned, there are a number of possible strategies which can be incorporated in the DMS framework and lead to further possible improvements. In fact, the performance of DMS is already remarkably good for the simple implementations tested which do not incorporate any dissemination or spreading techniques particularly designed for the determination of the Pareto front. Such techniques could be easily fitted into DMS by means of an appropriate search step (such as a swarm search step; see [44, 45] for $m = 1$).

In addition, one could also study the introduction of quadratic polynomial interpolation models in DMS to possibly improve the efficiency of DMS in what concerns the search step (see [11] for what has been done in this respect in single optimization). One could also think of incorporating linear polynomial interpolation models (i.e., simplex gradients) to possibly improve the efficiency of an opportunistic DMS poll step (see [10, 12] for the single objective case).

DMS could be parallelized in many different ways, one obvious one being the parallelization of polling. In fact, complete polling for MOO requires a total of $m|D_k|$ function evaluations, which could be distributed among the available processors. A search step could also lead to various parallelization schemes.

Finally, if the user of our methodology has some type of preference for an objective function (or for some of them), there are several places where such intention can be specified. In fact, there is flexibility to show preference (for some of the objective functions) in the initialization of the iterate list, in the search step, in the reordering of the list and selection of the iterate point

(poll center), in the form of polling, and, finally, in the way the trial list is selected from the filtered list.

Appendix A. Appendix

To illustrate how Algorithm 3.1 works, we will now describe in detail its application to problem SP1 [24], defined by:

$$\begin{aligned} \min \quad & F(x) \equiv \left((x_1-1)^2+(x_1-x_2)^2, (x_1-x_2)^2+(x_2-3)^2\right) \\ \text{s.t.} \quad & -1 \leq x_1 \leq 5, \\ & -1 \leq x_2 \leq 5. \end{aligned}$$

The ways used to select the trial list from the filtered one and to order the iterate list will be the ones described in Section 6.1. No search step will be performed.

Initialization. Let us set the initial point $x_0 = (1.5, 1.5)$, corresponding to $(f_1(x_0), f_2(x_0)) = (0.25, 2.25)$, and initialize the step size parameter as $\alpha_0 = 1$. The step size will be maintained at successful iterations and halved at unsuccessful ones, which corresponds to setting $\gamma = 1$ and $\beta_1 = \beta_2 = \frac{1}{2}$. Set $\mathcal{D} = D = [I_2 \ -I_2]$, where I_2 stands for the identity matrix of dimension 2. Initialize the iterate list of nondominated points as $L_0 = \{(x_0; 1)\}$.

Iteration 0. The algorithm starts by selecting a point from L_0, in this case the only available, $(x_0; \alpha_0)$. Since no search step is performed, the feasible points in the poll set $P_0 = \{(1.5, 1.5) + (1, 0), (1.5, 1.5) + (0, 1), (1.5, 1.5) + (-1, 0), (1.5, 1.5) + (0, -1)\}$ are evaluated (the pink diamonds plotted in Iteration 0 of Figure 19 represent the corresponding function values). In this case, all the poll points were feasible, thus

$$L_{add} = \{((2.5, 1.5); 1), ((1.5, 2.5); 1), ((0.5, 1.5); 1), ((1.5, 0.5); 1)\}.$$

The nondominated points are filtered from $L_0 \cup L_{add}$, resulting in $L_{filtered} = \{((1.5, 1.5); 1), ((1.5, 2.5); 1)\}$. Only one of the evaluated poll points remained unfiltered (the green circle in Iteration 0 of Figure 19 represents its corresponding function value). According to Algorithm 3.3, L_{trial} will coincide with $L_{filtered}$. Since there were changes in L_0, the iteration is declared successful, and $L_1 = L_{trial} = L_{filtered}$, being the step size maintained. The function values corresponding to the points in L_1 are represented by yellow

squares in Iteration 0 of Figure 19. Note that we move the poll point to the end of the list, yielding the new order $L_1 = \{((1.5, 2.5); 1), ((1.5, 1.5); 1)\}$.

