SIAM J. OPTIM.
Vol. 26, No. 4, pp. 2091–2119
© 2016 Society for Industrial and Applied Mathematics

A METHOD FOR CONSTRAINED MULTIOBJECTIVE OPTIMIZATION BASED ON SQP TECHNIQUES*

JÖRG FLIEGE† AND A. ISMAEL F. VAZ‡

Abstract. We propose a method for constrained and unconstrained nonlinear multiobjective optimization problems that is based on an SQP-type approach. The proposed algorithm maintains a list of nondominated points that is improved both for spread along the Pareto front and optimality by solving single-objective constrained optimization problems. These single-objective problems are derived as SQP problems based on the given nondominated points. Under appropriate differentiability assumptions we discuss convergence to local optimal Pareto points. We provide numerical results for a set of unconstrained and constrained multiobjective optimization problems in the form of performance and data profiles, where several performance metrics are used. The numerical results confirm the superiority of the proposed algorithm against a state-of-the-art multiobjective solver and a classical scalarization approach, both in the quality of the approximated Pareto front and in the computational effort necessary to compute the approximation.

Key words. sequential quadratic programming, constrained multiobjective optimization, Pareto front

AMS subject classifications. 90C29, 90C55, 90C30

DOI. 10.1137/15M1016424

1. Introduction.

In this paper we address the following inequality and equality constrained multiobjective optimization problem:

$$\min_{x\in\Omega} \quad f(x) = (f_1(x), \ldots, f_m(x))^T \tag{1.1}$$

with

$$\Omega = \{x \in \mathbb{R}^n : g_j(x) \leq 0,\, j = 1, \ldots, p, \quad h_l(x) = 0, l = 1, \ldots, q\}\,, \tag{1.2}$$

where $f_i : \mathbb{R}^n \to \mathbb{R}$ $(i = 1, \ldots, m)$ are assumed to be twice continuously differentiable objective functions all bounded from below, and $g_j : \mathbb{R}^n \to \mathbb{R}$ $(j = 1, \ldots, p)$ and $h_l : \mathbb{R}^n \to \mathbb{R}$ $(l = 1, \ldots, q)$ are assumed to be once continuously differentiable constraint functions. We allow $p = 0$ and $q = 0$, in which case problem (1.1) is unconstrained.

Although applications for this type of problem can be found in many fields of engineering and science (see, for example, in [5, Part III] and [16]), implementations of available algorithms are somewhat limited to evolutionary algorithms or similar heuristics, or deterministic single point algorithms based on, for example, scalarization approaches (see, e.g., [7, 11, 13] or the various scalarization methods described and analyzed in [10]). The former lack a convergence proof and have a low rate of convergence, but usually provide an approximation to the Pareto front. The latter rely on a multistart procedure in which different scalarization parameters are used

*Received by the editors April 13, 2015; accepted for publication (in revised form) August 17, 2016; published electronically October 13, 2016.
http://www.siam.org/journals/siopt/26-4/M101642.html
Funding: This work was supported by COMPETE: POCI-01-0145-FEDER-007043 and FCT - Fundação para a Ciência e a Tecnologia within the Project Scope UID/CEC/00319/2013.
†School of Mathematics, University of Southampton, Southampton, SO17 1BJ, UK (j.fliege@soton.ac.uk).
‡ALGORITMI Research Center, University of Minho, Campus of Gualtar, 4710-057 Braga, Portugal (aivaz@dps.uminho.pt).

in different restarts to obtain an approximation to the Pareto front and consequently often have difficulties in finding an approximation to the Pareto front that is of good quality in all areas of the front. Our algorithm is independent of any scalarization parameters and does not need any complex initialization procedures.

Herein we propose an algorithm for constrained multiobjective optimization that tracks and updates a finite set of points X_k from one iteration k to the next, in contrast to standard algorithms for single-objective optimization where just one point $x_k \in \mathbb{R}^n$ is used. The proposed mechanism, based on two main stages, allows the algorithm to compute a high-quality approximation to the Pareto front. The first stage is used to enrich the set of points X_k with additional nondominated points, while the second stage is used to drive each element in the given set of points to optimality. Both stages rely on standard derivative based optimization subproblems, allowing the overall algorithm to enjoin many of the good properties of derivative based optimization algorithms.

The remainder of this paper is as follows. In the next section, we introduce some definitions and motivate the proposed algorithm. Within this motivation, we develop appropriate techniques on how to handle constraints, how to improve a nondominated set of points, and how to drive each element of a given set of points to local Pareto optimality. Section 3 describes optimization subproblems that will be used to compute search directions in both algorithmic stages, setting the stage for the description of the full algorithm in section 4. Convergence of the algorithm is addressed in section 5, while section 6 describes details of the implementation. Finally, section 7 presents extensive numerical results, before we conclude with section 8.

2. Definitions and motivation for the algorithm. When several objective functions are present, as it is in our case for problem (1.1), obtaining a point that simultaneously minimizes every function is usually not possible, since it is common that the objectives conflict with each other. There are several approaches to addressing such a problem, such as those reported in [5], for example. Our approach relies on the Pareto dominance concept for comparing two points. To briefly describe the concept, we make use of two orders induced by the cones

$$\mathbb{R}^m_+ = \{z \in \mathbb{R}^m : z_i \geq 0 \ (i = 1, \ldots, m)\} \text{ and } \mathbb{R}^m_{++} = \{z \in \mathbb{R}^m : z_i > 0 \ (i = 1, \ldots, m)\},$$

defined by

$$f(x) \preceq_f f(y) \quad \Leftrightarrow \quad f(y) - f(x) \in \mathbb{R}^m_+$$

and

$$f(x) \prec_f f(y) \quad \Leftrightarrow \quad f(y) - f(x) \in \mathbb{R}^m_{++}.$$

Given two points x, y in Ω, we say that $x \prec y$ (x dominates y) if and only if $f(x) \prec_f f(y)$. We further say that x weakly dominates y ($x \preceq y$) if and only if $f(x) \preceq_f f(y)$. A set of points $X \subseteq \Omega$ is said to be nondominated (or indifferent) when no point in the set is dominated by any other point in the set, i.e.,

$$\forall x \in X : \quad \nexists y \in X : \quad f(y) \prec_f f(x).$$

The concept of Pareto dominance is also used to characterize global and local optimality, by defining the *Pareto front*. For this, we need the following definition.

DEFINITION 2.1. *A point $x_* \in \Omega$ is said to be a global Pareto minimizer of f in Ω if there does not exist a $y \in \Omega$ with $f(y) \preceq_f f(x_*)$ (i.e., $y \preceq x_*$) and $f(x_*) \neq f(y)$. If there exists a neighborhood $\mathcal{N}(x_*)$ of x_* such that the previous property holds in $\Omega \cap \mathcal{N}(x_*)$, then the point x_* is called a local Pareto minimizer of f.*

The global Pareto front is the image of the set of global Pareto minimizers under the function f. Likewise, the Pareto front is the image of the set of all Pareto minimizers, global or local under the function f.

The goal of our algorithm is to determine a set of Pareto optimal points of f, i.e., to determine a set X^* where each $x_* \in X^*$ is such that there exists a neighborhood $\mathcal{N}(x_*)$ of x_* with

$$\nexists y \in \Omega \cap \mathcal{N}(x_*): \quad f(y) \preceq_f f(x_*) \text{ and } f(y) \neq f(x_*)$$

and where X^* serves as a good approximation of the full set of Pareto points.

We next provide a necessary condition for Pareto optimality, i.e., a notion of criticality. The criticality definition presented herein is an extension of the criticality definition for unconstrained multiobjective optimization used in [14]. This criticality condition for a given feasible point $\bar{x}$ can be seen as the nonexistence of a feasible direction $d \in \mathbb{R}^n$ along which all the objective functions are decreasing. We can formalize this as follows. Define the *linearized cone* at $\bar{x}$ as usual, i.e.,

$$L(\bar{x}) := \{d \in \mathbb{R}^n : \nabla g_j(\bar{x})^T d \leq 0,\ j \in \mathcal{A}(\bar{x}), \quad \nabla h_l(\bar{x})^T d = 0,\ l = 1, \ldots, q\},$$

where $\mathcal{A}(\bar{x}) = \{j \in \{1, \ldots, p\} : g_j(\bar{x}) = 0\}$ is the index set of the inequality constraints active at $\bar{x}$. Then we say that a point is *Pareto critical* if for all $d \in L(\bar{x})$ there exists an $i = i(d) \in \{1, \ldots, m\}$ such that

$$\nabla f_i(\bar{x})^T d \geq 0.$$

Likewise, we call a Pareto critical point $\bar{x}$ *strongly Pareto critical* if for all $d \in L(\bar{x})$ we either have $\nabla f_i(\bar{x})^T d = 0$ for all objectives i, or there exists an index $i = i(d) \in \{1, \ldots, m\}$ such that

$$\nabla f_i(\bar{x})^T d > 0.$$

Denote the set of local Pareto points by P, the set of all Pareto critical points by PC, and the set of all strongly Pareto critical points by SC. Clearly, we have

$$SC \subseteq PC.$$

The relationship between Pareto points and Pareto critical points is given in the next theorem.

THEOREM 2.2. *Let $\bar{x}$ be a local Pareto point, and let Abadie's constraint qualification hold at $\bar{x}$. Then $\bar{x}$ is also a Pareto critical point. Thus, if Abadie's constraint qualification holds at all Pareto points, we have*

$$P \subseteq PC.$$

Proof. Let $\bar{x}$ be a local Pareto point. Suppose that $\bar{x}$ is not Pareto critical. Then there exists a $d \in L(\bar{x})$ with $\nabla f_i(\bar{x})^T d < 0$ for all objectives i. Under Abadie's constraint qualification, $L(\bar{x})$ is equal to the tangent cone of the set of feasible points

at $\bar{x}$, and thus there exist sequences $x_k \in \Omega$ and $t_k > 0$ such that $d = \lim_{k\to\infty}(x_k - \bar{x})/t_k$, $\lim_{k\to\infty} t_k = 0$, and $\lim_{k\to\infty} x_k = \bar{x}$. Accordingly, $\nabla f_i(\bar{x})^T(x_k - \bar{x}) < 0$ for all objectives i and all k large enough. Now $f_i(x_k) = f_i(\bar{x}) + \nabla f_i(\xi_{i,k})^T(x_k - \bar{x})$ with $\xi_{i,k}$ a point on the line segment between x_k and $\bar{x}$. A standard continuity argument shows that $f_i(x_k) < f_i(\bar{x})$ for all objectives and all sufficiently large k, a contradiction. ∎

An elementary example shows that not all Pareto points are strongly Pareto critical: in the unconstrained case, let $m = 2$, $n = 1$, and $f_1(x) = x^2$ as well as $f_2(x) = (x-1)^2$. Obviously, $P = [0, 1]$. For $x = 0 \in P$, the Jacobian of $f = (f_1, f_2)$ is $[0, -2]$, and so clearly $x \notin SC$, as $\nabla f_2(x)d = -2d < 0$ for any $d > 0$. We will show below that, under certain conditions, one can show that missing critical points of particular objective functions is the worst that can happen when considering strongly Pareto critical points instead of Pareto critical points.

Let C be the set of points which are critical in the classical sense for at least one of the objective functions; i.e., $x \in \Omega$ is critical if and only if there exists an $i \in \{1, \ldots, m\}$ such that for all directions $d \neq 0$ with $d \in L(x)$ we have $\nabla f_i(x)^T d \geq 0$. In case Abadie's constraint qualification holds at the optimal point x, then x is a KKT point. This shows that if Abadie's constraint qualification holds at all points in C, then the set C is closed, as the set of KKT points is closed, and C is a finite union of such sets.

We will make use of the set of *Geoffrion-optimal points*, characterized in the usual way, i.e.,

$$G = \left\{ x \in \mathbb{R}^n \mid \exists w \in \mathbb{R}^m_{++} : x \in \arg\min_{x\in\Omega} \sum_{i=1}^m w_i f_i \right\}.$$

Pareto optimality and criticality are related as follows.

THEOREM 2.3. *Let $x \in \Omega$ be a point such that there exists a $w \in \mathbb{R}^m_{++}$ such that x is a KKT point for the problem of minimizing $\sum_{i=1}^m w_i f_i$ over Ω. Then x is also Pareto-critical.*

Proof. Farkas's lemma provides us with $\sum_{i=1}^m w_i \nabla f_i(x)^T d \geq 0$ for all $d \in L(x)$, which means that for each $d \in L(x)$ there must be an i with $\nabla f_i(x)^T d \geq 0$, i.e., Pareto criticality holds at x. ∎

THEOREM 2.4. *If Abadie's constraint qualification holds at each point in $G \setminus C$, we have*

$$G \setminus C \subseteq SC \subseteq PC.$$

Moreover, if all functions f_i, g_j are convex and h is affine linear, we have

$$G \setminus C \subseteq P \setminus C \subseteq \mathrm{cl}(G) \setminus C = \mathrm{cl}(G) \setminus \mathrm{cl}(C) \subseteq \mathrm{cl}(G \setminus C) \subseteq \mathrm{cl}(SC) \subseteq \mathrm{cl}(PC).$$

Proof. Now let $x \in G \setminus C$ and assume that Abadie's constraint qualification holds at x. Then there exists a $w \in \mathbb{R}^m_{++}$ such that x minimizes $\sum_{i=1}^m w_i f_i$ over Ω. Let $d \in L(x)$. Then

$$\sum_{i=1}^m w_i (\nabla f_i(x))^T d \geq 0.$$

As such, we either have $(\nabla f_i(x))^T d = 0$ for each term of the sum, or there must be at least one index $\bar{i}$ with $(\nabla f_{\bar{i}}(x))^T d > 0$. As a consequence, $x \in SC$.

