A SQP type method for constrained multiobjective optimization

Jörg Fliege* and A. Ismael F. Vaz†

May 28, 2015

Abstract

We propose an SQP type method for constrained nonlinear multiobjective optimization. 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. 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, either in the quality of the approximated Pareto front or in the computational effort necessary to compute the approximation.

AMS subject classifications: 90C29 Multi-objective and goal programming, 90C55 Methods of successive quadratic programming type, 90C30 Nonlinear programming
Keywords: Sequential Quadratic Programming, constrained multiobjective optimization.

1 Introduction

In this paper we are addressing the following (inequality and equality) constrained multiobjective optimization problem:

$$\min_{x\in\Omega} \quad f(x) = (f_1(x), \ldots, f_m(x))^T \tag{1}$$

*School Of Mathematics, University Of Southampton, Southampton SO17 1BJ, U.K.; Email: j.fliege@soton.ac.uk

†ALGORITMI research center, University of Minho, Campus of Gualtar, 4710 - 057 Braga, Portugal; Email: aivaz@dps.uminho.pt

with

$$\Omega = \{x \in [\ell, u] \subseteq \mathbb{R}^n : g_j(x) \leq 0, j = 1, \ldots, p, \quad h_j(x) = 0, j = 1, \ldots, q\}, \tag{2}$$

where $\ell \in (\mathbb{R} \cup \{-\infty\})^n$, $u \in (\mathbb{R} \cup \{+\infty\})^n$, $f_i(x) : \mathbb{R}^n \to \mathbb{R}$, $i = 1, \ldots, m$, are assumed to be twice continuous differentiable objective functions all bounded from below, and $g_j(x) : \mathbb{R}^n \to \mathbb{R}$, $j = 1, \ldots, p$, and $h_j(x) : \mathbb{R}^n \to \mathbb{R}$, $j = 1, \ldots, q$, are assumed once continuous differentiable (inequality and equality) constraint functions. We allow $p = 0$ and $q = 0$ meaning that problem (1) may be unconstrained.

Although applications for this type of problems can be found in several engineering and science fields (see, for example, in [3, Part III] and [11]), implementation of available algorithms (solvers) are somehow limited to evolutionary algorithms (heuristics) or deterministic single point algorithms based on, for example, scalarization approaches (see the Optimization Tree for some available solvers `http://plato.asu.edu/sub/multiobj.html`). The former usually lack a convergence proof and have a low rate of convergence (as in the single objective case), but usually provide an approximation to the Pareto front, while the later rely on derivative based algorithms with a multistart procedure (e.g., setting different scalarization parameters) to obtain an approximation to the Pareto front.

Herein we propose an algorithm for (constrained) multiobjective optimization that tracks and updates a finite set of points X_k (at iteration k) from one iteration to the next, in opposition to standard algorithms for single-objective optimization where just one point $x_k \in \mathbb{R}^n$ is used. The proposed mechanism, based in two main stages, allows the algorithm to end with an approximation to the Pareto front. The first main stage is used to enrich the set of points X_k with additional nondominated points, while the second main stage is used to drive the set of points X_k 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 remaining of the paper is organized as follows. In the next section we introduce some definitions and motivation for the proposed algorithm. Included in the motivation of the proposed algorithm, we develop some ideas on how to handle constraints, how to improve a nondominated set of points, and how to drive a set of points to local Pareto optimality. Section 3 presents our selection for the optimization subproblems in both algorithmic stages, supporting the full algorithm described in Section 4. Section 6 describes some details used in our implementation. Before concluding, in Section 8, we present extensive numerical results in Section 7.

2 Definitions and motivation for the algorithm

When several objective functions are present, like in our case for problem (1), obtaining a point that simultaneously minimizes every function is usually not possible, since it is common to have several conflicting objective functions. There are several approaches to address such a problem, like those reported in [3], for example. This paper relies on the

Pareto dominance concept for comparing two points. To briefly describe the concept, we make use of two partial orders induced by the cones

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

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) when $f(x) \prec_f f(y)$. We further say that x weakly dominates y ($x \preceq y$) when $f(x) \preceq_f f(y)$. 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.*,

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

The concept of Pareto dominance is also used to characterize global and local optimality, by defining a 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 $\nexists y \in \Omega$ such that $y \preceq x_*$ 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 x_* is called a local Pareto minimizer of f.*

The (global) Pareto front is the set of (global) Pareto minimizers.

The goal of our algorithm is to determine a set of efficient points or Pareto optimal points of f, *i.e.*, to determine a set X^* where each $x^* \in X^*$ is such that

$$\nexists y \in \Omega : \quad f(y) \preceq_f f(x^*) \text{ and } f(y) \neq f(x^*),$$

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

For the rest of this section, we simplify the notation slightly by subsuming the box constraints on x into the constraint functions g (or into g and h, if $u_k = \ell_k$ for an $k \in \{1, \ldots, n\}$). 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 [10]. The criticality condition, for $\bar{x}$, can be seen as the nonexistence of a vector (direction) $d \in \mathbb{R}^n$ along which all the objective functions are decreasing while feasibility is kept. More formally, we say that a point is *Pareto critical* if for all $d \in \mathbb{R}^n$ there exists a $j = j(d) \in \{1, \ldots, m\}$ such that

$$\nabla f_j(\bar{x})^T d \geq 0, \quad \nabla g_i(\bar{x})^T d \leq 0, \quad i \in \mathcal{A}, \quad \nabla h_k(\bar{x})^T d = 0, \quad k = 1, \ldots, q$$

where $\mathcal{A} = \{i \in \{1, \ldots, p\} : g_i(\bar{x}) = 0\}$ is the inequality constraints active set.

In the unconstrained case, we can write this notion equivalently as

$$\mathcal{R}(\nabla f(\bar{x})) \cap (-\mathbb{R}^m_{++}) = \emptyset$$

where $\mathcal{R}(\nabla f(\bar{x}))$ denotes the range of the Jacobian $\nabla f(\bar{x}) \in \mathbb{R}^{m \times n}$.

Likewise, we call a point $\bar{x}$ *strongly Pareto critical* if for all $d \in \mathbb{R}^n$ with

$$\nabla g_i(\bar{x})^T d \leq 0, \quad i \in \mathcal{A}, \quad \nabla h_k(\bar{x})^T d = 0, \quad k = 1, \ldots, q$$

either we have $\nabla f_j(\bar{x})^T d = 0$ for all objectives j, or there exists a $j = j(d) \in \{1, \ldots, m\}$ such that

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

(Again, $\mathcal{A}$ is the same active constraint set as above.)

In the unconstrained case, this notion is equivalent to

$$\mathcal{R}(\nabla f(\bar{x})) \cap (-\mathbb{R}^m_{+} \setminus \{0\}) = \emptyset.$$

Denote the set of 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

$$SC \subseteq PC$$

as well as

$$P \subseteq PC.$$

Note that a point is strongly Pareto critical if and only if for all feasible directions $d \neq 0$, either $(\nabla f_j(x))^T d = 0$ for all indices j, or there exists an index $i = i(d) \in \{1, \ldots, m\}$ with $(\nabla f_i(x))^T d > 0$. The latter means

$$\max_{i=1}^{m} (\nabla f_i(x))^T d > 0.$$

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 $x = 0 \in P$, we have $DF(x) = [0; -2]$, and so clearly $x \notin SC$. We will show below that, under some mild condition, missing critical points of particular objective functions is the worst that can happen when moving from Pareto critical points to strongly 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

$$\nabla g_i(\bar{x})^T d \leq 0, \quad i \in \mathcal{A}, \quad \nabla h_k(\bar{x})^T d = 0, \quad k = 1, \ldots, q$$

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

Define the set of *Geoffrion-optimal points* in the usual way, *i.e.*

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

We have the following theorem.

Theorem 2.1 *Efficiency and criticality are related as follows. Under the assumption of* $f \in C^1(\mathbb{R}^n, \mathbb{R}^m)$ *we have*

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

and in the convex case

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

Proof.
Obviously,

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

It is well-known that

$$G \subseteq P$$

and thus also $G \subseteq PC$. In addition, if all functions f_i, g_j are convex and h is affine linear, we have

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

Now let $x \in G \setminus C$. Then, there exists an $w \in \mathbb{R}^m_{++}$ such that x minimizes $\sum_{i=1}^m w_i f_i$ over Ω. Let $d \in \mathbb{R}^n \setminus \{0\}$ be a feasible direcion of descent as above.

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

As such, we either have $(\nabla f_i(x))^T d = 0$ for each term of the sum, or there exists an index i with $(\nabla f_i(x))^T d \neq 0$. In the latter case, either $(\nabla f_i(x))^T d > 0$, or there must exist a further index j with $(\nabla f_j(x))^T d > 0$. As a consequence, $x \in SC$.

■

In the use of Theorem 2.1 we can thus concentrate on PC and SC, as requested by our constraint handling technique described in the next subsection. Additionally, the proposed algorithm relies on two different mechanisms that are described in the two following subsections. The first one consists in local improvements of a set of nondominated points, while the second one consists in driving a set of nondominated points to (local) Pareto optimality.

2.1 Handling constraints

Each inequality and equality constraint may be observed as an additional objective to optimize. While objective functions are to be minimized, constraint functions must achieve a specific value at the solution. Defining

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

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

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

For constrained optimization problems we further extend the weak Pareto dominance concept, saying that x weakly dominates y ($x \preceq y$) 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

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

The strong dominance concept is defined in a similar way.

Clearly, we have for feasible x and y

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

while an infeasible point never weakly dominates a feasible point, and a feasible point may weakly dominate an infeasible point.

Bound constraints are omitted in the theoretical results, as they can be considered as regular inequality constraints. From a practical point of view, bound constraints do not pose any algorithmic difficulty and, therefore, there is no need to consider them in this dominance concept extension. All points considered by our proposed algorithms always consider bound feasible points and bound feasibility is always enforced during the optimization process.

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

In what follows we need the concept of a merit function that will measure the progress we can make along a particular direction. We will use the following variant of the classical ℓ_1 merit function:

$$\phi(x, \sigma) = \sum_{i=1}^{m} f_i(x) + \sigma \left(\sum_{i=1}^{p} (g_i(x))^+ + \sum_{i=1}^{q} |h_i(x)| \right). \tag{3}$$

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

$$\phi_j(x, \sigma) = f_j(x) + \sigma \sum_{i=1}^{p} (g_i(x))^+ + \sigma \sum_{i=1}^{q} |h_i(x)|.$$

Note that ϕ as well as all ϕ_j are usually nondifferentiable, however they have directional derivative $\phi'(x, \sigma; d)$ at x in any direction $d \neq 0$. See Lemma 16.4 in [1] for additional details.

2.2 Improving a nondominated set of points

Let X be a set of (feasible or infeasible) nondominated points and $d \in \mathbb{R}^n$ be a (sufficient) descent direction for at least one objective function at one point in X that, locally, forces feasibility, *i.e.*, given $x_k \in X$, there exists a j such that

$$\nabla f_j(x_k)^T d < 0, \tag{4a}$$

$$g_i(x_k) + \nabla g_i(x_k)^T d \leq 0, \quad i = 1, \ldots, p \tag{4b}$$

$$h_l(x_k) + \nabla h_l(x_k)^T d = 0, \quad l = 1, \ldots, q \tag{4c}$$

and $\exists \bar{t}$, such that $\forall t \in]0, \bar{t}]$,

$$\phi_j(x_k + td, \sigma) \leq \phi_j(x_k, \sigma) + \mu t \phi_j'(x_k, \sigma; d) \tag{4d}$$

with $\mu \in]0, 1[$ and a sufficiently large σ value.

