



# A limited memory Quasi-Newton approach for multi-objective optimization

Matteo Lapucci<sup>1</sup> · Pierluigi Mansueto<sup>1</sup>

Received: 31 May 2022 / Accepted: 17 January 2023 / Published online: 02 March 2023  
© The Author(s) 2023, corrected publication 2024

## Abstract

In this paper, we deal with the class of unconstrained multi-objective optimization problems. In this setting we introduce, for the first time in the literature, a Limited Memory Quasi-Newton type method, which is well suited especially in large scale scenarios. The proposed algorithm approximates, through a suitable positive definite matrix, the convex combination of the Hessian matrices of the objectives; the update formula for the approximation matrix can be seen as an extension of the one used in the popular L-BFGS method for scalar optimization. Equipped with a Wolfe type line search, the considered method is proved to be well defined even in the nonconvex case. Furthermore, for twice continuously differentiable strongly convex problems, we state global and R-linear convergence to Pareto optimality of the sequence of generated points. The performance of the new algorithm is empirically assessed by a thorough computational comparison with state-of-the-art Newton and Quasi-Newton approaches from the multi-objective optimization literature. The results of the experiments highlight that the proposed approach is generally efficient and effective, outperforming the competitors in most settings. Moreover, the use of the limited memory method results to be beneficial within a global optimization framework for Pareto front approximation.

**Keywords** Multi-objective optimization · Quasi-Newton methods · Limited memory · Pareto front

**Mathematics Subject Classification** 90C26 · 90C29 · 90C30

---

✉ Pierluigi Mansueto  
pierluigi.mansueto@unifi.it  
Matteo Lapucci  
matteo.lapucci@unifi.it

<sup>1</sup> Global Optimization Laboratory, Department of Information Engineering, University of Florence, Via di Santa Marta, 3, 50139 Florence, Italy

## 1 Introduction

Multi-Objective Optimization (MOO) is a mathematical tool which has proved to be particularly well suited for many real-world problems over the years. Particular relevance of this framework is demonstrated, for example, by applications in statistics [1], design [2], engineering [3, 4], management science [5], space exploration [6]. The principal complexity that makes MOO problems difficult to handle is the general impossibility to reach a solution minimizing all the objective functions simultaneously. In this context, the definitions of optimality (global, local and stationarity) are based on Pareto's theory. The latter has complexity elements that make it difficult the creation of new methods and optimization processes.

A class of MOO methods that has been widely studied for the past two decades is the one concerning descent algorithms (either first-order, second-order and derivative-free). These approaches are basically extensions of the classical iterative scalar optimization algorithms. Steepest Descent [7], Newton [8, 9], Quasi-Newton [10], Augmented Lagrangian [11], Conjugate Gradient [12] are only a few methods of this family. In addition to having theoretically relevant convergence properties, these algorithms, when used on problems with reasonable regularity assumptions, proved to be valid alternatives to the scalarization approaches [13] and the evolutionary ones [14], especially as the problem size grows [15, 16]. In recent years, some of the descent methods were also extended to generate approximations of the Pareto front, instead of a single Pareto-stationary solution [15–18]. Moreover, following ideas developed for scalar optimization [19, 20], descent methods have also been used as local search procedures within memetic algorithms [21].

Quasi-Newton methods are among the most popular algorithms for unconstrained single-objective optimization. Based on a quadratic model of the objective function, they do not require the calculation of the second derivatives in order to find the search direction: the real Hessian is replaced by an approximation matrix, which is updated at each iteration considering the new generated solution and the previous one. The most famous update formula for the approximation matrix is the BFGS one, which is named after Broyden, Fletcher, Goldfarb and Shanno [22]. In the multi-objective setting, Quasi-Newton methods were proposed, for instance, in [10, 23–25].

Among the factors contributing to the success of Quasi-Newton methods in scalar optimization, the possibility of defining limited-memory variants of these approaches certainly stands out. The approximate Hessian matrix can in fact be roughly recovered only using a finite number  $M$  of previously generated solutions. In this way, its management in memory, which could be extremely inefficient and time-consuming, is avoided. In particular, the L-BFGS algorithm, firstly designed in [26], has managed over the years to achieve state-of-the-art performance in most settings, even with relatively small values for  $M$ .

This work concerns, to the best of our knowledge, the first attempt in the literature to define a multi-objective limited memory Quasi-Newton method. The key elements that characterize the proposed approach are:

- (i) a shared approximation of the Hessian matrices is employed to compute the search direction;

- (ii) the Hessian matrix approximation only requires information related to the most recent iterations to be computed;
- (iii) the method is in general well defined and, in the strongly convex case, is shown to possess R-linear global convergence properties to Pareto optimality.

The rest of the manuscript is organized as follows. In Sect. 2, we recall the main concepts related to MOO Quasi-Newton methods. In Sect. 3, we provide a description of the proposed limited memory Quasi-Newton approach; we then provide the convergence analysis in Sect. 4. In Sect. 5, we show through computational experiments the good performance of the limited memory approach w.r.t. main state-of-the-art Newton and Quasi-Newton methods. Moreover, in this section we show the performance of the new approach used as local search procedure in a global optimization method. Finally, in Sect. 6 we provide some concluding remarks.

## 2 Preliminaries

In this work, we consider the following unconstrained multi-objective optimization problem:

$$\min_{x \in \mathbb{R}^n} F(x) = (f_1(x), \dots, f_m(x))^T, \tag{1}$$

where  $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$  is a continuously differentiable function. We denote by  $J_F(\cdot) = (\nabla f_1(\cdot), \dots, \nabla f_m(\cdot))^T \in \mathbb{R}^{m \times n}$  the Jacobian matrix associated with  $F$ . Moreover, for all  $j \in \{1, \dots, m\}$ , the Hessian matrix of the function  $f_j(\cdot)$ , when it exists, is denoted by  $\nabla^2 f_j(\cdot)$ . In what follows, the Euclidean norm in  $\mathbb{R}^n$  will be denoted by  $\|\cdot\|$ .

Since we are in a multi-objective setting, we need a partial ordering in  $\mathbb{R}^m$ : considering two points  $u, v \in \mathbb{R}^m$ , we have that

$$\begin{aligned} u < v &\iff u_i < v_i, \quad \forall i = 1, \dots, m, \\ u \leq v &\iff u_i \leq v_i, \quad \forall i = 1, \dots, m. \end{aligned}$$

We can say that  $u$  dominates  $v$  if  $u \leq v$  and  $u \neq v$ . In this case, we use the following notation:  $u \preceq v$ . Similarly, we state that  $x \in \mathbb{R}^n$  dominates  $y \in \mathbb{R}^n$  w.r.t.  $F$  if  $F(x) \preceq F(y)$ .

In multi-objective optimization, a point minimizing all the objectives at once is unlikely to exist. For this reason, the concepts of optimality are based on Pareto's theory.

**Definition 1** A point  $\bar{x} \in \mathbb{R}^n$  is *Pareto optimal* for problem (1) if there does not exist  $y \in \mathbb{R}^n$  such that  $F(y) \preceq F(\bar{x})$ . If there exists a neighborhood  $\mathcal{N}(\bar{x})$  in which the previous property holds, then  $\bar{x}$  is *locally Pareto optimal*.

Since Pareto optimality is a strong property, it is hard to attain in practice. A slightly more affordable condition is weak Pareto optimality.

**Definition 2** A point  $\bar{x} \in \mathbb{R}^n$  is *weakly Pareto optimal* for problem (1) if there does not exist  $y \in \mathbb{R}^n$  such that  $F(y) < F(\bar{x})$ . If there exists a neighborhood  $\mathcal{N}(\bar{x})$  in which the previous property holds, then  $\bar{x}$  is *locally weakly Pareto optimal*.

We define the *Pareto set* as the set of all the Pareto optimal solutions. Moreover, we refer to the image of the Pareto set w.r.t.  $F$  as the *Pareto front*.

We can now introduce the concept of Pareto stationarity. Under differentiability assumptions, this condition is necessary for all types of Pareto optimality. Moreover, assuming the convexity of the objective functions in problem (1), Pareto stationarity is also a sufficient condition for Pareto optimality.

**Definition 3** A point  $\bar{x} \in \mathbb{R}^n$  is *Pareto-stationary* for problem (1) if we have that

$$\min_{d \in \mathbb{R}^n} \max_{j=1, \dots, m} \nabla f_j(\bar{x})^T d = 0.$$

The concepts of Pareto stationarity, Pareto optimality and convexity are related according to the following lemma.

**Lemma 1** ([8, Theorem 3.1]) *The following statements hold:*

- (i) *if  $\bar{x}$  is locally weakly Pareto optimal, then  $\bar{x}$  is Pareto-stationary for problem (1);*
- (ii) *if  $F$  is convex and  $\bar{x}$  is Pareto-stationary for problem (1), then  $\bar{x}$  is weakly Pareto optimal;*
- (iii) *if  $F$  is twice continuously differentiable,  $\nabla^2 f_j(x) \succ 0$  for all  $j \in \{1, \dots, m\}$  and all  $x \in \mathbb{R}^n$ , and  $\bar{x}$  is Pareto-stationary for problem (1), then  $\bar{x}$  is Pareto optimal.*

Lastly, we introduce a relaxation of Pareto stationarity, called  $\varepsilon$ -Pareto-stationarity. This concept is firstly introduced in [11]: here, we propose a slightly modified version.

**Definition 4** Let  $\varepsilon \geq 0$ . A point  $\bar{x} \in \mathbb{R}^n$  is  $\varepsilon$ -Pareto-stationary for problem (1) if

$$\min_{d \in \mathbb{R}^n} \max_{j=1, \dots, m} \nabla f_j(\bar{x})^T d + \frac{1}{2} \|d\|^2 \geq -\varepsilon.$$

In the following, we briefly review the basic concepts for Quasi-Newton algorithms in multi-objective optimization.

### 2.1 Quasi-Newton methods

If a point  $\bar{x} \in \mathbb{R}^n$  is not Pareto-stationary, then there exists a descent direction w.r.t. all the objectives. The Quasi-Newton direction can be introduced as the solution of the following problem [10]:

$$\min_{d \in \mathbb{R}^n} \max_{j=1, \dots, m} \nabla f_j(\bar{x})^T d + \frac{1}{2} d^T B_j d, \tag{2}$$

where  $B_j \in \mathbb{R}^{n \times n}$ , with  $j \in \{1, \dots, m\}$ , approximates the second derivatives  $\nabla^2 f_j(\bar{x})$ . If the approximation matrices are positive definite, i.e.,  $B_j \succ 0 \forall j \in \{1, \dots, m\}$ , then the function  $\nabla f_j(\bar{x})^T d + (1/2) d^T B_j d$  is strongly convex for each  $j \in \{1, \dots, m\}$ . In this case, problem (2) has a unique minimizer: we denote it by  $d_{QN}(\bar{x})$ . We also indicate with  $\theta_{QN}(\bar{x})$  the optimal value of problem (2) at  $\bar{x}$ . It is trivial to observe that  $\theta_{QN}(x) \leq 0$  for any  $x \in \mathbb{R}^n$ . If  $\bar{x}$  is Pareto-stationary, then  $\theta_{QN}(\bar{x}) = 0$ .

As in [24], we introduce the function  $\mathcal{D} : \mathbb{R}^n \times \mathbb{R}^n \rightarrow \mathbb{R}$ , defined by

$$\mathcal{D}(x, d) = \max_{j=1, \dots, m} \nabla f_j(x)^T d.$$

Any direction  $d$  such that  $\mathcal{D}(\bar{x}, d) < 0$  is a *descent direction* at  $\bar{x}$  for  $F$ . Moreover, the function  $\mathcal{D}(\cdot, \cdot)$  has some properties, which we report in the next lemma.

**Lemma 2** ([24, Lemma 2.2]) *The following statements hold:*

1. for any  $x \in \mathbb{R}^n$  and  $\alpha \geq 0$ , we have  $\mathcal{D}(x, \alpha d) = \alpha \mathcal{D}(x, d)$ .
2. the mapping  $(x, d) \rightarrow \mathcal{D}(x, d)$  is continuous.

As we thoroughly recall in Appendix A.1, the Lagrangian dual problem of (2) is given by:

$$\begin{aligned} \max_{\lambda \in \mathbb{R}^m} \quad & -\frac{1}{2} \lambda^T J_F(\bar{x}) \left[ \sum_{j=1}^m \lambda_j B_j \right]^{-1} J_F(\bar{x})^T \lambda \\ \text{s.t.} \quad & \sum_{j=1}^m \lambda_j = 1, \quad \lambda \geq \mathbf{0}. \end{aligned} \tag{3}$$

Regarding problems (2) and (3), strong duality holds and the Karush-Kuhn-Tucker conditions are sufficient and necessary for optimality. Moreover, denoting by  $\lambda^{QN}(\bar{x}) = (\lambda_1^{QN}(\bar{x}), \dots, \lambda_m^{QN}(\bar{x}))^T$  the optimal Lagrange multipliers vector, we have that

$$\sum_{j=1}^m \lambda_j^{QN}(\bar{x}) = 1, \quad \lambda^{QN}(\bar{x}) \geq \mathbf{0} \tag{4}$$

and

$$d_{QN}(\bar{x}) = - \left[ \sum_{j=1}^m \lambda_j^{QN}(\bar{x}) B_j \right]^{-1} J_F(\bar{x})^T \lambda. \tag{5}$$

Due to the presence of the inverse of the convex combination of the approximation matrices, i.e.,  $\left[ \sum_{j=1}^m \lambda_j B_j \right]^{-1}$ , problem (3) is difficult to solve.

If, for all  $j \in \{1, \dots, m\}$ ,  $B_j = I$ , where  $I \in \mathbb{R}^{n \times n}$  is the identity matrix, problem (2) is identical to the one proposed in [7] to find the steepest common descent direction. We denote the latter by  $d_{SD}(\bar{x})$  and the associated Lagrange multipliers vector by  $\lambda^{SD}(\bar{x}) = (\lambda_1^{SD}(\bar{x}), \dots, \lambda_m^{SD}(\bar{x}))^T$ . Obviously, equations (4)-(5) hold true in this particular case. We further recall a well-known result that will be used in our convergence analysis.

**Lemma 3** *The following statements hold:*

- (i) *the mapping  $d_{SD}(\cdot)$  is continuous;*
- (ii)  *$\bar{x} \in \mathbb{R}^n$  is Pareto-stationary for problem (1) if and only if  $d_{SD}(\bar{x}) = \mathbf{0}$ .*

**Proof** See [27, Lemma 3.3] and [8, Lemma 3.2]. □

Based on the concept of Quasi-Newton direction, a Quasi-Newton approach for multi-objective optimization of strongly convex objective functions is proposed in [10]. In this algorithm, a backtracking Armijo-type line search is used to guarantee the sufficient decrease w.r.t. all the objective functions. The result is formalized by the following lemma.

**Lemma 4** ([7, Lemma 4]) *If  $F$  is continuously differentiable and  $J_F(x)d < \mathbf{0}$ , then there exists some  $\varepsilon > 0$ , which may depend on  $x, d$  and  $\gamma \in (0, 1)$ , such that*

$$F(x + td) < F(x) + \gamma t J_F(x)d$$

for all  $t \in (0, \varepsilon]$ .