For the rest of the proof, note that

$$G \subseteq P,$$

and, due to Theorem 2.2, thus also $G \subseteq PC$ (see also, e.g., [15]). In addition, if all functions f_i, g_j are convex and h is affine linear, we have

$$P \subseteq \mathrm{cl}(G);$$

see, e.g., [20, p. 76]. ∎

Due to Theorems 2.3 and 2.4 we can thus, under appropriate assumptions, concentrate on the sets PC and SC, and we will make use of this in the next subsection. Additionally, the proposed algorithm relies on two different mechanisms that are described in the two subsections that follow. The first one consists of improvements of a set of nondominated points in the sense that points will be further spread along the Pareto front (instead of being clustered in a particular area), while the second consists in driving a given set of nondominated points to local Pareto optimality. Both mechanisms can be seen as improvement techniques for a particular notion of Pareto optimality that also handles constraints, to be introduced in the next subsection.

2.1. Handling constraints. Each inequality and equality constraint can be seen as an additional objective to optimize. While objective functions are to be minimized, constraint functions must achieve a specific value or stay in a certain range of values. Defining

$$c^+ = \max\{0, c\},$$

for arbitrary real numbers $c \in \mathbb{R}$ we have that each feasible point x satisfies

$$g_j^+(x) = 0, \quad j = 1, \ldots, p, \quad \text{and} \quad |h_l(x)| = 0, \quad l = 1, \ldots, q.$$

For constrained optimization problems we further extend the concept of weak Pareto dominance by saying that x weakly dominates y ($x \preceq y$) if and only if

$$\bar{f}(x) \preceq_{\bar{f}} \bar{f}(y) \quad \Leftrightarrow \quad \bar{f}(y) - \bar{f}(x) \in \mathbb{R}_+^{(m+p+q)}$$

with

$$\bar{f}(x)^T := (f(x)^T, \Phi(x)^T) \quad \text{and} \quad \Phi(x)^T := (g^+(x)^T, |h(x)^T|).$$

The strong dominance concept is defined in a similar way.

Clearly, for points x and y feasible for (1.1)–(1.2) we have that

$$f(x) \preceq_f f(y) \Leftrightarrow \bar{f}(x) \preceq_{\bar{f}} \bar{f}(y),$$

while a point infeasible for (1.1)–(1.2) never weakly dominates a point feasible for this problem, whilst a feasible point can weakly dominate an infeasible point.

This Pareto dominance concept extension for constrained multiobjective optimization allows the proposed algorithm to deal with points infeasible for (1.1)–(1.2) during the optimization process.

In what follows we need the concept of a merit function that will measure the progress over a sequence of iterates. We will use the following variant of the classical ℓ_1 merit function:

$$\phi(x, \sigma) = \sum_{i=1}^{m} f_i(x) + \sigma \left(\sum_{j=1}^{p} (g_j(x))^+ + \sum_{l=1}^{q} |h_l(x)| \right), \tag{2.1}$$

where, as usual, σ is a positive penalty parameter.

In a single-objective case, in which we want to minimize, say, the ith objective function ($i \in \{1, \ldots, m\}$), we will use

$$\phi_i(x, \sigma) = f_i(x) + \sigma \sum_{j=1}^{p} (g_j(x))^+ + \sigma \sum_{l=1}^{q} |h_l(x)|,$$

with σ playing the same role as above.

Note that ϕ as well as all ϕ_i are usually nondifferentiable, although directional derivatives $\phi'(x, \sigma; d)$ at x in any direction $d \neq 0$ exist. See Lemma 16.4 in [3] for additional details.

2.2. Improving a nondominated set of points. Let X be a set of nondominated points, with points in X feasible or infeasible for (1.1)–(1.2), and let $d \in \mathbb{R}^n$ be a direction of descent for at least one objective function at one point in X that, locally, forces feasibility; i.e., given some $x \in X$, there exists an i such that

$$\nabla f_i(x)^T d < 0, \tag{2.2a}$$

$$g_j(x) + \nabla g_j(x)^T d \leq 0, \quad j = 1, \ldots, p, \tag{2.2b}$$

$$h_l(x) + \nabla h_l(x)^T d = 0, \quad l = 1, \ldots, q, \tag{2.2c}$$

and there exists $\bar{t} > 0$ such that for all $t \in]0, \bar{t}]$ we have

$$\phi_i(x + td, \sigma) \leq \phi_i(x, \sigma) + \mu t \phi_i'(x, \sigma; d) \tag{2.2d}$$

for some $\mu \in]0, 1[$ and some $\sigma > 0$.

THEOREM 2.5. *Given $x_k \in X$ and a descent direction d satisfying equations* (2.2)*, there exists a $t_0 > 0$ such that*

$$\bar{f}(x_k) \not\preceq_{\bar{f}} \bar{f}(x_k + td) \quad \forall t \in]0, t_0], \tag{2.3}$$

i.e., $x_k + td$ is not dominated by x_k. Define $a \in \mathbb{R}^p$ and $b \in \mathbb{R}^q$ by

$$a_j := \begin{cases} 1 & : \quad g_j(x) = 0, \\ 0 & : \quad \textit{otherwise}, \end{cases} \qquad b_l := \begin{cases} 1 & : \quad h_l(x) = 0, \\ 0 & : \quad \textit{otherwise} \end{cases}$$

($j = 1, \ldots, p$, $l = 1, \ldots, q$), and $r := (a^T, b^T)^T$. Then for all $\epsilon > 0$ there exists a $t_0 > 0$ such that we have $\Phi(x_k) - \Phi(x_k + td) + \epsilon r \in \mathbb{R}^{p+q}_+$ for all $t \in]0, t_0]$.

Proof. The proof for (2.3) follows from the differentiability of f_i and (2.2a), since we have

$$\lim_{t \to 0} \frac{f_i(x_k + td) - f_i(x_k)}{t} = \nabla f_i(x_k)^T d < 0,$$

implying that there exists $t_1 > 0$ such that

$$f_i(x_k + td) < f_i(x_k) \quad \forall t \in]0, t_1]$$

holds, and by this we conclude that $\bar{f}(x_k) \not\preceq_{\bar{f}} \bar{f}(x_k + td)$.

We prove the second statement by considering each constraint type individually for a given $j = 1, \ldots, p$ or $l = 1, \ldots, q$.

- Suppose $g_j(x_k) > 0$. Since we have $\nabla g_j(x_k)^T d \leq -g_j(x_k)$, we arrive at $\nabla g_j(x_k)^T d < 0$, and by the differentiability of g_j we see that there exists a $t_2 > 0$ such that $g_j(x_k + td) \leq g_j(x_k)$ for all $t \in]0, t_2]$ (d is a descent direction for g_j), i.e., we have $g_j^+(x_k + td) \leq g_j^+(x_k)$ for all $t \in]0, t_2]$.
  If $g_j(x_k) < 0$, we have $g_j(x_k + td) < 0$ for all $t > 0$ smaller than some $t_3 > 0$, and thus $g_j^+(x_k + td) = g_j^+(x_k) = 0$ for all $t \in]0, t_3]$.
  Finally, suppose $g_j(x_k) = 0$, which results in

$$\lim_{t \to 0} \frac{g_j(x_k + td)}{t} \leq 0.$$

  If the limit attains a negative value, then we have $g_j(x_k + td) < 0$ for all $t \in]0, t_4]$ and $g_j^+(x_k + td) = g_j^+(x_k) = 0$ for all $t \in]0, t_4]$. It remains to consider the case where $g_j(x_k + td)$ approaches zero from above, i.e., $\lim_{t \to 0} g_j(x_k + td) = 0$ and $g_j(x_k + td) > 0$. For this case, recall that for all ϵ there exists a $t_5 > 0$ such that $|g_j(x_k + td)| < \epsilon$ for all $t \in]0, t_5]$, resulting in $g_j^+(x_k + td) - g_j^+(x_k) < \epsilon$ for all $t \in]0, t_5]$.
- As above, we consider three cases for the equality constraints as well. From (2.2c), we have $\nabla h_l(x_k)^T d = -h_l(x_k)$, resulting in

$$\lim_{t \to 0} \frac{h_l(x_k + td) - h_l(x_k)}{t} = -h_l(x_k).$$

  If $h_l(x_k) > 0$, then there exists a $t_6 > 0$ such that $h_l(x_k + td) - h_l(x_k) < 0$ for all $t \in]0, t_6]$, resulting in $h_l(x_k + td) < h_l(x_k)$. We may further assume, without loss of generality, that t_6 is such that $h_l(x_k + td) \geq 0$ for all $t \in]0, t_6]$, resulting in $|h_l(x_k + td)| < |h_l(x_k)|$ for all $t \in]0, t_6]$. If $h_l(x_k) < 0$, then there exists a $t_7 > 0$ such that $h_l(x_k + td) - h_l(x_k) > 0$ for all $t \in]0, t_7]$, resulting in $h_l(x_k + td) > h_l(x_k)$. We may further assume, without loss of generality, that t_7 is such that $h_l(x_k + td) \leq 0$ for all $t \in]0, t_7]$, which results in $|h_l(x_k + td)| < |h_l(x_k)|$ for all $t \in]0, t_7]$.
  Finally, if $h_l(x_k) = 0$, then

$$\lim_{t \to 0} \frac{h_l(x_k + td)}{t} = 0,$$

  and for all $\epsilon > 0$ there exists $t_8 > 0$ such that $|h_l(x_k + td)| < \epsilon$ for all $t \in]0, t_8]$, resulting in $|h_l(x_k + td)| - |h_l(x_k)| < \epsilon$ for all $t \in]0, t_8]$.

By considering all cases above and taking $t_0 = \min\{t_1, t_2, t_3, t_4, t_5, t_6, t_7, t_8\}$, the result follows. ∎

The computation of a particular search direction that satisfies conditions (2.2) is described in the next section.

By taking each point in X and computing, for each individual objective function, a search direction d that locally improves this point with respect to that objective, we can update the set X by all newly computed nondominated points; i.e., the set X is enriched with new nondominated points. It is easy to see that this procedure is indeed a heuristic to *spread* feasible points along the Pareto front, instead of allowing the set X to cover a small part of the front only: let x_k and $x_k + td$ be feasible for (1.1), let x_k be nonstationary for the ith objective function, and let $t \in]0, \bar{t}]$. Then we have from (2.2d) that $\|f(x_k) - f(x_k + td)\| \geq -\mu t \nabla f_i(x_k)^T d > 0$, i.e., as long as $x_k + td$ is not dominated by any other point in the set X, the vector $f(x_k + td)$ is not close to $f(x_k)$, and in this sense the spread of the set X is improved.

2.3. Driving a nondominated set of points to local Pareto optimality. The previous section provided us with a strategy to enrich a given set X with new nondominated points. However, there is of course no guarantee that after this enrichment the set X is formed of Pareto points only. While the search direction d considered in the previous section is a sufficient descent direction for at least one objective function, there is no control over what happens with the remaining objective function values. In order to drive a point $x \in X$ to Pareto optimality we now need to compute a search direction d that satisfies the following conditions.

There exists an i_0 such that

$$\nabla f_{i_0}(x_k)^T d < 0, \tag{2.4a}$$

$$\nabla f_i(x_k)^T d \leq 0, \quad i \in \{1, \ldots, m\} \setminus \{i_0\}, \tag{2.4b}$$

$$g_j(x_k) + \nabla g_j(x_k)^T d \leq 0, \quad j = 1, \ldots, p, \tag{2.4c}$$

$$h_l(x_k) + \nabla h_l(x_k)^T d = 0, \quad l = 1, \ldots, q, \tag{2.4d}$$

and there exists an $\bar{t}$ such that for all $t \in]0, \bar{t}\,]$,

$$\phi(x_k + td, \sigma) \leq \phi(x_k, \sigma) + \mu t \phi'(x_k, \sigma; d) \tag{2.4e}$$

with some $\mu \in]0, 1[$ and some $\sigma > 0$.

THEOREM 2.6. *Given $x_k \in X$ and a descent direction d satisfying equations* (2.4)*, define $w \in \mathbb{R}^m$ by*

$$w_i := \begin{cases} 1 & : \quad \nabla f_i(x_k)^T d = 0, \\ 0 & : \quad \textit{otherwise} \end{cases}$$

$(i = 1, \ldots, m)$, and $\bar{r} \in \mathbb{R}^{m+p+q}$ by $\bar{r} := (w^T, a^T, b^T)^T$, with a, b defined as in Theorem 2.5*. Then for all $\epsilon > 0$ there exists a $t_0 > 0$ such that*

$$\bar{f}(x_k + td) \preceq_{\bar{f}} \bar{f}(x_k) + \epsilon \bar{r} \quad \forall t \in]0, t_0]. \tag{2.5}$$

Proof. The proof is similar to that of Theorem 2.5. Equation (2.4a) results in $f_{i_0}(x_k + td) < f_{i_0}(x_k)$. Equations (2.4c) and (2.4d) result in $\Phi(x_k) - \Phi(x_k + td) + \epsilon r \in \mathbb{R}^{p+q}_+$. From (2.4b) we obtain that for $i = \{1, \ldots, m\} \setminus \{i_0\}$,

$$\lim_{t \to 0} \frac{f_i(x_k + td) - f_i(x_k)}{t} = \nabla f_i(x_k)^T d \leq 0.$$

If $\nabla f_i(x_k)^T d < 0$, then there exists a $t_9 > 0$ such that

$$f_i(x_k + td) < f_i(x_k) \quad \forall t \in]0, t_9].$$

But if $\nabla f_i(x_k)^T d = 0$, then for all $\epsilon > 0$ there exists a $t_{10} > 0$ such that

$$|f_i(x_k + td) - f_i(x_k)| \leq \epsilon \quad \Rightarrow \quad f_i(x_k + td) - f_i(x_k) \leq \epsilon,$$

and the proof follows. ∎

A particular technique to compute the search direction d that satisfies equations (2.4) is described in the next section.

