Journal of Optimization Theory and Applications (2021) 188:696–723
https://doi.org/10.1007/s10957-020-01803-w

An Efficient Descent Method for Locally Lipschitz Multiobjective Optimization Problems

Bennet Gebken[1] · **Sebastian Peitz**[1]

Received: 3 August 2020 / Accepted: 19 December 2020 / Published online: 13 January 2021

Abstract
We present an efficient descent method for unconstrained, locally Lipschitz multiobjective optimization problems. The method is realized by combining a theoretical result regarding the computation of descent directions for nonsmooth multiobjective optimization problems with a practical method to approximate the subdifferentials of the objective functions. We show convergence to points which satisfy a necessary condition for Pareto optimality. Using a set of test problems, we compare our method with the multiobjective proximal bundle method by Mäkelä. The results indicate that our method is competitive while being easier to implement. Although the number of objective function evaluations is larger, the overall number of subgradient evaluations is smaller. Our method can be combined with a subdivision algorithm to compute entire Pareto sets of nonsmooth problems. Finally, we demonstrate how our method can be used for solving sparse optimization problems, which are present in many real-life applications.

Keywords Multiobjective optimization · Nonsmooth optimization · Descent methods · Sparse optimization

Mathematics Subject Classification 90C29 · 90C30 · 90C56 · 49J52

Communicated by Xiaoqi Yang.

✉ Bennet Gebken
bgebken@math.upb.de

Sebastian Peitz
speitz@math.upb.de

[1] Chair of Applied Mathematics, Paderborn University, 33098 Paderborn, Germany

1 Introduction

In many scenarios in real life, the problem of optimizing multiple objectives at the same time arises. In engineering for example, one often wants to steer a physical system as close as possible to a desired state while minimizing the required energy at the same time. These problems are called *multiobjective optimization problems* (MOPs) and generally do not possess a single optimal solution. Instead, the solution is the set of all optimal compromises, the so-called *Pareto set*. Due to this, the numerical computation of solutions to MOPs is more challenging. In addition, there are numerous applications where the objectives are nonsmooth, for example contact problems in mechanics, which adds to the difficulty.

When addressing the above-mentioned difficulties, i.e., multiple objectives and nonsmoothness, separately, there exist a large number of solution methods. For smooth MOPs, the most popular methods include evolutionary [1,2] and scalarization methods [3]. Additionally, some methods from single-objective optimization have been generalized, like gradient descent methods [4–6] and Newton's method [7,8]. For the nonsmooth single-objective case, commonly used methods include subgradient methods [9], bundle methods [10] and gradient sampling methods [11]. More recently, in [12], a generalization of the steepest descent method to the nonsmooth case was proposed, which is based on an efficient approximation of the subdifferential of the objective function. For nonsmooth multiobjective optimization, the literature is a lot more scarce. Since classical scalarization approaches do not require the existence of gradients, they can still be used. In [13], a generalization of the steepest descent method was proposed for the case when the full subdifferentials of the objectives are known, which is rarely the case in practice. In [14,15], the subgradient method was generalized to the multiobjective case, but both articles report that their method is not suitable for real-life application due to inefficiency. In [16] (see also [17]), the proximal point method for single-objective optimization was generalized to convex vector optimization problems, where differentiability of the objectives is not required. In [18] (see also [19,20]), a multiobjective version of the proximal bundle method was proposed, which currently appears to be the most efficient solver.

In this article, we develop a new descent method for locally Lipschitz continuous MOPs by combining the descent direction from [13] with the approximation of the Clarke subdifferential from [12], which is based on the Goldstein ε-subdifferential [21]. In [13], it was shown that the element with the smallest norm in the negative convex hull of the subdifferentials of the objective functions is a common descent direction for all objectives. In [12], the subdifferential of the objective function was approximated by starting with a single subgradient and then systematically computing new subgradients until the element with the smallest norm in the convex hull of all subgradients is a direction of (sufficient) descent. Combining both approaches yields a descent direction for locally Lipschitz MOPs, and together with an Armijo step length, we obtain a descent method. We show convergence to points which satisfy a necessary condition for Pareto optimality (Sect. 3). We discuss the typical behavior of our method and present a modification which promises better efficiency in practice (Sect. 4.1). Using a set of test problems, we compare the performance of both methods to the multiobjective proximal bundle method from [18] (Sect. 4.2). The results indicate that our methods are inferior in terms of function evaluations, but superior in terms of subgradient evaluations. A globalization strategy to compute the complete Pareto set is discussed in Sect. 4.3. Finally, we demonstrate how our method can be used for sparse optimization in Sect. 4.4, a problem that is common to a large number of machine learning algorithms such as compressed sensing [22] or the sparse identification of nonlinear dynamics [23].

2 Nonsmooth Multiobjective Optimization

We consider the nonsmooth multiobjective optimization problem

$$\min_{x \in \mathbb{R}^n} f(x) = \min_{x \in \mathbb{R}^n} \begin{pmatrix} f_1(x) \\ \vdots \\ f_k(x) \end{pmatrix}, \tag{MOP}$$

where $f : \mathbb{R}^n \to \mathbb{R}^k$ is the *objective vector* with components $f_i : \mathbb{R}^n \to \mathbb{R}$ for $i \in \{1, \dots, k\}$, called *objective functions*. Throughout this article, we assume the objective functions to be *locally Lipschitz continuous*, i.e., for each $i \in \{1, \dots, k\}$ and $x \in \mathbb{R}^n$, there is some $L_i > 0$ and $\varepsilon > 0$ with

$$|f_i(y) - f_i(z)| \leq L_i \|y - z\| \quad \forall y, z \in \{y \in \mathbb{R}^n : \|x - y\| < \varepsilon\},$$

where $\|\cdot\|$ denotes the Euclidean norm in $\mathbb{R}^n$. Since (MOP) is an optimization problem with a vector-valued objective function, the classical concept of optimality from the scalar case cannot directly be conveyed. Instead, we are looking for the *Pareto set*, which is defined in the following way:

Definition 2.1 A point $x \in \mathbb{R}^n$ is called *Pareto optimal* for the problem (MOP), if there is no $y \in \mathbb{R}^n$ such that

$$\begin{aligned} f_i(y) &\leq f_i(x) \quad \forall i \in \{1, \dots, k\}, \\ f_j(y) &< f_j(x) \quad \text{for some } j \in \{1, \dots, k\}. \end{aligned}$$

The set of all Pareto optimal points is the *Pareto set*.