Theorem 2.2 *Given $x_k \in X$ and a descent direction d satisfying equations (4) then there exists a $t_0 > 0$ such that*

$$\bar{f}(x_k) \not\preceq_{\bar{f}} \bar{f}(x_k + td), \quad \forall t \in]0, t_1], \tag{5}$$

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

$$a_i := \begin{cases} 1 & : \quad g_i(x) = 0, \\ 0 & : \quad otherwise, \end{cases} \qquad b_j := \begin{cases} 1 & : \quad h_j(x) = 0, \\ 0 & : \quad otherwise, \end{cases}$$

$(i = 1, \ldots, p,\ j = 1, \ldots, q)$, and $r := (a^\top, b^\top)^\top$. 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 $t \in]0, t_0]$, i.e., *feasibility can be arbitrarily improved for $x_k + td$.*

Proof. The proof for (5) follows trivially from the differentiability of $f_j(x)$ and (4a), since we have that

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

implying that there exists $t_1 > 0$ such that

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

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

- If $g_i(x_k) > 0$, and since we have $\nabla g_i(x_k)^T d \leq -g_i(x_k)$, then $\nabla g_i(x_k)^T d < 0$ and from the differentiability of g_i we get that there exists a $t_2 > 0$ such that $g_i(x_k + td) \leq g_i(x_k)$, $\forall t \in]0, t_2]$ (d is a descent direction for g_i), *i.e.*, we have that $\forall t \in]0, t_2]$, $g_i^+(x_k + td) \leq g_i^+(x_k)$.

  On the other hand, if $g_i(x_k) < 0$, we have $g_i(x_k + td) < 0$ for all $t > 0$ smaller than some $t_3 > 0$, and thus $g_i^+(x_k + td) = g_i^+(x_k) = 0$, $\forall t \in]0, t_3]$.

  The last case is when $g_i(x_k) = 0$, which results in

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

  If the limit attains a negative value then we have $g_i(x_k + td) < 0$, $\forall t \in]0, t_4]$ and $g_i^+(x_k + td) = g_i^+(x_k) = 0$, $\forall t \in]0, t_4]$. It remains to prove the case where $g_i(x_k + td)$ approaches zero from above, *i.e.*, $\lim_{t \to 0} g_i(x_k + td) = 0$ and $g_i(x_k + td) > 0$. For this case we recall that $\forall \epsilon$ there exists $t_5 > 0$ such that $|g_i(x_k + td)| < \epsilon$, $\forall t \in]0, t_5]$, resulting in $g_i^+(x_k + td) - g_i^+(x_k) < \epsilon$, $\forall t \in]0, t_5]$.

- We also consider three cases for the equality constraints. From (4c) we have $\nabla h_j(x_k)^T d = -h_j(x_k)$, resulting in

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

If $h_j(x_k) > 0$ then there exists a $t_6 > 0$ such that $h_j(x_k + td) - h_j(x_k) < 0$, $\forall t \in]0, t_6]$, resulting in $h_j(x_k + td) < h_j(x_k)$. We may further assume, without loss of generality, that t_6 is such that $h_j(x_k + td) \geq 0$, getting that $|h_j(x_k + td)| < |h_j(x_k)|$, $\forall t \in]0, t_6]$. If $h_j(x_k) < 0$ then there exists a $t_7 > 0$ such that $h_j(x_k + td) - h_j(x_k) > 0$, $\forall t \in]0, t_7]$, resulting in $h_j(x_k + td) > h_j(x_k)$. We may further assume, without loss of generality, that t_7 is such that $h_j(x_k + td) \leq 0$, getting that $|h_j(x_k + td)| < |h_j(x_k)|$, $\forall t \in]0, t_7]$.

Last, if $h_j(x_k) = 0$ then

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

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

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

The computation of a particular search direction that satisfies conditions (4) is postponed to the next section.

By taking all point in X and computing a search direction d that locally improves each point with respect to each individual objective function allows to defined a new set X formed by all nondominated new computed points, *i.e.*, the set X is enriched with new nondominated points that are spread in the Pareto front.

2.3 Driving a nondominated set of points to local Pareto optimality

While the previous section provide us a strategy to enrich a given set X with new nondominated points, there is no guarantee that the set X is (only) formed by efficient Pareto points. However, the set X, hopefully, provides us a reasonable approximation to the Pareto front. While the search direction d addressed in the previous section is a sufficient descent direction for at least one objective function there is no control on what happens with the remaining objective function values. In order to drive a point $x_k \in X$ to Pareto optimality we need to compute a search direction d that satisfies the following conditions.

There exists a j_0 such that

$$\nabla f_{j_0}(x_k)^T d < 0, \tag{6a}$$

$$\nabla f_j(x_k)^T d \leq 0, \quad j \in \{1, ..., m\} \setminus \{j_0\} \tag{6b}$$

$$g_i(x_k) + \nabla g_i(x_k)^T d \leq 0, \quad i = 1, \ldots, p \tag{6c}$$

$$h_l(x_k) + \nabla h_l(x_k)^T d = 0, \quad l = 1, \ldots, q \tag{6d}$$

and $\exists \bar{t}$, such that $\forall t \in]0, \bar{t}]$,

$$\phi(x_k + td, \sigma) \leq \phi(x_k, \sigma) + \mu t \phi'(x_k, \sigma; d) \tag{6e}$$

with $\mu \in]0, 1[$ and a sufficiently large σ value.

Theorem 2.3 *Given $x_k \in X$ and a descent direction d satisfying equations (6), define $w \in \mathbb{R}^m$ by*

$$w_j := \begin{cases} 1 & : \quad \nabla f_j^\top(x_k) d = 0, \\ 0 & : \quad \textit{otherwise} \end{cases}$$

$(j = 1, \ldots, m)$ and $\bar{r} \in \mathbb{R}^{m+p+q}$ by $\bar{r} := (w^\top, a^\top, b^\top)^\top$, with a, b defined as in Theorem 2.2. Then, for all $\epsilon > 0$ there exists a $t_0 > 0$ such that

$$\bar{f}(x_k + td) \preceq_{\bar{f}} \bar{f}(x_k) + \epsilon \bar{r}, \quad \forall t \in]0, t_0], \tag{7}$$

i.e., *$x_k + td$ arbitrarily weakly dominates x_k.*

Proof. The proof is similar to the one provided for Theorem 2.2. Equation (6a) results in $f_{j_0}(x_k + td) < f_{j_0}(x_k)$. Equations (6c) and (6d) result in $\Phi(x_k) - \Phi(x_k + td) + \epsilon r \in \mathbb{R}^{p+q}_+$. From equation (6b) we obtain that for $j = \{1, \ldots, m\} \setminus \{j_0\}$,

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

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

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

On the other hand, if $\nabla f_j^T(x_k) d = 0$ then for all $\epsilon > 0$ there exists a $t_{10} > 0$ such that

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

and the proof follows trivially. ■

The proposal of a particular technique to compute the search direction d that satisfies equations (6) is described in the next section.

3 The search directions

There are several possibilities to obtain a search direction with properties described in equations (4) and (6). The herein selected strategy consists in using quadratic approximations to the objective functions and linear approximations to the equality and inequality constraints.

A search direction in equation (4), w.r.t. the ith objective function, can be obtained by solving the following quadratic optimization problem.

$$\begin{array}{ll} \min\limits_{d\in\mathbb{R}^n} & \nabla f_i(x_k)^T d + \dfrac{1}{2} d^T H_i d \\ \text{s.t.} & g_j(x_k) + \nabla g_j(x_k)^T d \leq 0, \quad j = 1, \ldots, p \\ & h_l(x_k) + \nabla h_l(x_k)^T d = 0, \quad l = 1, \ldots, q \\ & \ell \leq x_k + d \leq u \end{array} \tag{8}$$

where H_i is a symmetric positive definite matrix.

The following results relate problem (8) with the search direction (4).

Lemma 3.1 *For each $i = 1, \ldots, m$, let $\bar{d}$ be an optimal solution to problem (8). Then $\bar{d}$ satisfies the following conditions.*

1. *If x_k is feasible for problem (1) then $\nabla f_i(x_k)\bar{d} \leq 0$.*
2. *Let x_k be infeasible for problem (1) and let r be defined as in Theorem 2.2. Then for all $\epsilon > 0$ there exists a $\bar{t} > 0$ such that $\Phi(x_k) - \Phi(x_k + t\bar{d}) + \epsilon r \in \mathbb{R}^{p+q}_+$, $\forall t \in]0, \bar{t}]$.*
3. *Let $\bar{d} \neq 0$, let r be defined as in Theorem 2.2 and define $\hat{r} \in \mathbb{R}^{m+p+q}$ by $\hat{r} := (0_m^\top, r^\top)^\top$, with $0_m \in \mathbb{R}^m$. 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 (8) and $\nexists d$ feasible such that $\nabla f_i(x_k)^T d < 0$,* i.e., *no further improvement for the objective function f_i is possible.*
5. *If $d \neq 0$ then there exists a $\bar{t}$ such that for all $t \in [0, \bar{t}]$*

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

for $\mu \in]0, 1[$ and a sufficiently large σ.

Proof. We prove the conditions in the same enumerated order.

1. This results follows trivially by noting that if x_k is feasible to problem (1) then $d = 0$ is a feasible point for problem (8), and $\nabla f_i(x_k)^T\bar{d} \leq \nabla f_i(x_k)^T\bar{d} + \frac{1}{2}\bar{d}^T H_i\bar{d} \leq \nabla f_i(x_k)^T d + \frac{1}{2} d^T H_i d = 0$, since H_i is a positive definite matrix.
2. If x_k is infeasible to problem (1) then there exists a j or 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 2.2 we have that for all $\epsilon > 0$ there exists a $\bar{t}$ such that $g_j(x_k + t\bar{d}) < g_j(x_k)$ or $|h_l(x_k + t\bar{d})| < |h_l(x_k)|$, $\forall t \in]0, \bar{t}]$.

3. The result follows from item 1, item 2, and Theorem 2.2. If $\bar{d} \neq 0$ and x_k is feasible to problem (1) then $\bar{d}^T H_i \bar{d} > 0$, since H_i is positive definite, and using item 1 results in $\nabla f_i(x_k)^T \bar{d} < 0$, *i.e.* there exists a $\bar{t}$ such that $f(x_k + t\bar{d}) < f(x_k)$, $\forall t \in]0, \bar{t}]$. This result follows trivially by noting that the case where $\bar{d} \neq 0$ and x_k is infeasible falls in the case described in item 2.

4. Clearly if $\bar{d} = 0$ then x_k is feasible for problem (1) (since we get $g_j(x_k) \leq 0$, $j = 1, \ldots, p$ and $h_l(x_k) = 0$, $l = 1, \ldots, q$). We prove 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 to problem (1) we, trivially, have that $t\tilde{d}$, for $0 \leq t \leq 1$ is feasible to problem (8) 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 $\tilde{t}\left(\nabla f_i(x_k)^T \tilde{d} + \tilde{t}\frac{1}{2}\tilde{d}^T H_i \tilde{d}\right) < 0$, which contradicts $\bar{d} = 0$ (since $\tilde{t}\tilde{d}$ is feasible and attains a lower objective function value).