3. The search directions. There are several possibilities to obtain a search direction with properties described in equations (2.2) and (2.4). The strategy described herein consists in using quadratic approximations to the objective functions and linear approximations to the equality and inequality constraints, as in an SQP method.

A search direction in equations (2.2) with respect to the ith objective function can be obtained by solving the following quadratic optimization problem:

$$\begin{aligned} &\min_{d\in\mathbb{R}^n} && \nabla f_i^T(x_k)d + \frac{1}{2}d^T H_i d \\ &\text{subject to (s.t.)} && g_j(x_k) + \nabla g_j(x_k)^T d \le 0, \quad j = 1,\dots,p, \\ &&& h_l(x_k) + \nabla h_l(x_k)^T d = 0, \quad l = 1,\dots,q, \end{aligned} \tag{3.1}$$

where H_i is a symmetric positive definite matrix.

The following results relate problem (3.1) to the search direction (2.2) and the merit function in use. Note that item 5 of the theorem below, showing that the solution of the quadratic program above is a direction of descent for the merit function ϕ_i, is a direct consequence of known results for SQP methods.

LEMMA 3.1. *For each $i = 1,\dots,m$, let $\bar{d}$ be an optimal solution to problem* (3.1). *Then $\bar{d}$ satisfies the following conditions:*

1. *If x_k is feasible for problem* (1.1), *then $\nabla f_i(x_k)^T\bar{d} \le 0$.*
2. *Let x_k be infeasible for problem* (1.1), *and let r be defined as in Theorem* 2.5. *Then for all $\epsilon > 0$ there exists a $\bar{t} > 0$ such that $\Phi(x_k) - \Phi(x_k + t\bar{d}) + \epsilon r \in \mathbb{R}^{p+q}_+$ for all $t \in]0,\bar{t}\,]$.*
3. *Let $\bar{d} \neq 0$, let r be defined as in Theorem* 2.5, *and define $\hat{r} \in \mathbb{R}^{m+p+q}$ by $\hat{r} := (0_m^T, r^T)^T$, with $0_m = (0,\dots,0)^T \in \mathbb{R}^m$. Then for all $\epsilon > 0$ there exists a $\bar{t}$ such that*
$$\bar{f}(x_k) \not\prec_{\bar{f}} \bar{f}(x_k + t\bar{d}) - \epsilon\hat{r} \quad \forall t \in]0,\bar{t}\,].$$
4. *If $\bar{d} = 0$, then x_k is feasible for* (1.1) *and there does not exist a d feasible for* (3.1) *such that $\nabla f_i(x_k)^T d < 0$; i.e., no further improvement for the objective function f_i is possible along feasible directions.*
5. *If $d \neq 0$, then there exists a $\bar{t}$ such that for all $t \in [0,\bar{t}\,]$ and $\sigma > 0$ sufficiently large, we have*
$$\phi_i(x_k + td, \sigma) \le \phi_i(x_k, \sigma) + \mu\phi_i'(x_k, \sigma; d)$$
for all $\mu \in]0,1[$.

Proof.

1. This result follows by noting that if x_k is feasible to problem (1.1), then $d = 0$ is a feasible point for problem (3.1), and $\nabla f_i(x_k)^T\bar{d} \le \nabla f_i(x_k)^T\bar{d} + \frac{1}{2}\bar{d}^T H_i\bar{d} \le \nabla f_i(x_k)^T d + \frac{1}{2}d^T H_i d = 0$, since H_i is a positive definite matrix.
2. If x_k is infeasible to problem (1.1), then there exists a j or a l such that $g_j(x_k) > 0$ or $h_l(x_k) \neq 0$. By using a similar reasoning to that in the proof of Theorem 2.5, it follows that for all $\epsilon > 0$ there exists a $\bar{t}$ such that $g_j(x_k + t\bar{d}) < g_j(x_k)$ or $|h_l(x_k + t\bar{d})| < |h_l(x_k)|$ for all $t \in]0,\bar{t}\,]$.
3. The result follows from item 1, item 2, and Theorem 2.5 as follows. If $\bar{d} \neq 0$ and x_k is feasible for problem (1.1), then $\bar{d}^T H_i\bar{d} > 0$, since H_i is positive definite. Using item 1 results in $\nabla f_i(x_k)^T\bar{d} < 0$; i.e., there exists a $\bar{t}$ such that $f(x_k + t\bar{d}) < f(x_k)$ for all $t \in]0,\bar{t}\,]$. The result follows by noting that the case where $\bar{d} \neq 0$ and x_k is infeasible falls under the case described in item 2.

4. Clearly, if $\bar{d} = 0$, then $g_j(x_k) \leq 0$, $j = 1, \ldots, p$, and $h_l(x_k) = 0$, $l = 1, \ldots, q$, and thus x_k is feasible for problem (1.1). We show the remaining claim by contradiction. Suppose that there exists a feasible $\tilde{d}$ such that $\nabla f_i(x_k)^T \tilde{d} < 0$, with $\tilde{d} \neq 0$. Then $\tilde{d}$ is such that $g_j(x_k) + \nabla g_j(x_k)^T \tilde{d} \leq 0$ and $h_l(x_k) + \nabla h_l(x_k)^T \tilde{d} = 0$. Noting that x_k is feasible for problem (1.1), we have that for $0 \leq t \leq 1$ the vector $t\tilde{d}$ is feasible for problem (3.1) and there exists a $\tilde{t} > 0$ such that $\tilde{t}\frac{1}{2}\tilde{d}^T H_i \tilde{d} < -\nabla f_i(x_k)^T \tilde{d}$, resulting in $\tilde{t}\big(\nabla f_i(x_k)^T \tilde{d} + \tilde{t}\frac{1}{2}\tilde{d}^T H_i \tilde{d}\big) < 0$, which contradicts $\bar{d} = 0$, since $\tilde{t}\tilde{d}$ is feasible and attains a smaller objective function value.
5. By noting that problem (3.1) corresponds to the standard SQP step with respect to the ith objective function (see [3, equation (17.2)]), the result follows directly from Proposition 17.1 in [3]. Note also that $d \neq 0$ implies the noncriticality of x_k, H_i is symmetric positive definite by construction, and by σ large enough we mean that σ is no smaller than the absolute value of the largest Lagrange multiplier associated with the problem of minimizing f_i over Ω. ∎

Consider now the following optimization problem:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & \sum_{i=1}^{m} f_i(x) \\ \text{s.t.} \quad & g_j(x) \leq 0, \quad j = 1, \ldots, p, \\ & h_l(x) = 0, \quad l = 1, \ldots, q. \end{aligned} \tag{3.2}$$

For a given feasible point x_k, a direction v fulfilling (2.4) can be obtained by considering the SQP problem to (3.2),

$$\begin{aligned} \min_{v \in \mathbb{R}^n} \quad & \sum_{i=1}^{m} \nabla f_i(x_k)^T v + \frac{1}{2} v^T H_i v \\ \text{s.t.} \quad & g_j(x_k) + \nabla g_j(x_k)^T v \leq 0, \quad j = 1, \ldots, p, \\ & h_l(x_k) + \nabla h_l(x_k)^T v = 0, \quad l = 1, \ldots, q, \end{aligned} \tag{3.3}$$

where H_i are symmetric positive definite matrices.

In a certain sense, problem (3.3) tries the simultaneous minimization of quadratic approximation to all objective functions, while feasibility is maintained. Some properties of solutions $\bar{v}$ to problem (3.3) are addressed in the following lemma. Let $\vartheta_i(v) = \nabla f_i(x_k)^T v + (1/2) v^T H_i v$, $\bar{\vartheta}_i \equiv \bar{\vartheta}_i(\bar{v})$, $i = 1, \ldots, m$, and $\bar{\vartheta} = \sum_{i=1}^{m} \bar{\vartheta}_i$.

LEMMA 3.2. *Let $\bar{v}$ be a solution to problem* (3.3). *Then the following assertions hold:*

1. *If x_k is feasible to problem* (1.1), *then $\bar{\vartheta} \leq 0$.*
2. *Let x_k be infeasible for problem* (1.1), *and define $r \in \mathbb{R}^{p+q}$ as in Theorem* 2.5. *Then for every $\epsilon > 0$ there exists a $\bar{t} > 0$ such that $\Phi(x_k) - \Phi(x_k + t\bar{v}) + \epsilon r \in \mathbb{R}^{p+q}_+$ for all $t \in]0, \bar{t}\,]$.*
3. *Let $\bar{v} \neq 0$, let r be defined as in Theorem* 2.5, *and define $\hat{r} \in \mathbb{R}^{m+pq+q}$ by $\hat{r} := (0_m^T, r^T)^T$, with $0_m = (0, \ldots, 0)^T \in \mathbb{R}^m$. Then for all $\epsilon > 0$ there exists a $\bar{t}$ such that*

$$\bar{f}(x_k) \not\prec_{\bar{f}} \bar{f}(x_k + t\bar{v}) - \epsilon\hat{r}$$

for all $t \in]0, \bar{t}\,]$.

4. *Let $\hat{r} \in \mathbb{R}^{m+p+q}$ be defined as above. If $\bar{v} \neq 0$ and x_k is feasible for problem* (1.1)*, then for all $\epsilon > 0$ there exists a $\bar{t}$ such that*

$$\bar{f}(x_k + t\bar{v}) - \epsilon\hat{r} \preceq_{\bar{f}} \bar{f}(x_k)$$

for all $t \in]0, \bar{t}\,]$.

5. *If $\bar{v} = 0$, then x_k is feasible for problem* (1.1) *and there does not exist a $v \in L(x)$ such that $\nabla f_i(x_k)^T v < 0$, $i = 1, \ldots, m$; i.e., x_k is Pareto critical.*
6. *If $\bar{v} \neq 0$, then there exists a $\bar{t}$ such that for all $t \in]0, \bar{t}\,]$ and all $\sigma > 0$ sufficiently large, we have*

$$\phi(x_k + td, \sigma) \leq \phi(x_k, \sigma) + \mu t \phi'(x_k, \sigma; \bar{v})$$

for all $\mu \in]0, 1[$.

Proof.

1. If x_k is feasible for problem (1.1), then $(\bar{v}, \bar{\vartheta}) = 0$ is feasible for problem (3.3). Since we are minimizing the sum of all ϑ_i, $i = 1, \ldots, m$, the result follows.
2. This item follows by using the same reasoning as in item 2 of the proof of Lemma 3.1 and taking $\bar{v}$ as $\bar{d}$.
3. This item follows by using a reasoning similar to that of item 3 of the proof of Lemma 3.1. More precisely, if $\bar{v} \neq 0$ and x_k is feasible for problem (1.1), then $\nabla f_i(x_k)^T \bar{v} + \frac{1}{2}\bar{v}^T H_i \bar{v} \leq 0$, $i = 1, \ldots, m$, and, since H_i is positive definite, we have $\nabla f_i(x_k)^T \bar{v} < 0$, $i = 1, \ldots, m$, resulting in the existence of a $\bar{t}$ such that $f_i(x_k + t\bar{v}) < f_i(x_k)$, $i = 1, \ldots, m$, for all $t \in]0, \bar{t}\,]$. Noting that the case when x_k is infeasible for problem (1.1) is the case discussed in item 2, the result follows.
4. From the previous item we have that if $\bar{v} \neq 0$ and x_k is feasible for problem (1.1), then there exists a $t_1 > 0$ such that $f_i(x_k + t\bar{v}) < f_i(x_k)$, $i = 1, \ldots, m$, for all $t \in]0, t_1]$. Noting that for all $\epsilon > 0$ there exists a $t_2 > 0$ such that $\Phi(x_k + t\bar{v}) - \epsilon \leq 0$ for all $t \in]0, t_2]$, the result follows by taking $\bar{t} = \min\{t_1, t_2\}$.
5. If $\bar{v} = 0$, then we have that $g_j(x_k) \leq 0$, $j = 1, \ldots, p$, and $h_l(x_k) = 0$, $l = 1, \ldots, p$, i.e., x_k is feasible for problem (1.1). Clearly, $\bar{v} = 0$ results in $\bar{\vartheta}_i = 0$, $i = 1, \ldots, m$. The remaining claim is shown by contradiction. Suppose that there exists a feasible $\tilde{v}$ such that $\nabla f_i(x_k)^T \tilde{v} < 0$ for at least one $i \in \{1, \ldots, m\}$. Then we have that for $0 \leq t \leq 1$, the vector $t\tilde{v}$ is also feasible for problem (3.3). Thus, there exists a $\tilde{t} > 0$ such that $\tilde{t}\frac{1}{2}\tilde{v}^T H_i \tilde{v} < -\nabla f_i(x_k)^T \tilde{v}$, resulting in $\tilde{t}\left(\nabla f_i(x_k)^T \tilde{v} + \tilde{t}\frac{1}{2}\tilde{v}^T H_i \tilde{v}\right) < 0$. The contradiction results from noting that there exists a $\tilde{\vartheta}_i$ such that $\nabla f_i(x_k)^T \tilde{t}\tilde{v} + \frac{1}{2}\tilde{t}\tilde{v}^T H_i \tilde{t}\tilde{v} \leq \tilde{\vartheta}_i < 0$, i.e., $\tilde{t}\tilde{v}$ is feasible and attains a lower objective function value than $\bar{v}$. Pareto criticality arrives from the fact that there is no feasible search direction that can locally improve at least one objective function without increasing other objective function values.
6. Using the same reasoning as in Lemma 3.1, item 5, we arrive at

$$\phi'(x_k, \sigma; \bar{v}) < 0$$

from Proposition 17.1 in [3], and the result follows immediately. ∎

4. A multiobjective SQP-type algorithm. In this section we present the full algorithm that takes advantage of the previous described techniques for spreading and improving a given set of nondominated points.