In practice, to check whether a given point is Pareto optimal, we need optimality conditions. In the smooth case, there are the well-known *Karush–Kuhn–Tucker (KKT) conditions* (cf. [3]), which are based on the gradients of the objective functions. In case the objective functions are merely locally Lipschitz, the KKT conditions can be generalized using the concept of *subdifferentials*. In the following, we recall the required definitions and results from nonsmooth analysis. For a more detailed introduction, we refer to [24], Chapter 2.

Definition 2.2 For $i \in \{1, \ldots, k\}$ let $\Omega_i \subseteq \mathbb{R}^n$ be the set of points where f_i is not differentiable. Then,

$$\partial f_i(x) := \operatorname{conv}(\{\xi \in \mathbb{R}^n : \exists (x_j)_j \in \mathbb{R}^n \backslash \Omega_i \text{ with } x_j \to x \text{ and } \nabla f_i(x_j) \to \xi \text{ for } j \to \infty\})$$

is the *(Clarke) subdifferential of* f_i *in* x. $\xi \in \partial f_i(x)$ is a *subgradient*.

Remark 2.1 The original definition of the Clarke subdifferential (on Banach spaces) was based on so-called *generalized directional derivatives* (cf. Chapter 2 in [24]). In the finite-dimensional case, Theorem 2.5.1 in [24] shows that the Clarke subdifferential is given by the expression used in Definition 2.2. Since we will only work with this expression, we omitted the original definition.

It is easy to see that, if an objective function is continuously differentiable, then the Clarke subdifferential is the set containing only the gradient. We will later use the following technical result on some properties of the Clarke subdifferential.

Lemma 2.1 *$\partial f_i(x)$ is nonempty, convex and compact for all $i \in \{1, \ldots, k\}$.*

Proof See [24], Prop. 2.1.2. □

Using the subdifferential, we can state a necessary optimality condition for locally Lipschitz MOPs.

Theorem 2.1 *Let $x \in \mathbb{R}^n$ be Pareto optimal. Then,*

$$0 \in \operatorname{conv}\left(\bigcup_{i=1}^{k} \partial f_i(x)\right). \tag{1}$$

Proof See [25], Thm. 12. □

In the smooth case, (1) reduces to the classical multiobjective KKT conditions. Note that in contrast to the smooth case, the optimality condition (1) is numerically challenging to work with, as subdifferentials are difficult to compute. Thus, in numerical methods, (1) is only used implicitly.

The method we are presenting in this paper is a *descent method*, which means that, starting from a point $x_1 \in \mathbb{R}^n$, we want to generate a sequence $(x_j)_j \in \mathbb{R}^n$ in which each point is an improvement over the previous point. This is done by computing directions $v_j \in \mathbb{R}^n$ and step lengths $t_j \in \mathbb{R}^{>0}$ such that $x_{j+1} = x_j + t_j v_j$ and

$$f_i(x_{j+1}) < f_i(x_j) \quad \forall j \in \mathbb{N},\ i \in \{1, \ldots, k\}.$$

For the computation of v_j, we recall the following basic result from convex analysis that forms the theoretical foundation for descent methods in the presence of multiple (sub)gradients. Let $\| \cdot \|$ be the Euclidean norm in $\mathbb{R}^n$.

Theorem 2.2 *Let $W \subseteq \mathbb{R}^n$ be convex and compact and*

$$\bar{v} := \operatorname*{argmin}_{\xi \in -W} \|\xi\|^2. \tag{2}$$

Then, either $\bar{v} \neq 0$ and

$$\langle \bar{v}, \xi \rangle \leq -\|\bar{v}\|^2 < 0 \quad \forall \xi \in W, \tag{3}$$

or $\bar{v} = 0$ and there is no $v \in \mathbb{R}^n$ with $\langle v, \xi \rangle < 0$ for all $\xi \in W$.

Proof Follows from [26], Lemma. □

Roughly speaking, Theorem 2.2 states that the element of minimal norm in the convex and compact set $-W$ is directionally opposed to all elements of W. To be more precise, $\bar{v}$ is contained in the intersection of all half-spaces induced by elements of $-W$. In the context of optimization, this result has several applications:

(i) In the smooth, single-objective case, $W = \{\nabla f(x)\}$ trivially yields the classical steepest descent method.
(ii) In the smooth, multiobjective case, $W = \operatorname{conv}(\{\nabla f_1(x), \ldots, \nabla f_k(x)\})$ yields the descent direction from [4] (after dualization) and [5].
(iii) In the nonsmooth, single-objective case, $W = \partial f(x)$ yields the descent direction from [24], Prop. 6.2.4.
(iv) In the nonsmooth, multiobjective case, $W = \operatorname{conv}\left(\bigcup_{i=1}^k \partial f_i(x)\right)$ yields the descent direction from [13].