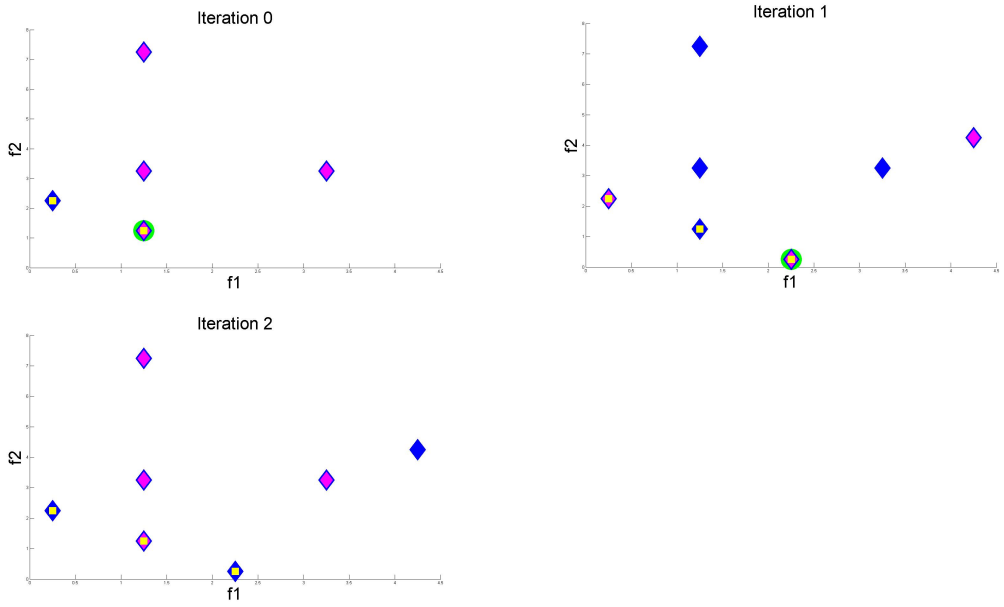

FIGURE 19. First three iterations of one instance of Algorithm 3.1, when applied to the MOO problem SP1. The blue diamonds represent the function values corresponding to all the evaluated points since the beginning of the optimization process. The pink diamonds represent the function values corresponding to the poll points evaluated at the current iteration. In green circles are represented the nondominated points which were evaluated at the current iteration, and in yellow squares the current iterate list of nondominated points.

Iteration 1. At the beginning of the new iteration, the algorithm selects a point from the two stored in L_1. Suppose the point $(x_1; \alpha_1) = ((1.5, 2.5); 1)$ was selected. In this case, the poll set $P_1 = \{(2.5, 2.5), (1.5, 3.5), (0.5, 2.5), (1.5, 1.5)\}$ is evaluated (again, the corresponding function values are represented by the pink diamonds in Iteration 1 of Figure 19). Note that two of the poll points correspond to the same function values. The list

$$L_{add} = \{((2.5, 2.5); 1), ((1.5, 3.5); 1), ((0.5, 2.5); 1), ((1.5, 1.5); 1)\}$$

is formed and $L_1 \cup L_{add}$ is filtered. Again, only one of the poll points was nondominated (the corresponding function values are represented in green in

Iteration 1 of Figure 19). Thus, the iteration was successful, the step size was maintained, and the new list is

$$L_2 = L_{trial} = L_{filtered} = \{((1.5, 2.5); 1), ((1.5, 1.5); 1), ((2.5, 2.5); 1)\}$$

(the corresponding function values are represented by the yellow squares in Iteration 1 of Figure 19). Again, we move the poll point (in this case, $((1.5, 2.5); 1)$) to the end of the list.

Iteration 2. The next iteration begins by selecting $(x_2; \alpha_2) = ((1.5, 1.5); 1)$ from the list L_2 (a previous poll center). After evaluating the corresponding poll points, all of them are dominated, thus $L_{trial} = L_2$, the iteration is declared as unsuccessful, the corresponding step size is halved, and $L_3 = \{((1.5, 1.5); 0.5), ((2.5, 2.5); 1), ((1.5, 2.5); 1)\}$ (the corresponding function values are represented by the yellow squares in Iteration 2 of Figure 19).