Since the algorithm deals, iteratively, with a set of points X_k ($k = 0, 1, 2, \ldots$), a point $x_k \in X_k$ is labeled as *stopped* when no further progress is possible for x_k. This is the case when x_k has already been used to compute new candidate points and has reached criticality, or further progress towards optimality is no longer possible. Details are found in the algorithm below.

The solution of problem (3.2) is a Pareto critical point, as can be shown by taking $w_i = 1$, $i = 1, \ldots, m$, in Theorem 2.3. However, performing SQP steps using points in X_k as initial guesses may destroy the spread already obtained in the previous stage, especially in cases where problem (3.2) has, e.g., a unique solution. To overcome this issue, we introduce a reference point $\hat{x}_k$ for each point in the set X_k and consider the following optimization problem:

$$\begin{aligned} \min_{x \in \mathbb{R}^n} \quad & \sum_{i=1}^{m} f_i(x) \\ \text{s.t.} \quad & f_i(x) \leq f_i(\hat{x}_k), \quad i = 1, \ldots, m, \\ & g_j(x) \leq 0, \quad j = 1, \ldots, p, \\ & h_l(x) = 0, \quad l = 1, \ldots, q. \end{aligned} \tag{4.1}$$

Vectors $f(\hat{x}_k)$ can be seen as worst case acceptable outcomes of the optimization problem considered, while the problem itself drives towards Pareto optimality via its objective $\sum_{i=1}^{m} f_i(x)$. The SQP problem of (4.1) at an arbitrary point x_k is

$$\begin{aligned} \min_{v \in \mathbb{R}^n} \quad & \sum_{i=1}^{m} \nabla f_i(x_k)^T v + \frac{1}{2} v^T H_i v \\ \text{s.t.} \quad & f_i(x_k) - f_i(\hat{x}_k) + \nabla f_i(x_k)^T v \leq 0, \quad i = 1, \ldots, m, \\ & g_j(x_k) + \nabla g_j(x_k)^T v \leq 0, \quad j = 1, \ldots, p, \\ & h_l(x_k) + \nabla h_l(x_k)^T v = 0, \quad l = 1, \ldots, q. \end{aligned} \tag{4.2}$$

PROPOSITION 4.1. *Let $\bar{v}$ be a solution to* (4.2)*; then item* 1 *and items* 3–6 *of Lemma* 3.2 *hold for $\bar{v}$.*

As mentioned in the introduction, our algorithm improves several different points $x_k \in X_k$ in a parallel fashion. In what follows, we will use the notation $\sigma(x_k)$ to indicate that we use different penalty parameters for different points x_k. Computing a $\bar{\sigma} > 0$ such that $\phi'(x, \sigma; d) < 0$ for all $\sigma \geq \bar{\sigma}$ is somewhat nontrivial: typically, $\bar{\sigma}$ has to be larger than the largest absolute Lagrange multiplier's absolute value. There exist several strategies to compute such $\sigma(x_k)$ (see, e.g., [3, 18]), but below we have chosen to keep our algorithm as general as possible by not specifying details on how to compute such values.

We are now able to describe our full algorithm.

ALGORITHM 4.1 (multiobjective SQP method).

1. *Initialization. Define $\mu \in]0, 1/2[$ and $\beta \in]0, 1[$. Set $k = 0$.*
2. *First stage (Initialization)*
   Choose a nonempty, finite set of points $X_0 \subset \mathbb{R}^n$. Let $\bar{n}$ be the number of points in X_0. Consider all point in X_0 as nonstopped.
3. *Second stage (Spread)*
   Perform a finite number of iterations on the set X_k as follows.
   (a) *Set $\mathcal{T} = \emptyset$.*
   For each point $x_k \in X_k$ not stopped do

i. *For each* $i = 1, \dots, m$ *do*
   A. *Compute the search direction* $\bar{d}$ *that corresponds to an SQP step for objective function* i *by solving the quadratic problem* (3.1).
   B. *If* $\|\bar{d}\|$ *is small, go to step* 3aiG.
   C. *Compute* $\sigma(x_k)$.
   D. *Find the largest* $\alpha \in \{1, \beta, \beta^2, \dots\}$ *such that we have*

$$\phi_i(x_k + \alpha\bar{d}, \sigma(x_k)) \leq \phi_i(x_k, \sigma(x_k)) + \mu\alpha\phi_i'(x_k, \sigma(x_k); \bar{d}).$$

   E. *If* α *is small, go to step* 3aiG.
   F. *Set* $\mathcal{T} = \mathcal{T} \bigcup \{x_k + \alpha\bar{d}\}$.
   G. *Continue with the next* i.

ii. *Set* x_k *as a stopped point. Continue with the next point in* X_k.

(b) *Set all points in* $\mathcal{T}$ *as nonstopped points and* X_{k+1} *as the set of nondominated points in* $X_k \bigcup \mathcal{T}$. *Set* $k = k + 1$. *Continue with step* 3a *if the prespecified finite number of iterations has not been reached.*

4. *Third stage (Optimality—refining stage)*
Set all points in X_k *as nonstopped.*
For each point $x_k \in X_k$, *set* $\hat{x}_k := x_k$ *as the reference point to be considered for problems* (4.1).
While not all points in X_k *are stopped do:*
(a) *For all nonstopped points* $x_k \in X_k$ *do*
   i. *Compute* $\bar{v}$ *by solving the optimization problem* (4.2).
   ii. *If* $\|\bar{v}\|$ *is small, then set* x_k *as stopped and go to step* 4avii.
   iii. *Compute* $\sigma(x_k)$.
   iv. *Find the largest* $\alpha \in \{1, \beta, \beta^2, \dots\}$ *element such that*

$$\phi(x_k + \alpha\bar{v}, \sigma(x_k)) \leq \phi(x_k, \sigma(x_k)) + \mu\alpha\phi'(x_k, \sigma(x_k); \bar{v}).$$

   v. *If* α *is small, set the point* x_k *as a stopped point and go to step* 4avii.
   vi. *Replace* x_k *by* $x_k + \alpha\bar{v}$ *in the set* X_k.
   vii. *Continue with the next point in* X_k.

(b) *Set* X_{k+1} *as the set of all nondominated points in* X_k. *Set* $k = k + 1$. *Continue with step* 4a.

Phase 2, the initialization stage, is entirely concerned with providing some initial guesses for Pareto points in the set X_0. We provide details for a generic procedure for providing such guesses in our section on implementation details. Obviously, a user can use domain-specific knowledge to provide such guesses.

Note that in the spread stage the algorithm computes a set of points X_k such that the set of images $\{f(x) \mid x \in X_k\}$ is spread along the Pareto front. The "spread" does not try to construct a converging sequence of points, as this task is solely referred to as the third stage.

5. Convergence. Lemmas 3.1 and 3.2 provide further justification for our choices of stopping criteria (small $\|\bar{d}\|$ and small $\|\bar{v}\|$, or small α) in the second and third stages of Algorithm 4.1.

Algorithm 4.1 first solves a finite number of single-objective optimization problems in the first stage and then performs a finite number of iterations in the second stage. We have then computed a set X_k of nondominated points that are used as initial guesses for the third stage. Step 4 is coupled with a strategy to address some numerical difficulties that can arise in computing the search direction for the next

iterate. Nondominated points will be saved in the set X_{k+1}. Step 4 can be considered as a standard SQP algorithm applied to individual points in the set X_k. Convergence results follow immediately by considering an SQP framework as described in [3]. We provide details in what follows.

Step 4 of Algorithm 4.1 can be described as follows. For each given $x_k \in X_k$, problem (4.2) is considered, and iterations of an SQP-type method for this problem are carried out, with x_k as a starting point.

Denote by $(x_k^{(\zeta)})_\zeta$ the sequence of points thus generated ($\zeta = 0, 1, 2, \ldots$) with $x_k^{(0)} = x_k$. The corresponding subproblems, solved in each step, are of the form

$$
\begin{aligned}
\min_{v \in \mathbb{R}^n} \quad & \sum_{i=1}^{m} \nabla f_i(x_k^{(\zeta)})^T v + \frac{1}{2} v^T H_k^{(\zeta)} v \\
\text{s.t.} \quad & f_i(x^{(\zeta)}) - f_i(\hat{x}_k) + \nabla f_i(x_k^{(\zeta)})^T v \leq 0, \quad i = 1, \ldots, m, \\
& g_j(x_k^{(\zeta)}) + \nabla g_j(x_k^{(\zeta)})^T v \leq 0, \quad j = 1, \ldots, p, \\
& h_l(x_k^{(\zeta)}) + \nabla h_l(x_k^{(\zeta)})^T v = 0, \quad l = 1, \ldots, q.
\end{aligned} \tag{5.1}
$$

The theorem below provides a justification for this strategy.

THEOREM 5.1. *The following hold:*

1. *Let x_* be a (global) minimum of* (4.1). *Then x_* is a (global) Pareto point of problem* (1.1).
2. *Let x_* be a KKT point of problem* (4.1). *Then x_* is a Pareto critical point.*

Proof.

1. This follows directly from Theorem 2.21 of [15].
2. Let λ, μ, ν be relevant Lagrange multipliers, i.e., $\lambda, \mu \geq 0$ and

$$\sum_{i=1}^{m} \nabla f_i(x_*) + \sum_{i=1}^{m} \lambda_i \nabla f_i(x_*) + \sum_{j=1}^{p} \mu_j \nabla g_j(x_*) + \sum_{l=1}^{q} \nu_l \nabla h_l(x_*) = 0.$$

Define $w \in \mathbb{R}^m_{++}$ by $w_i := 1 + \lambda_i > 0$. Then x_* is a KKT point for the problem of optimizing $\sum_{i=1}^{m} w_i f_i$ over Ω, and the result follows with Theorem 2.3. ∎

As long as a point $x_k^{(\zeta)}$ is not dominated by another feasible point, it is included in the next set X_{k+1}. While not strictly necessary, this helps excluding points converging to local but not global Pareto optima.

Step 4 deals *simultaneously* with several SQP steps induced by the possible many points in the set X_k. Since the number of iterations performed for each point x_k may differ, we consider points to be stopped whenever optimality or a sufficient good approximation is obtained. Therefore, step 4 either generates a finite number of iterations, proving a set of *approximate* Pareto critical points, or it generates an infinite number of iterations, constructing at least one infinite sequence of points, whose convergence is now analyzed. Convergence analysis may concentrate on the single sequence of points $(x_k^{(\zeta)})_\zeta$ generated for each point in X_k.

The main convergence result follows readily from standard convergence results in the theory of SQP methods: let $(x_k^{(\zeta)})_\zeta$ be an arbitrary infinite sequence of points generated in step 4.