In (i) and (ii), the solution of problem (2) is straightforward, since W is a convex polytope with the gradients as vertices. In (iii), the solution of (2) is non-trivial due to the difficulty of computing the subdifferential. In subgradient methods [9], the solution is approximated by using a single subgradient instead of the entire subdifferential. As a result, it cannot be guaranteed that the solution is a descent direction and in particular, (2) cannot be used as a stopping criterion. In gradient sampling methods [11], the subdifferential is approximated by the convex hull of gradients of the objective function in randomly sampled points around the current point. Due to the randomness, it cannot be guaranteed that the resulting direction yields sufficient descent. Additionally, a check for differentiability of the objective is required, which can pose a problem [27]. In (iv), the solution of (2) gets even more complicated due to the presence of multiple subdifferentials. So far, the only methods that deal with (2) in this case are multiobjective versions of the subgradient method [14,15], which were reported unsuitable for real-life applications.

In the following section, we will describe a new way to compute descent directions for nonsmooth MOPs by systematically computing an approximation of $\operatorname{conv}\left(\cup_{i=1}^k \partial f_i(x)\right)$ that is sufficient to obtain a "good enough" descent direction from (2).

3 Descent Method for Nonsmooth MOPs

In this section, we will present a method to compute descent directions of nonsmooth MOPs that generalizes the method from [12] to the multiobjective case. As described in the previous section, when computing descent directions via Theorem 2.2, the problem of computing subdifferentials arises. Since this is difficult in practice, we will instead replace W in Theorem 2.2 by an approximation of $\operatorname{conv}\left(\cup_{i=1}^{k} \partial f_i(x)\right)$ such that the resulting direction is guaranteed to have sufficient descent. To this end, we replace the Clarke subdifferential by the so-called ε-*subdifferential* and take a finite approximation of the latter.

3.1 The Epsilon-Subdifferential

By definition, $\partial f_i(x)$ is the convex hull of the limits of the gradient of f_i in all sequences near x that converge to x. Thus, if we evaluate ∇f_i in a number of points close to x (where it is defined) and take the convex hull, we expect the resulting set to be an approximation of $\partial f_i(x)$. To formalize this, we introduce the following definition [21,28].

Definition 3.1 Let $\varepsilon \geq 0$, $x \in \mathbb{R}^n$, $B_\varepsilon(x) := \{y \in \mathbb{R}^n : \|x - y\| \leq \varepsilon\}$ and $i \in \{1, \dots, k\}$. Then,

$$\partial_\varepsilon f_i(x) := \operatorname{conv}\left(\bigcup_{y \in B_\varepsilon(x)} \partial f_i(y)\right)$$

is the *(Goldstein)* ε-*subdifferential of* f_i *in* x. $\xi \in \partial_\varepsilon f_i(x)$ is an ε-*subgradient*.

Note that $\partial_0 f_i(x) = \partial f_i(x)$ and $\partial f_i(x) \subseteq \partial_\varepsilon f_i(x)$. For $\varepsilon \geq 0$ we define for the multiobjective setting

$$F_\varepsilon(x) := \operatorname{conv}\left(\bigcup_{i=1}^{k} \partial_\varepsilon f_i(x)\right).$$

To be able to choose $W = F_\varepsilon(x)$ in Theorem 2.2, we first need to establish some properties of $F_\varepsilon(x)$.

Lemma 3.1 $\partial_\varepsilon f_i(x)$ *is nonempty, convex and compact for all* $i \in \{1, \dots, k\}$. *In particular, the same holds for* $F_\varepsilon(x)$.

Proof For $\partial_\varepsilon f_i(x)$, this was shown in [21], Prop. 2.3. For $F_\varepsilon(x)$, it then follows directly from the definition. □

We immediately get the following corollary from Theorems 2.1 and 2.2.

Corollary 3.1 *Let* $\varepsilon \geq 0$.

(a) *If x is Pareto optimal, then*

$$0 \in F_\varepsilon(x). \tag{4}$$

(b) *Let $x \in \mathbb{R}^n$ and*

$$\bar{v} := \operatorname*{argmin}_{\xi \in -F_\varepsilon(x)} \|\xi\|^2. \tag{5}$$

Then, either $\bar{v} \neq 0$ and

$$\langle \bar{v}, \xi \rangle \leq -\|\bar{v}\|^2 < 0 \quad \forall \xi \in F_\varepsilon(x), \tag{6}$$

or $\bar{v} = 0$ and there is no $v \in \mathbb{R}^n$ with $\langle v, \xi \rangle < 0$ for all $\xi \in F_\varepsilon(x)$.

The previous corollary states that when working with the ε-subdifferential instead of the Clarke subdifferential, we still have a necessary optimality condition and a way to compute descent directions, although the optimality conditions are weaker and the descent direction has a less strong descent. This is illustrated in the following example.

Example 3.1 Consider the locally Lipschitz function

$$f : \mathbb{R}^2 \to \mathbb{R}^2, \quad x \mapsto \begin{pmatrix} (x_1 - 1)^2 + (x_2 - 1)^2 \\ x_1^2 + |x_2| \end{pmatrix}.$$

The set of nondifferentiable points of f is $\mathbb{R} \times \{0\}$. For $\varepsilon > 0$ and $x \in \mathbb{R}^2$ we have

$$\nabla f_1(x) = \begin{pmatrix} 2(x_1 - 1) \\ 2(x_2 - 1) \end{pmatrix} \quad \text{and} \quad \partial_\varepsilon f_1(x) = 2B_\varepsilon(x) - \begin{pmatrix} 2 \\ 2 \end{pmatrix}.$$

For $x \in \mathbb{R} \times \{0\}$ we have

$$\partial f_2(x) = \{2x_1\} \times [-1, 1] \quad \text{and} \quad \partial_\varepsilon f_2(x) = \{2x_1 + [-2\varepsilon, 2\varepsilon]\} \times [-1, 1].$$

Figure 1 shows the Clarke subdifferential (a), the ε-subdifferential (b) for $\varepsilon = 0.2$ and the corresponding sets $F_\varepsilon(x)$ for $x = (1.5, 0)^\top$.