**Remark 1** By the definition of  $\mathcal{D}(\cdot, \cdot)$ , we have that  $J_F(x)d \leq \mathbf{1}\mathcal{D}(x, d)$ . Moreover, if  $B_j \succ 0 \forall j \in \{1, \dots, m\}$ , then it follows that  $\mathcal{D}(x, d) < \theta_{QN}(x)$ . Using Lemma 4 and these results, we trivially obtain that, for all  $t \in (0, \varepsilon]$ ,

$$\begin{aligned} F(x + td) &< F(x) + \gamma t J_F(x)d \\ &\leq F(x) + \mathbf{1}\gamma t \mathcal{D}(x, d) \\ &< F(x) + \mathbf{1}\gamma t \theta_{QN}(x). \end{aligned}$$

In many works for MOO [10, 25], the BFGS update formula is independently used for all  $B_j$ , with  $j \in \{1, \dots, m\}$ :

$$B_j^{k+1} = B_j^k - \frac{B_j^k s_k s_k^T B_j^k}{s_k^T B_j^k s_k} + \frac{y_j^k (y_j^k)^T}{s_k^T y_j^k},$$

where  $s_k = x_{k+1} - x_k$  and  $y_j^k = \nabla f_j(x_{k+1}) - \nabla f_j(x_k)$ . We also introduce the formula for updating the inverse of the approximation matrix  $B_j$ , which we denote by  $H_j$ :

$$H_j^{k+1} = \left( I - \rho_j^k y_j^k s_k^T \right)^T H_j^k \left( I - \rho_j^k y_j^k s_k^T \right) + \rho_j^k s_k s_k^T, \tag{6}$$

where  $\rho_j^k = 1 / (s_k^T y_j^k)$ .

Similar to the scalar case [28], for each  $j \in \{1, \dots, m\}$ , if  $s_k^T y_j^k > 0$  and  $B_j^k > 0$ , then  $B_j^{k+1}$  is positive definite. The same property holds true if  $\{H_j^k\}$  is considered. When the objective functions are strictly convex, the condition  $s_k^T y_j^k > 0$  is always satisfied for any pair  $(x_{k+1}, x_k)$  and for each  $j \in \{1, \dots, m\}$ . However, this property is not guaranteed to hold in the general case. In order to overcome this issue, in Quasi-Newton methods for scalar optimization, Wolfe conditions are imposed at each iteration [28].

The Wolfe conditions have been extended to MOO in [12]:

$$\begin{aligned} F(x_k + \alpha d_{QN}(x_k)) &\leq F(x_k) + \mathbf{1}\gamma\alpha\mathcal{D}(x_k, d_{QN}(x_k)), \\ \mathcal{D}(x_k + \alpha d_{QN}(x_k), d_{QN}(x_k)) &\geq \sigma\mathcal{D}(x_k, d_{QN}(x_k)). \end{aligned} \tag{7}$$

Assuming that  $d_{QN}(x_k)$  is a descent direction for  $F$  at  $x_k$  and there exists  $\mathcal{A} \in \mathbb{R}^m$  such that  $F(x_k + \alpha d_{QN}(x_k)) \geq \mathcal{A}$  for all  $\alpha > 0$ , an interval of values exists satisfying these conditions [12, Proposition 3.2]. The theoretical result can be further improved assuming the boundedness of at least one objective function [29, Proposition 1].

However, even if Wolfe conditions are satisfied, it may occur that  $s_k^T y_j^k \leq 0$  for some  $j \in \{1, \dots, m\}$ . In other words, considering that

$$s_k = x_{k+1} - x_k = \alpha_k d_{QN}(x_k), \tag{8}$$

we may have that, for some  $j \in \{1, \dots, m\}$ ,

$$[\nabla f_j(x_{k+1}) - \nabla f_j(x_k)]^T d_{QN}(x_k) \leq 0,$$

which can be also re-written in the form

$$\nabla f_j(x_{k+1})^T d_{QN}(x_k) \leq \nabla f_j(x_k)^T d_{QN}(x_k). \tag{9}$$

For this reason, a different formula for updating  $B_j$  is introduced in [24]. The corresponding update formula for  $H_j$  remains similar to (6), except that  $\rho_j^k$  is now defined as

$$\rho_j^k = \begin{cases} 1 / (s_k^T y_j^k) & \text{if } s_k^T y_j^k > 0, \\ 1 / [\mathcal{D}(x_{k+1}, s_k) - \nabla f_j(x_k)^T s_k] & \text{otherwise.} \end{cases} \tag{10}$$

Using the above update rule,  $\rho_j^k$  is proved to be strictly positive even when  $s_k^T y_j^k \leq 0$ . Thus,  $H_j^{k+1}$  and, consequently,  $B_j^{k+1}$  always remain positive definite.

### 2.2 Single Hessian matrix approximation

The use of a single positive definite matrix  $B$  was proposed in [23] to approximate  $\nabla^2 f_1(x), \dots, \nabla^2 f_m(x)$ . In this case, problem (2) becomes

$$\min_{d \in \mathbb{R}^n} \max_{j=1, \dots, m} \nabla f_j(\bar{x})^T d + \frac{1}{2} d^T B d,$$

while the dual (3) changes into

$$\begin{aligned} \max_{\lambda \in \mathbb{R}^m} \quad & -\frac{1}{2} \lambda^T J_F(\bar{x}) B^{-1} J_F(\bar{x})^T \lambda \\ \text{s.t.} \quad & \sum_{j=1}^m \lambda_j = 1, \quad \lambda \geq \mathbf{0}. \end{aligned} \tag{11}$$

Now, the descent direction can accordingly be computed as

$$d_{MQN}(\bar{x}) = -B^{-1} J_F(\bar{x})^T \lambda^{MQN}(\bar{x}),$$

where  $\lambda^{MQN}(\bar{x}) = (\lambda_1^{MQN}(\bar{x}), \dots, \lambda_m^{MQN}(\bar{x}))^T$  indicate the Lagrange multipliers vector.

The difficult term  $[\sum_{j=1}^m \lambda_j B_j]^{-1}$  appearing in (3) is replaced by  $B^{-1}$ . As a consequence, problem (11) reduces to a linearly constrained, convex quadratic program which is easy to solve.

The unique matrix  $B$  is obtainable as the approximation of a convex combination of matrices. For this purpose, slightly modified BFGS update formulas are introduced in [23]:

$$B^{k+1} = B^k - \frac{B^k s_k s_k^T B^k}{s_k^T B^k s_k} + \frac{u_k u_k^T}{s_k^T u_k}; \tag{12}$$

$$H^{k+1} = \left( I - \rho^k u_k s_k^T \right)^T H^k \left( I - \rho^k u_k s_k^T \right) + \rho^k s_k s_k^T, \tag{13}$$

with  $\rho^k = 1 / (s_k^T u_k)$  and  $u_k = \sum_{j=1}^m \lambda_j^{MQN}(x_k) y_j^k$ .

### 3 A limited memory Quasi-Newton method

In this section, we introduce a new Limited Memory Quasi-Newton approach for MOO, whose algorithmic scheme is reported in Algorithm 1.



**Algorithm 1** Limited Memory Quasi-Newton Method

**Require:**  $F : \mathbb{R}^n \rightarrow \mathbb{R}^m, x_0 \in \mathbb{R}^n, \gamma \in (0, 1/2), \sigma \in (\gamma, 1), H^0 \succ 0, M \in \mathbb{N}_+$ .

- 1: **for**  $k = 0, 1, 2, \dots$  **do**
- 2:   Estimate  $\mathcal{R}^k = H^k J_F(x_k)^T \in \mathbb{R}^{n \times m}$  (two-loop recursive procedure)
- 3:   Compute

$$\begin{aligned} \theta_{LM}(x_k) = \max_{\lambda \in \mathbb{R}^m} & -\frac{1}{2} \lambda^T J_F(x_k) \mathcal{R}^k \lambda \\ \text{s.t.} & \sum_{j=1}^m \lambda_j = 1, \quad \lambda \geq \mathbf{0} \end{aligned} \tag{14}$$

- 4:   Let  $\lambda^{LM}(x_k) = (\lambda_0^{LM}(x_k), \dots, \lambda_m^{LM}(x_k))^T$  be the Lagrange multipliers vector related to  $\theta_{LM}(x_k)$
- 5:   Let  $d_{LM}(x_k) = -\mathcal{R}^k \lambda^{LM}(x_k)$
- 6:   Choose  $\alpha_k > 0$  (trying first  $\alpha_k = 1$ ) such that

$$F(x_k + \alpha_k d_{LM}(x_k)) \leq F(x_k) + \mathbf{1} \gamma \alpha_k \mathcal{D}(x_k, d_{LM}(x_k)), \tag{15}$$

$$\mathcal{D}(x_k + \alpha_k d_{LM}(x_k), d_{LM}(x_k)) \geq \sigma \mathcal{D}(x_k, d_{LM}(x_k)) \tag{16}$$

- 7:   Let  $x_{k+1} = x_k + \alpha_k d_{LM}(x_k)$
- 8:   **if**  $k \geq M$  **then**
- 9:     Discard vectors pair  $(s_{k-M}, u_{k-M})$  from storage
- 10:   **end if**
- 11:   Compute and save

$$s_k = x_{k+1} - x_k \tag{17}$$

$$u_k = \sum_{j=1}^m \lambda_j^{LM}(x_k) [\nabla f_j(x_{k+1}) - \nabla f_j(x_k)] \tag{18}$$

12: **end for**

In the proposed approach, we use a single positive definite matrix  $H^k$  at each iteration  $k$ . In Sect. 3.2, we introduce the update formula for  $H^k$ , which is slightly different w.r.t. the one introduced in [23]. As in L-BFGS for scalar optimization, we maintain only a finite number  $M$  of vectors pairs  $\{(s_i, u_i)\}$  in memory: the oldest one is discarded each time a new vectors pair is calculated. These pairs are used in a two-loop recursive procedure to efficiently carry out the matrix multiplication  $\mathcal{R}^k = H^k J_F(x_k)^T$  (Sect. 3.1). This procedure is essentially an extension for MOO of the one used in L-BFGS [28]. The matrix  $\mathcal{R}^k$  is then used in problem (14) at step 3 of the algorithm: the latter is simply derived from problem (11) substituting  $H^k J_F(x_k)^T$  with  $\mathcal{R}^k$ . We denote by  $\theta_{LM}(x_k)$  the optimal value of problem (14) at  $x_k$ . Moreover, we respectively denote by  $\lambda^{LM}(x_k)$  (Line 4) and  $d_{LM}(x_k)$  (Line 5) the Lagrange multipliers vector and the direction corresponding to  $\theta_{LM}(x_k)$ . Note that (4) is valid in this context too. Finally, in Line 6, a Wolfe line search is carried out to find a step size  $\alpha_k$  along the direction  $d_{LM}(x_k)$ , satisfying the Wolfe conditions for MOO (Sect. 3.3).

In the following, we deeply analyze the various aspects of Algorithm 1.

### 3.1 Two-loop recursive procedure for MOO

In L-BFGS, one of the most relevant features is the two-loop recursive procedure which, at any iteration  $k$ , given the vectors pairs saved in memory, allows to efficiently compute the product  $H^k \nabla f(x_k)$ , where  $f(\cdot)$  indicates the objective function [28]. We remind, indeed, that in scalar optimization the negative of this product identifies the Quasi-Newton descent direction:  $d(x_k) = -H^k \nabla f(x_k)$ . Using this procedure, we do not need to store the matrix  $H$  in memory. This property could be crucial when high dimensional problems are considered: in these cases, maintaining and updating the matrix  $H$ , which is dense in general, could be extremely inefficient. In this work, we propose an extension of this procedure for MOO: the algorithmic scheme is reported in Algorithm 2.

---

#### Algorithm 2 Two-Loop Recursive Procedure

---

**Require:**  $k \in \mathbb{N}$ ,  $M \in \mathbb{N}_+$ ,  $\{(s_i, u_i) \mid i \in [\max\{0, k - M\}, k - 1]\}$ ,  $J_F(x_k) \in \mathbb{R}^{m \times n}$ ,  $H^0 > 0$ .

- 1:  $q = J_F(x_k)^T$
- 2: **if**  $k = 0$  **then**
- 3:    $\mathcal{R}^0 = H^0 q$
- 4: **else**
- 5:   **for**  $i = k - 1, \dots, \max\{0, k - M\}$  **do**
- 6:      $\alpha_i = \rho^i q^T s_i$
- 7:      $q = q - u_i \alpha_i^T$
- 8:   **end for**
- 9:    $\mathcal{R}^k = H^0 q$
- 10:   **for**  $i = \max\{0, k - M\}, \dots, k - 1$  **do**
- 11:      $\beta_i = \rho^i (\mathcal{R}^k)^T u_i$
- 12:      $\mathcal{R}^k = \mathcal{R}^k + s_i (\alpha_i - \beta_i)^T$
- 13:   **end for**
- 14: **end if**
- 15: **return**  $\mathcal{R}^k$

---

With respect to the scalar optimization case, this procedure computes the product  $H^k J_F(x_k)^T$ . The same result could be obtained repeating  $m$  times the procedure for scalar optimization to find  $H^k \nabla f_j(x_k)$  for all  $j \in \{1, \dots, m\}$ . In both cases,  $m(4Mn + n)$  multiplications are required. However, Algorithm 2 allows to exploit the optimized operations of software libraries for vector calculus.

The employment of Algorithm 2 is possible thanks to some properties of (13). Indeed, the latter can be re-written in the following form [28]:

$$\begin{aligned}
 H^k &= \left[ (V^{k-1})^T \dots (V^{k-M})^T \right] H^{k-M} \left[ V^{k-M} \dots V^{k-1} \right] \\
 &\quad + \rho^{k-M} \left[ (V^{k-1})^T \dots (V^{k-M+1})^T \right] s_{k-M} s_{k-M}^T \left[ V^{k-M+1} \dots V^{k-1} \right] \\
 &\quad + \rho^{k-M+1} \left[ (V^{k-1})^T \dots (V^{k-M+2})^T \right] s_{k-M+1} s_{k-M+1}^T \left[ V^{k-M+2} \dots V^{k-1} \right] \\
 &\quad + \dots
 \end{aligned}$$

$$+\rho^{k-1}s_{k-1}s_{k-1}^T,$$

where  $V^i = I - \rho^i u_i s_i^T$ . As in L-BFGS, the *exact* matrix  $H^{k-M}$  is substituted by a suitable sparse positive definite matrix  $H^0$ . From this last equation, the two-loop recursive procedure to compute the product  $H^k J_F(x_k)^T$  is derived. We refer the reader to [28] for more details.

### 3.2 Definition of H

In the proposed approach, we use a single positive definite matrix  $H$ . As in [23] the update formula (13) is used. However, taking inspiration from (10), we use a different definition of  $\rho^k$ :

$$\rho^k = \begin{cases} 1/(s_k^T u_k) & \text{if } s_k^T u_k > 0, \\ 1/\left\{\sum_{j=1}^m \lambda_j^{LM}(x_k) [\mathcal{D}(x_{k+1}, s_k) - \nabla f_j(x_k)^T s_k]\right\} & \text{otherwise.} \end{cases} \tag{19}$$

As in [24], we carry out a line search to find a step size satisfying the Wolfe conditions for MOO (Sect. 3.3). However, recalling the reasoning in Sect. 2.1, in order to ensure that  $H^{k+1} \succ 0$ , we force through (19)  $\rho_k$  to be positive even when  $s_k^T u_k \leq 0$ . We formalize this statement in the following proposition.