In Figure 20 we can observe the evolution of the optimization process after 10, 20, and 100 iterations. The number of points in the Pareto front is steadily increasing and, after 100 iterations, the corresponding curve is well defined.

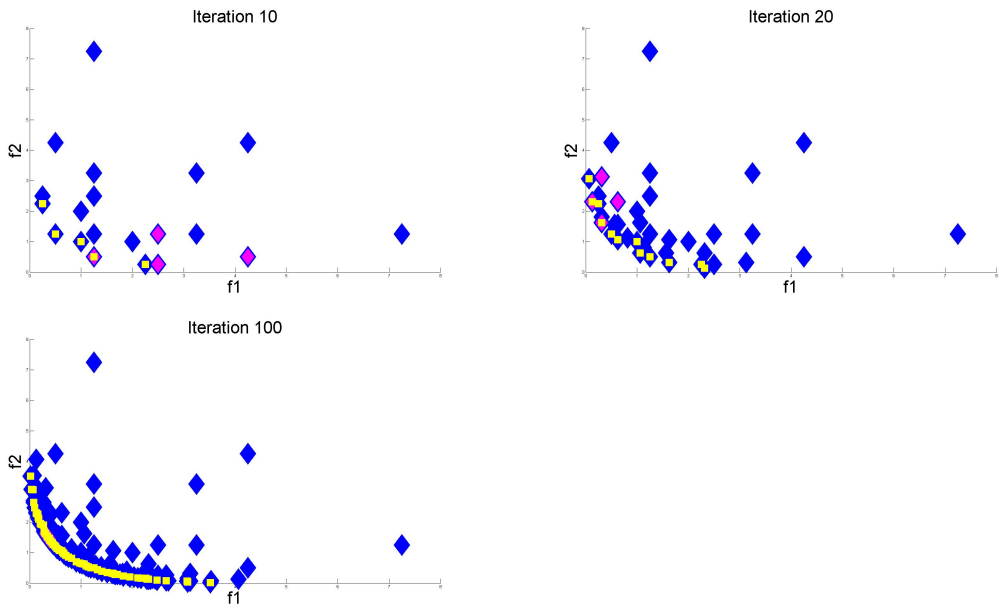

FIGURE 20. Iterations 10, 20, and 100 of one instance of Algorithm 3.1, when applied to the MOO problem SP1. See the caption of Figure 19 for details.

Acknowledgments. The authors would like to thank John Dennis (Rice University) and Jörg Fliege (University of Southampton) for interesting discussions on derivative-free multiobjective optimization.

References

[1] C. Audet and J. E. Dennis Jr. Analysis of generalized pattern searches. *SIAM J. Optim.*, 13:889–903, 2002.

[2] C. Audet and J. E. Dennis Jr. Mesh adaptive direct search algorithms for constrained optimization. *SIAM J. Optim.*, 17:188–217, 2006.

[3] C. Audet, G. Savard, and W. Zghal. Multiobjective optimization through a series of single-objective formulations. *SIAM J. Optim.*, 19:188–210, 2008.

[4] S. Bandyopadhyay, S. K. Pal, and B. Aruna. Multiobjective GAs, quantative indices, and pattern classification. *IEEE Transactions on Systems, Man, and Cybernetics – Part B: Cybernetics*, 34:2088–2099, 2004.

[5] S. Bandyopadhyay, S. Saha, U. Maulik, and K. Deb. A simulated anneling-based multiobjective optimization algorithm: AMOSA. *IEEE Transactions on Evolutionary Computation*, 12:269–283, 2008.

[6] F. Y. Cheng and X. S. Li. Generalized center method for multiobjective engineering optimization. *Engineering Optimization*, 31:641–661, 1999.

[7] F. H. Clarke. *Optimization and Nonsmooth Analysis*. John Wiley & Sons, New York, 1983. Reissued by SIAM, Philadelphia, 1990.

[8] C. A. Coello Coello and M. S. Lechuga. Mopso: A proposal for multiple objective particle swarm optimization. In *Congress on Evolutionary Computation (CEC'2002)*, volume 2, pages 1051–1056, Los Alamitos,USA, 2002. IEEE Computer Society.