Additionally, the corresponding solutions of (5) are shown. In this case, the predicted descent $-\|\bar{v}\|^2$ (cf. (3)) is approximately -3.7692 in (a) and -2.4433 in (b).

Figure 2 shows the same scenario for $x = (0.5, 0)^\top$.

Here, the Clarke subdifferential still yields a descent, while $\bar{v} = 0$ for the ε-subdifferential. In other words, x satisfies the necessary optimality condition (4) but not (1).

The following lemma shows that for the direction from (5), there is a lower bound for a step size up to which we have guaranteed descent in each objective function f_i.

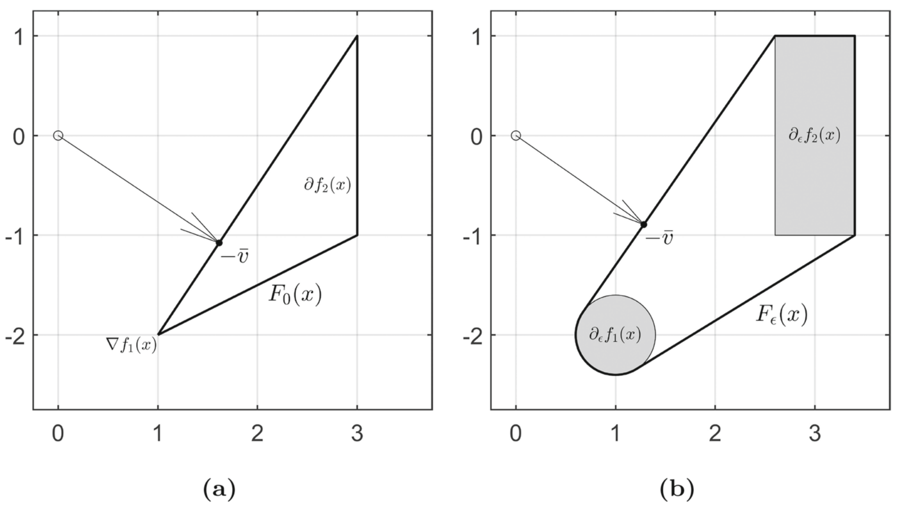

Fig. 1 Clarke subdifferentials (**a**), ε-subdifferentials (**b**) for $\varepsilon = 0.2$ and the corresponding sets $F_\varepsilon(x)$ for $x = (1.5, 0)^\top$ in Example 3.1

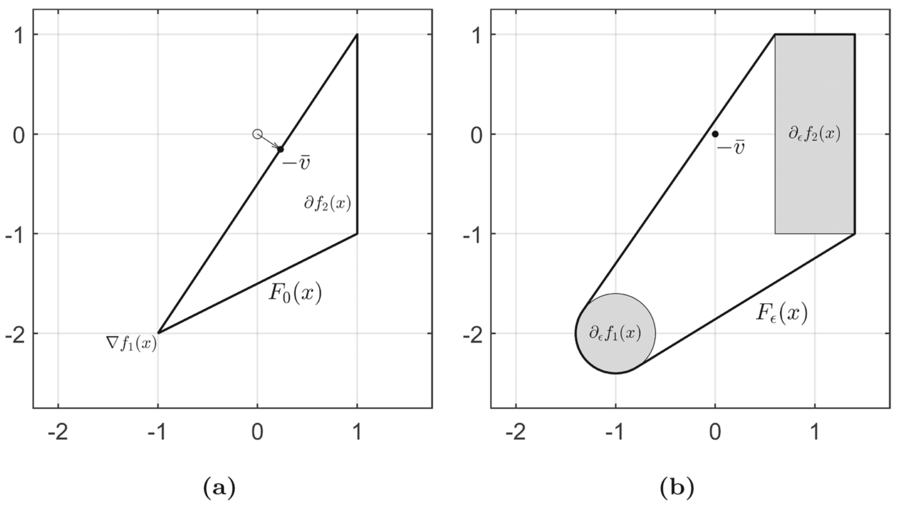

Fig. 2 Clarke subdifferentials (**a**), ε-subdifferentials (**b**) for $\varepsilon = 0.2$ and the corresponding sets $F_\varepsilon(x)$ for $x = (0.5, 0)^\top$ in Example 3.1

Lemma 3.2 *Let $\varepsilon \geq 0$ and $\bar{v}$ be the solution of* (5). *Then,*

$$f_i(x + t\bar{v}) \leq f_i(x) - t\|\bar{v}\|^2 \quad \forall t \leq \frac{\varepsilon}{\|\bar{v}\|}, i \in \{1, \dots, k\}.$$

Proof Let $t \leq \frac{\varepsilon}{\|\bar{v}\|}$. Since f_i is locally Lipschitz continuous on $\mathbb{R}^n$, it is in particular Lipschitz continuous on an open set containing $x + [0, t]\bar{v}$. By applying the mean value theorem (cf. [24], Thm. 2.3.7), we obtain the existence of some $r \in]0, t[$ with

$$f_i(x + t\bar{v}) - f_i(x) \in \langle \partial f_i(x + r\bar{v}), t\bar{v} \rangle.$$

From $\|x - (x + r\bar{v})\| = r\|\bar{v}\| < \varepsilon$ it follows that $\partial f_i(x + r\bar{v}) \subseteq \partial_\varepsilon f_i(x)$. This means that there is some $\xi \in \partial_\varepsilon f_i(x)$ such that

$$f_i(x + t\bar{v}) - f_i(x) = t\langle \xi, \bar{v} \rangle.$$

Combined with (6) we obtain

$$\begin{aligned} & f_i(x + t\bar{v}) - f_i(x) \le -t\|\bar{v}\|^2 \\ \Leftrightarrow \quad & f_i(x + t\bar{v}) \le f_i(x) - t\|\bar{v}\|^2, \end{aligned}$$

which completes the proof. □

In the following, we will describe how we can obtain a good approximation of (5) without requiring full knowledge of the ε-subdifferentials.