**Proposition 1** *Considering a generic iteration  $k$  of Algorithm 1, let  $x_k \in \mathbb{R}^n$ ,  $d_{LM}(x_k) \in \mathbb{R}^n$  be a direction such that  $\mathcal{D}(x_k, d_k) < 0$ ,  $\alpha_k > 0$  be a step size along  $d_{LM}(x_k)$  and  $\lambda^{LM}(x_k)$  be the Lagrange multipliers vector obtained solving problem (14). If  $\rho^k$  is updated by (19), then  $\rho^k$  is positive.*

**Proof** See Appendix A.2. □

**Remark 2** In the single objective case, the update formula (13) for  $H^k$  coincides with the classical BFGS rule. Indeed, it is sufficient to realize that, since  $\lambda^{LM}(x_k)$  lies in the unit simplex by (4), then  $u_k = y^k$ . Moreover, the same reasoning can be applied with (19) to get that  $\rho^k = 1/(s_k^T y^k)$ . Hence, the two-loop recursive procedure reduces to that of L-BFGS. In turn, the overall Algorithm 1 is nothing but L-BFGS, since  $d^k = -H^k \nabla f(\bar{x})$  and Wolfe conditions are imposed by the line search.

**Remark 3** The procedure in Algorithm 2 cannot be used if we consider an approximation matrix for each objective function, as in problem (2). In such case, both in the primal and in the dual problem (3) the matrices are tied to the problem variables; for example, when solving (3), the product  $[\sum_{j=1}^m \lambda_j B_j]^{-1} J_F(\bar{x})^T$  would be recomputed any time a different solution  $\lambda$  is considered. The use of a single positive definite matrix prevents this issue: matrix multiplication  $H J_F(\bar{x})^T$  can be computed only once, before solving subproblem (11), making it possible to exploit the efficiency of the two-loop recursive procedure.

### 3.3 Wolfe line search

In this section, we introduce a simple line search scheme to find a step size  $\alpha$  along a given direction  $d_k$  satisfying the Wolfe conditions:

$$F(x_k + \alpha d_k) \leq F(x_k) + \mathbf{1}\gamma\alpha\mathcal{D}(x_k, d_k), \quad (20)$$

$$\mathcal{D}(x_k + \alpha d_k, d_k) \geq \sigma\mathcal{D}(x_k, d_k). \quad (21)$$

Before proceeding, we make a reasonable assumption. Then, we prove that there exists an interval of values satisfying the Wolfe conditions. Note that an analogous result has been obtained in [12, 29] under the different assumptions also reported in Sect. 2.1 of this manuscript.

**Assumption 1** The objective function  $F$  has bounded level sets in the multi-objective sense, i.e., the set  $\mathcal{L}_F(z) = \{x \in \mathbb{R}^n \mid F(x) \leq z\}$  is bounded for any  $z \in \mathbb{R}^m$ .

**Proposition 2** Let Assumption 1 hold. Let  $x_k \in \mathbb{R}^n$  and assume that  $d_k \in \mathbb{R}^n$  is a direction such that  $\mathcal{D}(x_k, d_k) < 0$ ,  $\gamma \in (0, 1/2)$  and  $\sigma \in (\gamma, 1)$ . Then, there exists an interval of values  $[\alpha_l, \alpha_u]$ , with  $0 < \alpha_l < \alpha_u$ , such that for all  $\alpha \in [\alpha_l, \alpha_u]$  equations (20) and (21) hold.

**Proof** See Appendix B. □

After proving the existence of an interval of values satisfying the Wolfe conditions, we report the algorithmic scheme of the considered line search.

---

#### Algorithm 3 Wolfe Line Search

---

**Require:**  $F : \mathbb{R}^n \rightarrow \mathbb{R}^m$ ,  $x_k \in \mathbb{R}^n$ ,  $d_k \in \mathbb{R}^n$ ,  $\gamma \in (0, 1/2)$ ,  $\sigma \in (\gamma, 1)$ .

```

1:  $\alpha_l^0 = 0, \alpha_u^0 = \infty, \alpha^0 = 1$ 
2: for  $t = 0, 1, 2, \dots$  do
3:   if  $\exists j$  s.t.  $f_j(x_k + \alpha^t d_k) > f_j(x_k) + \gamma\alpha^t \mathcal{D}(x_k, d_k)$  then
4:      $\alpha_u^{t+1} = \alpha^t$ 
5:      $\alpha_l^{t+1} = \alpha_l^t$ 
6:   else
7:      $\alpha_u^{t+1} = \alpha_u^t$ 
8:     if  $\mathcal{D}(x_k + \alpha^t d_k, d_k) < \sigma\mathcal{D}(x_k, d_k)$  then
9:        $\alpha_l^{t+1} = \alpha^t$ 
10:    else
11:      return  $\alpha^t$ 
12:    end if
13:  end if
14:  Choose  $\alpha^{t+1} \in (\alpha_l^{t+1}, \alpha_u^{t+1})$ 
15: end for

```

---

Starting from  $\alpha_l^0 = 0, \alpha_u^0 = \infty$ , the core idea of the line search is that of reducing the interval  $[\alpha_l^t, \alpha_u^t]$  until a valid step size  $\alpha^t$  is found. At the beginning of the for-loop, the Wolfe sufficient decrease condition (20) is checked. If it is not satisfied by  $\alpha^t$ , we update  $\alpha_u^t$  and we maintain the same value for  $\alpha_l^t$  (Lines 4 and 5). Otherwise,  $\alpha_u^t$  is not

updated (Line 7) and we check if the Wolfe curvature condition (21) is satisfied by  $\alpha^t$ : if it is, both Wolfe conditions are satisfied and, then, the current step size value is returned; else  $\alpha_l^t$  is updated according to Line 9. After updating  $\alpha_u^t$  or  $\alpha_l^t$ , a new value for the step size  $\alpha^t$  is chosen in the interval  $(\alpha_l^t, \alpha_u^t)$  and the process is repeated.

In the next lemma, we state some properties related to the interval upper and lower bounds  $\alpha_u^t$  and  $\alpha_l^t$ .

**Lemma 5** Consider a generic iteration  $t$  of Algorithm 3. Let  $x_k \in \mathbb{R}^n$  and  $d_k$  be a direction such that  $\mathcal{D}(x_k, d_k) < 0$ . Then, we have the following properties:

(i) if  $\alpha_u^t < \infty$ , then

$$\exists j(\alpha_u^t) \text{ s.t. } f_{j(\alpha_u^t)}(x_k + \alpha_u^t d_k) > f_{j(\alpha_u^t)}(x_k) + \gamma \alpha_u^t \mathcal{D}(x_k, d_k); \tag{22}$$

(ii)  $\alpha_l^t$  is such that

$$F(x_k + \alpha_l^t d_k) \leq F(x_k) + \mathbf{1} \gamma \alpha_l^t \mathcal{D}(x_k, d_k), \tag{23}$$

$$\mathcal{D}(x_k + \alpha_l^t d_k, d_k) < \sigma \mathcal{D}(x_k, d_k). \tag{24}$$

**Proof** See Appendix B. □

In the following proposition, we state that the proposed line search is well defined, i.e., it terminates after a finite number of iterations returning a step size satisfying the Wolfe conditions.

**Proposition 3** Let Assumption 1 hold,  $\delta \in [1/2, 1)$ ,  $\eta > 1$  and let  $\{\alpha_l^t, \alpha_u^t, \alpha^t\}$  be the sequence generated by Algorithm 3. Assume that:

1.  $d_k \in \mathbb{R}^n$  is a descent direction for  $F$  at  $x_k \in \mathbb{R}^n$ ;
2. for all  $t > 0$ , the step size  $\alpha^t$  is chosen so that

a) if  $\alpha_u^t = \infty$ ,

$$\alpha^t \geq \eta \max \{ \alpha_l^t, \alpha^0 \}, \tag{25}$$

b) if  $\alpha_u^t < \infty$ ,

$$\max \{ (\alpha^t - \alpha_l^t), (\alpha_u^t - \alpha^t) \} \leq \delta (\alpha_u^t - \alpha_l^t).$$

Then Algorithm 3 is well defined, i.e., it stops after a finite number of iterations returning a step size  $\hat{\alpha}$  satisfying the Wolfe conditions for MOO.

**Proof** See Appendix B. □

**Remark 4** To the best of our knowledge, the first Wolfe line search for MOO was proposed in [29]. Our line search is just a simpler algorithm that is guaranteed to produce a point satisfying the Wolfe conditions. In fact, we think that not using an inner solver, as done in [29], could be a performance disadvantage and, in addition,

smarter strategies to set the trial step size may be integrated. We decided not to compare the two line searches, since finding new efficient methodologies to find the step size is not the focus of our work. Moreover, we are confident that the experimental results of Sect. 5 would be similar regardless the employed Wolfe line search.

### 4 Convergence analysis

In this section, we show the convergence properties of our Limited Memory Quasi-Newton approach. Before proceeding, similarly to what is done in [30], we need to make some assumptions about the objective function  $F$  and the initial approximation matrix  $H^0$ .

**Assumption 2** We assume that:

- (i)  $F$  is twice continuously differentiable;
- (ii) the set  $\mathcal{L}_F(F(x_0)) = \{x \in \mathbb{R}^n \mid F(x) \leq F(x_0)\}$  is convex;
- (iii)  $\exists a, b \in \mathbb{R}_+$  such that, for all  $j \in \{1, \dots, m\}$ ,

$$a \|z\|^2 \leq z^T \nabla^2 f_j(x) z \leq b \|z\|^2, \quad \forall z \in \mathbb{R}^n, \forall x \in \mathcal{L}_F(F(x_0)).$$

**Assumption 3** The matrix  $H^0$  is chosen such that the norms  $\|H^0\|$  and  $\|B^0\|$  are bounded.

**Remark 5** Assumption 2 implies Assumption 1. Indeed,  $f_j(\cdot)$  is strongly convex for all  $j \in \{1, \dots, m\}$ , and thus has all the level sets bounded. Also, by Assumption 1, we have that Propositions 2 and 3 concerning the line search remain valid.

**Remark 6** By Assumption 2, we have  $s_k^T y_j^k > 0$  for any  $k$  and for all  $j \in \{1, \dots, m\}$ . Then, considering (18) and since  $\lambda^{LM}(x_k)$  satisfies (4), we have that  $s_k^T u_k > 0$ . Then, according to (19),  $\rho^k = 1 / (s_k^T u_k)$  and, thus, we update  $B^k$  and  $H^k$  using (12) and (13), respectively.

In order to carry out the theoretical analysis, we take as reference Algorithm 4, which is mathematically equivalent to Algorithm 1 but makes it more explicit how the approximation of  $H^k$  is computed, i.e., applying  $M$  times the update rule (13) starting from  $H^0$ . In the remainder of the section, we will consider the approximation matrix  $B^k$  for the sake of clarity; the results are obviously the same if we consider the matrix  $H^k$ . Finally, note that Algorithm 4 is only used in this section, since, unlike Algorithm 1, it requires to store the entire matrix in memory.

For the theoretical analysis, we also need to introduce the formula for the trace and the determinant of the matrix  $B^{k+1}$ :

$$\text{Tr}(B^{k+1}) = \text{Tr}(B^k) - \frac{\|B^k s_k\|^2}{s_k^T B^k s_k} + \frac{\|u_k\|^2}{s_k^T u_k}, \tag{26}$$

$$\det(B^{k+1}) = \det(B^k) \frac{s_k^T u_k}{s_k^T B^k s_k}. \tag{27}$$

Note that these expressions hold when (12) is used to update the matrix  $B^k$ , which is always the case here by Assumption 2. We also introduce some basic notation that will be useful in the following analysis.

**Notation:** We will denote by  $\Omega(B^k)$  the eigenvalues set of the matrix  $B^k$ ; by  $\omega_m(B^k)$  and  $\omega_M(B^k)$  we indicate the minimum and the maximum eigenvalue, respectively; we refer by  $\beta^k$  to the angle between the vectors  $s_k$  and  $B^k s_k$ . Concerning  $\beta^k$ , we also recall the formula of the cosine:

$$\cos \beta^k = \frac{s_k^T B^k s_k}{\|s_k\| \|B^k s_k\|}. \tag{28}$$

---

**Algorithm 4** Limited Memory Quasi-Newton Method

---

**Require:**  $F : \mathbb{R}^n \rightarrow \mathbb{R}^m, x_0 \in \mathbb{R}^n, \gamma \in (0, 1/2), \sigma \in (\gamma, 1), B^0 > 0, M \in \mathbb{N}_+$ .

- 1: Let  $h = 0$
- 2: **for**  $k = 0, 1, 2, \dots$  **do**
- 3:    Compute  $\theta_{LM}(x_k)$  and  $\lambda^{LM}(x_k)$  solving problem (11)
- 4:    Let  $d_{LM}(x_k) = -\left(B^k\right)^{-1} J_F(x_k)^T \lambda^{LM}(x_k)$
- 5:    Choose  $\alpha_k > 0$  s.t. (15) and (16) hold
- 6:    Let  $x_{k+1} = x_k + \alpha_k d_{LM}(x_k)$
- 7:    **if**  $k \geq M$  **then**
- 8:        Discard vectors pair  $(s_{k-M}, u_{k-M})$  from storage
- 9:        Set  $h = k - M + 1$
- 10:    **end if**
- 11:    Compute and save  $(s_k, u_k)$  according to (17)-(18)
- 12:    Let  $B_{(0)}^k = B^0$
- 13:    **for**  $l = 0, \dots, \min\{k, M - 1\}$  **do**
- 14:        Set

$$B_{(l+1)}^k = B_{(l)}^k - \frac{B_{(l)}^k s_{l+h} s_{l+h}^T B_{(l)}^k}{s_{l+h}^T B_{(l)}^k s_{l+h}} + \frac{u_{l+h} u_{l+h}^T}{s_{l+h}^T u_{l+h}} \tag{29}$$

- 15:    **end for**
  - 16:    Let  $B^{k+1} = B_{(\min\{k, M-1\}+1)}^k$
  - 17: **end for**
- 

We are now able to begin the convergence analysis with three technical lemmas.