5. By noting that problem (8) corresponds to the standard SQP step w.r.t. the ith objective function (see [1], equation (17.2)) the results follows trivially from Proposition 17.1 in [1]. We further clarify that $d \neq 0$ implies the non criticality of x_k, H_i is symmetric positive definite by construction, and by σ large enough we mean that σ is no lower than the absolute value of the largest Lagrange multiplier.

■

A search direction for d in equation (6) can be obtained by solving the following nonlinear (quadratic constrained) optimization problem.

$$\begin{aligned} \min_{(v,\vartheta)\in\mathbb{R}^{n+m}} \quad & \sum_{i=1}^{m} \vartheta_i \\ \text{s.t.} \quad & \nabla f_i(x_k)^T v + \frac{1}{2} v^T H_i v \leq \vartheta_i, \quad i = 1, \ldots, m \\ & g_j(x_k) + \nabla g_j(x_k)^T v \leq 0, \quad j = 1, \ldots, p \\ & h_l(x_k) + \nabla h_l(x_k)^T v = 0, \quad l = 1, \ldots, q \\ & \ell \leq x_k + v \leq u \end{aligned} \tag{9}$$

where H_i is a symmetric positive definite matrix.

Problem (9) provides the simultaneous minimization of quadratic approximation to all objective functions, while feasibility is enforced. Some properties for problem (9) are addressed in the following lemma.

Lemma 3.2 *Let $(\bar{\vartheta}, \bar{v})$ be a solution to problem (9). Then, the following hold.*

1. *If x_k is feasible to problem (1) then $\bar{\vartheta} \leq 0$ (componentwise).*

2. *Let x_k be infeasible to problem (1) and define $r \in \mathbb{R}^{p+q}$ as in Theorem 2.2. Then for every $\epsilon > 0$ there exists a $\bar{t} > 0$ such that $\Phi(x_k) - \Phi(x_k + t\bar{v}) + \epsilon r \in \mathbb{R}^{p+q}_+$, $\forall t \in]0, \bar{t}]$.*

3. *Let $\bar{v} \neq 0$, let r be defined as in Theorem 2.2 and define $\hat{r} \in \mathbb{R}^{m+pq+q}$ by $\hat{r} := (0_m^\top, r^\top)^\top$, with $0_m \in \mathbb{R}^m$. Then, for all $\epsilon > 0$ there exists a $\bar{t}$ such that*

$$\bar{f}(x_k) \not\prec_{\bar{f}} \bar{f}(x_k + t\bar{v}) - \epsilon\hat{r}, \quad \forall t \in]0, \bar{t}].$$

4. *Let $\hat{r} \in \mathbb{R}^{m+p+q}$ be defined as above. If $\bar{v} \neq 0$ and x_k is feasible to problem (1), then for all $\epsilon > 0$ there exists a $\bar{t}$ such that*

$$\bar{f}(x_k + t\bar{v}) - \epsilon\hat{r} \preceq_{\bar{f}} \bar{f}(x_k), \quad \forall t \in]0, \bar{t}].$$

5. *If $\bar{v} = 0$ then x_k is feasible and $\nexists v$ feasible such that $\nabla f_i(x_k)^T v < 0$, $i = 1, \ldots, m$,* i.e. *x_k is Pareto critical.*

6. *If $d \neq 0$ then there exists a $\bar{t}$, such that $\forall t \in]0, \bar{t}]$,*

$$\phi(x_k + td, \sigma) \leq \phi(x_k, \sigma) + \mu t \phi'(x_k, \sigma; d)$$

with $\mu \in]0, 1[$ and a sufficiently large σ value.

Proof.

1. If x_k is feasible to problem (1) then $(\bar{v}, \bar{\vartheta}) = 0$ is a feasible point to problem (9). Since we are minimizing the sum of all ϑ_i, $i = 1, \ldots, m$, the result follows trivially.

2. This item follows the same reasoning as in item 2 of Theorem 3.1 proof by taking $\bar{v}$ as $\bar{d}$.

3. This item follows a similar reasoning as in item 3 of Theorem 3.1 proof. If $\bar{v} \neq 0$ and x_k is feasible to problem (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 $f_i(x_k + t\bar{v}) < f_i(x_k)$, $i = 1, \ldots, m$, $\forall t \in]0, \bar{t}]$. Noting that the case when x_k is infeasible to problem (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 to problem (1) then there exists a t_1 such that $f_i(x_k + t\bar{v}) < f_i(x_k)$, $i = 1, \ldots, m$, $\forall t \in]0, t_1]$. Noting that for all $\epsilon > 0$ there exists a t_2 such that $\Phi(x_k + t\bar{v}) - \epsilon \leq 0$, for all $t \in]0, t_2]$ the result follows, 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 to problem (1). Clearly $\bar{v} = 0$ results in $\bar{\vartheta}_i = 0$, $i = 1, \ldots, m$. The remaining claim is proved by contradiction. Suppose, by contradiction, that there exists a feasible $\tilde{v}$ such that $\nabla f_i(x_k)^T \tilde{v} < 0$, for at least one $i \in \{1, \ldots, m\}$. Therefore we have that $t\tilde{v}$, $0 \leq t \leq 1$, is also feasible to problem (9) (since x_k is feasible to problem (1)) and there exists a $\tilde{t} > 0$ such that $\tilde{t}\frac{1}{2}\tilde{v}^T H_i \tilde{v} < -\nabla f_i(x_k)^T \tilde{v} (> 0)$ 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 attain a lower objective function value than $\bar{v}$. Pareto criticality arrives from the fact that there is no search direction that can, locally, improve at least one objective function without increasing other objective function values.

6. We re-write problem (9) as the following quadratic optimization problem.

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

Using the same reasoning as in Lemma 3.1, item 6, we get

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

from Proposition 17.1 in [1], and the result follows immediately.

■

4 A multiobjective SQP-type algorithm

We present, in this section, the full algorithm that takes advantage of the previous described techniques for spreading and improving a given set of nondominated points.

We need an additional concept before presenting the full algorithm. Since the algorithm deals, iteratively, with a set of points X_k, each point $x_k \in X_k$ is labeled as *stopped* when no further progress is possible for x_k (x_k has been used to obtain new points, has reached optimality/stationarity, or progress is no longer possible).

Algorithm 4.1 ***Multiobjective SQP method***

1. *Initialization. Define $\tau > 0$, $\bar{n}$ integer, $\mu \in]0,1[$, and $\beta \in]0,1[$. Set $k = 0$.*

2. *First stage (Initialization)*

   Taking as initial guess

$$(x_0)_j = \begin{cases} \frac{u_j - \ell_j}{2} & \text{if } \ell_j, u_j \in \mathbb{R} \\ u_j & \text{if } \ell_j = -\infty \text{ and } u_j \in \mathbb{R} \\ \ell_j & \text{if } u_j = +\infty \text{ and } \ell_j \in \mathbb{R} \\ 0 & \text{if } \ell_j = -\infty \text{ and } u_j = +\infty \end{cases}, j = 1, \ldots, n, \tag{10}$$

   compute all the extreme Pareto front points by solving the following single-objective problems for each $i = 1, \ldots, m$.

$$
\begin{aligned}
x_i^* \in arg\min_{x\in\mathbb{R}^n} \quad & f_i(x) \\
s.t. \quad & g_j(x) \leq 0, \quad j = 1,\ldots,p \\
& h_l(x) = 0, \quad l = 1,\ldots,q \\
& \ell \leq x \leq u
\end{aligned}
\tag{11}
$$

3. *Second stage (Spread)*

   Initialize the set X_k with $\bar{n}$ bound feasible points (we may allow users to provide an initial set of points). Consider all points in X_k as non stopped.

   Perform a limited number of iterations in the set X_k as follows.

   (a) *Set $\mathcal{T} = \emptyset$.*

   For each non stopped point x_l, $l = 1,\ldots,\bar{n}$, in X_k do

   i. *For each $i = 1,\ldots,m$ do*

   A. *Compute the $\bar{d}$ search direction that corresponds to the individual minimization of the objective function i by solving the quadratic problem* (8).

   B. *If problem* (8) *has no feasible solution then enter a (feasibility) restoration procedure. If the restoration procedure obtains a feasible $\bar{x}_l$ point then set $\mathcal{T} = \mathcal{T} \bigcup \{\bar{x}_l\}$ and go to step 3(a)iG.*

   C. *If $\|\bar{d}\|$ is small ($\|\bar{d}\| < \sqrt[4]{\tau}$) go to 3(a)iG.*

   D. *Find the first $\alpha \in \{1, \beta, \beta^2, \ldots\}$ element such that (for big enough σ,* e.g. *the first element in $\{1, 10, 100, \ldots\}$ such that $\phi_i'(x_l, \sigma; \bar{d}) < 0$)*

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

   E. *If $\alpha < \sqrt{\tau}$ go to step 3(a)iG.*

   F. *Set $\mathcal{T} = \mathcal{T} \bigcup \{x_l + \alpha\bar{d}\}$.*

   G. *Continue with next i.*

   ii. *Set x_l as a stopped point. Continue with next l.*

   (b) *Set all points in $\mathcal{T}$ as non stopped points and X_{k+1} as the set of nondominated points in $X_k \bigcup \mathcal{T}$. If necessary proceed with a* Cleanup *procedure in the set X_{k+1} by removing points that are two close (e.g. using a crowding distance [5]) in the Pareto front. Set $k = k + 1$. Continue with step 3a.*

   Add the x_i^* points (obtained in previous stage) the to the set X_k.

4. *Third stage (Optimality – refining stage)*

   Take all points in X_k as non stopped and while not all points in X_k are stopped do:

(a) For all non stopped points x_l, $l = 1, \ldots, \bar{n}$, in the set X_k do

i. Compute $\bar{v}$ and $\bar{\vartheta}$ by solving the optimization problem (9).

ii. If problem (9) has no feasible solution then enter a (feasibility) restoration procedure. If a feasible $\bar{x}_l$ point is found during the restoration procedure then replace x_l by $\bar{x}_l$ and go to step 4(a)vii. Otherwise set x_l as a stopped point.

iii. If $\|\bar{v}\|$ is very small ($\|\bar{v}\| < \tau$) then mark x_l as stopped and go to step 4(a)vii.

iv. Find the first $\alpha \in \{1, \beta, \beta^2, \ldots\}$ element such that (for big enough σ)

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

v. If α is very small ($\alpha < \sqrt{\tau}$) set x_l as a stopped point and go to step 4(a)vii.

vi. Replace x_l by $x_l + \alpha\bar{v}$ in the set X_k.

vii. Continue with next l.

(b) Set X_{k+1} as the set of points in X_k where all points that are dominated by feasible points are removed. Set $k = k + 1$. Continue with step 4a.

The second stage in Algorithm 4.1 computes a new set of nondominated points $\mathcal{T}$. The updated set X_{k+1} can grow up to $(m+1)\bar{n}$ points ($\bar{n}$ previous points and an additional $m \times \bar{n}$ new points). However, in order to keep the set X_k at a reasonable size, when the set X_k reaches $\bar{n}$ points and an additional point is to be inserted in the set, a *Cleanup* procedure is taken, which consists in the computation of the crowding distance [5] for each point and the removal of the 10 points with the lowest one. The crowing distance provides an estimate of the density of solutions surrounding a given point, taking the average distance of the two neighboring points (on either side) along each of the objectives. Clearly, points with a lower crowding distance are positioned in a *crowded* region and are good candidates to be dropped.