THEOREM 5.2. *Consider some $x_k \in X_k$ and the sequence $(x_k^{(\zeta)})_\zeta$ ($\zeta = 0, 1, 2, \ldots$) generated in step* 4 *of Algorithm* 4.1 *with $x_k^{(0)} = x_k$. Suppose the following assumptions hold:*

1. *The functions f, g, h are twice continuously differentiable on $\mathbb{R}^n$.*
2. *The update rule for $\sigma_\zeta(x_k)$ fulfils the following conditions:*
   (a) *Let $\varrho_k^{(\zeta)} = (\lambda_k^{(\zeta)}, \mu_k^{(\zeta)}, \nu_k^{(\zeta)})$ be an optimal Lagrange multiplier for problem (5.1), and let $\bar\sigma_k > 0$ be a constant. Let $\sigma_\zeta(x_k) \geq \|\varrho_k^{(\zeta)}\| + \bar\sigma_k$ hold for all ζ.*
   (b) *There exists an index $\bar\zeta$ such that if $\zeta \geq \bar\zeta$ and $\sigma_{\zeta-1}(x_k) \geq \|\varrho_k^{(\zeta)}\| + \bar\sigma_k$, then $\sigma_\zeta(x_k) = \sigma_{\zeta-1}(x_k)$.*
   (c) *If the sequence $(\sigma_\zeta(x_k))_\zeta$ is bounded, then $\sigma_\zeta(x_k)$ is modified finitely often.*
3. *The problems (5.1) are feasible for all ζ.*
4. *The matrices $H_k^{(\zeta)}$ are positive definite for all ζ, and the sequences $(H_k^{(\zeta)})_\zeta$ and $((H_k^{(\zeta)})^{-1})_\zeta$ are bounded.*

Then one of the following holds:

1. *The sequence $(\sigma_\zeta(x_k))_\zeta$ is unbounded. In this case, the corresponding sequence of Lagrange multipliers $(\varrho_k^{(\zeta)})_\zeta$ is also unbounded.*
2. *There exists an index $\bar\zeta$ such that for $\zeta \geq \bar\zeta$ we have $\sigma_\zeta(x_k) =: \sigma_k$ constant. In this case, one of the following holds:*
   (a) $\lim_{\zeta\to+\infty} \phi(x_k^{(\zeta)}, \sigma_k) = -\infty$.
   (b) *Every cluster point of the sequence $(x_k^{(\zeta)})_\zeta$ is a Pareto critical point.*

Proof. The assumptions allow one to invoke Theorem 17.2 of [3], by which one of the following holds:

1. $\lim_{\zeta\to+\infty} \phi(x_k^{(\zeta)}, \sigma_k) = -\infty$.
2. The distance between $x_k^{(\zeta)}$ and the set of nondifferentiable points of the problem functions f, g, h converges to zero.
3. Let $\mathcal{L}_k$ be the Lagrangian of problem (4.1), and recall $\Phi(x) = (g^+(x), |h(x)|)$. Then $\lim_{\zeta\to+\infty} \nabla_x \mathcal{L}_k(x_k^{(\zeta)}, \varrho_k^{(\zeta)}) = 0$, $\lim_{\zeta\to+\infty} \Phi(x_k^{(\zeta)}) = 0$, and

$$\lim_{\zeta\to+\infty} \max\{0, f_i(x_k^{(\zeta)}) - f_i(x_k)\} = 0 \ (i = 1, \ldots, m).$$

   Moreover, if $\mu_k^{(\zeta,j)}$ denotes the Lagrange multiplier to the constraint $g_j(x_k^{(\zeta)}) + \nabla g_j(x_k^{(\zeta)})^T v \leq 0$ in (5.1) and $\nu_k^{(\zeta,\ell)}$ denotes the Lagrange multiplier to the constraint $h_l(x_k^{(\zeta)}) + \nabla h_l(x_k^{(\zeta)})^T v = 0$ in (5.1), then

$$\lim_{\zeta\to\infty} \mu_k^{(\zeta,j)} g_j(x_k^{(\zeta)}) = 0 \quad \text{and} \quad \lim_{\zeta\to\infty} \nu_k^{(\zeta,\ell)} h_\ell(x_k^{(\zeta)}) = 0.$$

The second assertion cannot hold, as we have assumed that f, g, h are sufficiently smooth everywhere. The third assertion means that every cluster point of $(x_k^{(\zeta)})_\zeta$ is a KKT point of problem (4.1). The result then follows with Theorem 5.1. ∎

Note that an update rule for $\sigma_\zeta(x_k)$ fulfilling the conditions of the theorem is easy to implement; see, e.g., [18] for details.

The last stage of the proposed algorithm can be seen as a mechanism akin to a multistart procedure, using all the points available in the set X_k at the end of the spread stage. For the nonconvex case we can also expect convergence to local Pareto points or merely Pareto critical points of problem (4.1), and not to global Pareto points. Recall also that Algorithm 4.1 is not described as a primal-dual method in the sense that we do not address the computation of any Lagrange multipliers. However,

it is straightforward to include such a mechanism and thus to implement a strategy to properly handle the penalty parameter σ. Estimates of Lagrange multipliers are available from problems (3.1) and (4.2), and a possible strategy for updating the merit function parameter σ based on estimates of Lagrange multipliers is described in [3, p. 294].

6. Implementation details. A prototype implementation of Algorithm 4.1 is available in MATLAB [17] from http://www.norg.uminho.pt/aivaz/mosqp. The implementation is able to address both constrained and unconstrained optimization problems. While the purpose of our contribution is to provide a solver for nonlinearly constrained multiobjective optimization problems, we also report numerical results for simple bound constrained problems separately.

In our implementation, we use the `fmincon` MATLAB solver for nonlinear constrained optimization problems for solving the subproblems described in (6.1) and feasibility restoration, which is described below. Since the subproblems (4.2) and (3.1) are quadratic optimization problems, `quadprog` is the MATLAB solver of choice. Furthermore, `exitflag` is used to decide whether a corresponding subproblem has been solved successfully or not and when to enter a feasibility restoration procedure.

The first stage (Initialization) is entirely concerned with providing an initial set of points for the overall procedure. Our implementation allows users to provide an initial set of points that are used to initialize the set X_0. Since the set X_0 is a set of nondominated points, some user-provided points might be discarded in the initialization procedure. We provide two strategies to initialize the remaining points for problems with $-\infty < \ell_i,\ u_i < +\infty$, $i = 1, \ldots, n$. The `line` strategy initializes the remaining points by considering the line segment between ℓ and u, i.e., points of the form $x_i = \ell + i\frac{u-\ell}{2\bar{n}}$, $i = 1, \ldots, 2\bar{n}$. The `rand` strategy initializes the remaining point using a uniform random distribution on each of intervals $[\ell_i, u_i]$, $i = 1, \ldots, n$, generating at most $2\bar{n}$ random points. Additionally, our implementation computes all the "extreme" Pareto front points by solving the following single-objective problems for each $i = 1, \ldots, m$:

$$
\begin{aligned}
(x_*)_i \in \arg\min_{x \in \mathbb{R}^n} \quad & f_i(x) \\
\text{s.t.} \quad & g_j(x) \le 0, \quad j = 1, \ldots, p, \\
& h_l(x) = 0, \quad l = 1, \ldots, q.
\end{aligned}
\tag{6.1}
$$

The second stage of Algorithm 4.1 is performed for a maximum of 20 iterations or until all points in the set X_k are considered to be stopped. Points in X_k are declared as stopped if $\|\bar{d}\|$ or $\|\bar{v}\|$ is small, as justified by Lemmas 3.1 and 3.2. In our implementation, we use a tolerance parameter $\tau \in]0, 1[$ and check for $\|\bar{d}\| < \sqrt[4]{\tau}$, $\alpha < \sqrt{\tau}$, and $\|\bar{v}\| < \tau$, respectively, as each stage has it own purpose: the second stage to obtain points that are spread over the Pareto front and the third stage to obtain Pareto critical points. This approach seems to work well on the set of problems considered. A value of $\tau = 10^{-5}$ has been used for our numerical tests.

The second stage in Algorithm 4.1 computes a new set of nondominated points $\mathcal{T}$. The updated set X_{k+1} can grow to $(m+1)\bar{n}$ points ($\bar{n}$ points already in X_k and an additional $m\bar{n}$ new points). However, in order to keep the set X_k at a reasonable size, one should consider a *Cleanup* procedure whenever the cardinality of X_k reaches a certain upper bound. *Cleanup* consists in the computation of the crowding distance [8] for each point and the removal of the 10 points with the smallest crowding distances. The crowding distance provides an estimate of the density of solutions surrounding a

given point, taking the average distance of the two closest neighboring points. Points with a small crowding distance are positioned in a *crowded* region and are good candidates to be eliminated from the set under consideration. In our implementation, *Cleanup* is performed whenever the cardinality of X_k reaches $\bar{n}+1$.

For the single-objective problems in (6.1) we consider the starting point x_0. If the constraints include bound constraints of the form $-\infty < \ell_i \leq x_i \leq u_i < +\infty$, then $(x_0)_i := (u_i + \ell_i)/2$. If the constraints include a lower bound $-\infty < \ell_i \leq x_i$ but no upper bound on x_i, then $(x_0)_i := \ell_i$. Likewise, if the constraints include an upper bound $x_i \leq u_i < +\infty$ but no lower bound on x_i, then $(x_0)_i := u_i$. In all other cases, set $(x_0)_i := 0$.

We consider $d = 0$ and $(v, \vartheta) = 0$ as the initial guesses for subproblems (3.3) and (3.1), while x_k is used as an initial guess for the feasibility restoration problem.

Regarding the quadratic approximations to the objective functions, we consider three cases. In the first case we simply consider $H_i = I_m$, $i = 1, \ldots, m$, in both stages of the algorithm, where I_m is the identity matrix of dimension m. The second case uses $H_i = \nabla^2 f_i(x_k) + E_i$ in both stages of the algorithm, where E_i is obtained by a modified Cholesky algorithm as proposed in [12, 21]. The matrix E_i is diagonal and ensures that H_i is positive definite; note that $E_i = 0$ if $\nabla^2 f_i(x_k)$ is already positive definite. Finally, the last case uses $H_i = I_m$ in the second stage of the algorithm and $H_i = \nabla^2 f_i(x_k) + E_i$ as in the second case for the third stage.

Note that the quadratic problems (3.1) and (4.2) may fail to provide us with a direction of sufficient decrease, and a robust implementation of our algorithm needs to deal with possible failures after steps 3aiA and 4ai. As such, after step 3aiA, if problem (3.1) has no feasible solution, we enter a feasibility restoration procedure. If the restoration procedure obtains a point $\bar{x}_k$ feasible for (3.1), then we set $\mathcal{T} = \mathcal{T} \bigcup \{\bar{x}_k\}$. Otherwise, problem (3.1) has no feasible solution, a feasible $\bar{x}_k$ was not obtained, or numerical difficulties prevent us from solving problem (3.1). In any of these cases we proceed to step 3aiG. Likewise, after step 4ai, if problem (4.2) has no feasible solution, we also enter a feasibility restoration procedure. If a feasible point $\bar{x}_k$ is found during such a restoration procedure, we replace x_k by $\bar{x}_k$. Otherwise, problem (4.2) has no feasible solution, a feasible $\bar{x}_k$ was not obtained, or numerical difficulties prevent us from solving problem (4.2). We then set x_k as a stopped point. In both cases we proceed to step 4avii.

A typical restoration procedure consists in the minimization of an infeasibility measure, for example the sum of the violated inequality constraints at the current point, plus the sum of squares of the violated equality constraints, plus a regularization term. Further numerical difficulties may also arise in solving this nonlinear optimization problem. In that case, we consider the current point as stopped. While other more complex restoration procedures could be considered, we chose to include this simple procedure before giving up on the current point to provide progress to the algorithm. See [23] for a discussion of a more complex restoration procedure.

7. Numerical results. In this section we report on the numerical results of our implementation. We use a set of multiobjective test problems collected from the literature and modeled in AMPL [2]. AMPL allows us to obtain, through automatic differentiation, first and second derivatives for objectives and constraints (although second derivatives of the constraints are not used in our implementation). Approximations to Pareto fronts computed are used to compare performance with other solvers via performance profiles and data profiles. In the next subsection we describe test problems used as well as the performance profiles by which we will ascertain the

TABLE 1
Bound constrained multiobjective problems.

Problem	m	n	Problem	m	n	Problem	m	n	Problem	m	n
BK1	2	2	DTLZ3	3	12	Jin4_a	2	2	MOP6	2	2
CEC09_1	2	30	DTLZ3n2	2	2	KW2	2	2	MOP7	3	2
CEC09_10	3	30	DTLZ4	3	12	lovison1	2	2	SK1	2	1
CEC09_2	2	15	DTLZ4n2	2	2	lovison2	2	2	SK2	2	4
CEC09_3	2	30	DTLZ5_a	3	12	lovison3	2	2	SP1	2	2
CEC09_7	2	30	DTLZ5n2_a	2	2	lovison4	2	2	SSFYY1	2	2
CEC09_8	3	30	DTLZ6	3	22	lovison5	3	3	SSFYY2	2	1
CL1	2	4	DTLZ6n2	2	2	lovison6	3	3	TKLY1	2	4
Deb41	2	2	ex005	2	2	LRS1	2	2	VFM1	3	2
Deb513	2	2	Far1	2	2	MHHM1	3	1	VU1	2	2
Deb521a_a	2	2	Fonseca	2	2	MHHM2	3	2	VU2	2	2
Deb521b	2	2	GE2	2	40	MLF1	2	1	ZDT1	2	30
DG01	2	1	GE5	3	3	MLF2	2	2	ZDT2	2	30
DG02_a	2	1	IKK1	3	2	MOP1	2	1	ZDT3	2	30
DTLZ1	3	7	IM1	2	2	MOP2	2	4	ZDT4	2	10
DTLZ1n2	2	2	Jin1	2	2	MOP3	2	2	ZDT6_a	2	10
DTLZ2	3	12	Jin2_a	2	2	MOP5	3	2	ZLT1	3	10
DTLZ2n2	2	2	Jin3	2	2						

overall performance of the implementation. After that, we present numerical results obtained on our set of test problems as well as on real-word applications.

7.1. Test problems and profiles.

7.1.1. Test problems. We consider those test problems available in [6] that are sufficiently smooth. These problems are available by following the instructions at http://www.mat.uc.pt/dms. We enrich the test problem database with further test problems provided in [8, 11, 14, 24] as well as those available in the MacMOOP database (http://wiki.mcs.anl.gov/leyffer/index.php/MacMOOP). The full test set is presented in Table 1 for bound constrained problems, i.e., for problems where the feasible set is $\Omega = \{x \in \mathbb{R}^n : \ell \leq x \leq u\}$ for some $\ell \in (\mathbb{R} \cup \{-\infty\})^n$, $u \in (\mathbb{R} \cup \{\infty\})^n$, and in Table 2 for nonlinearly constrained optimization problems. In both tables, m is the number of objective functions, n is the number of variables, "Linear" is the number of linear constraints, and "Nonlinear" is the number of nonlinear constraints. We provide our full problem database at www.norg.uminho.pt/aivaz/mosqp.