3.2 Efficient Computation of Descent Directions

In this part, we will describe how the solution of (5) can be approximated when only a single subgradient can be computed at every $x \in \mathbb{R}^n$. Similar to the gradient sampling approach, the idea behind our method is to replace $F_\varepsilon(x)$ in (5) by the convex hull of a finite number of ε-subgradients $\xi_1, \dots, \xi_m$ from $F_\varepsilon(x)$ for $m \in \mathbb{N}$. Since it is impossible to know a priori how many and which ε-subgradients are required to obtain a good descent direction, we solve (5) multiple times in an iterative approach to enrich our approximation until a satisfying direction has been found. To this end, we have to specify how to enrich our current approximation $\mathrm{conv}(\{\xi_1, \dots, \xi_m\})$ and how to characterize an acceptable descent direction.

Let $W = \{\xi_1, \dots, \xi_m\} \subseteq F_\varepsilon(x)$ and

$$\tilde{v} := \operatorname*{argmin}_{v \in -\mathrm{conv}(W)} \|v\|^2. \tag{7}$$

Let $c \in]0, 1[$. Motivated by Lemma 3.2, we regard $\tilde{v}$ as an *acceptable* descent direction, if

$$f_i\left(x + \frac{\varepsilon}{\|\tilde{v}\|}\tilde{v}\right) \le f_i(x) - c\varepsilon\|\tilde{v}\| \quad \forall i \in \{1, \dots, k\}. \tag{8}$$

If the set $I \subseteq \{1, \dots, k\}$ for which (8) is violated is nonempty, then we have to find a new ε-subgradient $\xi' \in F_\varepsilon(x)$ such that $W \cup \{\xi'\}$ yields a better descent direction. Intuitively, (8) being violated means that the local behavior of f_i, $i \in I$, in x in the direction $\tilde{v}$ is not sufficiently captured in W. Thus, for each $i \in I$, we expect that there exists some $t' \in]0, \frac{\varepsilon}{\|\tilde{v}\|}]$ such that $\xi' \in \partial f_i(x + t'\tilde{v})$ improves the approximation of $F_\varepsilon(x)$. This is proven in the following lemma.

Lemma 3.3 *Let $c \in \,]0,1[$, $W = \{\xi_1, \dots, \xi_m\} \subseteq F_\varepsilon(x)$ and $\tilde{v}$ be the solution of (7). If*

$$f_i\left(x + \frac{\varepsilon}{\|\tilde{v}\|}\tilde{v}\right) > f_i(x) - c\varepsilon\|\tilde{v}\|$$

for some $i \in \{1, \dots, k\}$, then there is some $t' \in \,]0, \frac{\varepsilon}{\|\tilde{v}\|}]$ and $\xi' \in \partial f_i(x + t'\tilde{v})$ such that

$$\langle \tilde{v}, \xi' \rangle > -c\|\tilde{v}\|^2. \tag{9}$$

In particular, $\xi' \in F_\varepsilon(x) \setminus \operatorname{conv}(W)$.

Proof Assume that for all $t' \in \,]0, \frac{\varepsilon}{\|\tilde{v}\|}]$ and $\xi' \in \partial f_i(x + t'\tilde{v})$ we have

$$\langle \tilde{v}, \xi' \rangle \leq -c\|\tilde{v}\|^2. \tag{10}$$

By applying the mean value theorem as in Lemma 3.2, we obtain

$$f_i\left(x + \frac{\varepsilon}{\|\tilde{v}\|}\tilde{v}\right) - f_i(x) \in \langle \partial f_i(x + r\tilde{v}), \frac{\varepsilon}{\|\tilde{v}\|}\tilde{v} \rangle$$

for some $r \in \,]0, \frac{\varepsilon}{\|\tilde{v}\|}[$. This means that there is some $\xi' \in \partial f_i(x + r\tilde{v})$ such that

$$f_i\left(x + \frac{\varepsilon}{\|\tilde{v}\|}\tilde{v}\right) - f_i(x) = \langle \xi', \frac{\varepsilon}{\|\tilde{v}\|}\tilde{v} \rangle = \frac{\varepsilon}{\|\tilde{v}\|}\langle \xi', \tilde{v} \rangle.$$

By (10) it follows that

$$\begin{aligned} & f_i\left(x + \frac{\varepsilon}{\|\tilde{v}\|}\tilde{v}\right) - f_i(x) \leq -c\varepsilon\|\tilde{v}\| \\ \Leftrightarrow \quad & f_i\left(x + \frac{\varepsilon}{\|\tilde{v}\|}\tilde{v}\right) \leq f_i(x) - c\varepsilon\|\tilde{v}\|, \end{aligned}$$

which is a contradiction. In particular, (3) yields $\xi' \in F_\varepsilon(x) \setminus \operatorname{conv}(W)$. □

The following example visualizes the previous lemma.

Example 3.2 Consider f as in Example 3.1, $\varepsilon = 0.2$ and $x = (0.75, 0)^\top$. The dashed lines in Fig. 3 show the ε-subdifferentials, $F_\varepsilon(x)$ and the resulting descent direction (cf. Figs. 1 and 2).