**Lemma 6** *Let Assumption 2 hold and consider the sequences  $\{x_k\}$  and  $\{d_{LM}(x_k)\}$  generated by Algorithm 4. Then,*

$$\sum_{k \geq 0} \frac{\mathcal{D}(x_k, d_{LM}(x_k))^2}{\|d_{LM}(x_k)\|^2} < \infty.$$

**Proof** The result follows as in Proposition 3.3 in [24], as the assumptions made in the latter are trivially implied by Assumption 2. □

**Lemma 7** Assume that Assumption 2 holds. Let  $\{x_k\}$  be the sequence generated by Algorithm 4. Then, for all  $k \geq 0$ , we have that

$$\mathcal{D}(x_k, d_{LM}(x_k)) \leq -\frac{\cos \beta^k}{2} \|d_{LM}(x_k)\| \|d_{SD}(x_k)\|.$$

**Proof** The proof is analogous to the one of Lemma 4.2 in [24], taking into account that we have a single approximation matrix  $B^k$ . □

**Lemma 8** Let Assumptions 2 and 3 hold. Moreover, let  $\{x_k\}$  be the sequence generated by Algorithm 4. Then, there exists a constant  $\delta > 0$  such that, for all  $k \geq 0$ , we have that

$$\cos \beta^k \geq \delta.$$

**Proof** Let us consider  $k \geq 0$ ,  $\tau \in [0, 1]$  and the point  $x_k + \tau s_k$ . By Assumption 2 and equations (15) and (17), we have that  $x_k + \tau s_k \in \mathcal{L}_F(F(x_0))$ . Also recalling that  $\lambda^{LM}(x_k)$  satisfies (4), we obtain for any  $z \in \mathbb{R}^n$  that

$$\int_0^1 a \|z\|^2 d\tau \leq \int_0^1 z^T \sum_{j=1}^m \lambda_j^{LM}(x_k) \nabla^2 f_j(x_k + \tau s_k) z d\tau \leq \int_0^1 b \|z\|^2 d\tau$$

and, then,

$$a \|z\|^2 \leq z^T \int_0^1 \sum_{j=1}^m \lambda_j^{LM}(x_k) \nabla^2 f_j(x_k + \tau s_k) z d\tau \leq b \|z\|^2. \tag{29}$$

For  $z = s_k$  we thus obtain

$$a \|s_k\|^2 \leq s_k^T \int_0^1 \sum_{j=1}^m \lambda_j^{LM}(x_k) \nabla^2 f_j(x_k + \tau s_k) s_k d\tau \leq b \|s_k\|^2. \tag{30}$$

Defining

$$I_k = \int_0^1 \sum_{j=1}^m \lambda_j^{LM}(x_k) \nabla^2 f_j(x_k + \tau s_k) d\tau \tag{31}$$

and recalling (18), we solve the integral:

$$\begin{aligned} I_k s_k &= \sum_{j=1}^m \lambda_j^{LM}(x_k) \int_0^1 \nabla^2 f_j(x_k + \tau s_k) s_k d\tau \\ &= \sum_{j=1}^m \lambda_j^{LM}(x_k) [\nabla f_j(x_{k+1}) - \nabla f_j(x_k)] = u_k. \end{aligned} \tag{32}$$



Given this last result and equation (30), we obtain that

$$a \|s_k\|^2 \leq s_k^T u_k \leq b \|s_k\|^2$$

and, thus, considering the left-hand side,

$$\frac{s_k^T u_k}{\|s_k\|^2} \geq a. \tag{33}$$

Furthermore, if we consider  $z = I_k^{1/2} s_k$  in (29), with  $I_k^{1/2}$  being the positive definite square root of  $I_k$ , we get

$$a \|I_k^{1/2} s_k\|^2 \leq (I_k^{1/2} s_k)^T \int_0^1 \sum_{j=1}^m \lambda_j^{LM}(x_k) \nabla^2 f_j(x_k + \tau s_k) d\tau (I_k^{1/2} s_k) \leq b \|I_k^{1/2} s_k\|^2$$

and, recalling (31),

$$a (s_k^T I_k s_k) \leq s_k^T I_k^2 s_k \leq b (s_k^T I_k s_k).$$

Then, given Remark 6 and equation (32), focusing on the right-hand side, we have

$$\frac{\|u_k\|^2}{s_k^T u_k} \leq b.$$

Now, recalling Assumption 3 and equation (29), we apply recursively (26) and we obtain that

$$\begin{aligned} \text{Tr}(B^{k+1}) &= \text{Tr}(B_{(0)}^k) - \sum_{l=0}^{\min\{k, M-1\}} \frac{\|B_{(l)}^k\|_{s_{l+h}}^2}{s_{l+h}^T B_{(l)}^k s_{l+h}} + \sum_{l=0}^{\min\{k, M-1\}} \frac{\|u_{l+h}\|^2}{s_{l+h}^T u_{l+h}} \\ &\leq \text{Tr}(B^0) + \sum_{l=0}^{\min\{k, M-1\}} \frac{\|u_{l+h}\|^2}{s_{l+h}^T u_{l+h}} \\ &\leq \text{Tr}(B^0) + (\min\{k, M-1\} + 1) b \leq \tilde{b}, \end{aligned} \tag{34}$$

for some  $\tilde{b} > 0$ , where the inequalities come from the fact that, for all  $k \geq 0$  and  $l = 0, \dots, \min\{k, M-1\}$ ,  $B_{(l)}^k$  is positive definite (cf. the instructions of Algorithm 4 and Remark 6). We can apply a similar reasoning with the determinant formula (27):

$$\begin{aligned} \det(B^{k+1}) &= \det(B_{(0)}^k) \prod_{l=0}^{\min\{k, M-1\}} \frac{s_{l+h}^T u_{l+h}}{s_{l+h}^T B_{(l)}^k s_{l+h}} \\ &= \det(B^0) \prod_{l=0}^{\min\{k, M-1\}} \frac{s_{l+h}^T u_{l+h}}{\|s_{l+h}\|^2} \frac{\|s_{l+h}\|^2}{s_{l+h}^T B_{(l)}^k s_{l+h}}. \end{aligned}$$

From (34), we deduce that the greatest eigenvalue of  $B^k_{(l)}$  is smaller than  $\tilde{b}$ . Thus, given Assumption 3 and equation (33), we get that

$$\det(B^{k+1}) \geq \det(B^0) \left(\frac{a}{\tilde{b}}\right)^{\min\{k, M-1\}+1} \geq \tilde{a}, \tag{35}$$

where  $\tilde{a} > 0$ .

Then, by (28), the min-max theorem and the triangle inequality, we have:

$$\cos \beta^k = \frac{s_k^T B^k s_k}{\|s_k\| \|B^k s_k\|} \geq \frac{\omega_m(B^k) \|s_k\|^2}{\|B^k\| \|s_k\|^2} = \frac{\omega_m(B^k)}{\|B^k\|}.$$

We know that:

- by definition of trace and determinant, recalling (34) and (35), we get

$$\det(B^k) = \prod_{\omega \in \Omega(B^k)} \omega \leq (n-1)\omega_M(B^k) \omega_m(B^k),$$

and thus

$$\omega_m(B^k) \geq \frac{\det(B^k)}{(n-1)\omega_M(B^k)} \geq \frac{\tilde{a}}{(n-1)\text{Tr}(B^k)} \geq \frac{\tilde{a}}{(n-1)\tilde{b}};$$

- considering the euclidean norm and that  $B^k$  is a real positive definite matrix,

$$\|B^k\| \leq \omega_M(B^k) \leq \text{Tr}(B^k) \leq \tilde{b}.$$

Joining the last three results, we obtain that

$$\cos \beta^k \geq \frac{\omega_m(B^k)}{\|B^k\|} \geq \frac{\tilde{a}}{(n-1)\tilde{b}^2} > 0,$$

where the last inequality comes from the definitions of  $\tilde{a}$  and  $\tilde{b}$ . Thus, we get the thesis choosing

$$\delta = \frac{\tilde{a}}{(n-1)\tilde{b}^2}.$$

□

In the next proposition, we state that the sequence of points produced by Algorithm 4 converges to a Pareto optimal point.

**Proposition 4** *Let Assumptions 2 and 3 hold. Assume that  $\{x_k\}$  is the sequence generated by Algorithm 4. Then,  $\{x_k\}$  converges to a Pareto optimal point  $x^*$  for problem (1).*

**Proof** By Lemmas 7 and 8, we know that there exists a constant  $\delta > 0$  such that, for all  $k \geq 0$ ,

$$\mathcal{D}(x_k, d_{LM}(x_k)) \leq -\frac{\cos \beta^k}{2} \|d_{LM}(x_k)\| \|d_{SD}(x_k)\| \leq -\frac{\delta}{2} \|d_{LM}(x_k)\| \|d_{SD}(x_k)\|.$$

Considering this last result and Lemma 6, we obtain that

$$\infty > \sum_{k \geq 0} \frac{\mathcal{D}(x_k, d_{LM}(x_k))^2}{\|d_{LM}(x_k)\|^2} \geq \sum_{k \geq 0} \frac{\delta^2}{4} \|d_{SD}(x_k)\|^2,$$

and, thus,

$$\lim_{k \rightarrow \infty} d_{SD}(x_k) = \mathbf{0}. \tag{36}$$

By (15), we know that, for all  $k \geq 0$ ,  $x_k \in \mathcal{L}_F(F(x_0))$ . Since  $\mathcal{L}_F(F(x_0))$  is compact (Remark 5), there exists a subsequence  $K \subseteq \{0, 1, \dots\}$  such that

$$\lim_{\substack{k \rightarrow \infty \\ k \in K}} x_k = x^*. \tag{37}$$

Recalling Lemma 3 and equation (36), we have that  $d_{SD}(x^*) = \mathbf{0}$  and, thus,  $x^*$  is Pareto-stationary for problem (1). Therefore, by Lemma 1 and Assumption 2, we conclude that  $x^*$  is Pareto optimal.

Now, let us assume, by contradiction, that there exists another subsequence  $\tilde{K} \subseteq \{0, 1, \dots\}$  such that

$$\lim_{\substack{k \rightarrow \infty \\ k \in \tilde{K}}} x_k = \tilde{x}, \tag{38}$$

with  $\tilde{x} \neq x^*$ .

We prove that  $F(\tilde{x}) \neq F(x^*)$ . If it were false, since by Assumption 2  $F$  is strongly convex and  $\mathcal{L}_F(F(x_0))$  is convex, for all  $t \in (0, 1)$ , we would get that

$$F(t\tilde{x} + (1-t)x^*) < tF(\tilde{x}) + (1-t)F(x^*) = F(x^*).$$

But, in this case, we would contradict the fact that  $x^*$  is Pareto optimal.

Then, given that  $x^*$  is Pareto optimal and that  $F(\tilde{x}) \neq F(x^*)$ ,

$$\exists \tilde{j} \in \{1, \dots, m\} \text{ such that } f_{\tilde{j}}(x^*) < f_{\tilde{j}}(\tilde{x}).$$

Now, recalling (37) and (38), there exist  $k \in K$  and  $\tilde{k} \in \tilde{K}$  such that  $k < \tilde{k}$  and

$$f_{\tilde{j}}(x_k) < f_{\tilde{j}}(x_{\tilde{k}}).$$

But, since (15) holds at each iteration of Algorithm 4, we implicitly have that the sequence  $\{f_j(x_k)\}$  is decreasing, for all  $j \in \{1, \dots, m\}$ . Thus, we get a contradiction and we conclude that

$$\lim_{k \rightarrow \infty} x_k = x^*,$$

with  $x^*$  being Pareto optimal. □

In the rest of the section, we discuss the convergence rate of Algorithm 4. We first have to provide a technical result.

**Lemma 9** *Let Assumptions 2 and 3 hold. Moreover, let  $\{x_k\}$  be the sequence generated by Algorithm 4 and  $x^*$  be the Pareto optimal point to which the sequence converges. Then, for all  $k \geq 0$ ,*

- (i)  $\|x_k - x^*\| \leq \frac{2}{a} \|d_{SD}(x_k)\|;$
- (ii)  $\|s_k\| \geq \frac{(1-\sigma)}{2b} \cos \beta^k \|d_{SD}(x_k)\|.$

**Proof** The proof is analogous to the one of Lemma 4.4 in [24], recalling that here a single approximation matrix  $B^k$  is considered. □

We are now ready to prove that the sequence of points generated by Algorithm 4 R-linearly converges to Pareto optimality.

**Proposition 5** *Let Assumptions 2 and 3 hold. Furthermore, let  $\{x_k\}$  be the sequence generated by Algorithm 4 and  $x^*$  be the Pareto optimal limit point of the sequence. Then,  $\{x_k\}$  R-linearly converges to  $x^*$ . In addition, we have that*

$$\sum_{k \geq 0} \|x_k - x^*\| < \infty. \tag{39}$$

**Proof** We first introduce the function  $f^* : \mathbb{R}^n \rightarrow \mathbb{R}$ , defined as

$$f^*(x) = \sum_{j=1}^m \lambda_j^{SD}(x^*) f_j(x), \tag{40}$$

where  $\lambda^{SD}(x^*)$  is the multipliers vector associated with the steepest common descent direction at  $x^*$ . Recalling Lemmas 1 and 3, that  $x^*$  is Pareto optimal and that (5) holds for  $d_{SD}(x^*)$ , we have that

$$\nabla f^*(x^*) = \sum_{j=1}^m \lambda_j^{SD}(x^*) \nabla f_j(x^*) = -d_{SD}(x^*) = \mathbf{0}. \tag{41}$$

Now, for all  $k \geq 0$  and  $j \in \{1, \dots, m\}$ , by Assumption 2 and using Taylor’s theorem, we get

$$\frac{a}{2} \|x_k - x^*\|^2 \leq f_j(x_k) - f_j(x^*) - \nabla f_j(x^*)^T (x_k - x^*) \leq \frac{b}{2} \|x_k - x^*\|^2.$$

Multiplying this result by  $\lambda_j^{SD}(x^*)$ , summing over  $j \in \{1, \dots, m\}$ , recalling (4), which is valid for  $\lambda^{SD}(x^*)$ , and (41), we obtain that

$$\frac{a}{2} \|x_k - x^*\|^2 \leq f^*(x_k) - f^*(x^*) \leq \frac{b}{2} \|x_k - x^*\|^2. \tag{42}$$

Given Lemma 9, from the right-hand side of the last result we get

$$f^*(x_k) - f^*(x^*) \leq \frac{2b}{a^2} \|d_{SD}(x_k)\|^2. \tag{43}$$

On the other side, (4), (15) and (40) imply that, for all  $k \geq 0$ ,

$$f^*(x_{k+1}) \leq f^*(x_k) + \gamma \alpha_k \mathcal{D}(x_k, d_{LM}(x_k))$$

which, by subtracting the term  $f^*(x^*)$  in both sides and taking into account Lemmas 7 and 9, changes into

$$\begin{aligned} f^*(x_{k+1}) - f^*(x^*) &\leq f^*(x_k) - f^*(x^*) - \frac{\gamma \cos \beta^k}{2} \|s_k\| \|d_{SD}(x_k)\| \\ &\leq f^*(x_k) - f^*(x^*) - \frac{\gamma (1 - \sigma) \cos^2 \beta^k}{4b} \|d_{SD}(x_k)\|^2. \end{aligned}$$

Joining this last result and (43), we obtain that

$$f^*(x_{k+1}) - f^*(x^*) \leq \left(1 - \frac{\gamma (1 - \sigma) a^2 \cos^2 \beta^k}{8b^2}\right) (f^*(x_k) - f^*(x^*)). \tag{44}$$

Now, for all  $k \geq 0$ , we define

$$r_k = 1 - \frac{\gamma (1 - \sigma) a^2 \cos^2 \beta^k}{8b^2}.$$

It is easy to see that, by the definitions of  $\gamma$  and  $\sigma$ , Assumption 2 and Lemma 8,  $r_k \in (0, 1)$ . In addition, by Lemma 8, we also have that there exists a constant  $\delta > 0$  such that, for all  $k \geq 0$ ,

$$r_k \leq 1 - \frac{\gamma (1 - \sigma) a^2 \delta^2}{8b^2} = \bar{r} < 1.$$

Then, recursively applying equation (44) and taking into account that, combining (4), (15) and (40),  $f^*(x_0) - f^*(x^*) > 0$ , we get

$$\begin{aligned} f^*(x_{k+1}) - f^*(x^*) &\leq \left[ \prod_{l=0}^k r_l \right] (f^*(x_0) - f^*(x^*)) \\ &\leq \left[ \prod_{l=0}^k \bar{r} \right] (f^*(x_0) - f^*(x^*)) \\ &= \bar{r}^{k+1} (f^*(x_0) - f^*(x^*)). \end{aligned}$$

Considering this last result and the left-hand side of (42), we obtain that

$$\|x_{k+1} - x^*\| \leq (\bar{r}^{k+1})^{1/2} \left[ \frac{2}{a} (f^*(x_0) - f^*(x^*)) \right]^{1/2},$$

and, thus, the sequence  $\{x_k\}$  R-linearly converges to  $x^*$ .