The test set includes problems with both a convex and a nonconvex Pareto front. About 50% of the unconstrained or simple bound constrained biobjective problems have a convex Pareto front. Four biobjective problems with nonlinear constraints exhibit a nonconvex Pareto front.

7.1.2. Performance profiles. We provide most of our numerical results in the form of performance profiles. Performance profiles were first proposed for single-objective optimization problems in [9] and later adapted to multiobjective optimization in [6]. Performance profiles are defined by a cumulative function $\rho(\tau)$ presenting a performance ratio with respect to a given metric for a given set of solvers. Given a set of solvers $\mathcal{SO}$ and a set of problems $\mathcal{P}$, let $t_{p,s}$ be the performance of solver s on solving problem p. Usually, the metric $t_{p,s}$ is an algorithmic performance metric like the number of iterations needed to attain a solution. The performance ratio is then defined as $r_{p,s} := t_{p,s} / \min_{\bar{s} \in \mathcal{SO}} t_{p,\bar{s}}$. The cumulative function $\rho_s(\tau)$ ($s \in \mathcal{SO}$) is defined as the percentage of problems whose performance ratio is below or equal to τ, i.e., $\rho_s(\tau) := |\{p \in \mathcal{P} : r_{p,s} \leq \tau\}|/|\mathcal{P}|$. For a solver s, we can interpret $\rho_s(\tau)$ as the

TABLE 2
Linear and nonlinearly constrained multiobjective problems.

Problem	m	n	Linear	Nonlinear	Problem	m	n	Linear	Nonlinear
ABC_comp	2	2	2	1	KW1	2	5	2	3
BNH	2	2	0	2	liswetm	2	7	5	0
CEC09_C10	3	10	0	1	MOLPg_001	3	8	8	0
CEC09_C3	2	10	0	1	MOLPg_002	3	12	13	0
CEC09_C9	3	10	0	1	MOLPg_003	3	10	12	0
ex001	2	5	2	3	MOQP_0001	3	20	10	0
ex002	2	5	0	4	MOQP_0002	3	20	9	0
ex003	2	2	0	2	MOQP_0003	3	20	10	0
ex004	2	2	2	0	OSY	2	6	4	2
GE1	2	2	0	1	SRN	2	2	1	1
GE3	2	2	0	2	TNK	2	2	0	2
GE4	3	3	0	1	WeldedBeam	2	4	1	3
hs05x	3	5	6	0					

probability of the performance of s on an arbitrary problem to be within a factor of τ of the best performance. We compare the values of $\rho_s(1)$ for all solvers $s \in \mathcal{SO}$ when looking for the most efficient solver, while we consider $\rho(\tau)$ for $\tau \to \infty$ to compare solvers with respect to robustness, as the solver that attains a larger $\rho(\tau)$ for large τ is the one able to solve a larger percentage of problems.

Multiobjective optimization problems pose a particular challenge to the notion of performance profiles, as it is not a priori clear what an appropriate metric for quality of solution is. In general, the output of a multiobjective optimization solver is a set of points, and not just one point, and one of the tasks of the solver is to compute a set of points serving as an approximation of the full Pareto set. Here we consider the following metrics: the *Purity* metric [6], two *spread* metrics [6], and the popular *hypervolume* metric [26].

Purity metric. Let $F_{p,s}$ be the approximation to the Pareto front computed by solver s for problem p. Let F_p be the approximation to the Pareto front obtained by the union of all individual Pareto approximation, $\cup_{s\in\mathcal{S}} F_{p,s}$, where all dominated points are removed. Since the true Pareto front is not known for all problems in our problems database, we consider F_p in place of the true Pareto front. We define the purity metric as the number of points in F_p divided by the number of points solver s is able to compute that are not dominated by any other point computed, i.e.,

$$t_{p,s} = \frac{|F_p|}{|F_{p,s} \cap F_p|}.$$

The purity metric measures the inverse of how many nondominated points a solver is able to compute from the set of all nondominated points computed. In our version of the metric, small values are better, as necessitated when using performance profiles. In case $|F_{p,s} \cap F_p| = 0$ we set $t_{p,s} := \infty$, meaning that solver s was unable to provide even a single nondominated point for problem p. See [6] for further details and discussions.

Spread metrics. While the purity metric measures how well a solver is able to compute nondominated points, the purity metric is unable to provide any information about how points are spread over the Pareto front. In order to understand whether a given solver is able to provide an approximation to the Pareto front whose points are "well distributed," we consider two additional metrics for our performance profiles. Let the approximated Pareto front computed by solver s for problem p be formed

of N points $x_1, \ldots, x_N$, and let these points be sorted by objective function j, i.e., $f_j(x_i) \leq f_j(x_{i+1})$ $(i = 1, \ldots, N)$. Furthermore, let x_0 and x_{N+1} be the extreme values for objective j; i.e., x_0 is the best known approximation to a global minimum of f_j, and x_{N+1} is the best known approximation to a global maximum of f_j, computed over all Pareto front approximations obtained. Define $\delta_{i,j} = |f_j(x_{i+1}) - f_j(x_i)|$, and let $\bar{\delta}_j$ $(j = 1, \ldots, m)$ be the average of the distances $\delta_{i,j}$. The $\Gamma > 0$ and $\Delta > 0$ metrics are then defined as

$$\Gamma_{p,s} = \max_{j \in \{1,\ldots,m\}} \max_{i \in \{0,\ldots,N\}} \delta_{i,j}$$

and

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

Including x_0 and x_{N+1} in the above is important, as $f(x_1)$ and $f(x_N)$ may be close to each other but far away from the true Pareto front extremes. This inclusion ensures that the metric Γ is always well defined, while Δ is not defined in the case $N = 1$, $x_0 = x_1 = x_{N+1}$.

While the Γ metric measures the largest gap in the Pareto front, the Δ metric measures the scaled deviation from the average gap in the Pareto front. Further details on these metrics are available in [6]. We set $t_{p,s} = \Gamma_{p,s}$ or $t_{p,s} = \Delta_{p,s}$ depending on the selected metric.

Hypervolume metric (S-metric). The hypervolume metric proposed in [26, 25] corresponds to the volume of the dominated space, enclosed by the nondominated points and the origin. A more detailed description is available in [4].

We use the code from ftp://ftp.tik.ee.ethz.ch/pub/people/zitzler/hypervol.c to compute the hypervolume metric, and we have interfaced this code with MATLAB for the performance profile generation.

Using the hypervolume as defined in [26] may result in an invalid value for the metric, since the hypervolume measure is zero for a Pareto front formed by only one point. The performance profile modification used in [22] was used to address this possibility. This modification corresponds to defining $t_{p,s} := t_{p,s} + 1 - \min_{\bar{s} \in \mathcal{SO}} t_{p,\bar{s}}$ when $\min_{\bar{s} \in \mathcal{SO}} t_{p,\bar{s}} < 0.001$ and leaving $t_{p,s}$ unchanged in all other cases.

7.1.3. Data profiles. Data profiles were originally proposed for derivative-free single-objective optimization algorithms [19] and later extended to derivative-free multiobjective optimization algorithms in [6]. A data profile can be interpreted as a cumulative distribution function $d_s(\sigma)$ that reports the percentage of problems solver s is able to solve, given a budget of σ objective function evaluations. Since data profiles allow us to discuss the computational resources an algorithm needs to solve a problem, we describe further numerical results using this approach.

Formally, the data profile function is defined as follows:

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

where $h_{p,s}$ is the number of function and gradient evaluations needed to solve problem p by solver s.

A further issue in multiobjective optimization arises from the lack of a simple definition for having solved a particular multiobjective optimization problem. Again,

this stems from the fact that we have to approximate a whole set of points, and do not just need to compute one point, as in single-objective optimization. As in [6], we use the following notion. A solver s is said to have solved problem p with accuracy ε if the following holds. Let F_p be a set of points that we consider as a high-quality approximation of the Pareto front. In our numerical experiments, the set F_p is computed by running the solver s for a limit of 5000 function evaluations, ignoring any other stopping criteria. The solver s has then solved problem p with accuracy ε if

$$\frac{|F_{p,s} \cap F_p|}{|F_p|/|\mathcal{SO}|} \geq 1 - \varepsilon.$$

7.2. Numerical results. As described in the last subsection, we present our numerical results in the form of performance and data profiles. We compare our implementation with the `NSGA II` solver [8] (C version 1.1) for all test problems and with a classical scalarization approach for biobjective test problems. A population of 100 points was used for `NSGA II`. To generate the corresponding performance profiles for this solver, we used 200 generations, corresponding to a total of 20,000 objective and constraint function evaluations, since objectives and constraints are evaluated for each point in each generation. Each objective function evaluation corresponds to an evaluation of the vector function f, and each constraint function evaluation corresponds to an evaluation of all the linear and nonlinear constraint functions.

Further, a scalarization approach for biobjective optimization was implemented in MATLAB, allowing us to compare our implementation against a widely used derivative based approach. For this, we solve problems of the form

$$\min_{x \in \Omega} \quad w f_1(x) + (1 - w) f_2(x) \tag{7.1}$$

for $w = k/100$, $k = 1, \ldots, 100$, using `fmincon` for each of the problems (7.1). First and second derivatives are provided to the solver through AMPL, although the selected SQP algorithm within `fmincon` only uses first derivatives. We denote this solution strategy by `MOScalar`. While this weighted-sum approach represents a rather elementary scalarization strategy, and more sophisticated methods exist, weighted-sum is widely popular in practice and thus represents a baseline approach against which all practical methods have to be compared.

Let us further denote solvers based on our proposed algorithm with the prefix `MOSQP`. We consider six variations of our solver, which correspond to the combinations of the initialization strategy and the selected H_i, as described above. For example, `MOSQP` ($H = \nabla^2 f$, `rand`) corresponds to the `rand` initialization strategy where the true objective function Hessian is considered in both stages, while `MOSQP` ($H = (I, \nabla^2 f)$, `line`) corresponds to the `line` initialization strategy where the identity matrix is used in the second algorithmic stage and the true objective Hessian is used in the third algorithmic stage. Since `NSGA II` is a stochastic algorithm and we provide an implementation with a stochastic initialization, we have performed 10 runs for each solver with a stochastic element, namely for `NSGA II`, `MOSQP` ($H = I$, `rand`), `MOSQP` ($H = \nabla^2 f$, `rand`), and `MOSQP` ($H = (I, \nabla^2 f)$, `rand`). For each stochastic solver we computed the *best* and *worst* Pareto fronts, i.e., the *best* Pareto front for a given stochastic solver is the computed Pareto front that has the higher number of nondominated points when compared with the approximated Pareto front F_p, while the *worst* is the one with the smallest number of nondominated points when compared with F_p.

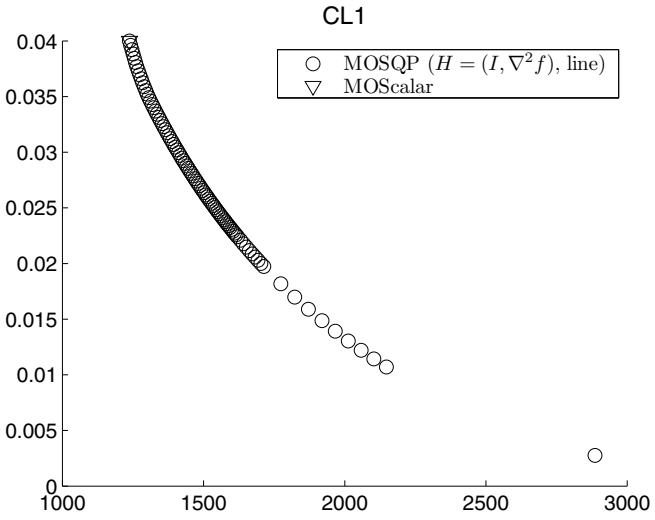

FIG. 1. *Pareto fronts obtained with the* `MOSQP` *and* `MOScalar` *for problem* `CL1`. *Note that* `MOScalar` *finds only one Pareto point here.*

As such, we are considering a significant number of performance and data profiles. To keep matters succinct, we will discuss in what follows only the most important findings. The interested reader can access the full set of profiles and Pareto front plots at the solver webpage http://www.norg.uminho.pt/aivaz/mosqp.

7.2.1. Approximate Pareto fronts. Before discussing the performance and data profiles obtained, we set the stage by providing illuminating examples of Pareto front approximations computed by the solvers considered. Pareto front approximations obtained by `MOSQP (`$H = (I, \nabla^2 f)$`, line)` and `MOScalar` for the `CL1` problem are depicted in Figure 1. As can be clearly seen, `MOSQP (`$H = (I, \nabla^2 f)$`, line)` is able to capture the overall front well, while `MOScalar` finds only a single point. The weighted-sum objective for problem `CL1` is

$$200w(2x_1 + \sqrt{2}x_2 + \sqrt{x_3} + x_4) + 10^{-2}w\left(\frac{2}{x_1} + \frac{2\sqrt{2}}{x_2} - \frac{2\sqrt{2}}{x_3} + \frac{2}{x_4}\right)$$

with constraints $1 \leq x_1, x_4 \leq 3$ and $\sqrt{2} \leq x_2, x_3 \leq 3$. For parameters $w = k/100$, $k = 1, \ldots, 100$, the optimal solution of all these parameterized problems is always $x = (1, \sqrt{2}, \sqrt{2}, 1)$. While this might appear to be a somewhat extreme example, subsection 7.3 shows a real-world test case exhibiting exactly the same phenomenon; see especially Figure 8. Further depictions of Pareto front approximations can be found in subsection 7.3. As a final illustrative example, Figure 2 shows a case where the `MOSQP` solver has difficulties in computing a good approximation to the Pareto front for the given budget of iterations and function evaluations.