Let $y = (0.94, -0.02)^\top$. Then, we have $\|x - y\| \approx 0.191 \leq \varepsilon$, so $y \in B_\varepsilon(x)$ and

$$\begin{aligned} \partial_\varepsilon f_1(x) \supseteq \partial f_1(y) &= \left\{ \begin{pmatrix} -0.12 \\ -2.04 \end{pmatrix} \right\} =: \{\xi_1\}, \\ \partial_\varepsilon f_2(x) \supseteq \partial f_2(y) &= \left\{ \begin{pmatrix} 1.88 \\ -1 \end{pmatrix} \right\} =: \{\xi_2\}. \end{aligned}$$

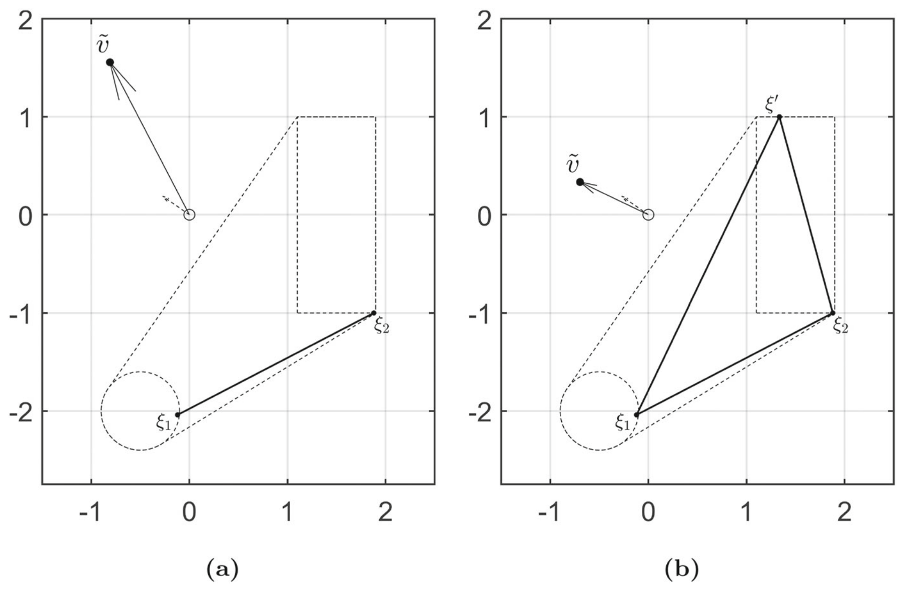

Fig. 3 Approximations of $F_\varepsilon(x)$ for $\varepsilon = 0.2$ and $x = (0.75, 0)^\top$ in Example 3.2. $F_\epsilon(x)$ is approximated by $\text{conv}(\{\xi_1, \xi_2\})$ in **a** and by $\text{conv}(\{\xi_1, \xi_2, \xi'\})$ in **b**

Let $W := \{\xi_1, \xi_2\}$ and $\text{conv}(W)$ be the approximation of $F_\varepsilon(x)$, shown as the solid line in Fig. 3a. Let $\tilde{v}$ be the solution of (7) for this W and choose $c = 0.25$. Checking (8), we have

$$f_2\left(x + \frac{\varepsilon}{\|\tilde{v}\|}\tilde{v}\right) \approx 0.6101 > 0.4748 \approx f_2(x) - c\varepsilon\|\tilde{v}\|.$$

By Lemma 3.3, this means that there is some $t' \in]0, \frac{\varepsilon}{\|\tilde{v}\|}]$ and $\xi' \in \partial f_2(x + t'\tilde{v})$ such that

$$\langle \tilde{v}, \xi' \rangle > -c\|\tilde{v}\|^2.$$

In this case, we can take for example $t' = \frac{1}{2}\frac{\varepsilon}{\|\tilde{v}\|}$, resulting in

$$\partial f_2(x + t'v) \approx \left\{\begin{pmatrix} 1.4077 \\ 1 \end{pmatrix}\right\} =: \{\xi'\},$$
$$\langle \tilde{v}, \xi' \rangle \approx 0.4172 > -0.7696 \approx -c\|\tilde{v}\|^2.$$

Figure 3b shows the improved approximation $W \cup \{\xi'\}$ and the resulting descent direction $\tilde{v}$. By checking (8) for this new descent direction, we see that $\tilde{v}$ is acceptable. (Note that in general, a single improvement step like this will not be sufficient to reach an acceptable direction.)

Note that Lemma 3.3 only shows the existence of t' and ξ' without stating a way how to actually compute them. To this end, let $i \in \{1, \dots, k\}$ be the index of an objective function for which (8) is not satisfied, define

$$h_i : \mathbb{R} \to \mathbb{R}, \quad t \mapsto f_i(x + t\tilde{v}) - f_i(x) + ct\|\tilde{v}\|^2 \tag{11}$$

(cf. [12]) and consider Algorithm 1. If f_i is continuously differentiable around x, then (9) is equivalent to $h_i'(t') > 0$, i.e., h_i being monotonically increasing around t'. Thus, the idea of Algorithm 1 is to find some t such that h_i is monotonically increasing around t, while checking whether (9) is satisfied for a subgradient $\xi \in f_i(x + t\tilde{v})$.

Algorithm 1 Compute new subgradient

```
Require: Current point $x \in \mathbb{R}^n$, direction $\tilde{v}$, tolerance $\varepsilon$, Armijo parameter $c \in ]0, 1[$.
1: Set $a = 0$, $b = \frac{\varepsilon}{\|\tilde{v}\|}$ and $t = \frac{a+b}{2}$.
2: Compute $\xi' \in \partial f_i(x + t\tilde{v})$.
3: If $\langle \tilde{v}, \xi' \rangle > -c\|\tilde{v}\|^2$ then stop.
4: If $h_i(b) > h_i(t)$ then set $a = t$. Otherwise set $b = t$.
5: Set $t = \frac{a+b}{2}$ and go to step 2.
```

Although in the general setting, we cannot guarantee that Algorithm 1 yields a subgradient satisfying (9), we can at least show that after finitely many iterations, a factor t is found such that $\partial f_i(x + t\tilde{v})$ contains a subgradient that satisfies (9).