Algorithm 4.1 already addresses the possibility of the quadratic problems (8) and (9) fail to obtain a feasible direction (the quadratic problem might be infeasible), so some feasibility *restoration* procedure is needed. The restoration procedure consists in the minimization of a measure of constraint violation with a regularization term subject to the non violated constraints. See [18] for a discussion of a more complex restoration procedure.

5 Convergence

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

For the local and global convergence of Algorithm 4.1 we will assume that H_i corresponds to the Hessian of the Lagrangian of the corresponding minimization problem,

i.e.

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

and

$$\mathcal{L}_i(x,\lambda,\delta) = f_i(x) + \sum_{j=1}^{p} \lambda_j g_j(x) + \sum_{l=1}^{q} \delta_l h_l(x).$$

While other approximations could be considered (e.g. a quasi-Newton approximation to the Hessian of the Lagrangian) this is the setting that allow us to have a quadratic rate of convergence (please note that we have already assumed differentiability conditions on f, g, and h).

A local approximation of the objective function and constraints may lead to an optimization problem where a feasible solution is numerically difficult to obtain. Therefore, the proposed algorithm uses a restoration phase in order to drive x_k to feasibility (minimizing a measure of constraints violation while keeping feasibility with respect to the already not violated constraints).

Since the number of operations performed in the first and second stages are finite, it remain to show that the limit points generated in the third stage are Pareto critical. However, the second stage, by itself is rich enough to prove some useful results under common assumptions.

Assumption 5.1 *The level set $L(x_0) = \cup_{i=1}^{m} L_i(x_0)$ is compact, where $L_i(x_0) = \{x \in [\ell,u] : f_i(x) \leq f_i(x_0)\}$, $i = 1,\ldots,m$ are the objective functions individual level sets. Furthermore the objective functions components are bounded below and above in $L(x_0)$.*

Consider Algorithm 4.1 where the second stage is allowed for an infinite number of iterations. Then we are able to prove the following theorem.

Theorem 5.1 *Under Assumption 5.1 and the allowance for an infinite number of iterations (take also $\tau = 0$) of the second stage in Algorithm 4.1 we have that*

1. *the algorithm terminates in a finite number of iterations and all the obtained points are Pareto critical, or*

2. *every limit point x_* is a Pareto point.*

Proof. The finite termination of the algorithm implies that $\bar{d} = 0$ or $\alpha = 0$ for all points in the set X_k and the Pareto criticality results trivially from Lemma 3.1.

For the second claim we have that all iterates remain in a compact level set where a sufficient decrease is imposed to all the objective functions, which are present in a finite number, and are bounded from below and above in the compact set.

More formally we can see that problem (8) is the same problem as (15.4) in [1], and, therefore iterates from our algorithm are iterates from a SQP algorithm for the optimization problem (11) which converges (quadratically) to the optimal solution x_*. Clearly x_* is a (local) Pareto point. ■

From an algorithmic point of view our algorithm is not performing the iterates consecutively for a single objective function, but the convergence still apply since we have a finite number of objective functions. During these iterates computations the algorithm is collecting the nondominated point in the set X_k.

While the previous theorem shows that convergence to (extreme) Pareto points and Lemma 3.1 support the computation of nondominated points during the iterates, the computation of all the Pareto points is unpractical and computationally impossible. The second stage is kept finite so a reasonable set (well spread) of approximated Pareto points is obtained.

Lets now analyze the third stage convergence, *i.e.* the full Algorithm 4.1 convergence.

Theorem 5.2 *Under our differentiability assumptions, the use of the ϕ merit function, and with a proper selection rule for σ, Algorithm 4.1 third stage converges to a Pareto critical point from an arbitrary starting point.*

Proof. This result follows trivially by noting that problem (9) corresponds to problem (17.2) in [1]. This implies that Algorithm 4.1 refining stage is in fact a SQP method applied to the optimization problem

$$\begin{aligned} \min_{x\in\mathbb{R}^n} \quad & \sum_{i=1}^{m} f_i(x) \\ \text{s.t.} \quad & g_j(x) \leq 0, \quad j = 1,\ldots,p \\ & h_l(x) = 0, \quad l = 1,\ldots,q \\ & \ell \leq x \leq u \end{aligned} \tag{12}$$

whose optimal value is a Pareto point. The result follows trivially from Theorem 17.2 in [1]. ■

A local convergence results may also be obtained for our proposed algorithm.

Theorem 5.3 *Suppose that the strict complementarity hold at the Pareto critical point x_* of (12), with associated Lagrange multipliers λ_* and δ_*. Under our assumption on f, g, and h differentiability in a neighborhood V of a x_* then Algorithm 4.1 generates a sequence $\{x_k, \lambda_k, \delta_k\}$ converging quadratically to $\{x_*, \lambda_*, \delta_*\}$.*

Proof. This result is a straightforward application of Theorem 15.2 in [1]. ■

While convergence can only be proved to a particular Pareto point (Problem (12) solution) the proposed algorithm includes a mechanism that can be observed as a multistart procedure (using all the points available at the set X_k after the spread stage). For the nonconvex case we can also expect convergence to local Pareto optimal point of Problem (12).

While in the numerical results we observe the computation of an approximate Pareto front, theoretically we cannot prove that the algorithm converges to different Pareto points.

Recall that Algorithm 4.1 is not described as being a primal-dual SQP method in the sense that we do not address the Lagrange multipliers computation. However it is straightforward to include them and to implement a strategy to properly handle the σ parameter. The Lagrange multipliers estimates are available from problems (8) and (9), and a possible strategy for the σ merit function parameter is described in [1, p. 294].

6 Implementation details

A prototype implementation of Algorithm 4.1 is available in MATLAB [13]. The implementation is able to address both constrained and unconstrained optimization problems. While our purpose it to provide a solver for constrained optimization we are also reporting numerical results for simple bound constrained problems.

The implementation uses the `fmincon` MATLAB solver for nonlinear constrained optimization problems for solving the subproblems described in (9), (11), and feasibility restoration. Since subproblems (8) are quadratically constrained problems, `quadprog` is the MATLAB solver of choice. Furthermore `exitflag` is used to decide if a corresponding subproblem has been solved successfully or not, and when to enter a feasibility restoration procedure.

The second stage of Algorithm (4.1) is performed for a maximum of 20 iterations or until all points in the set $\mathcal{S}$ are considered to be stopped. Points in $\mathcal{S}$ are declared as stopped if $\|\bar{d}\|$ (or $\|\bar{v}\|$) is small, as justified by Lemma 3.1 and Lemma 3.2. We use different tolerances ($\sqrt[4]{\tau}$ resp. τ) in different stages, since each stage has it own purpose (the second stage to obtain point that are spread over a Pareto front and the third stage to obtain Pareto critical points).

We consider $d = 0$ and $(v, \vartheta) = 0$ as the initial guesses for subproblems (8) and (9), while x_k is used as an initial guess for the feasibility restoration problem.The initial guess for problems (11) is built, componentwise, based on the simples bounds, as described in (10). Our implementation allows users to provide an initial set of points that are used to initialize the set X_k. Since the set X_k is a set of nondominated points some user provided points may be discarded in the initialization procedure. We provide two strategies to initialize the remaining points (until a maximum of $\bar{n}$ points is reached). The `line` strategy simple initializes the remaining points by considering a line between ℓ and u ($x_i = \ell + i\frac{u-\ell}{2n_{\mathcal{S}}}$, $i = 1, \ldots, 2\bar{n}$). The `rand` strategy simply initializes the remaining point using a uniform (ℓ, u) random distribution (generating, at most, $\bar{n}$ random points).

Regarding the quadratic approximations to the objective function we consider three cases. In the first case we simple 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 proposed in [9, 15]. The E_i is a diagonal matrix that ensures that H_i positive definite; note that $E_i = 0$ if $\nabla^2 f_i(x_k)$ is already positive definite. The last case

uses $H_i = I_m$ in the second stage of the algorithm and $H_i = (\nabla^2 f_i(x_k) + E_i)$ for the third stage.

7 Numerical Results

In this section we report on the numerical results of our implementation. We coined the implementation of the *MultiObjective Sequential Quadratic Programming* as `MOSQP`.

In order to conduct a proper testing we used a set of multiobjetive test problems collected from the literature and modeled in AMPL [12]. The use of AMPL allows the solver to obtain, through automatic differentiation, first and second derivatives for the objective functions and constraints (second derivatives of the constraints are not used in our implementation). To compare performance with other solvers, we use performance profiles and data profiles. We describe in the next subsection the test problems used and the profiles used. We end this section presenting the obtained numerical results with the test problems collection and a real-word application.

7.1 Test problems and profiles

7.1.1 Test problems

We consider the test problems available at [4], which can be obtained by following the instructions at `http://www.mat.uc.pt/dms`. While this test set is provided in the context of bound constrained multiobjective derivative-free optimization, many test problems are indeed differentiable. We enriched the test problems database with test problems provided in [6, 8, 10, 19] 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 and in Table 2 for constrained optimization problems. In both tables "Problem" is the problem name under our database, 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 problems database (modeled in AMPL) at `www.norg.uminho.pt/aivaz/mosqp`. Problem whose names have the suffix "`_a`" where changed so that all objective functions are differentiable at all points of the feasible domain (*e.g.* taking $\ell = 0.001$ instead of $\ell = 0$).

7.1.2 Performace profiles

We provide most of our numerical results in the form of performance profiles. Performance profiles were firstly proposed for uniobjective optimization problems in [7] and later on adapted to multiobjective optimization in [4]. Performance profiles are defined by a cumulative function $\rho(\tau)$ presenting a performance ratio (with respect to a given metric) for different solvers. Given $\mathcal{SO}$, a set of solvers, and $\mathcal{P}$, a set of problems, we define $t_{p,s}$ as the performance of solver s in solving problem p and the performance ratio to be $r_{p,s} = \frac{t_{p,s}}{\min_{\bar{s}\in\mathcal{SO}} t_{p,\bar{s}}}$. The cumulative function $\rho_s(\tau)$, $s \in \mathcal{SO}$ is defined to be the percentage

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						

Table 1: Bound 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					

Table 2: Linear and nonlinear constrained multiobjective problems

of problems whose performance ration is below or equal to τ, *i.e.*, $\rho_s(\tau) = \frac{|\{p \in \mathcal{P}: r_{p,s} \leq \tau\}|}{|\mathcal{P}|}$. Clearly $\rho_s(1)$ is the percentage of problems solved by solver s with the best metric value. Thus, we compare the values of $\rho_s(1)$ for all solvers $s \in \mathcal{SO}$ when looking for the most efficient solver (in the sense that attains a higher percentage of problems solved with the best metric value). By observing $\rho(\tau)$ when $\tau \to \infty$ we can compare the solver with respect to robustness, since the solver that attains a higher ration is the one able to solve a higher percentage of problems.

Usually the metric $t_{p,s}$ is an algorithmic performance metric like the number of iterations needed to attain a solution. Performance profiles defined in this way consider the lower the metric value the better.

The metric selected for the comparison is of most importance, since, for example, it is meaningless to consider the number of iterations when solvers computational work per iteration is not the same.