7.2.2. Solver performance. We discuss the performance of our approach in three separate parts: the use of computational resources (number of function evaluations), the quality of the approximation found (purity metric, spread metrics, and hypervolume metric), and data profiles (number of problems solved per solver).

Computational resources. First we discuss the computational resources needed by various solvers to solve multiobjective optimization problems. We measure the

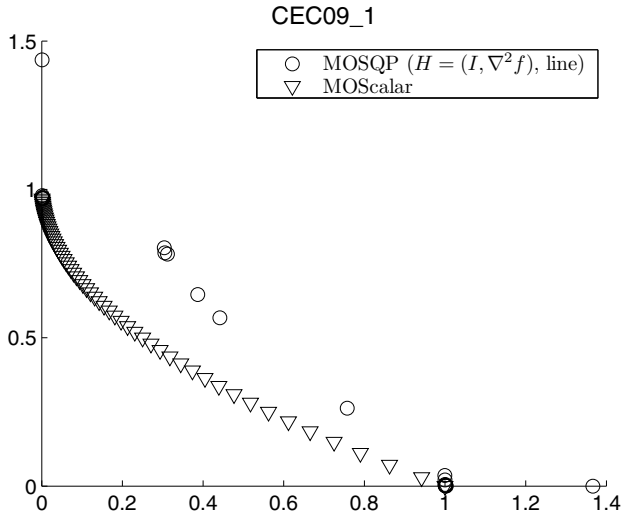

FIG. 2. *Pareto fronts obtained with the* `MOSQP` *and* `MOScalar` *for problem* `CEC09_1`. `MOSQP` *has difficulties computing a good approximation to the Pareto front.*

TABLE 3

Number of objective function, gradient, and Hessian evaluations for the bound constrained test set. Note that in our experiments, `NSGA II` *always uses* $20,000$ *function evaluations but no gradient or Hessian evaluations.*

	$\#f$			$\#\nabla f$			$\#\nabla^2 f$		
MOSQP version	max	*Avg*	min	max	*Avg*	min	max	*Avg*	min
$H = \nabla^2 f$, line	3639.5	643.7	207.0	1489.0	218.4	6.0	1483.0	209.8	2.0
$H = I$, line	3782.0	945.4	207.0	1544.0	266.7	6.0	0.0	0.0	0.0
$H = (I, \nabla^2 f)$, line	3643.5	757.2	207.0	1163.5	249.3	6.0	1041.0	130.9	0.0
$H = \nabla^2 f$, rand	4013.5	598.9	203.0	1309.0	157.5	3.0	1303.0	149.0	2.0
$H = I$, rand	12241.5	857.9	203.0	1578.0	235.5	3.0	0.0	0.0	0.0
$H = (I, \nabla^2 f)$, rand	11564.0	712.5	203.0	1544.0	199.2	3.0	652.0	102.8	0.0
MOScalar	33568.0	3383.9	200.0	15383.0	1360.7	100.0	0.0	0.0	0.0

resources needed by objective and constraint function evaluations, as well as their derivatives. Tables 3 and 4 present the maximum, average, and minimum numbers of function evaluations ($\#f$), the number of objective gradient function evaluations ($\#\nabla f$), the number of objective Hessian function evaluations ($\#\nabla^2 f$), the number of nonlinear constraint function evaluations ($\#c$), and the number of nonlinear constraint gradient evaluations ($\#\nabla c$) that variants of `MOSQP` need on the bound constrained and the linearly and nonlinearly constrained test sets, respectively. The same figures are presented for the `MOScalar` solver, considering only biobjective optimization problems. We do not provide information on the number of linear constraint evaluations performed, as the corresponding computational effort is negligible. Tables 3 and 4 show in each entry the number of function, gradient, and Hessian evaluations divided by the number of objective functions, since the solvers report the number of individual objective function evaluations. Note that in our setting `NSGA II` always performs 20,000 function evaluations and 20,000 constraint evaluations on each problem. As can be seen, all variants of `MOSQP` consistently outperform both `MOScalar` and `NSGA II`.

TABLE 4

Number of objective function, gradient, and Hessian evaluations for the linearly and nonlinearly constrained test sets. Note that in our experiments, `NSGA II` *always uses* $20,000$ *function evaluations but no gradient or Hessian evaluations.*

	$\#f$			$\#\nabla f$			$\#\nabla^2 f$		
MOSQP version	max	*Avg*	min	max	*Avg*	min	max	*Avg*	min
$H = \nabla^2 f$, line	4013.0	1057.2	230.5	1641.5	351.6	23.5	1634.0	340.8	15.0
$H = I$, line	3962.0	1427.8	249.5	1826.5	472.4	29.5	0.0	0.0	0.0
$H = (I, \nabla^2 f)$, line	3919.0	1281.2	245.5	1826.5	443.1	25.5	537.0	107.4	0.0
$H = \nabla^2 f$, rand	4401.3	1009.8	224.5	1104.0	301.8	16.5	1073.0	291.0	8.0
$H = I$, rand	3823.0	1346.9	262.5	1629.5	458.5	27.5	0.0	0.0	0.0
$H = (I, \nabla^2 f)$, rand	3745.0	1301.0	271.5	1639.5	436.7	30.5	427.0	100.1	0.0
MOScalar	49865.0	5316.9	600.0	15836.0	1813.5	300.0	0.0	0.0	0.0

	$\#c$			$\#\nabla c$		
MOSQP version	max	*Avg*	min	max	*Avg*	min
$H = \nabla^2 f$, line	4040.0	1160.8	0.0	1885.0	486.6	0.0
$H = I$, line	6751.0	1692.5	0.0	2803.0	614.0	0.0
$H = (I, \nabla^2 f)$, line	6751.0	1528.3	0.0	2418.0	568.4	0.0
$H = \nabla^2 f$, rand	9663.0	1108.3	0.0	2689.0	412.1	0.0
$H = I$, rand	14299.0	1553.9	0.0	5102.0	596.2	0.0
$H = (I, \nabla^2 f)$, rand	16431.0	1555.7	0.0	5280.0	585.9	0.0
MOScalar	49865.0	5181.9	0.0	15836.0	1746.1	0.0

Quality metrics. Next, we turn our attention to the quality of solutions provided, i.e., how good the set of points produced by various methods approximates the true Pareto front, measured by the various metrics introduced above. All `MOSQP` solver variations were able to perform well when compared with other approaches. Overall, we found that `MOSQP` ($H = (I, \nabla^2 f)$, `line`) provided the best performance, which is the reason why in what follows we focus on comparing this variant against other approaches.

Considering first classical scalarization techniques, we compare `MOScalar` against `MOSQP` on biobjective optimization test problems. In terms of quality of solutions provided, Figure 3 shows that `MOSQP` ($H = (I, \nabla^2 f)$, `line`) outperforms `MOScalar` robustness on biobjective problems with respect to the Purity metric while being slightly outperformed with respect to efficiency on the same metric. The purity metric is sensitive to the number of solvers considered in the profile, and only two solvers should be compared with each other with this metric.

Next, we compare variants of `MOSQP` with `NSGA II`. We present in Figure 3 a comparison between our implementation and `NSGA II` for all test problems from Tables 1 and 2, using the Purity metric in the performance profile. For the best `NSGA II` run we conclude from Figure 3 that `MOSQP` ($H = (I, \nabla^2 f)$, `line`) is able to solve about 75% of the problems with the best metric, while `NSGA II` solves about 45% of the problems with the best metric. Considering $\tau \to \infty$, we can further conclude that `MOSQP` ($H = (I, \nabla^2 f)$, `line`) is able to solve all problems, while `NSGA II` solves about 90% of the problems given. Note also that the `MOSQP` implementations make use of local approximation methods, and thus convergence to local optimal points is guaranteed. In contrast to this, `NSGA II` is at best associated with a probabilistic notion of convergence. This further favors the use of `MOSQP` in one of its variants.

Regarding the Δ and Γ spread metrics, we note that these metrics are independent of any solver or any given approximation to any Pareto front. Therefore, all solvers can

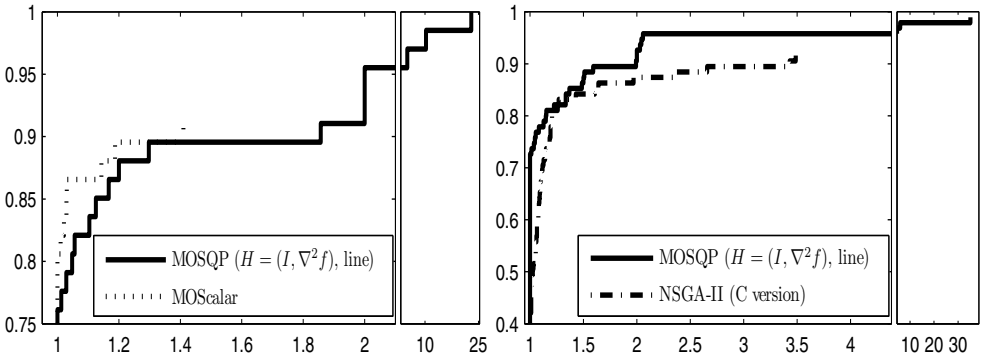

FIG. 3. *Purity performance profiles. Left: bound constrained and linearly and nonlinearly constrained biobjective test sets. Right: bound constrained and linearly and nonlinearly constrained test problems, with the* best *Pareto front returned by* `NSGA II` *over* 10 *runs.*

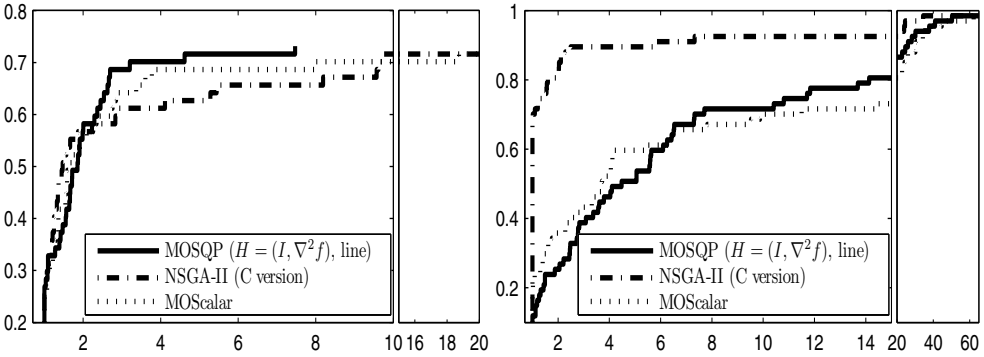

FIG. 4. *Spread average* Δ *(left) and* Γ *(right) performance profiles. Biobjective bound constrained and linearly and nonlinearly constrained test sets.*

be included in a single profile and one can consider maximum, average, and minimum values of these metrics if several runs of the same stochastic solver are given. However, for ease of exposition, we have chosen to include only a limited number of solvers in each performance profile depicted.

We present in Figure 4 performance profiles with the Δ and the Γ metric for the bound constrained and constrained test sets. Note that the Γ metric is always well defined and, therefore, we have $\rho(\tau) \to 1$ for $\tau \to \infty$ for all solvers. For these metrics, it is clear that `MOSQP` variants do not perform as well as `NSGA II`, in particular for the Γ metric. This can be explained by the fact that at present our implementation has no implicit control of the spread metric during the optimization procedure, while `NSGA II` uses certain measures to select offsprings with a better spread. Similar results were obtained with the full test set for both metrics, not depicted here as a matter of brevity.

Performance profiles for the hypervolume metric are provided in Figure 5. We can observe that `MOSQP (`$H = (I, \nabla^2 f)$`, line)` is again the most robust and efficient solver, clearly outperforming the `NSGA II` solver for the full test set.