Summing the last result for all  $k \geq 0$  and recalling that  $\bar{r} < 1$ , we get that (39) holds.  $\square$

## 5 Computational experiments

In this section, we report the results of thorough computational experiments, comparing the performance of the proposed approach and other state-of-the-art methods from the literature. All the tests were run on a computer with the following characteristics: Ubuntu 20.04 OS, Intel Xeon Processor E5-2430 v2 6 cores 2.50 GHz, 16 GB RAM. For all algorithms, the code was implemented in Python3. <sup>1</sup> Finally, in order to solve the optimization problems to determine the descent direction, e.g., problem (14), the Gurobi Optimizer (Version 9) was employed.

### 5.1 Experiments settings

In the next subsections, we provide detailed information on the settings used for the experiments.

#### 5.1.1 Algorithms and parameters

We chose to compare the new limited memory Quasi-Newton approach, which we call LM-Q-NWT for the rest of the section, with some state-of-the-art Newton and Quasi-Newton methods for MOO from the literature.

The first competitor is the Multi-Objective Newton method (NWT) proposed in [8], which is an extension of the classical Newton method to multi-objective optimization. In this algorithm, the problem for finding the search direction is similar to problem (2):

<sup>1</sup> The implementation code can be found at [https://github.com/pierlumanzu/limited\\_memory\\_method\\_for\\_MOO](https://github.com/pierlumanzu/limited_memory_method_for_MOO) [37].

the difference is on the use of the real Hessian  $\nabla^2 f_j(\bar{x})$  instead of the approximation matrix  $B_j$ , for all  $j \in \{1, \dots, m\}$ . Since this method is not designed to handle unconstrained multi-objective non-convex problems, we evaluated its performance only on the convex test instances.

The other two competitors are the Quasi-Newton approach (Q-NWT) proposed in [24] and the Modified Quasi-Newton method (MQ-NWT) presented in [23]. We refer the reader back to Sect. 2.1 for further details. At the first iteration of all the Quasi-Newton approaches, including LM-Q-NWT, the approximation matrix/matrices is/are set equal to the identity matrix.

In order to make the comparisons as fair as possible, we decided to use the same line search strategy for all the approaches. In particular, we employed the proposed Wolfe line search (Algorithm 3). The values for the line search parameters were chosen according to some experiments on a subset of the tested problems and are as follows:  $\gamma = 10^{-4}$ ,  $\sigma = 10^{-1}$ ,  $\eta = 2.5$  and  $\delta = 0.5$ . We do not report these preliminary results for the sake of brevity. In order to efficiently use the proposed line search in MQ-NWT, we used equation (19) to compute  $\rho_k$  at each iteration  $k$ .

Finally, the choice for the parameter  $M$  of the new limited memory approach is separately discussed in Sect. 5.2. Since it denotes the number of vectors pairs maintained in memory during the iterations, it is the most critical among the LM-Q-NWT parameters.

### 5.1.2 Problems

In Table 1, we list the tested problems. In particular, we compared the algorithms in 78 convex and 83 non-convex problems. All the test instances are characterized by objective functions that are at least continuously differentiable almost everywhere. If a problem is characterized by singularities, these latter ones were counted as Pareto-stationary points. All the problems have objective functions that let Assumption 1 hold: the latter is essential to guarantee the finite termination of the proposed Wolfe line search.

Some problem names are characterized by the prefix  $M$ -. These problems are rescaled versions of the original ones and their formulations are provided in Appendix C. In this appendix, we also introduce a new convex test problem, which we call MAN\_2.

For each algorithm, we tested each problem with 100 different initial points chosen from a uniform distribution. The latter was defined through lower and upper bounds specified for each problem. Since in this work we consider unconstrained multi-objective optimization problems, these bounds were only used to choose the random initial points. For the M-FDS\_1, MMR\_5, M-MOP\_2, MOP and CEC problems, the lower and upper bounds values can be found in the referenced papers. For the others, the values are provided in Table 2.

Finally, starting from an initial point, we decided to let the algorithms run until one of the following stopping conditions was met:

- the current solution is  $\varepsilon$ -Pareto-stationary (Definition 4); in the experiments,

$$\varepsilon = 5\text{eps}^{1/2},$$

**Table 1** Problems used in the computational experiments

Type	Source	Problem	$m$	$n$
Convex	[31]	JOS_1a, JOS_1b JOS_1c	2	2, 5, 10, 20, 30, 40, 50, 100, 200, 500, 1000
	[21]	M-MAN_1		
	[32]	SLC_2		
	[8]	M-FDS_1	3	
	–	MAN_2		
	[33]	MOP_7	3	2
Non convex	[34]	MMR_5	2	2, 5, 10, 20, 30, 40, 50, 100, 200, 500, 1000
	[33]	M-MOP_2		
		MOP_3	2	2
	[35]	CEC09_1, CEC09_2 CEC09_3, CEC09_7 CEC09_8, CEC09_10	2	5, 10, 20, 30, 40, 50, 100, 200, 500, 1000

**Table 2** Bounds used to choose the initial points

Problem	Bounds
JOS_1a	$[-10, 10]^n$
JOS_1b	$[-10^2, 10^2]^n$
JOS_1c	$[10^{-2}, 1]^n$
SLC_2	$[-10^2, 10^2]^n$
M-MAN_1	$[-10, 10]^n$
MAN_2	$[-1, 1]^n$

where  $\text{eps}$  denotes the machine precision;

- a time limit of 2 min is reached.

### 5.1.3 Metrics

For each algorithm and problem, the main metrics to be computed are the following.

- $N_\varepsilon$ : the percentage of runs ended with an  $\varepsilon$ -Pareto-stationary point.
- $T$ : the computational time to reach the  $\varepsilon$ -Pareto-stationarity from an initial point. If the  $\varepsilon$ -Pareto-stationarity is not reached within the time limit, the value of  $T$  related to that point is set to  $\infty$ .
- $T_M$ : the mean of the finite  $T$  values.



In Sect. 5.4, we employed the metrics proposed in [17]: *purity*,  $\Gamma$ -*spread* and  $\Delta$ -*spread*. These metrics are used to evaluate the quality of Pareto front approximations. On the one hand, the *purity* metric indicates the ratio of the number of non-dominated points that a method obtained w.r.t. a reference front over the number of the points produced by that method. The reference front is obtained by combining the fronts retrieved by all the considered algorithms and by discarding the dominated points. On the other hand, the *spread* metrics measure the uniformity of the generated fronts in the objectives space. In particular, the  $\Gamma$ -*spread* is defined as the maximum  $\ell_\infty$  distance in the objectives space between adjacent points of the Pareto front, while the  $\Delta$ -*spread* is similar to the standard deviation of this distance.

Finally, we employed the performance profiles introduced in [36], which are an useful tool to appreciate the relative performance and robustness of the considered algorithms. The performance profile of a solver w.r.t. a certain metric is the (cumulative) distribution function of the ratio of the score obtained by the solver over the best score among those obtained by all the considered solvers. In other words, it is the probability that the score achieved by a method in a problem is within a factor  $\tau \in \mathbb{R}$  of the best value obtained by any of the algorithms in that problem. We refer the reader to [36] for additional information about this tool. Since  $N_\varepsilon$  and *purity* have increasing values for better solutions, the performance profiles w.r.t. these metrics were produced based on the inverse of the obtained values. All the performance profiles were plotted with specific axes ranges in order to remark the differences among the considered solvers.

## 5.2 Selection of the parameter $M$

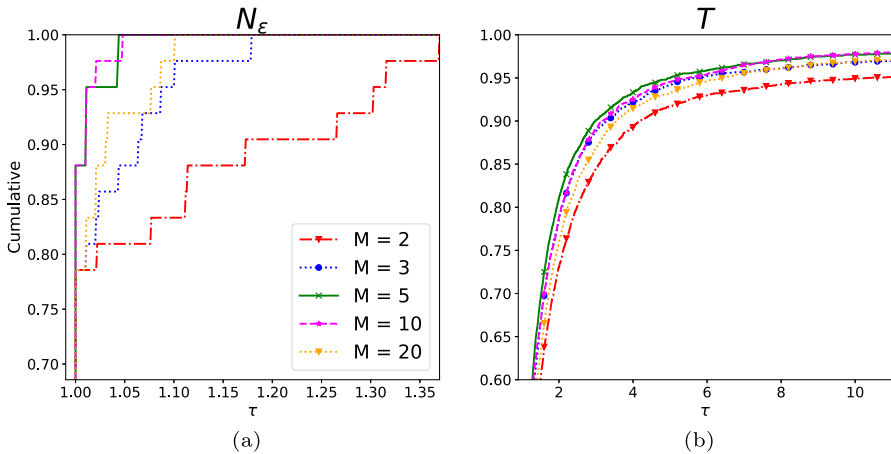
The parameter  $M$  indicates how many vectors pairs  $\{(s_i, u_i)\}$  are maintained in memory at each iteration of LM-Q-NWT. A bad value for this parameter might compromise the overall performance of the approach, making it too slow or not capable of reaching  $\varepsilon$ -Pareto-stationary points within the time limit.

In order to select a proper value for  $M$ , we analyzed the performance of LM-Q-NWT with  $M \in \{2, 3, 5, 10, 20\}$  on a subset of the tested problems.

- 2 convex problems: SLC\_2 ( $m = 2$ ), MAN\_2 ( $m = 3$ ).
- 2 non-convex problems: CEC09\_1 ( $m = 2$ ), CEC09\_10 ( $m = 3$ ).

In Fig. 1, we report the performance profiles for the five variants of the new limited memory method. The solvers with  $M \in \{5, 10\}$  turned out to be the best w.r.t. both  $N_\varepsilon$  and  $T$ , while the variant with  $M = 2$  was outperformed by all the other methods. We conclude that too little information on the previous steps can compromise the performance of LM-Q-NWT. On the other hand, the management of too many vectors pairs and the use of the two-loop recursive procedure can require great computational costs. A demonstration of this fact is the performance of the proposed approach with  $M = 20$  on the  $T$  metric. Although this solver performed well w.r.t.  $N_\varepsilon$ , it is only the fourth most robust algorithm in terms of computational time.

After analyzing the performance profiles, we decided to use the new limited memory approach with  $M = 5$  for the rest of the section. However, the variant with  $M = 10$  appears to be a good choice too.



**Fig. 1** Performance profiles for the LM-Q-NWT algorithm with  $M \in \{2, 3, 5, 10, 20\}$  on the SLC\_2, MAN\_2, CEC09\_1 and CEC09\_10 problems (for interpretation of the references to color in text, the reader is referred to the electronic version of the article). Performance metric: **a**  $N_\varepsilon$ ; **b**  $T$

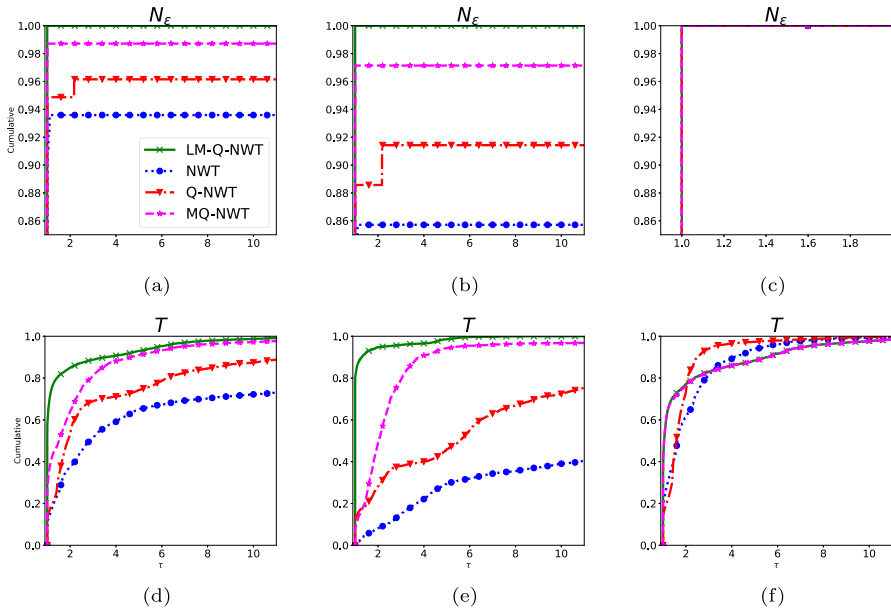
### 5.3 Overall comparisons

In this section, we compare the proposed approach with the Newton and Quasi-Newton algorithms described in Sect. 5.1.1. As already mentioned, we tested NWT only on the convex problems. Then, we separately report the performance profiles for the convex and non-convex problems in Figs. 2 and 3 respectively. In order to better remark the differences among the methods, for each metric we show three plots concerning different sets of values for  $n$ .

- Figs. 2a, 2d, 3a, 3d: all the  $n$  values.
- Figs. 2b, 2e, 3b, 3e:  $n \geq 50$ .
- Figs. 2c, 2f, 3c, 3f:  $n < 50$ .

Regarding the performance on the convex problems for all the  $n$  values, the proposed approach proved to be the best algorithm, outperforming the competitors w.r.t. both the metrics. Moreover, the gap between LM-Q-NWT and the others is sharper when taking into account the non-convex problems or the high dimensional ones. For high  $n$  values, the NWT and Q-NWT algorithms proved to suffer the maintenance of the Hessians and the approximation matrices respectively. As a consequence, they turned out to be the least robust w.r.t. both the metrics. Using a single approximation matrix allowed the MQ-NWT approach to perform better. However, in extremely high dimensional problems, even managing a single matrix proved to be an expensive job. In these cases, the performance of the limited memory approach was remarkable.

On the low dimensional problems, the NWT and Q-NWT algorithms had a good performance. The proposed approach similarly behaved w.r.t. the  $N_\varepsilon$  metric, but it was generally outperformed by these algorithms in terms of  $T$ . Managing the real Hessians or the approximation matrices turned out to be a tractable task when  $n$  is small enough. Moreover, by definition, these matrices provide more accurate information about the curvature of the objective functions than the matrix of LM-Q-NWT and



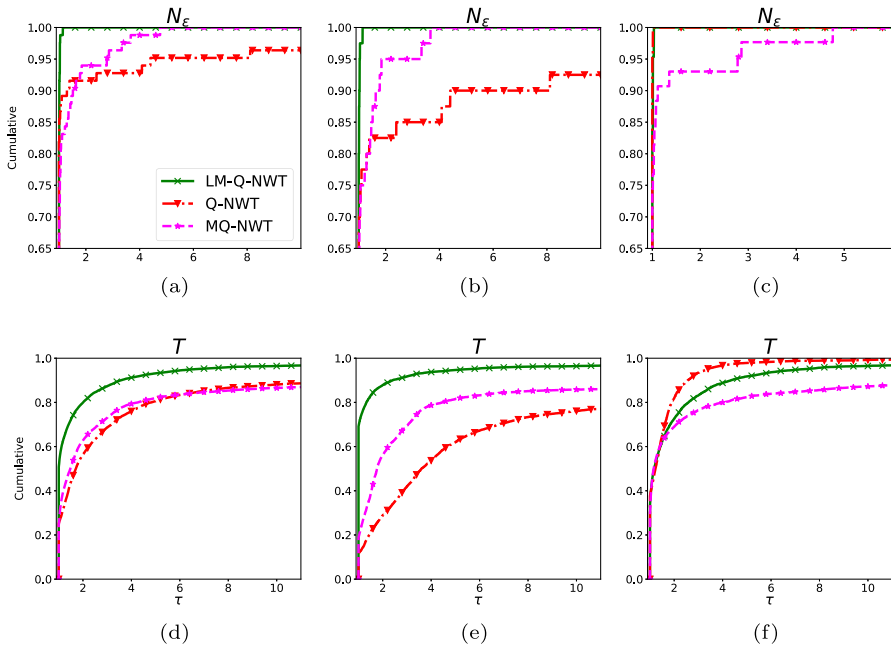
**Fig. 2** Performance profiles for the LM-Q-NWT, NWT, Q-NWT and MQ-NWT algorithms on the convex problems of Table 1 (for interpretation of the references to color in text, the reader is referred to the electronic version of the article). Performance metric: **a–c**  $N_\varepsilon$ ; **d–f**  $T$ . Values for  $n$ : **a, d** All; **e**  $n \geq 50$ ; **c, f**  $n < 50$

MQ-NWT. However, these two algorithms still proved to be competitive, obtaining good  $T$  metric results in most of the problems.