Additionally, multiobjective optimization problems pose a new challenge to performance profiles in the metric selection. As in [4] we chose to use the *Purity* and two *spread* metrics. The hypervolume metric as defined in [21] is also a popular metric in multiobjective optimization and is therefore included in our numerical results.

Purity metric. Let $F_{p,s}$ be the approximation to the Pareto front computed by solver s for problem p, and F_p 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 metric to be the inverse of the number of points in the true Pareto front over the number of points solver s is able to compute that are not dominated by the true Pareto front, *i.e.*,

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

The purity metric measures (in an inverse ratio) how many nondominated points one solver is able to compute (in percentage). See [4] for further details and discussions.

Spread metrics. While the purity metric measures how well a solver is able to computed nondominated points, this metric is unable to provide any information about how points are spread over the Pareto front. In order to understand if a given solver is able to provide a Pareto front where points are well distributed (spread) we consider two additional metrics for the performance profiles. To define the Pareto front extension we need to consider the *extreme points* that correspond to the global minimization of problems (11). Again, and since the true *extreme points* are not available, we take the computed objective function minimizer (w.r.t. all solvers) to be approximations to the true *extreme points*. Consider the approximated Pareto front computed by solver s for problem p formed by N points, indexed by $1, \ldots, N$, to which we add the extreme points (indexed by 0 and $N+1$). The

$\Gamma > 0$ and $\Delta > 0$ metrics are then defined as

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

where $\delta_{i,j} = f_j(x_{i+1}) - f_j(x_i)$ (and we assume the values of f to be sorted in an increasing order for each objective function j), and

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

where $\bar{\delta}_j$, $j = 1, \ldots, m$ is the average of the distances $\delta_{i,j}$.

While the Γ metric measures the biggest 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 [4]. 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 [21] (a more detailed description can be obtained in [2] or [20]) corresponds to the volume of the dominated space, enclosed by the nondominated points and the origin.

The used code to compute the metric can be found at `ftp://ftp.tik.ee.ethz.ch/pub/people/zitzler/hypervol.c`, and we have interfaced it with MATLAB for being used in the performance profiles generation.

Since many problems objective functions may attain negative values (or $f_i >> 0$, $i = 1, \ldots, m$) we apply a translation of the objective functions for each problem by using the extreme Pareto points computed from all the obtained Pareto fronts (*i.e.* a translation base on the ideal objective vector), so that all nondominated points are in the positive cone.

Taking $t_{p,s}$ as the hypervolume may result in an invalid $r_{p,s}$ value, since the hypervolume measure is null for a Pareto front formed by only one (ideal) point. The performance profiles modification used in [17] was used to address this possibility. This modification corresponds to redefining $t_{p,s} := t_{p,s} + 1 - \min_{s \in \mathcal{SO}} t_{p,s}$ when $\min_{s \in \mathcal{SO}} t_{p,s} < 0.001$ and leaving $t_{p,s}$ unchanged in all other cases.

7.1.3 Data profiles

Data profiles were originally proposed for derivative-free optimization algorithms [14]. A data profiles can be interpreted as a cumulative function $d_s(\sigma)$ that reports the percentage of problems solver s is able to solver given a budget of σ objective function evaluations. Data profiles were used in [4] in the context of derivative-free multiobjective optimization. Since data profiles allows us to discuss the (computational) resources in terms of function evaluations an algorithm needs to solve a problem, we describe numerical results using this approach.

The data profile cumulative 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 evaluations needed to solve problem p by solver s.

A critical issue in multiobjective optimization is related with when to declare that problem p is solved by solver s. As in [4] we use the following criteria to declare success. A solver s is declared as solving problem p, with accuracy ε, when

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

i.e., if the percentage of points obtained in the Pareto front F_p is equal to or greater than $1 - \varepsilon$.

The approximated Pareto front F_p is computed by running all the solver for a limit of 5000 function evaluations (thus removing any other stopping criteria). When all the points obtained by the solvers are nondominated, $|F_p|/|\mathcal{SO}|$ is the average number of points obtained by a solver, and $|F_{p,s} \cap F_p|$ is the number of nondominated points obtained by solver s.

Following [4] and [14] we also divided σ by $n+1$ (the number of points needed to build a *simplex gradient*). For the derivative based solvers we define

$$h_{p,s} = \#f + (n)\#\nabla f + \left(\frac{(n+1)(n+2)}{2}\right)\#\nabla^2 f, \tag{13}$$

where $\#f$ is the number of objective functions evaluations, $\#\nabla f$ is the number of objective gradients evaluations, and $\#\nabla^2 f$ is the number of objective Hessians evaluations. Equation (13) accounts for the number of function evaluations since n is the number of additional function evaluations needed to build a finite difference approximation to the gradient and $(n+1)(n+2)/2$ is the number of function evaluation needed to build a finite difference approximation to the Hessian (full quadratic model).

7.2 Numerical results

We present our numerical results in the form of performance and data profiles. We compare our implementation with the `NSGA II` solver [5] (C version 1.1). To the best of our knowledge there is no derivative based solver publicly available for constrained multiobjective optimization and therefore we chose to use the (widely used) solver described in [4] whose performance was considered best among various stochastic solvers. A population of 100 points was used for `NSGA II` and for the performance profiles we used 200 generations, corresponding to a total of 20000 objective and constraint functions evaluations (since the objective and constraints are evaluated for each point in each generation). Each objective functions evaluation corresponds to an evaluation of the vector function f and each constraint functions evaluations corresponds to an evaluation of all the linear and nonlinear functions.

We consider six variations of our solver, which correspond to the combinations of the initialization strategy and the selected H_i, e.g., `MOSQP` ($H = \nabla^2 f$, `rand`) corresponds to the `rand` initialization strategy where the true objective functions 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 objectives 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 *stochastic* solver, 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 front, *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 optimal Pareto front (F_p), while *worst* is the one with the lower number of nondominated points.

Therefore, we have computed a significant number of performance (and data) profiles, but to keep matters succint, we are discussing here only the most important ones. The reader can see the full set of profiles at the solver webpage `www.norg.uminho.pt/aivaz/mosqp`.

While all the `MOSQP` solver variations where able to perform reasonable well when compared with the `NSGA II` solver, the `MOSQP` ($H = \nabla^2 f$, `line`) was the one with best performace in both best and worst `NSGA II` solver runs (recall that `MOSQP` ($H = \nabla^2 f$, `line`) is completely deterministic and therefore it was run only once). We present in Figure 1 a compare between our implementation and `NSGA II` for the bound constrained problems in Table 1, using the Purity metric in the performance profiles. The purity metric is sensitive to the number of solvers considered in the profile, so only a two by two comparison should be done.

For the best `NSGA II` run we can conclude, from Figure 1, that `MOSQP` ($H = \nabla^2 f$, `line`) is able to solve about 70% of the problems with the best metric, while `NSGA II` solves about 50% of the problems with the best metric. Observing $\tau \to \infty$ we can conclude that `MOSQP` ($H = \nabla^2 f$, `line`) is able to solve all problems, while `NSGA II` solves about 95% of the problems. By noting that the `MOSQP` solver uses local approximation methods (*i.e.*, convergence to local optimal points) makes the results even more favorable to our implementation, as `NSGA II` is often associated with a probabilistic convergence to global (Pareto) optimal points. The purity metric is clearly influenced by the presence of local or global optimal points, as some global optimal points will dominate local optimal points that are not globally optimal.

Figure 2 provides the performance profiles with the Purity metric for the constrained test problems described in Table 2. As it can be observed from the figure, these results for the constrained test set are more favorable to our implementation, both in efficiency as well as in robustness.

Regarding the Δ and Γ spread metrics, we note that these metrics are solver and Pareto front independent (*i.e.*, given an approximate Pareto front we are able to compute the corresponding metric). Therefore, all the solver may be included in a single profile and we are able to compute maximum, average, and minimum metrics when we are in

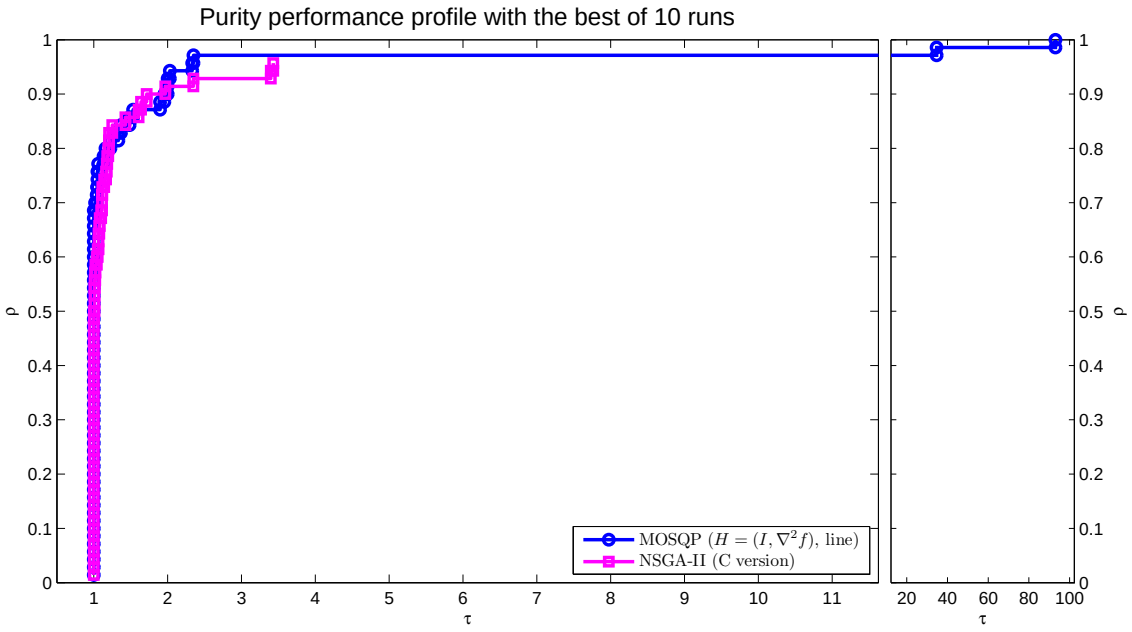

Figure 1: Purity performance profile with the *best* Pareto front. Bound constrained test set.