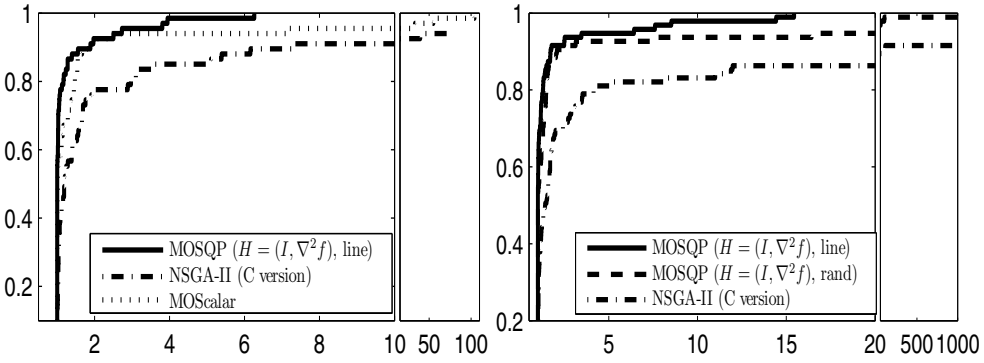

FIG. 5. *Average hypervolume performance profile for bound constrained and linearly and nonlinearly constrained test problems. Left: biobjective test problems. Right: all test problems.*

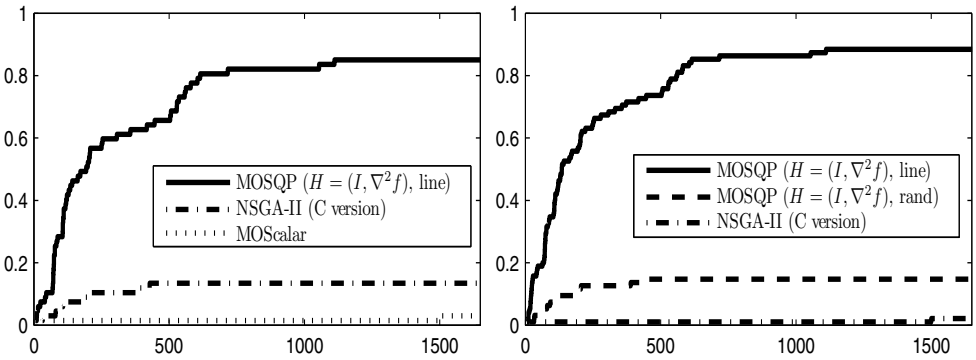

FIG. 6. *Data profiles for biobjective test problems (left) and all test problems (right).* `NSGA II` *results are reported for an average over* 10 *runs* ($\epsilon = 0.99$).

Data profiles. Let us now turn our attention to data profiles. We use again the approximated Pareto front F_p previously described in section 7.1.3, which is computed by running all solvers for 5000 function evaluations. We use $\epsilon = 0.99$ as the parameter which controls the accuracy to which a problem is considered to be solved. As we also consider stochastic solvers, we provide data profiles based on the average number of objective function evaluations over 10 runs.

Figure 6 depicts the relevant profiles. Clearly, `MOSQP` ($H = (I, \nabla^2 f)$, `line`) solves more problems than any other solver considered. Note also that all `MOSQP` variants consistently outperform `NSGA II` and `MOScalar`.

7.3. Real-world applications. To complete our numerical test, we provide results on difficult real-world problems from space engineering. We focus on two problems from trajectory optimization, as trajectory optimization plays a substantial role in overall mission design, and many trajectory optimization problems are known to be exceptionally difficult in their numerical treatment.

First, we consider the Cassini-1 multiple gravity assist problem; see [1] for further details. Here we adapt this problem for biobjective optimization. The first objective

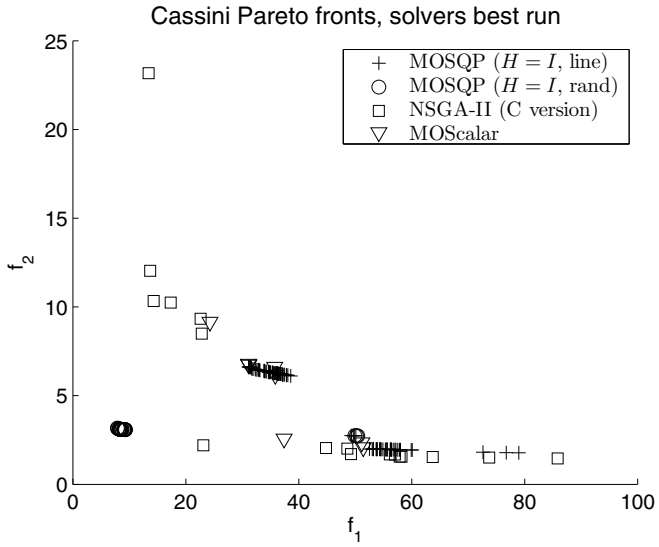

FIG. 7. *Best Pareto front approximations for the Cassini problem obtained by* `MOSQP`, `MOScalar`, *and* `NSGA II`.

function, f_1, is the total ΔV, a widely considered performance measure in space engineering. For a given trajectory design, ΔV is the "change in velocity," the amount of impulse needed to perform the mission and as such a measure for the amount of fuel needed. The second objective function, f_2, is the squared total travel time to reach the final destination, a Saturn orbit. The travel time has been squared to form the objective to penalize exceptionally long trajectories. First derivatives of objective function f_1 and the constraints have been computed by finite differences. For the second objective function, exact derivatives were readily available. All solvers were used with the same algorithmic parameters as in the previous section. Again, stochastic solvers have been executed 10 times and we are reporting results for their "best run," i.e., the run with the least number of points dominated by points produced by other runs of the same algorithm. Figure 7 shows the Pareto fronts thus obtained. We can observe that `MOSQP` ($H = I$, `rand`) provides high-quality Pareto points around the "knee" of the Pareto front but does not provide an approximation with a broad spread of points. Both `NSGA II` and `MOSQP` ($H = I$, `line`) provide approximations to Pareto points with small f_2 function values.

For the second real-word problem we consider the Rosetta space mission [1], another spacecraft trajectory design problem with multiple gravity assists but with the goal to reach a different celestial body (in this case Comet 67P/Churyumov-Gerasimenko instead of Saturn). Objective functions are the same as for the Cassini mission, with the same availability of derivatives. As before, all solvers were used with the same algorithmic parameters as in the previous section.

As can be seen from Figure 8, good results are obtained by `MOSQP` ($H = I$, `line`), whose approximation to the Pareto front computed is well spread, except near what appears to be the minimum of f_2, where `MOSQP` ($H = I$, `rand`) provides a better approximation, and near what appears to be the minimum of f_1, where `MOScalar` computes a few good points. Interestingly, the situation appears to be similar to that depicted in Figure 1, showing again the problematic behavior of `MOScalar`. `NSGA II`

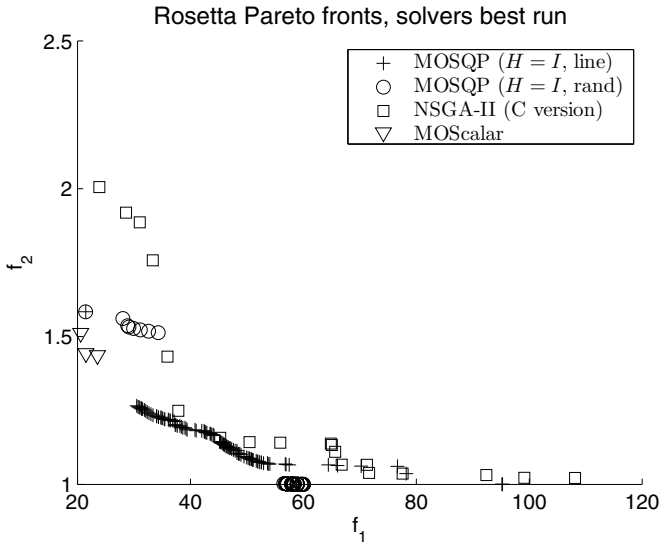

FIG. 8. *Best Pareto front approximations for the Rosetta problems obtained by* `MOSQP`*,* `MOScalar`*, and* `NSGA II`*.*

does not compute any point that is not dominated by another point computed by one of the `MOSQP` variants.

For both problems, variants of `MOSQP` are able to approximate important parts of the Pareto fronts when compared to `MOScalar` and `NSGA II`, thus showing their competitiveness when addressing hard real-world problems.

8. Conclusions. We develop and analyze a novel method for constrained multiobjective optimization. Under appropriate differentiability assumptions on the objective and constraint functions we can show local and global convergence to local Pareto critical points.

We provide a publicly available implementation of the proposed algorithm, which we denote by `MOSQP`. The algorithmic framework is flexible enough to be investigated further to improve algorithmic performance, including the use of quasi-Newton-type methods and adaptive measures taken to improve the spread of the computed approximation to the Pareto front.

Extensive numerical results are provided for our implementation, and we compare several variants of our implementation with the state-of-the-art `NSGA II` solver and an implementation of a classical scalarization approach for biobjective problems. Numerical results indicate that our implementation performs well for a significant number of test problems, making the proposed solver the preferred solution framework for multiobjective optimization problems when derivatives of objective and constraint functions are available.

Acknowledgment. The authors are in debt to the two anonymous referees, whose comments and suggestions have substantially improved the paper.

REFERENCES

[1] ADVANCED CONCEPTS TEAM, *Multiple Gravity Assist with One Deep Space Manouvre (MGA-1DSM)*, http://www.esa.int/gsp/ACT/inf/projects/gtop/mga1dsm.html, 2014.

[2] AMPL Optimization LLC, *AMPL: A Modeling Language for Mathematical Programming*, http://www.ampl.com, 2012.
[3] J. F. Bonnans, J. C. Gilbert, C. Lemaréchal, and C. A. Sagastizábal, *Numerical Optimization, Theoretical and Practical Aspects*, 2nd ed., Springer, Berlin, 2006.
[4] L. Bradstree, *The Hypervolume Indicator for Multi-Objective Optimisation: Calculation and Use*, Ph.D. thesis, Department of Computer Science & Software Engineering, The University of Western Australia, Perth, Australia, 2011.
[5] Y. Collette and P. Siarry, *Multiobjective Optimization: Principles and Case Studies*, Springer, Berlin, Heidelberg, 2004.
[6] A. L. Custódio, J. F. A. Madeira, A. I. F. Vaz, and L. N. Vicente, *Direct multisearch for multiobjective optimization*, SIAM J. Optim., 21 (2011), pp. 1109–1140, https://doi.org/10.1137/10079731X.
[7] I. Das and J. E. Dennis, *Normal-boundary intersection: A new method for generating the Pareto surface in nonlinear multicriteria optimization problems*, SIAM J. Optim., 8 (1998), pp. 631–657, https://doi.org/10.1137/S1052623496307510.
[8] K. Deb, *Multi-objective genetic algorithms: Problem difficulties and construction of test problems*, Evol. Comput., 7 (1999), pp. 205–230.
[9] E. D. Dolan and J. J. Moré, *Benchmarking optimization software with performance profiles*, Math. Program., 91 (2002), pp. 201–213.
[10] G. Eichfelder, *Adaptive Scalarization Methods in Multiobjective Optimization*, Springer-Verlag, Berlin, Heidelberg, 2008.
[11] G. Eichfelder, *An adaptive scalarization method in multiobjective optimization*, SIAM J. Optim., 19 (2009), pp. 1694–1718, https://doi.org/10.1137/060672029.
[12] E. Eskow and R. B. Schnabel, *Algorithm* 695: *Software for a new modified Cholesky factorization*, ACM Trans. Math. Software, 17 (1991), pp. 306–312.
[13] J. Fliege, *Gap-free computation of Pareto-points by quadratic scalarizations*, Math. Methods Oper. Res., 59 (2004), pp. 69–89.
[14] J. Fliege, L. M. G. Drummond, and B. F. Svaiter, *Newton's method for multiobjective optimization*, SIAM J. Optim., 20 (2009), pp. 602–626, https://doi.org/10.1137/08071692X.
[15] A. Göpfert and R. Nehse, *Vektoroptimierung*, Teubner Verlagsgesellschaft, Leipzig, 1990.
[16] J. Knowles, D. Corne, and K. Deb, eds., *Multiobjective Problem Solving from Nature: From Concepts to Applications*, Springer, Berlin, Heidelberg, 2008.
[17] MATLAB, *version* 7.10.0 *(R*2010*a)*, The MathWorks Inc., Natick, MA, 2010.
[18] D. Q. Mayne and E. Polak, *A superlinearly convergent algorithm for constrained optimization problems*, Math. Programming Stud., 16 (1982), pp. 45–61.
[19] J. J. Moré and S. M. Wild, *Benchmarking derivative-free optimization algorithms*, SIAM J. Optim., 20 (2009), pp. 172–191, https://doi.org/10.1137/080724083.
[20] H. Nakayama, Y. Sawaragi, and T. Tanino, *Theory of Multiobjective Optimization*, Academic Press, Orlando, FL, 1985.
[21] R. B. Schnabel and E. Eskow, *A revised modified Cholesky factorization algorithm*, SIAM J. Optim, 9 (1999), pp. 1135–1148, https://doi.org/10.1137/S105262349833266X.
[22] A. I. F. Vaz and L. N. Vicente, *A particle swarm pattern search method for bound constrained global optimization*, J. Global Optim., 39 (2007), pp. 197–219.
[23] A. Wächter and L. T. Biegler, *On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming*, Math. Program., 106 (2005), pp. 25–57.
[24] Q. Zhang, A. Zhou, S. Zhaoy, P. N. Suganthany, W. Liu, and S. Tiwariz, *Multiobjective Optimization Test Instances for the CEC* 2009 *Special Session and Competition*, Tech. Report CES-487, The School of Computer Science and Electronic Engineering, University of Essex, Colchester, UK, 2009.
[25] E. Zitzler, *Evolutionary Algorithms for Multiobjective Optimization: Methods and Applications*, Vol. 63, Shaker, Ithaca, NY, 1999.
[26] E. Zitzler and L. Thiele, *Multiobjective optimization using evolutionary algorithms – a comparative case study*, in Parallel Problem Solving from Nature – PPSN V, Lecture Notes in Comput. Sci. 1498, A. E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, eds., Springer, Berlin, Heidelberg, 1998, pp. 292–301.