In order to analyze the performance of the algorithms more deeply, in Tables 3, 4, 5 and 6 we report the metrics values obtained in two convex and two non-convex problems respectively. In particular, we show the results for  $n \in \{5, 20, 50, 200, 500, 1000\}$ .

Regarding the  $N_\varepsilon$  metric, the proposed method outperformed the competitors regardless the values for  $n$  and  $m$ . As in the performance profiles, the differences between LM-Q-NWT and the other approaches are clearer on the high dimensional problems. In some of these, NWT, Q-NWT and MQ-NWT were not able to obtain any  $\varepsilon$ -Pareto-stationary point.

On the problems with two objective functions, almost all the best results in terms of the  $T_M$  metric were obtained by the proposed approach. However, the same performance was not obtained on the problems with  $m = 3$  and low value for  $n$ . The use of a single matrix seems not to provide accurate enough information about the functions curvature when the objectives are more than two. An additional demonstration of this fact could be also the similar performance of the MQ-NWT algorithm. On the other hand, the use of the real Hessian/an approximation matrix for each objective function seems to overcome the issue: indeed, NWT and Q-NWT had the best performance in terms of  $T_M$  in these cases. LM-Q-NWT still obtained great results for this metric on the problems with three objective functions and high value for  $n$ , outperforming the other competitors. Even with  $m = 3$ , the employment of a single matrix turned out



**Fig. 3** Performance profiles for the LM-Q-NWT, Q-NWT and MQ-NWT algorithms on the non-convex problems of Table 1 (for interpretation of the references to color in text, the reader is referred to the electronic version of the article). Performance metric: **a–c**  $N_\epsilon$ ; **d–f**  $T$ . Values for  $n$ : **a, d** All; **b, e**  $n \geq 50$ ; **c, f**  $n < 50$

**Table 3** Metrics values achieved by the LM-Q-NWT, NWT, Q-NWT and MQ-NWT algorithms on the convex M-MAN\_1 problem ( $m = 2$ ) for  $n \in \{5, 20, 50, 200, 500, 1000\}$

$n$	$N_\epsilon$				$T_M$			
	LM-Q-NWT	NWT	Q-NWT	MQ-NWT	LM-Q-NWT	NWT	Q-NWT	MQ-NWT
5	<b>1.0</b>	<b>1.0</b>	<b>1.0</b>	<b>1.0</b>	<b>0.029</b>	0.126	0.054	0.031
20	<b>1.0</b>	<b>1.0</b>	<b>1.0</b>	<b>1.0</b>	0.285	0.288	0.605	<b>0.273</b>
50	<b>1.0</b>	<b>1.0</b>	<b>1.0</b>	<b>1.0</b>	<b>0.538</b>	0.978	1.332	0.54
200	<b>1.0</b>	<b>1.0</b>	<b>1.0</b>	<b>1.0</b>	<b>1.959</b>	25.099	22.118	5.882
500	<b>1.0</b>	0.03	0.0	<b>1.0</b>	<b>18.367</b>	127.817	–	42.801
1000	<b>1.0</b>	0.0	0.0	0.0	<b>39.593</b>	–	–	–

A value marked in bold is the best obtained for a metric on a specific problem

to be essential in high dimensional problems. Like the proposed approach, MQ-NWT proved to perform better than NWT and Q-NWT with  $m = 3$  and high value for  $n$ , resulting the second best algorithm in these cases.

**Table 4** Metrics values achieved by the LM-Q-NWT, NWT, Q-NWT and MQ-NWT algorithms on the convex M-FDS\_1 problem ( $m = 3$ ) for  $n \in \{5, 20, 50, 200, 500, 1000\}$

$n$	$N_\epsilon$				$T_M$			
	LM-Q-NWT	NWT	Q-NWT	MQ-NWT	LM-Q-NWT	NWT	Q-NWT	MQ-NWT
5	<b>1.0</b>	<b>1.0</b>	<b>1.0</b>	<b>1.0</b>	0.771	<b>0.132</b>	0.176	0.758
20	<b>1.0</b>	<b>1.0</b>	<b>1.0</b>	<b>1.0</b>	2.413	<b>0.32</b>	0.403	2.44
50	<b>1.0</b>	<b>1.0</b>	<b>1.0</b>	<b>1.0</b>	3.376	0.891	<b>0.688</b>	3.287
200	<b>1.0</b>	<b>1.0</b>	<b>1.0</b>	<b>1.0</b>	5.333	13.416	<b>5.292</b>	13.113
500	<b>1.0</b>	0.95	<b>1.0</b>	<b>1.0</b>	<b>23.353</b>	109.853	36.66	39.028
1000	<b>1.0</b>	0.0	0.0	0.99	<b>32.868</b>	–	–	109.541

A valuemarked in bold is the best obtained for a metric on a specific problem

**Table 5** Metrics values achieved by the LM-Q-NWT, Q-NWT and MQ-NWT algorithms on the non-convex M-MOP\_2 problem ( $m = 2$ ) for  $n \in \{5, 20, 50, 200, 500, 1000\}$

$n$	$N_\epsilon$			$T_M$		
	LM-Q-NWT	Q-NWT	MQ-NWT	LM-Q-NWT	Q-NWT	MQ-NWT
5	<b>1.0</b>	<b>1.0</b>	<b>1.0</b>	<b>0.057</b>	0.093	<b>0.057</b>
20	<b>1.0</b>	<b>1.0</b>	<b>1.0</b>	0.116	0.166	<b>0.113</b>
50	<b>1.0</b>	<b>1.0</b>	<b>1.0</b>	<b>0.142</b>	0.225	0.145
200	<b>1.0</b>	<b>1.0</b>	<b>1.0</b>	<b>0.324</b>	1.158	0.495
500	<b>1.0</b>	<b>1.0</b>	<b>1.0</b>	<b>1.013</b>	5.766	1.673
1000	<b>1.0</b>	<b>1.0</b>	<b>1.0</b>	<b>1.787</b>	32.464	5.244

A value marked in bold is the best obtained for a metric on a specific problem

**Table 6** Metrics values achieved by the LM-Q-NWT, Q-NWT and MQ-NWT algorithms on the non-convex CEC09\_8 problem ( $m = 3$ ) for  $n \in \{5, 20, 50, 200, 500, 1000\}$

$n$	$N_\epsilon$			$T_M$		
	LM-Q-NWT	Q-NWT	MQ-NWT	LM-Q-NWT	Q-NWT	MQ-NWT
5	<b>1.0</b>	<b>1.0</b>	<b>1.0</b>	1.036	<b>0.62</b>	1.472
20	<b>1.0</b>	<b>1.0</b>	0.99	3.4	<b>1.357</b>	3.72
50	<b>1.0</b>	<b>1.0</b>	<b>1.0</b>	5.531	<b>2.013</b>	6.213
200	<b>1.0</b>	<b>1.0</b>	<b>1.0</b>	<b>6.565</b>	12.028	11.798
500	<b>0.99</b>	0.9	0.98	<b>26.099</b>	80.713	36.693
1000	<b>0.94</b>	0.0	0.73	<b>32.367</b>	–	77.311

A valuemarked in bold is the best obtained for a metric on a specific problem

### 5.4 Results in a global optimization setting

In the previous section, we compared the LM-Q-NWT method with strongly related approaches from the state-of-the-art, in terms of efficiency and effectiveness at reaching approximate Pareto-stationarity. Now, we show the (positive) impact that the

proposed procedure may have if used within a global multi-objective optimization framework.

In particular, here we consider the memetic algorithm proposed in [21], which is named NSMA. Starting with an initial population of  $N$  points, this method aims at approximating the Pareto front of the considered problem, combining the genetic operators of the NSGA-II algorithm [14] with a front-based projected gradient method FMOPG. The latter is employed, every  $n_{opt}$  iterations of NSMA, to refine selected solutions up to  $\varepsilon$ -Pareto-stationarity. Moreover, the selection of the points to be optimized is based on the ranking and the crowding distance values assigned to the population at each iteration. Finally, the algorithm exploits a front-based variant of the Armijo-Type Line Search for MOO defined in [7]. For additional information about NSMA, we refer the reader to [21].

For the experiments of the present paper, we consider possible modifications of the NSMA algorithm:

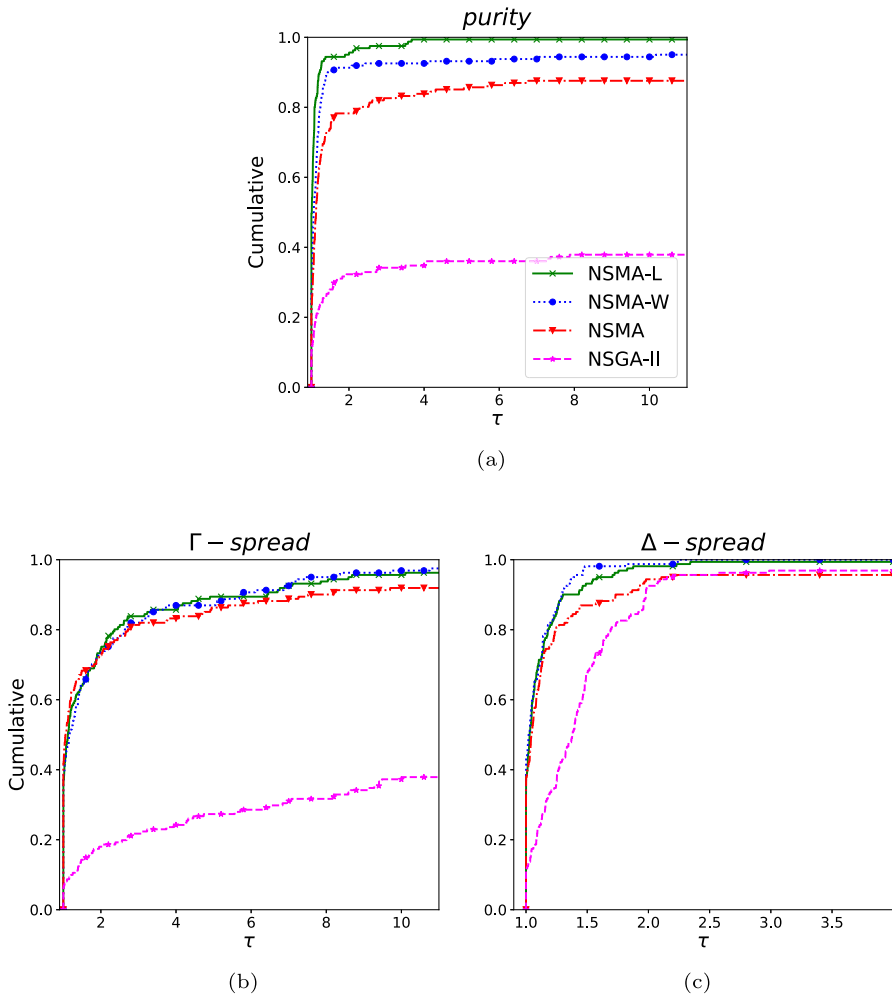
- NSMA-W, which employs the proposed Wolfe line search (Algorithm 3) in the FMOPG method;
- NSMA-L, which uses the new limited memory approach (Algorithm 1) as the local optimization procedure.

We compared these two approaches with NSGA-II and the original version of NSMA.

In the experiments, as in [21], we set  $N = 100$  and  $n_{opt} = 5$ . Moreover, the points selected as starting solutions for the local search procedures were only optimized w.r.t. all the objective functions. In the original version of the NSMA method, the points can be also refined w.r.t. a subset of the objective functions  $I \subset \{1, \dots, m\}$ . However, Assumption 1 may not hold for some subset  $I$  and, then, when trying to optimize a point w.r.t.  $I$ , the Wolfe line search would continue its execution for an infinite number of steps.

In Fig. 4, we report the performance profiles for the NSMA-L, NSMA-W, NSMA and NSGA-II algorithms on the problems listed in Table 1. The considered performance metrics are *purity*,  $\Gamma$ -*spread* and  $\Delta$ -*spread*. In order to make the comparisons as independent from random operations as possible, for each algorithm and problem five tests characterized by different seeds for the pseudo-random number generator were executed. The five resulting Pareto front approximations were then compared based on the *purity* metric: the best among them was chosen as the output of the algorithm for the problem at hand.

In terms of *purity*, NSMA-L and NSMA-W turned out to be the two most robust algorithms. The proposed Wolfe line search allowed to improve the results of the original NSMA. In fact, the use of the limited memory approach allowed to obtain the best possible performance. Regarding the *spread* metrics, NSMA-L and NSMA-W had a similar performance. NSMA results on  $\Gamma$ -*spread* are comparable with the ones of the two variants. However, it was slightly outperformed in terms of  $\Delta$ -*spread*. NSGA-II did not perform well w.r.t. all the metrics: the variants of NSMA turned out to be capable in finding more accurate and uniform Pareto front approximations.



**Fig. 4** Performance profiles for the NSMA-L, NSMA-W, NSMA and NSGA-II algorithms on the problems of Table 1 (for interpretation of the references to color in text, the reader is referred to the electronic version of the article). Performance metric: **a** purity; **b**  $\Gamma$ -spread; **c**  $\Delta$ -spread

## 6 Conclusions

In this paper we proposed a new limited memory Quasi-Newton algorithm for unconstrained multi-objective optimization. To the best of our knowledge, it is the first attempt to define such an approach for MOO. As in [23], we use a single approximation matrix, contrarily to what is done in the other Quasi-Newton approaches. The idea of a single matrix, whose update formula is slightly modified from the one used in the scalar case, allowed us to extend the L-BFGS two-loop recursive procedure to multi-objective optimization: the Hessian matrix approximation does not need to be maintained and managed in memory, but it is computed using a finite number  $M$  of

previously generated solutions. This feature proves to be crucial, especially when the approximation matrix is dense and/or high dimensional problems are handled. For the proposed approach, under assumptions similar to the ones made for L-BFGS in the strongly convex scalar case, we stated properties of R-linear convergence to the Pareto optimality of the produced sequence of points.

The results of thorough computational experiments show that the new limited memory algorithm consistently outperforms the state-of-the-art Newton and Quasi-Newton methods for MOO. Moreover, we show the substantial benefits of using the proposed algorithm as local search procedure within a global optimization framework.

**Acknowledgements** We would like to thank the two anonymous reviewers for their precious comments, that allowed us to significantly improve the quality of this manuscript. We would also like to thank Prof. Fabio Schoen for the useful discussions at the early stages of the work presented in this paper.