Lemma 3.4 *Let $i \in \{1, \dots, k\}$ be an index for which (8) is violated. Let $(t_j)_j$ be the sequence generated in Algorithm 1.*

(a) *If $(t_j)_j$ is finite, then some ξ' was found satisfying (9).*
(b) *If $(t_j)_j$ is infinite, then it converges to some $\bar{t} \in]0, \frac{\varepsilon}{\|\tilde{v}\|}]$ with $h_i(\bar{t}) \geq h_i(\frac{\varepsilon}{\|\tilde{v}\|})$ such that*

  (i) *there is some $\xi' \in \partial f_i(x + \bar{t}\tilde{v})$ satisfying (9) or*
  (ii) *$0 \in \partial h_i(\bar{t})$, i.e., $\bar{t}$ is a* critical point *of h_i.*

Proof Let $(t_j)_j$ be finite with last element $\bar{t} \in]0, \frac{\varepsilon}{\|\tilde{v}\|}[$. Then, Algorithm 1 must have stopped in step 3, i.e., some $\xi' \in \partial f_i(x + \bar{t}\tilde{v})$ satisfying (9) was found.
Now let $(t_j)_j$ be infinite. By construction, $(t_j)_j$ is a Cauchy sequence in the compact set $[0, \frac{\varepsilon}{\|\tilde{v}\|}]$, so it has to converge to some $\bar{t} \in [0, \frac{\varepsilon}{\|\tilde{v}\|}]$. Additionally, since (8) is violated for the index i by assumption, we have

$$h_i(0) = 0 \quad \text{and} \quad h_i\left(\frac{\varepsilon}{\|\tilde{v}\|}\right) > 0.$$

Let $(a_j)_j$ and $(b_j)_j$ be the sequences corresponding to a and b in Algorithm 1 (at the start of each iteration in step 2).
Assume that $\bar{t} = 0$. Then, we must have $h_i(t_j) \geq h_i(b_j)$ in step 4 for all $j \in \mathbb{N}$. Thus,

$$h_i(t_j) \geq h_i(b_j) = h(t_{j-1}) \geq h_i(b_{j-1}) = \dots = h_i(t_1) \geq h_i(b_1) = h_i\left(\frac{\varepsilon}{\|\tilde{v}\|}\right) > 0$$

for all $j \in \mathbb{N}$. This is a contradiction to the continuity of h_i (in 0). So we must have $\bar{t} > 0$. Additionally, $\lim_{j\to\infty} b_j = \bar{t}$ and $(h_i(b_j))_j$ is non-decreasing, so

$$h_i(\bar{t}) = \lim_{j\to\infty} h_i(b_j) \geq h_i(b_1) = h_i\left(\frac{\varepsilon}{\|\tilde{v}\|}\right).$$

By construction we have $h_i(a_j) < h_i(b_j)$ for all $j \in \mathbb{N}$. Thus, by the mean value theorem, there has to be some $r_j \in]a_j, b_j[$ such that

$$0 < h_i(b_j) - h_i(a_j) \in \langle \partial h_i(r_j), b_j - a_j\rangle = \partial h_i(r_j)(b_j - a_j).$$

In particular, $\lim_{j\to\infty} r_j = \bar{t}$ and since $a_j < b_j$, $\partial h_i(r_j) \cap \mathbb{R}^{>0} \neq \emptyset$ for all $j \in \mathbb{N}$. By upper semicontinuity of ∂h there must be some $\theta \in \partial h_i(\bar{t})$ with $\theta \geq 0$. By the chain rule, we have

$$0 \leq \theta \in \partial h_i(\bar{t}) \subseteq \langle \tilde{v}, \partial f_i(x + \bar{t}\tilde{v})\rangle + c\|\tilde{v}\|^2. \tag{12}$$

Thus, there must be some $\xi \in \partial f_i(x + \bar{t}\tilde{v})$ with

$$0 \leq \langle \tilde{v}, \xi\rangle + c\|\tilde{v}\|^2 \quad \Leftrightarrow \quad \langle \tilde{v}, \xi\rangle \geq -c\|\tilde{v}\|^2.$$

If there exists $\xi' \in \partial f_i(x + \bar{t}\tilde{v})$ such that this inequality is strict, i.e., such that $\langle \tilde{v}, \xi'\rangle > -c\|\tilde{v}\|^2$, then b)(i) holds. Otherwise, if we have $\langle \tilde{v}, \xi'\rangle \leq -c\|\tilde{v}\|^2$ for all $\xi' \in \partial f_i(x + \bar{t}\tilde{v})$, then (12) implies $\partial h_i(\bar{t}) \subseteq \mathbb{R}^{\leq 0}$. This means that $0 = \theta \in \partial h_i(\bar{t})$, so b)(ii) holds. □

In the following remark, we will briefly discuss the implication of Lemma 3.4 for practical use of Algorithm 1.

Remark 3.1 Let $i \in \{1, \dots, k\}$ and $(t_j)_j$ be as in Lemma 3.4. Assume that $(t_j)_j$ is infinite with limit $\bar{t} \in]0, \frac{\varepsilon}{\|\tilde{v}\|}]$. In the proof of Lemma 3.4, we showed that there is a sequence $(r_j)_j \in]0, \frac{\varepsilon}{\|\tilde{v}\|}[$ with limit $\bar{t}$ such that $\partial h_i(r_j) \cap \mathbb{R}^{>0} \neq \emptyset$ for all $j \in \mathbb{N}$. By the definition of the Clarke subdifferential, this implies that there is a sequence $(s_j)_j \in]0, \frac{\varepsilon}{\|\tilde{v}\|}[$ with $\lim_{j\to\infty} s_j = \bar{t}$ such that h_i is differentiable in s_j and $h_i'(s_j) > 0$ for all $j \in \mathbb{N}$. Due to the upper semicontinuity of the Clarke subdifferential, each s_j has an open neighborhood U_j with

$$\partial h_i(s) \subseteq \mathbb{R}^{>0} \quad \forall s \in U_j,\ j \in \mathbb{N}.$$

As in the proof of Lemma 3.4, it follows that for all $s \in U_j$ with $j \in \mathbb{N}$, there is some $\xi' \in \partial f_i(x + s\tilde{v})$ satisfying (9). Thus, roughly speaking, even if we are in case b)(ii) of Lemma 3.4, there are open sets arbitrarily close to $\bar{t}$ on which we can find new ε-subgradients (as in case b)(i)).