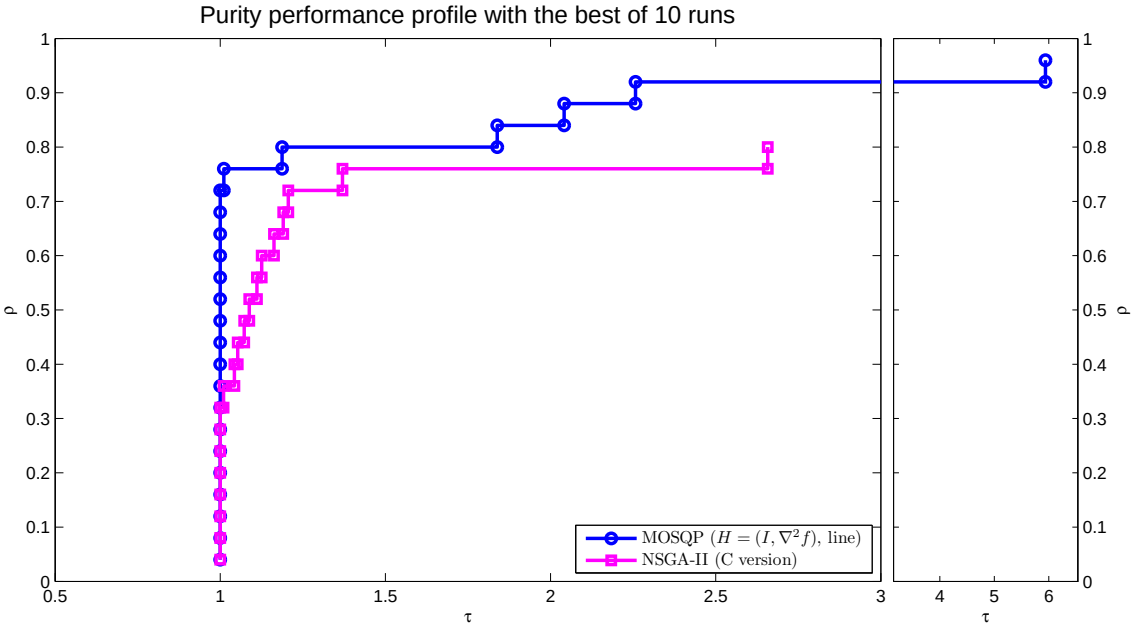

Figure 2: Purity performance profile with the *best* Pareto front. Constrained test set.

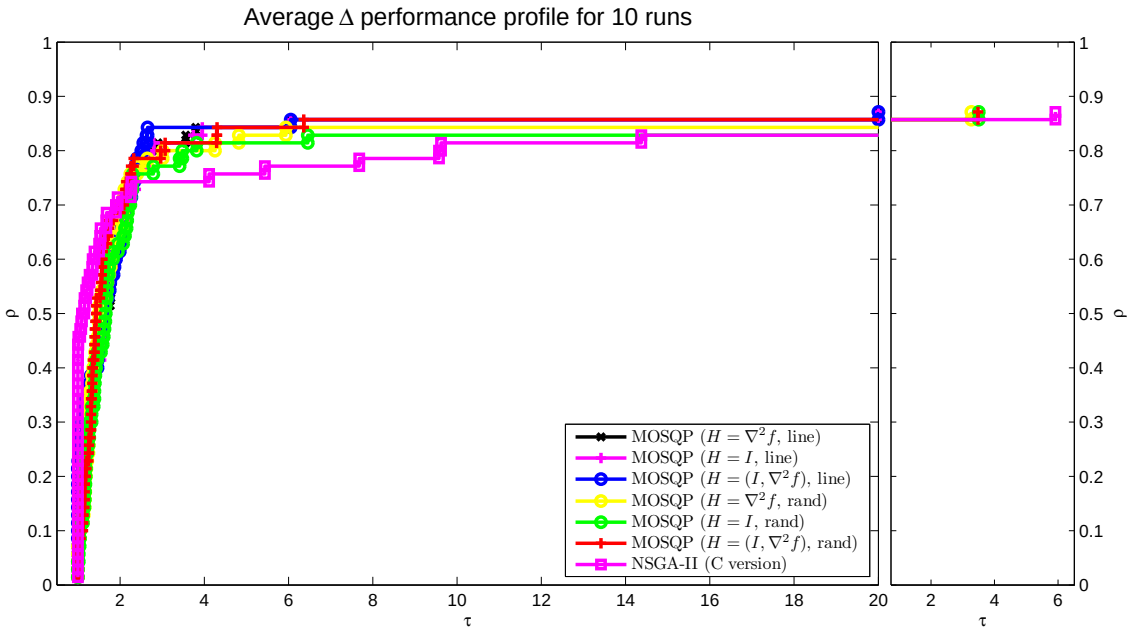

Figure 3: Spread average Δ performance profile. Bound constrained test set.

the presence of several runs. Thus, we present in Figure 3 and 4 the performance profiles with the spread Δ and Γ metrics for the bound constrained test. While these results may not look favorable to our implementation (specially for the Γ metric) we recall that our implementation has no implicitly control of the spread metric along the optimization procedure (except when the nondominated list of points reaches the maximum allowed size, and a cleanup procedure is performed), while `NSGA II` uses the crowding distance to select offsprings with a better spread.

Figure 5 and 6 depict the spread metrics Δ and Γ on our set of constrained test problems.

We end the performance analysis by presenting the performance profiles for the hypervolume metric, which is pictured in Figure 7 for the bound constrained test set. We can observe that the `MOSQP` ($H = \nabla^2 f$, `line`) is again the more robust and efficient solver, with a slight advantage over the other `MOSQP` variants, and more detached from the `NSGA II` solver.

For the constrained test set `NSGS II` is comparable to the `MOSQP` variants with random initialization, while `MOSQP` variants with the `line` initialization are detached from the remaining solvers, as it can be observed in Figure 8.

Before starting to present the computational effort analysis for all the solvers, presented in the form of data profiles, we address the number of objective and constraints functions and derivatives evaluations done when obtaining the performance profiles. Tables 3 and 4 present the (maximum, average, and minimum) number of functions evaluations ($\#f$), number of objective gradient functions evaluations ($\#\nabla f$), number of objective Hessian

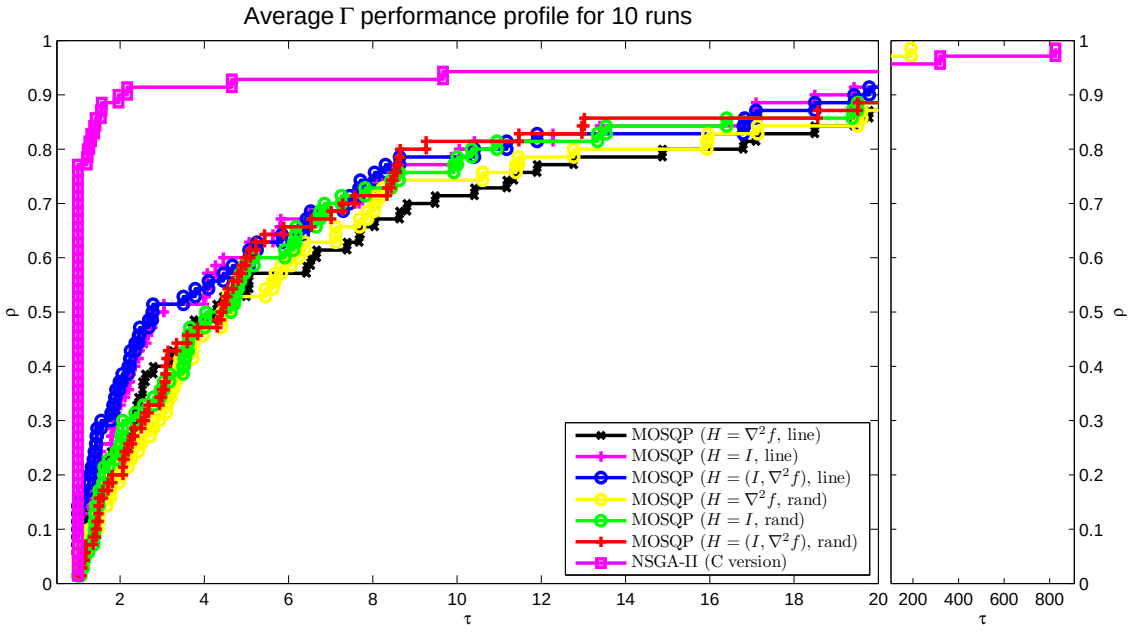

Figure 4: Spread average Γ performance profile. Bound constrained test set.

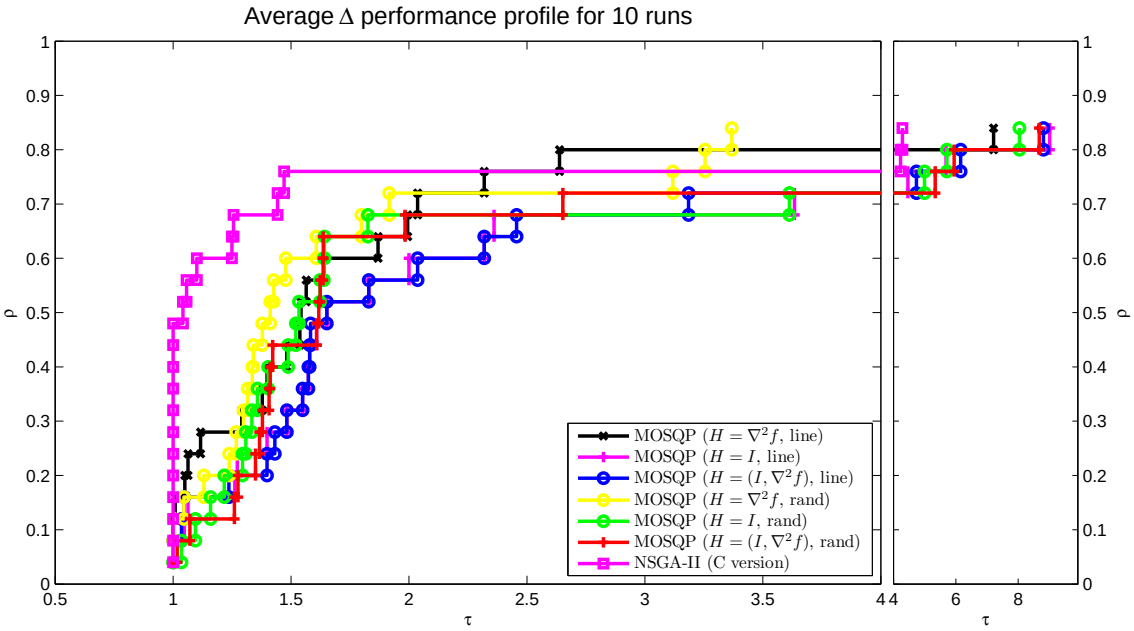

Figure 5: Spread average Δ performance profile. Constrained test set.

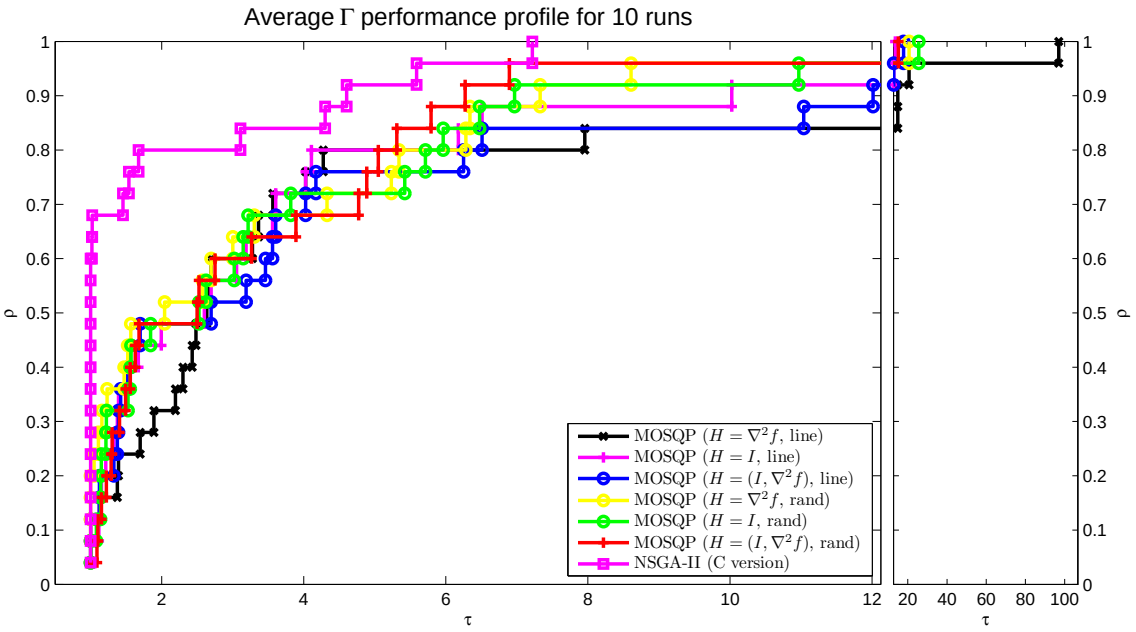

Figure 6: Spread average Γ performance profile. Constrained test set.