[9] A. R. Conn, K. Scheinberg, and L. N. Vicente. *Introduction to Derivative-Free Optimization*. MPS-SIAM Series on Optimization. SIAM, Philadelphia, 2009.

[10] A. L. Custódio, J. E. Dennis Jr., and L. N. Vicente. Using simplex gradients of nonsmooth functions in direct search methods. *IMA J. Numer. Anal.*, 28:770–784, 2008.

[11] A. L. Custódio, H. Rocha, and L. N. Vicente. Incorporating minimum Frobenius norm models in direct search. *Comput. Optim. Appl.*, 46:265–278, 2010.

[12] A. L. Custódio and L. N. Vicente. Using sampling and simplex derivatives in pattern search methods. *SIAM J. Optim.*, 18:537–555, 2007.

[13] I. Das and J. E. Dennis Jr. A closer look at drawbacks of minimizing weighted sums of objectives for pareto set generation in multicriteria optimization problems. *Structural Optimization*, 14:63–69, 1997.

[14] I. Das and J. E. Dennis Jr. Normal-boundary intersection: a new method for generating the pareto surface in nonlinear multicriteria optimization problems. *SIAM J. Optim.*, 8:631–657, 1998.

[15] K. Deb. Multi-objective genetic algorithms: Problem difficulties and construction of test problems. *Evolutionary Computation*, 7:205–230, 1999.

[16] K. Deb, M. Mohan, and S. Mishra. Towards a quick computation of well-spread pareto-optimal solutions. In *Evolutionary Multi-Criterion Optimization (EMO 2003)*, volume 2632 of *Lecture Notes in Computer Science*, pages 222–236. 2003.

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

[18] K. Deb, A. Sinha, and S. Kukkonen. Multi-objective test problems, linkages, and evolutionary methodologies. In *Proceedings of the Genetic and Evolutionary Computation GECCO'06*, pages 1141–1148, New York, USA, 2006. ACM.

[19] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Math. Program.*, 91:201–213, 2002.

[20] C. M. Fonseca and P. J. Fleming. Multiobjective optimization and multiple constraint handling with evolutionary algorithms – Part I: A unified formulation. *IEEE Transactions on Systems, Man, and Cybernetics – Part A: Systems and Humans*, 28:26–37, 1998.

[21] R. Fourer, D. M. Gay, and B. W. Kernighan. A modeling language for mathematical programming. *Management Sci.*, 36:519–554, 1990.

[22] A. L. Hoffmann, A. Y. D. Siem, D. den Hertog, J. H. A. M. Kaanders, and H. Huizenga. Derivative-free generation and interpolation of convex pareto optimal IMRT plans. *Phys. Med. Biol.*, 51:6349–6369, 2006.

[23] S. Huband, L. Barone, L. While, and P. Hingston. A scalable multi-objective test problem toolkit. In C. A. Coello Coello et al., editor, *Evolutionary Multi-Criterion Optimization (EMO 2005)*, volume 3410 of *Lecture Notes in Computer Science*, pages 280–295. Springer-Verlag, Berlin, Germany, 2005.

[24] S. Huband, P. Hingston, L. Barone, and L. While. A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Transactions on Evolutionary Computation*, 10:477–506, 2006.

[25] C.-L. Hwang and A. S. Md. Masud. Multiple objective decision making, methods and applications: a state-of-the-art survey. Lecture Notes in Economics and Mathematical Systems, page 281. Springer-Verlag, Berlin, Germany, 1979.

[26] J. Jahn. *Introduction to the Theory of Nonlinear Optimization*. Springer-Verlag, Berlin, 1996.

[27] Y. Jin, M. Olhofer, and B. Sendhoff. Dynamic weighted aggregation for evolutionary multiobjective optimization: Why does it work and how? In L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke, editors, *Proceedings of Genetic and Evolutionary Computation Conference GECCO'01*, pages 1042–1049, San Francisco, USA, 2001. Morgan Kaufmann.

[28] D. Jones and T. Merhdad. *Practical Goal Programming*. Springer, 2010.