**Funding** No funding was received for conducting this study.

**Data Availability** Data sharing not applicable to this article as no datasets were generated or analyzed during the current study.

## Declarations

**Conflict of interest** The authors have no competing interests to declare that are relevant to the content of this article.

**Code availability statement** The full code of the experiments can be found at [https://github.com/pierlumanzu/limited\\_memory\\_method\\_for\\_MOO](https://github.com/pierlumanzu/limited_memory_method_for_MOO) (DOI: 10.5281/zenodo.7533784).

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## A Basic Results

In this appendix, we propose a further analysis on problem (2) and proofs of technical results used in the paper.

### A.1 Quasi-Newton direction problem

Problem (2) can be re-written as a quadratically-constrained convex one:

$$\begin{aligned} \min_{\substack{t \in \mathbb{R} \\ d \in \mathbb{R}^n}} t \\ \text{s.t. } \nabla f_j(\bar{x})^T d + \frac{1}{2} d^T B_j d - t \leq 0, \quad \forall j \in \{1, \dots, m\}. \end{aligned}$$



In this case, the Lagrangian function is of the following form:

$$L(t, d, \lambda) = t + \sum_{j=1}^m \lambda_j \left[ \nabla f_j(\bar{x})^T d + \frac{1}{2} d^T B_j d - t \right],$$

where  $\lambda_1, \dots, \lambda_m$  are the Lagrange multipliers. Denoting by  $\lambda \in \mathbb{R}^m$  the vector of all the Lagrange multipliers, we also introduce the dual problem:

$$\begin{aligned} \max_{\lambda \in \mathbb{R}^m} \quad & \inf_{\substack{t \in \mathbb{R} \\ d \in \mathbb{R}^t}} L(t, d, \lambda) \\ \text{s.t.} \quad & \lambda \geq \mathbf{0}. \end{aligned}$$

As already mentioned in Sect. 2.1, if  $B_j \succ 0 \forall j \in \{1, \dots, m\}$ , then problem (2) has a unique solution. Moreover, the problem is convex and has a Slater point, i.e.,  $(t, d) = (1, \mathbf{0})$ . Then, strong duality holds and the Karush-Kuhn-Tucker conditions are sufficient and necessary for optimality. We thus obtain:

$$\frac{\partial L(t, d, \lambda)}{\partial t} = 1 - \sum_{j=1}^m \lambda_j = 0, \quad \frac{\partial L(t, d, \lambda)}{\partial d} = \sum_{j=1}^m \lambda_j [\nabla f_j(\bar{x}) + B_j d] = \mathbf{0},$$

which leads to

$$\sum_{j=1}^m \lambda_j = 1, \quad d = - \left[ \sum_{j=1}^m \lambda_j B_j \right]^{-1} J_F(\bar{x})^T \lambda. \tag{A1}$$

Considering equation (A1), given  $\lambda \geq 0$ , we can state that in MOO the Quasi-Newton direction depends on convex combinations of both the approximation matrices and the gradients. In scalar optimization, the second formula of (A1) reduces to the classical Quasi-Newton direction  $d(\bar{x}) = -B^{-1} \nabla f(\bar{x})$ , where  $f(\cdot)$  indicates the objective function.

Equation (A1) also leads to re-write the dual function; in particular,  $t$  disappears since  $\sum_{j=1}^m \lambda_j = 1$ , while  $d$  is substituted:

$$\inf_{\substack{t \in \mathbb{R} \\ d \in \mathbb{R}^n}} L(t, d, \lambda) = -\frac{1}{2} \lambda^T J_F(\bar{x}) \left[ \sum_{j=1}^m \lambda_j B_j \right]^{-1} J_F(\bar{x})^T \lambda$$

Finally, using this last equation, we can retrieve the final form of the dual problem:

$$\begin{aligned} \max_{\lambda \in \mathbb{R}^m} \quad & -\frac{1}{2} \lambda^T J_F(\bar{x}) \left[ \sum_{j=1}^m \lambda_j B_j \right]^{-1} J_F(\bar{x})^T \lambda \\ \text{s.t.} \quad & \sum_{j=1}^m \lambda_j = 1, \quad \lambda \geq \mathbf{0}. \end{aligned}$$

## A.2 Proofs of technical results

### Proposition 1

**Proof** First, the case  $s_k^T u_k > 0$  is trivial.

Then, let  $s_k^T u_k \leq 0$  and let us consider

$$\mathcal{D}(x_{k+1}, s_k) - \nabla f_j(x_k)^T s_k,$$

with  $j \in \{1, \dots, m\}$ . Considering that  $\mathcal{D}(x_k, s_k) \geq \nabla f_j(x_k)^T s_k$  by definition of  $\mathcal{D}(\cdot, \cdot)$  and equation (8), we have that

$$\begin{aligned} & \mathcal{D}(x_{k+1}, s_k) - \nabla f_j(x_k)^T s_k \\ & \geq \mathcal{D}(x_{k+1}, s_k) - \mathcal{D}(x_k, s_k) \\ & = \mathcal{D}(x_{k+1}, \alpha_k d_{LM}(x_k)) - \mathcal{D}(x_k, \alpha_k d_{LM}(x_k)). \end{aligned} \tag{A2}$$

In Algorithm 1, the Wolfe conditions are imposed at each iteration  $k$  (Line 6): in particular,  $\mathcal{D}(x_{k+1}, d_{LM}(x_k)) \geq \sigma \mathcal{D}(x_k, d_{LM}(x_k))$ . Then, also considering Lemma 2, we obtain that

$$\begin{aligned} & \mathcal{D}(x_{k+1}, \alpha_k d_{LM}(x_k)) - \mathcal{D}(x_k, \alpha_k d_{LM}(x_k)) \\ & = \alpha_k [\mathcal{D}(x_{k+1}, d_{LM}(x_k)) - \mathcal{D}(x_k, d_{LM}(x_k))] \\ & \geq \alpha_k (\sigma - 1) \mathcal{D}(x_k, d_{LM}(x_k)) > 0, \end{aligned} \tag{A3}$$

where the last inequality comes from the fact that  $\alpha_k > 0$ ,  $\sigma - 1 < 0$  and  $\mathcal{D}(x_k, d_{LM}(x_k)) < 0$ . Using (A2) and (A3), we conclude that

$$\mathcal{D}(x_{k+1}, s_k) - \nabla f_j(x_k)^T s_k > 0. \tag{A4}$$

Since we have considered a generic  $j \in \{1, \dots, m\}$ , it trivially follows that this last equation is verified for all  $j$ .

Now, given (4), which is valid for the Lagrange multipliers vector  $\lambda^{LM}(x_k)$ , and (A4), we can state that

$$\sum_{j=1}^m \lambda_j^{LM}(x_k) [\mathcal{D}(x_{k+1}, s_k) - \nabla f_j(x_k)^T s_k] > 0.$$

Then, if the second formula in (19) is used,  $\rho^k > 0$ . □

## B Proofs of Wolfe line search analysis

### Proposition 2

**Proof** Given  $F$  continuously differentiable and  $d_k$  descent direction for  $F$  at  $x_k$  and recalling Lemma 4, we can state that there exists  $\varepsilon > 0$  such that, for all  $\alpha \in (0, \varepsilon]$ , we have

$$F(x_k + \alpha d_k) < F(x_k) + \gamma \alpha J_F(x_k) d_k \leq F(x_k) + \mathbf{1} \gamma \alpha \mathcal{D}(x_k, d_k), \tag{B1}$$

where the last inequality follows from Remark 1. We now assume, by contradiction, that for all  $\alpha > 0$  we have

$$F(x_k + \alpha d_k) < F(x_k) + \mathbf{1} \gamma \alpha \mathcal{D}(x_k, d_k).$$

This last result indicates that we have  $\{x_k + \alpha d_k \mid \alpha > 0\} \subseteq \mathcal{L}_F(F(x_k))$ , which is absurd since Assumption 1 holds. Therefore, there exists  $\hat{\alpha} > \varepsilon$  and  $\hat{j} \in \{1, \dots, m\}$  such that

$$f_{\hat{j}}(x_k + \hat{\alpha} d_k) = f_{\hat{j}}(x_k) + \gamma \hat{\alpha} \mathcal{D}(x_k, d_k) \tag{B2}$$

and, by the continuity of  $F$ , equation (B1) holds for all  $\alpha < \hat{\alpha}$ .

Using the Mean-value Theorem, we have that

$$f_{\hat{j}}(x_k + \hat{\alpha} d_k) = f_{\hat{j}}(x_k) + \hat{\alpha} \nabla f_{\hat{j}}(x_k + t \hat{\alpha} d_k)^T d_k, \tag{B3}$$

with  $t \in (0, 1)$ . Combining (B2) and (B3) we get

$$\nabla f_{\hat{j}}(x_k + t \hat{\alpha} d_k)^T d_k = \gamma \mathcal{D}(x_k, d_k) > \sigma \mathcal{D}(x_k, d_k), \tag{B4}$$

where the last inequality follows from the fact that  $\sigma > \gamma$  and  $\mathcal{D}(x_k, d_k) < 0$ .

By definition of  $\mathcal{D}(\cdot, \cdot)$ , we have the following condition:

$$\mathcal{D}(x_k + t \hat{\alpha} d_k, d_k) \geq \nabla f_{\hat{j}}(x_k + t \hat{\alpha} d_k)^T d_k.$$

Using the latter and (B4), we obtain that

$$\mathcal{D}(x_k + t \hat{\alpha} d_k, d_k) > \sigma \mathcal{D}(x_k, d_k).$$

Then, by the continuity of  $\mathcal{D}(\cdot, \cdot)$ , there exists an interval  $[\alpha_l, \alpha_u] \subseteq (0, \hat{\alpha})$  such that, for all  $\alpha \in [\alpha_l, \alpha_u]$ , we have that

$$\mathcal{D}(x_k + \alpha d_k, d_k) > \sigma \mathcal{D}(x_k, d_k).$$

Moreover, since for all  $\alpha \in [0, \hat{\alpha}]$  equation (20) holds and  $[\alpha_l, \alpha_u] \subseteq (0, \hat{\alpha})$ , the proof is complete.  $\square$

**Lemma 5**

**Proof** (1) Since  $\alpha_u^0 = \infty$  and  $\alpha_u^t < \infty$ , the interval upper bound has been updated at least once though Line 4. Let  $\bar{t}$  be the iteration in which this update takes place: it follows that  $0 \leq \bar{t} < t$ . In this case, we have that

$$\exists j \left( \alpha^{\bar{t}} \right) \in \{1, \dots, m\} \text{ s.t. } f_{j(\alpha^{\bar{t}})}(x_k + \alpha^{\bar{t}}d_k) > f_{j(\alpha^{\bar{t}})}(x_k) + \gamma \alpha^{\bar{t}}\mathcal{D}(x_k, d_k)$$

and

$$\alpha_u^{\bar{t}+1} = \alpha^{\bar{t}}.$$

Then, it trivially follows that equation (22) holds for  $\alpha_u^{\bar{t}+1}$ .

Now, suppose that  $\bar{t} + 1 < t$  and consider the iteration  $\bar{t} + 1$ . By the instructions of the algorithm,  $\alpha_u^{\bar{t}+2}$  is updated either by  $\alpha_u^{\bar{t}+2} = \alpha^{\bar{t}+1}$  (Line 4) or  $\alpha_u^{\bar{t}+2} = \alpha_u^{\bar{t}+1}$  (Line 7). The first case is identical to the one of iteration  $\bar{t}$ . In the second case, since (22) is satisfied for  $\alpha_u^{\bar{t}+1}$ , it is also for  $\alpha_u^{\bar{t}+2}$ . For  $t > \bar{t} + 2$ , the property follows by induction.

(2) Given  $\sigma \in (0, 1)$  and  $\mathcal{D}(x_k, d_k) < 0$ , it is easy to prove that (23) and (24) hold for  $\alpha_l^0 = 0$ .

Then, consider the first iteration of Algorithm 3. Similarly to the upper bound,  $\alpha_l^1$  is set equal to either  $\alpha_l^0$  (Line 5) or  $\alpha^0$  (Line 9). The latter case occurs if

$$F(x_k + \alpha^0d_k) \leq F(x_k) + \mathbf{1}\gamma\alpha^0\mathcal{D}(x_k, d_k)$$

and

$$\mathcal{D}(x_k + \alpha^0d_k, d_k) < \sigma\mathcal{D}(x_k, d_k).$$

Thus, it simply follows that conditions (23) and (24) are satisfied in both cases.

For  $t > 1$ , we get the thesis by induction.  $\square$

**Proposition 3**

**Proof** By contradiction, we assume that the thesis is false, i.e., the algorithm does not stop in a finite number of iterations.

First, we consider the case in which  $\alpha_u^t = \infty$  for all  $t$ , i.e., the interval upper bound is never updated. In this case, by the instructions of Algorithm 3, the Wolfe sufficient decrease condition is always satisfied, i.e., for each step size  $\alpha^t$  we have that

$$F(x_k + \alpha^t d_k) \leq F(x_k) + \mathbf{1}\gamma\alpha^t \mathcal{D}(x_k, d_k). \tag{B5}$$

By Assumption 2.a, for all  $t$ , if  $\alpha_u^t = \infty$ , then  $\alpha^t$  is updated according to (25). Using the latter, we obtain that

$$\alpha^t \geq \eta \max \left\{ \alpha_l^t, \alpha^0 \right\} \geq \eta \alpha_l^t. \tag{B6}$$

Moreover, Line 9 is executed at every iteration, i.e.,  $\alpha_l^t = \alpha^{t-1}$ . Then, (B6) turns into

$$\alpha^t \geq \eta \alpha^{t-1} \geq \eta^2 \max \left\{ \alpha_l^{t-1}, \alpha^0 \right\} \geq \dots \geq (\eta)^t \alpha^0.$$

Since  $\eta > 1$  and  $\alpha^0 = 1$ , it follows that  $\lim_{t \rightarrow \infty} \alpha^t = \infty$ .

Then, there exists an infinite sequence of points  $\{x_k + \alpha d_k\}_{\alpha \geq \alpha_0}$  satisfying (B5). This fact is in contradiction with Assumption 1.

Now, we consider the case in which  $\exists \tilde{t}$  such that  $\alpha_u^{\tilde{t}} \leq M$ . Then, the bounded and monotone sequences  $\{\alpha_l^t\}_{t \geq \tilde{t}}$ , with  $\alpha_l^t \geq 0$ , and  $\{\alpha_u^t\}_{t \geq \tilde{t}}$ , with  $\alpha_u^t \leq M$ , are generated. From Lemma 5, it follows that, for  $t \geq \tilde{t}$ ,

$$\exists j(\alpha_u^t) \text{ s.t. } f_j(\alpha_u^t)(x_k + \alpha_u^t d_k) > f_j(\alpha_u^t)(x_k) + \gamma \alpha_u^t \mathcal{D}(x_k, d_k), \tag{B7}$$

$$\forall j \in \{1, \dots, m\}, f_j(x_k + \alpha_l^t d_k) \leq f_j(x_k) + \gamma \alpha_l^t \mathcal{D}(x_k, d_k). \tag{B8}$$

Let  $\mathcal{T} \subseteq \{\tilde{t}, \tilde{t} + 1, \dots\}$  be a subsequence such that, for all  $t \in \mathcal{T}$ ,  $j(\alpha_u^t) = \hat{j}$ .