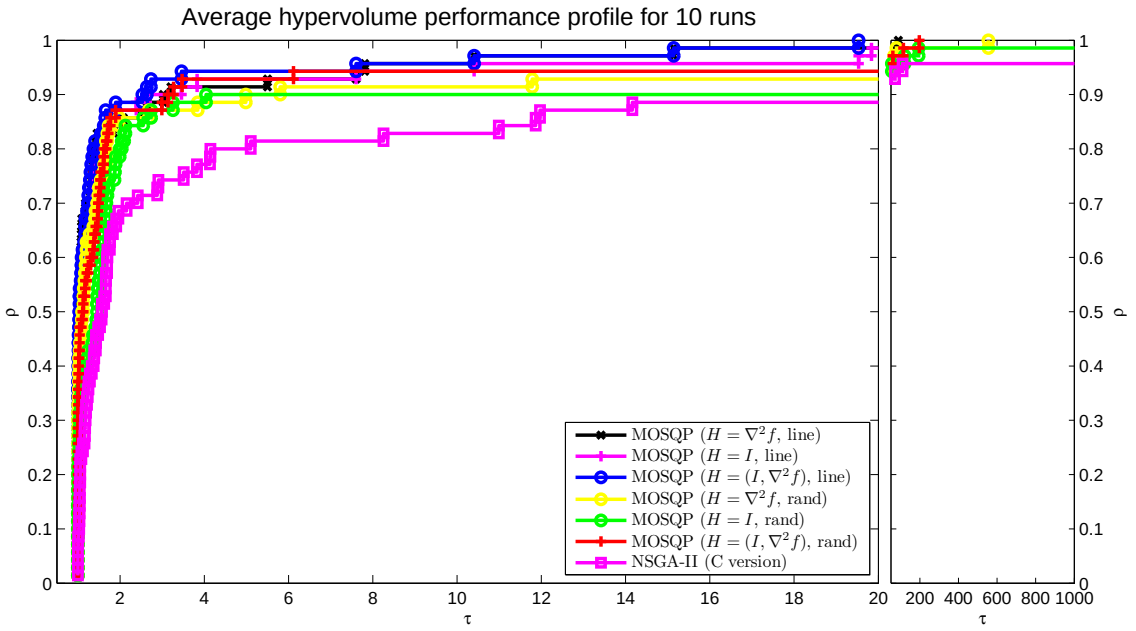

Figure 7: Average hypervolume performance profile. Bound constrained test set.

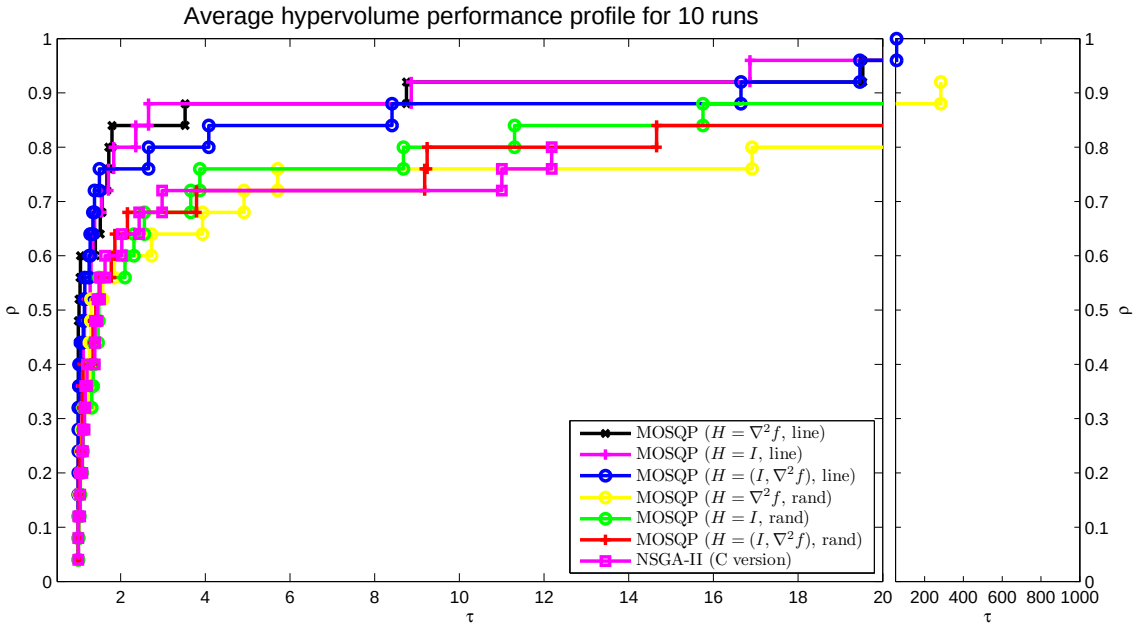

Figure 8: Average hypervolume performance profile. Constrained test set.

functions evaluations ($\#\nabla^2 f$), number of nonlinear constraint functions evaluations ($\#c$), and number of nonlinear constraint gradient evaluations ($\#\nabla c$). Our implementation, in some stages of the algorithm (*e.g.* on problems (11) and (8)), considers the minimization of individual objective functions and, therefore, we are reporting the number of individual objective, gradient, and Hessian functions evaluations divided by m. We take advantage on the existence of linear constraints to provide them in a matrix/vector form to `fmincon`. Therefore, we are unable to control how many linear constraints evaluations are performed. However, such evaluations can be seen as taking an effort that is computationally negligible; we therefore do not report the corresponding number of linear constraint evaluations. Table 4 shows the number of nonlinear constraint function and gradient evaluations divided by the number of nonlinear constraints. Recall that `NSGA II` always performs 20000 function evaluations and 20000 constraint evaluations (on each problem).

For the data profiles, and since the computed F_p may be a crude approximation to the true Pareto front, the approximated Pareto fronts obtained in the numerical results for the performance profiles could also be used. However, using a very good approximation to the Pareto front makes $|F_{p,s} \cap F_p|$ to be zero for reasonable budgets of function evaluations, *i.e.*, the worst solvers would not be visible in the data profile plot. So, we chose to use the approximated Pareto front F_p previously described in section 7.1.3, which is computed by running all the solver for a limit of 5000 functions evaluations (corresponding to $5000m$ individual objective function evaluations). For the `MOSQP` solver we consider the number of individual evaluations in accordance to (13). Recall that the approximate Pareto front is used for all the solver in the analysis and, therefore, it is still a fair compare for all solvers.

MOSQP version	$\#f$ max	$\#f$ *Avg*	$\#f$ min	$\#\nabla f$ max	$\#\nabla f$ *Avg*	$\#\nabla f$ min	$\#\nabla^2 f$ max	$\#\nabla^2 f$ *Avg*	$\#\nabla^2 f$ min
$H = \nabla^2 f$, line	1546.7	517.1	205.0	1250.0	199.5	6.0	1250.0	199.5	6.0
$H = I$, line	2032.0	684.0	205.0	1599.0	274.8	6.0	114.3	10.1	1.0
$H = (I, \nabla^2 f)$, line	2032.0	624.4	205.0	980.3	230.2	6.0	884.3	120.3	1.0
$H = \nabla^2 f$, rand	2713.3	458.0	203.0	1127.0	139.0	3.0	1127.0	139.0	3.0
$H = I$, rand	12047.5	644.0	203.0	1598.0	244.0	3.0	114.3	10.0	1.0
$H = (I, \nabla^2 f)$, rand	11364.3	558.0	203.0	1546.7	178.0	3.0	985.3	90.0	2.0

Table 3: Number of objective function, gradient, and Hession evaluations for the bound constrained test set.

MOSQP version	$\#f$ max	$\#f$ *Avg*	$\#f$ min	$\#\nabla f$ max	$\#\nabla f$ *Avg*	$\#\nabla f$ min	$\#\nabla^2 f$ max	$\#\nabla^2 f$ *Avg*	$\#\nabla^2 f$ min
$H = \nabla^2 f$, line	2119.0	765.6	229.0	1640.5	317.8	23.5	1634.0	299.7	6.0
$H = I$, line	3802.0	1111.9	251.0	1825.5	445.2	32.5	0.0	0.0	0.0
$H = (I, \nabla^2 f)$, line	3802.0	1050.5	244.0	1825.5	422.3	25.5	468.0	78.4	0.0
$H = \nabla^2 f$, rand	4877.3	759.6	223.0	1155.0	269.1	16.5	1070.0	251.0	9.0
$H = I$, rand	3505.0	1161.5	264.0	1628.5	446.1	30.5	0.0	0.0	0.0
$H = (I, \nabla^2 f)$, rand	3414.0	1079.6	270.0	1638.5	406.9	30.5	416.0	61.5	0.0

MOSQP version	$\#c$ max	$\#c$ *Avg*	$\#c$ min	$\#\nabla c$ max	$\#\nabla c$ *Avg*	$\#\nabla c$ min
$H = \nabla^2 f$, line	4555.0	1244.8	176.7	2337.0	655.0	31.0
$H = I$, line	28637.0	2928.2	214.7	15324.0	1472.2	40.0
$H = (I, \nabla^2 f)$, line	28637.0	2870.7	215.0	15324.0	1450.2	33.0
$H = \nabla^2 f$, rand	18889.0	1415.3	107.7	9877.0	714.0	24.0
$H = I$, rand	39524.0	2172.2	112.0	22296.0	1041.4	28.3
$H = (I, \nabla^2 f)$, rand	31642.0	2120.0	118.3	18522.0	1034.1	32.3

Table 4: Number of objective function, gradient, and Hession evaluations for the constrained test set.