[29] J. D. Knowles and D. W. Corne. Approximating the nondominated front using the pareto archived evolution strategy. *Evolutionary Computation*, 8:149–172, 2000.

[30] T. G. Kolda, R. M. Lewis, and V. Torczon. Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Rev.*, 45:385–482, 2003.

[31] F. Kursawe. A variant of evolution strategies for vector optimization. In H. P. Schwefel and R. Männer, editors, *Parallel Problem Solving from Nature – PPSN I*, volume 496 of *Lecture Notes in Computer Science*, pages 193–197. Springer-Verlag, Berlin, Germany, 1990.

[32] G. Liuzzi, S. Lucidi, F. Parasiliti, and M. Villani. Multiobjective optimization techniques for the design of induction motors. *IEEE Transactions on Magnetics*, 39:1261–1264, 2003.

[33] A. Lovison. A synthetic approach to multiobjective optimization. arxiv: `http://arxiv.org/abs/1002.0093`, 2010.

[34] R. T. Marler and J. S. Arora. Survey of multi-objective optimization methods for engineering. *Structural Multidisciplinary Optimization*, 26:369–395, 2004.

[35] M. D. McKay, R. J. Beckman, and W. J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 42:55–61, 2000.

[36] K. Miettinen. *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, New York, 1999.

[37] J. J. Moré and S. M. Wild. Benchmarking derivative-free optimization algorithms. *SIAM J. Optim.*, 20:172–191, http://www.mcs.anl.gov/~more/dfo, 2009.
[38] T. Okabe, Y. Jin, M. Olhofer, and B. Sendhoff. On test functions for evolutionary multi-objective optimization. In *Parallel Problem Solving from Nature – PPSN VIII*, volume 3242 of *Lecture Notes in Computer Science*, pages 792–802. Springer-Verlag, Berlin, Germany, 2004.
[39] T. J. Santner, B. J. Williams, and W. I. Notz. *The Design and Analysis of Computer Experiments*. Springer-Verlag, New York, 2003.
[40] J. D. Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the First International Conference on Genetic Algorithms and their Applications*, pages 93–100, Hillsdale, USA, 1985. Lawrence Erlbaum Associates.
[41] R. Steuer. *Multiple criteria optimization: theory, computation and application.* John Wiley & Sons, New York, 1986.
[42] A. Suppapitnarm, K. A. Seffen, G. T. Parks, and P. Clarkson. A simulated annealing algorithm for multiobjective optimization. *Engineering Optimization*, 33:59–85, 2000.
[43] V. Torczon. On the convergence of pattern search algorithms. *SIAM J. Optim.*, 7:1–25, 1997.
[44] A. I. F. Vaz and L. N. Vicente. A particle swarm pattern search method for bound constrained global optimization. *J. Global Optim.*, 39:197–219, 2007.
[45] A. I. F. Vaz and L. N. Vicente. PSwarm: A hybrid solver for linearly constrained global derivative-free optimization. *Optim. Methods Softw.*, 24:669–685, 2009.
[46] L. N. Vicente and A. L. Custódio. Analysis of direct searches for discontinuous functions. Technical Report 09-38, Department of Mathematics, University of Coimbra, 2009.
[47] E. Zitzler, K. Deb, and L. Thiele. Comparison of multiobjective evolutionary algorithms: Empirical results. *Evolutionary Computation*, 8:173–195, 2000.

A. L. Custódio
Department of Mathematics, FCT-UNL, Quinta da Torre, 2829-516 Caparica, Portugal (alcustodio@fct.unl.pt).

J. F. A. Madeira
IDMEC-IST, TU-Lisbon, Av. Rovisco Pais, 1040-001 Lisboa, Portugal and ISEL, Rua Conselheiro Emídio Navarro, 1, 1959-007 Lisboa (jaguilar@dem.ist.utl.pt).

A. I. F. Vaz
Department of Systems and Production, University of Minho, Campus de Gualtar, 4710-057, Portugal (aivaz@dps.uminho.pt).

L. N. Vicente
CMUC, Department of Mathematics, University of Coimbra, 3001-454 Coimbra, Portugal (lnv@mat.uc.pt) — corresponding author.