By the instructions of the algorithm, the upper and lower bounds are updated in one of the following ways:

- $\alpha_u^{t+1} = \alpha^t, \alpha_l^{t+1} = \alpha_l^t;$
- $\alpha_u^{t+1} = \alpha_u^t, \alpha_l^{t+1} = \alpha^t.$

In the first case, we can state that

$$\begin{aligned} \max \left\{ (\alpha^t - \alpha_l^t), (\alpha_u^t - \alpha^t) \right\} &= \max \left\{ (\alpha_u^{t+1} - \alpha_l^{t+1}), (\alpha_u^t - \alpha_u^{t+1}) \right\} \\ &\geq \alpha_u^{t+1} - \alpha_l^{t+1}. \end{aligned} \tag{B9}$$

An analogous result can be also achieved for the second case. Using (B9) and Assumption 2.b, we obtain that

$$\alpha_u^{t+1} - \alpha_l^{t+1} \leq \max \left\{ (\alpha^t - \alpha_l^t), (\alpha_u^t - \alpha^t) \right\} \leq \delta (\alpha_u^t - \alpha_l^t).$$

Recalling that  $\delta \in [1/2, 1)$ ,  $\{\alpha_l^t\}$  and  $\{\alpha_u^t\}$  are monotone and bounded sequences and that, for all  $t$ , we have  $\alpha_l^t \leq \alpha^t \leq \alpha_u^t$ , the above equation implies that  $\alpha_u^t - \alpha_l^t$  goes to zero as  $t \rightarrow \infty$ , with  $t \in \mathcal{T}$ . Moreover, it follows that

$$\lim_{\substack{t \rightarrow \infty \\ t \in \mathcal{T}}} \alpha_u^t = \lim_{\substack{t \rightarrow \infty \\ t \in \mathcal{T}}} \alpha_l^t = \lim_{t \in \mathcal{T}} \alpha^t = \bar{\alpha}. \tag{B10}$$

Given (B7) and (B8), the definition of  $\hat{j}$  and the continuity of  $F$ , by taking the limit for  $t \rightarrow \infty$ , with  $t \in \mathcal{T}$ , we obtain that

$$f_{\hat{j}}(x_k + \bar{\alpha}d_k) = f_{\hat{j}}(x_k) + \gamma\bar{\alpha}\mathcal{D}(x_k, d_k). \quad (\text{B11})$$

Taking into account this result and (B7), we have that, for all  $t \in \mathcal{T}$ ,  $\alpha_u^t > \bar{\alpha}$ .

On the other hand, for  $t \in \mathcal{T}$ , equation (B7) can be re-written in the following way:

$$f_{\hat{j}}(x_k + \alpha_u^t d_k) > f_{\hat{j}}(x_k) + \gamma(\bar{\alpha} + \alpha_u^t - \bar{\alpha})\mathcal{D}(x_k, d_k).$$

Using (B11) and by simple algebraic manipulations we get

$$f_{\hat{j}}(x_k + \alpha_u^t d_k) > f_{\hat{j}}(x_k + \bar{\alpha}d_k) + \gamma(\alpha_u^t - \bar{\alpha})\mathcal{D}(x_k, d_k)$$

and, then,

$$\frac{f_{\hat{j}}(x_k + \alpha_u^t d_k) - f_{\hat{j}}(x_k + \bar{\alpha}d_k)}{\alpha_u^t - \bar{\alpha}} > \gamma\mathcal{D}(x_k, d_k).$$

Now, by taking the limit for  $t \rightarrow \infty$ , with  $t \in \mathcal{T}$ , and recalling (B10) and the continuous differentiability of  $F$ , we obtain that

$$\nabla f_{\hat{j}}(x_k + \bar{\alpha}d_k)^T d_k \geq \gamma\mathcal{D}(x_k, d_k). \quad (\text{B12})$$

Since  $\mathcal{D}(x_k + \bar{\alpha}d_k, d_k) \geq \nabla f_{\hat{j}}(x_k + \bar{\alpha}d_k)^T d_k$  by definition of  $\mathcal{D}(\cdot, \cdot)$ ,  $\sigma > \gamma$  and  $\mathcal{D}(x_k, d_k) < 0$ , from (B12) we have that

$$\mathcal{D}(x_k + \bar{\alpha}d_k, d_k) > \sigma\mathcal{D}(x_k, d_k). \quad (\text{B13})$$

However, from Lemma 5, it follows that

$$\mathcal{D}(x_k + \alpha_u^t d_k, d_k) < \sigma\mathcal{D}(x_k, d_k).$$

By taking the limit for  $t \rightarrow \infty$ , with  $t \in \mathcal{T}$ , and recalling (B10) and the continuity of  $\mathcal{D}(\cdot, \cdot)$ , we get from the last equation that

$$\mathcal{D}(x_k + \bar{\alpha}d_k, d_k) \leq \sigma\mathcal{D}(x_k, d_k).$$

The latter is in contradiction with (B13). We thus get the thesis.  $\square$

## C Definition of new test problems

In this appendix, we introduce the new convex test problem which we call MAN\_2. Moreover, we report the formulation of the rescaled versions of MAN\_1 [21], FDS\_1

[8] and MOP\_2 [33]. We remind that the reported lower and upper bounds were only used to choose the starting points for the algorithms (Sect. 5.1.2).

- MAN\_2:

$$f_1(x) = \sum_{i=1}^n \frac{i(x_i - i)^2}{n^2}$$

$$\min_{x \in \mathbb{R}^n} f_2(x) = \sum_{i=1}^n e^{-x_i} + x_i, \quad x \in [-1, 1]^n.$$

$$f_3(x) = \sum_{i=1}^n e^{x_i^2}$$

- M-MAN\_1:

$$f_1(x) = \sum_{i=1}^n \frac{(x_i - i)^2}{n}$$

$$\min_{x \in \mathbb{R}^n} f_2(x) = \sum_{i=1}^n e^{-x_i} + x_i, \quad x \in [-10, 10]^n.$$

- M-FDS\_1:

$$f_1(x) = \sum_{i=1}^n \frac{i(x_i - i)^4}{n^4}$$

$$\min_{x \in \mathbb{R}^n} f_2(x) = e^{\sum_{i=1}^n x_i/n} + \|x\|_2^2, \quad x \in [-2, 2]^n.$$

$$f_3(x) = \sum_{i=1}^n \frac{i(n - i + 1)e^{-x_i}}{n(n + 1)}$$

- M-MOP\_2:

$$\min_{x \in \mathbb{R}^n} f_1(x) = 1 - e^{-\sum_{i=1}^n (x_i - 1/\sqrt{n})^2/n}$$

$$f_2(x) = 1 - e^{-\sum_{i=1}^n (x_i + 1/\sqrt{n})^2/n}, \quad x \in [-4, 4]^n.$$

## References

1. Carrizosa, E., Frenk, J.B.G.: Dominating sets for convex functions with some applications. *J. Optim. Theory Appl.* **96**(2), 281–295 (1998). <https://doi.org/10.1023/A:1022614029984>
2. Shan, S., Wang, G.G.: An efficient Pareto set identification approach for multiobjective optimization on black-box functions. *J. Mech. Des.* **127**(5), 866–874 (2005). <https://doi.org/10.1115/DETC2004-57194>
3. Liuzzi, G., Lucidi, S., Parasiliti, F., Villani, M.: Multiobjective optimization techniques for the design of induction motors. *IEEE Trans. Magn.* **39**(3), 1261–1264 (2003). <https://doi.org/10.1109/TMAG.2003.810193>

4. Campana, E.F., Diez, M., Liuzzi, G., Lucidi, S., Pellegrini, R., Piccialli, V., Rinaldi, F., Serani, A.: A multi-objective direct algorithm for ship hull optimization. *Comput. Optim. Appl.* **71**(1), 53–72 (2018). <https://doi.org/10.1007/s10589-017-9955-0>
5. White, D.: Epsilon-dominating solutions in mean-variance portfolio analysis. *Eur. J. Oper. Res.* **105**(3), 457–466 (1998). [https://doi.org/10.1016/S0377-2217\(97\)00056-8](https://doi.org/10.1016/S0377-2217(97)00056-8)
6. Tavana, M.: A subjective assessment of alternative mission architectures for the human exploration of Mars at NASA using multicriteria decision making. *Comput. Oper. Res.* **31**(7), 1147–1164 (2004). [https://doi.org/10.1016/S0305-0548\(03\)00074-1](https://doi.org/10.1016/S0305-0548(03)00074-1)
7. Fliege, J., Svaiter, B.F.: Steepest descent methods for multicriteria optimization. *Math. Methods Oper. Res.* **51**(3), 479–494 (2000). <https://doi.org/10.1007/s001860000043>
8. Fliege, J., Drummond, L.G., Svaiter, B.F.: Newton’s method for multiobjective optimization. *SIAM J. Optim.* **20**(2), 602–626 (2009). <https://doi.org/10.1137/08071692X>
9. Gonçalves, M.L.N., Lima, F.S., Prudente, L.F.: Globally convergent newton-type methods for multiobjective optimization. *Comput. Optim. Appl.* **83**(2), 403–434 (2022). <https://doi.org/10.1007/s10589-022-00414-7>
10. Povalej, Z.: Quasi-Newton’s method for multiobjective optimization. *J. Comput. Appl. Math.* **255**, 765–777 (2014). <https://doi.org/10.1016/j.cam.2013.06.045>
11. Cocchi, G., Lapucci, M.: An augmented Lagrangian algorithm for multi-objective optimization. *Comput. Optim. Appl.* **77**(1), 29–56 (2020). <https://doi.org/10.1007/s10589-020-00204-z>
12. Lucambio Pérez, L.R., Prudente, L.F.: Nonlinear conjugate gradient methods for vector optimization. *SIAM J. Optim.* **28**(3), 2690–2720 (2018). <https://doi.org/10.1137/17M1126588>
13. Pascoletti, A., Serafini, P.: Scalarizing vector optimization problems. *J. Optim. Theory Appl.* **42**(4), 499–524 (1984). <https://doi.org/10.1007/BF00934564>
14. Deb, K., Pratap, A., Agarwal, S., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.* **6**(2), 182–197 (2002). <https://doi.org/10.1109/4235.996017>
15. Cocchi, G., Liuzzi, G., Lucidi, S., Sciandrone, M.: On the convergence of steepest descent methods for multiobjective optimization. *Comput. Optim. Appl.* (2020). <https://doi.org/10.1007/s10589-020-00192-0>
16. Cocchi, G., Lapucci, M., Mansueto, P.: Pareto front approximation through a multi-objective augmented Lagrangian method. *EURO J. Comput. Optim.* (2021). <https://doi.org/10.1016/j.ejco.2021.100008>
17. Custódio, A.L., Madeira, J.A., Vaz, A.I.F., Vicente, L.N.: Direct multisearch for multiobjective optimization. *SIAM J. Optim.* **21**(3), 1109–1140 (2011). <https://doi.org/10.1137/10079731X>
18. Cocchi, G., Liuzzi, G., Papini, A., Sciandrone, M.: An implicit filtering algorithm for derivative-free multiobjective optimization with box constraints. *Comput. Optim. Appl.* **69**(2), 267–296 (2018). <https://doi.org/10.1007/s10589-017-9953-2>
19. Locatelli, M., Maischberger, M., Schoen, F.: Differential evolution methods based on local searches. *Comput. Oper. Res.* **43**, 169–180 (2014). <https://doi.org/10.1016/j.cor.2013.09.010>
20. Mansueto, P., Schoen, F.: Memetic differential evolution methods for clustering problems. *Pattern Recogn.* **114**, 107849 (2021). <https://doi.org/10.1016/j.patcog.2021.107849>
21. Lapucci, M., Mansueto, P., Schoen, F.: A memetic procedure for global multi-objective optimization. *Math. Program. Comput.* (2022). <https://doi.org/10.1007/s12532-022-00231-3>
22. Bertsekas, D.: *Nonlinear Programming*, vol. 4, 2nd edn. Athena Scientific, Belmont, Massachusetts (2016)
23. Ansary, M.A.T., Panda, G.: A modified Quasi-Newton method for vector optimization problem. *Optimization* **64**(11), 2289–2306 (2015). <https://doi.org/10.1080/02331934.2014.947500>
24. Prudente, L.F., Souza, D.R.: A Quasi-Newton method with Wolfe line searches for multiobjective optimization. *J. Optim. Theory Appl.* **194**(3), 1107–1140 (2022). <https://doi.org/10.1007/s10957-022-02072-5>
25. Qu, S., Goh, M., Chan, F.T.S.: Quasi-Newton methods for solving multiobjective optimization. *Oper. Res. Lett.* **39**(5), 397–399 (2011). <https://doi.org/10.1016/j.orl.2011.07.008>
26. Nocedal, J.: Updating quasi-Newton matrices with limited storage. *Math. Comput.* **35**(151), 773–782 (1980). <https://doi.org/10.1090/S0025-5718-1980-0572855-7>
27. Graña Drummond, L.M., Svaiter, B.F.: A steepest descent method for vector optimization. *J. Comput. Appl. Math.* **175**(2), 395–414 (2005). <https://doi.org/10.1016/j.cam.2004.06.018>
28. Nocedal, J., Wright, S.J.: *Quasi-Newton Methods*. In: *Numerical Optimization*, pp. 135–163. Springer, New York, NY (2006). [https://doi.org/10.1007/978-0-387-40065-5\\_6](https://doi.org/10.1007/978-0-387-40065-5_6)



29. Lucambio Pérez, L.R., Prudente, L.F.: A Wolfe line search algorithm for vector optimization. *ACM Trans. Math. Softw.* (2019). <https://doi.org/10.1145/3342104>
30. Liu, D.C., Nocedal, J.: On the limited memory BFGS method for large scale optimization. *Math. Program.* **45**(1), 503–528 (1989). <https://doi.org/10.1007/BF01589116>
31. Jin, Y., Olhofer, M., Sendhoff, B.: Dynamic weighted aggregation for evolutionary multi-objective optimization: Why does it work and how? In: *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1042–1049 (2001)
32. Schütze, O., Lara, A., Coello, C.C.: The directed search method for unconstrained multi-objective optimization problems. In: *Proceedings of the EVOLVE—A Bridge Between Probability, Set Oriented Numerics, and Evolutionary Computation*, 1–4 (2011)
33. Huband, S., Hingston, P., Barone, L., While, L.: A review of multiobjective test problems and a scalable test problem toolkit. *IEEE Trans. Evol. Comput.* **10**(5), 477–506 (2006). <https://doi.org/10.1109/TEVC.2005.861417>
34. Miglierina, E., Molho, E., Recchioni, M.C.: Box-constrained multi-objective optimization: a gradient-like method without “a priori” scalarization. *Eur. J. Oper. Res.* **188**(3), 662–682 (2008). <https://doi.org/10.1016/j.ejor.2007.05.015>
35. Zhang, Q., Zhou, A., Zhao, S., Suganthan, P.N., Liu, W., Tiwari, S., et al.: Multiobjective optimization test instances for the CEC 2009 special session and competition. *Univ. Essex, Colch., UK Nanyang Tech. Univ., Singap., Spec. Sess. Perform. Assess. Multi-Object. Optim. Algorithms, Tech. Rep.* **264**, 1–30 (2008)
36. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. *Math. Program.* **91**(2), 201–213 (2002). <https://doi.org/10.1007/s101070100263>
37. Mansueto, P.: LM-Q-NWT: a limited memory Quasi-Newton approach for multi-objective optimization (2023). <https://doi.org/10.5281/zenodo.7533784>

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.