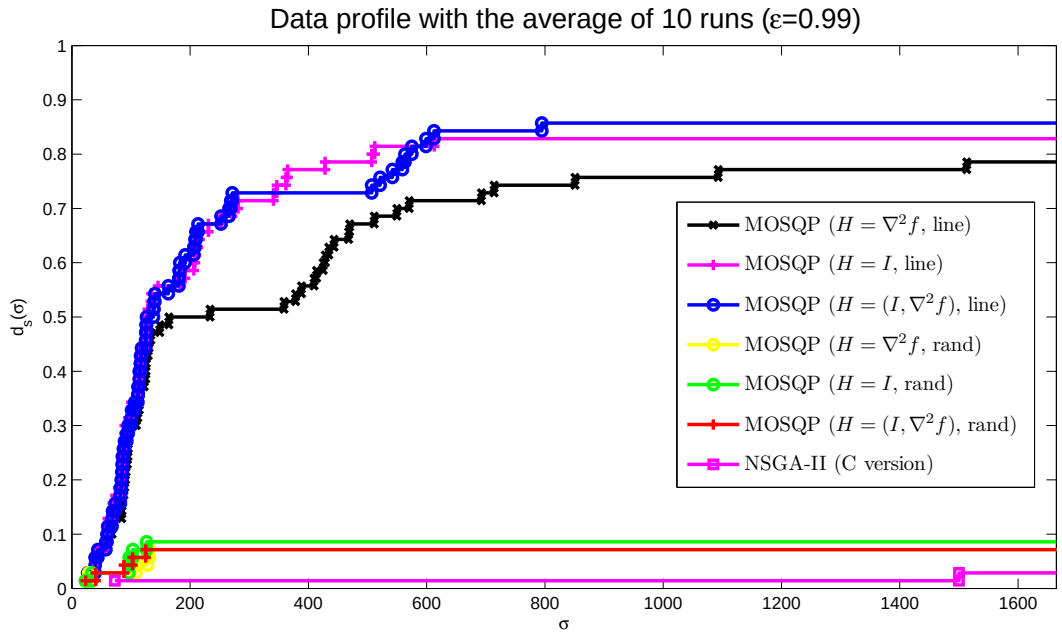

Figure 9: Data profile for bound constrained test problems.

Other detail related with the data profile is the ϵ parameter, which controls the accuracy to which a problem is considered to be solved. Usually the data profiles are represented for some typical values of ϵ, say, for example, $\epsilon = 0.05$ (meaning that if the obtained solution, for the given budget, is 95% near the optimal solution, then the problem is considered to be solver). For our multiobjective optimization solvers it means that the problem is solved, for a given budget, if the obtained Pareto front has 95% of points that are not dominated by the approximated Pareto front. Since we are in the presence of stochastic solver we can build data profiles for the best (bigger number of nondominated points), average, and worst (lower number of nondominated points) performance in 10 runs for bound constrained and constrained test problems. Again, for a matter of brevity, we provide only the ones based on the average number of objective functions evaluations.

Since the data profiles provided for $\epsilon = 0.05$ are not very informative, we provide data profiles for $\epsilon = 0.99$, which allows us the same conclusions with better visibility. We provide in Figure 9 a data profile for the bound constrained test problems and Figure 10 for the constrained test problems, based on the average objective functions evaluations. Clearly, we are able to see that the solver `MOSQP` ($H = I$, `line`) is the one able to solve more problems for a budget of 400 function evaluations, immediately followed by the solver `MOSQP` ($H = (I, \nabla^2 f)$, `line`).

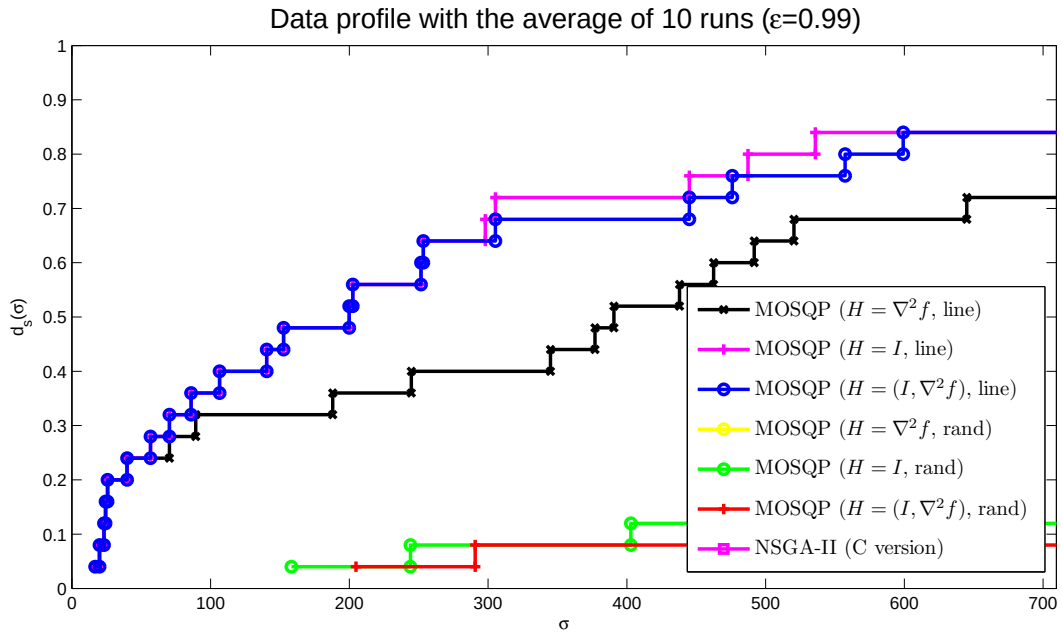

Figure 10: Data profile for constrained test problems.

7.3 A real-world application

To complete the numerical testing we provide some numerical results with a hard real-world problem adapted for bi-objective optimization. We consider the Cassini 1 Multiple Gravity Assist (MGA) problem, which is related to the Cassini spacecraft trajectory design problem (see the ESA problem web page [16] for further details). The problem provides a nonlinear objective and four nonlinear inequality constraints. The (first) objective function (f_1) is the total ΔV accumulated during the mission; these values are proportional to fuel consumption. The second objective function is total travel time to reach the final destination (Saturn orbit).

Gradients and Jacobians of the constraints were computed by finite differences. Second derivatives (Hessian) of the objective functions are not considered, *i.e.* we only considered the `MOSQP` solver with `MOSQP (`$H = I$`, line)` and `MOSQP (`$H = I$`, rand)`. Table 5 reports the hypervolume, Γ, and Δ metric. Again we have performed 10 runs for the stochastic solver (using the same algorithmic parameters as in the previous section) and we report on the maximum, minimum, and average for each metric. From the numerical results we can conclude that `MOSQP` has a comparable performance to `NSGA II`, since it performed well for the hypervolume metric, while `NSGA II` has an advantage with respect to the Γ and Δ metrics. In Figure 7.3 we provide a picture of the best hypervolume metric Pareto fronts obtained by the solvers. The `MOSQP` is able to provide better objective function values for both the objective functions, while it reveals some difficulty in spreading point along the Pareto front.

	MOSQP ($H = I$, line)	MOSQP ($H = I$, rand)	NSGA-II (C version)
Hyper max	66493.43	121285.33	95527.89
Hyper avg	66493.43	64944.58	72281.57
Hyper min	66493.43	10862.25	56784.03
Γ max	3257.75	5089.97	2628.22
Γ avg	3257.75	3433.52	2326.91
Γ min	3257.75	2040.40	1874.94
Δ max	0.47786	1.73859	1.08101
Δ avg	0.47786	1.26121	1.06785
Δ min	0.47786	0.98216	1.05907

Table 5: Metrics for the Cassini bi-objective optimization problem

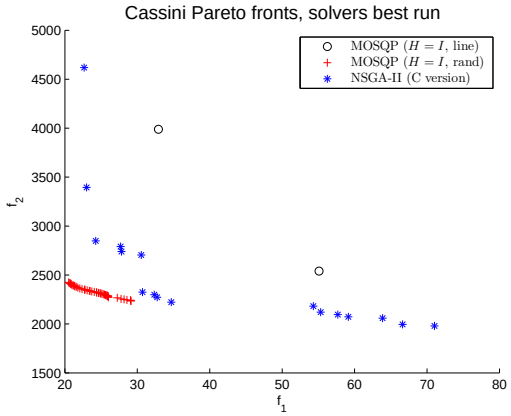

Figure 11: Best Pareto fronts obtained by `MOSQP` and `NSGA II` solvers

8 Conclusions

We develop and analyze a Newton type method for constrained multiobjective optimization. Under appropriate differentiability assumptions on the objective and constraint functions we prove local and global convergence to local Pareto critical points.

We provide a publicly available implementation of the proposed algorithm, which we coined as `MOSQP`. While the proposed implementation complies with the proposed algorithm framework there are many further research avenues to improve the algorithmic performance, like, e.g., the use of quasi-Newton type methods.

Extensive numerical results are provided for our implementation of the `MOSQP` algorithm, comparing it with the `NSGA II` solver. Numerical results show that `MOSQP` performs well for a significant set of bound and constrained test problems, making the `MOSQP` solver the preferred solver for multiobjective optimization problems when objective and constraint functions derivatives are available.

References

[1] J.F. Bonnans, J.C. Gilbert, C. Lemaréchal, and C.A. Sagastizábal, *Numerical Optimization, Theoretical and Practical Aspects*, Springer, second ed., 2000.

[2] L. Bradstree, *The hypervolume indicator for multi-objective optimisation: calculation and use*, PhD thesis, Department of Computer Science & Software Engineering, The University of Western Australia, 2011.

[3] Y. Collette and P. Siarry, *Multiobjective Optimization: Principles and Case Studies*, Springer, 2004.

[4] A.L. Custódio, J.F.A. Madeira, A.I.F. Vaz, and L.N. Vicente, *Direct multisearch for multiobjective optimization*, SIAM Journal on Optimization, 21 (2011), pp. 1109–1140.

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

[6] Kalyanmoy Deb, *Multi-objective genetic algorithms: Problem difficulties and construction of test problems*, Evol. Comput., 7 (1999), pp. 205–230.

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

[8] G. Eichfelder, *An adaptive scalarization method in multiobjective optimization*, SIAM J. Optim., 19 (2009), pp. 1694–1718.

[9] E. Eskow and R.B. Schnabel, *Algorithm 695: Software for a new modified cholesky factorization*, ACM Transactions on Mathematical Software, 17 (1991), pp. 306 – 312.

[10] J. Fliege, L. Drummond, and B. Svaiter, *Newton's method for multiobjective optimization*, SIAM Journal on Optimization, 20 (2009), pp. 602–626.

[11] J. Knowles, D. Corne, and K. Deb, eds., *Multiobjective Problem Solving from Nature: From Concepts to Applications*, Springer, 2008.

[12] AMPL Optimization LLC, *AMPL: A modeling language for mathematical programming.* `http://www.ampl.com`, 2012.

[13] MATLAB, *version 7.10.0 (R2010a)*, The MathWorks Inc., Natick, Massachusetts, 2010.

[14] J. J. Moré and S. M. Wild, *Benchmarking derivative-free optimization algorithms*, SIAM J. Optim., 20 (2009), pp. 172–191, `http://www.mcs.anl.gov/~more/dfo`.

[15] R.B. Schnabel and E. Eskow, *A revised modified cholesky factorization algorithm*, SIAM J. Optim, 9 (1999), pp. 1135–1148.

[16] Advanced Concepts Team, *Cassini 1 MGA problem.* `http://www.esa.int/gsp/ACT/inf/projects/gtop/cassini1.html`, 2014.

[17] 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.

[18] A. Wächter and L.T. Biegler, *On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming*, Mathematical Programming, 106 (2005), pp. 25–57.

[19] 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 Engieering University of Essex, Colchester, C04, 3SQ, UK, 2009.

[20] Eckart Zitzler, *Evolutionary algorithms for multiobjective optimization: Methods and applications*, vol. 63, Shaker Ithaca, 1999.

[21] Eckart Zitzler and Lothar Thiele, *Multiobjective optimization using evolutionary algorithms – a comparative case study*, in Parallel Problem Solving from Nature – PPSN V, A.E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, eds., vol. 1498 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 1998, pp. 292–301.