Motivated by the previous remark, we will from now on assume that Algorithm 1 stops after finitely many iterations and thus yields a new subgradient satisfying (9).

Algorithm 2 Compute descent direction

```
Require: Current point $x \in \mathbb{R}^n$, tolerances $\varepsilon, \delta > 0$, Armijo parameter $c \in ]0,1[$.
1: Compute $\xi_1^i \in \partial_\varepsilon f_i(x)$ for all $i \in \{1,...,k\}$. Set $W_1 = \{\xi_1^1,...,\xi_1^k\}$ and $l = 1$.
2: Compute $v_l = \operatorname{argmin}_{v \in -\operatorname{conv}(W_l)} \|v\|^2$.
3: If $\|v_l\| \leq \delta$ then stop.
4: Find all objective functions for which there is insufficient descent:
       $I_l = \left\{ j \in \{1,...,k\} : f_j\left(x + \frac{\varepsilon}{\|v_l\|} v_l\right) > f_j(x) - c\varepsilon\|v_l\| \right\}.$
   If $I_l = \emptyset$ then stop.
5: For each $j \in I_l$, compute $t \in ]0, \frac{\varepsilon}{\|v_l\|}]$ and $\xi_l^j \in \partial f_j(x + t v_l)$ such that
       $\langle v_l, \xi_l^j \rangle > -c\|v_l\|^2$
   via Algorithm 1.
6: Set $W_{l+1} = W_l \cup \{\xi_l^j : j \in I_l\}$, $l = l + 1$ and go to step 2.
```

We can use this method of finding new subgradients to construct an algorithm that computes descent directions of nonsmooth MOPs, namely Algorithm 2.

The following theorem shows that Algorithm 2 stops after a finite number of iterations and produces an acceptable descent direction (cf. (8)).

Theorem 3.1 *Algorithm 2 terminates. In particular, if $\tilde{v}$ is the last element of $(v_l)_l$, then either $\|\tilde{v}\| \leq \delta$ or $\tilde{v}$ is an acceptable descent direction, i.e.,*

$$f_i\left(x + \frac{\varepsilon}{\|\tilde{v}\|}\tilde{v}\right) \leq f_i(x) - c\varepsilon\|\tilde{v}\| \quad \forall i \in \{1,...,k\}.$$

Proof Assume that Algorithm 2 does not terminate, i.e., $(v_l)_{l\in\mathbb{N}}$ is an infinite sequence. Let $l > 1$ and $j \in I_{l-1}$. Since $\xi_{l-1}^j \in W_l$ and $-v_{l-1} \in W_{l-1} \subseteq W_l$, we have

$$\begin{aligned}\|v_l\|^2 &\leq \| - v_{l-1} + s(\xi_{l-1}^j + v_{l-1})\|^2 \\ &= \|v_{l-1}\|^2 - 2s\langle v_{l-1}, \xi_{l-1}^j + v_{l-1}\rangle + s^2\|\xi_{l-1}^j + v_{l-1}\|^2 \\ &= \|v_{l-1}\|^2 - 2s\langle v_{l-1}, \xi_{l-1}^j\rangle - 2s\|v_{l-1}\|^2 + s^2\|\xi_{l-1}^j + v_{l-1}\|^2 \qquad (13)\end{aligned}$$

for all $s \in [0,1]$. Since $j \in I_{l-1}$, we must have

$$\langle v_{l-1}, \xi_{l-1}^j\rangle > -c\|v_{l-1}\|^2 \qquad (14)$$

by step 5. Let L be a common Lipschitz constant of all f_i, $i \in \{1,...,k\}$, on the closed ε-ball $B_\varepsilon(x)$ around x. Then by [24], Prop. 2.1.2, and the definition of the ε-subdifferential, we must have $\|\xi\| \leq L$ for all $\xi \in F_\varepsilon(x)$. So in particular,

$$\|\xi_{l-1}^j + v_{l-1}\| \leq 2L. \qquad (15)$$

Combining (13) with (14) and (15) yields

$$\begin{aligned}\|v_l\|^2 &< \|v_{l-1}\|^2 + 2sc\|v_{l-1}\|^2 - 2s\|v_{l-1}\|^2 + 4s^2L^2 \\ &= \|v_{l-1}\|^2 - 2s(1-c)\|v_{l-1}\|^2 + 4s^2L^2.\end{aligned}$$

Let $s := \frac{1-c}{4L^2}\|v_{l-1}\|^2$. Since $1-c \in \,]0,1[$ and $\|v_{l-1}\| \leq L$ we have $s \in \,]0,1[$. We obtain

$$\begin{aligned}\|v_l\|^2 &< \|v_{l-1}\|^2 - 2\frac{(1-c)^2}{4L^2}\|v_{l-1}\|^4 + \frac{(1-c)^2}{4L^2}\|v_{l-1}\|^4 \\ &= \left(1 - \frac{(1-c)^2}{4L^2}\|v_{l-1}\|^2\right)\|v_{l-1}\|^2.\end{aligned}$$

Since Algorithm 2 did not terminate, it holds $\|v_{l-1}\| > \delta$. It follows that

$$\|v_l\|^2 < \left(1 - \left(\frac{1-c}{2L}\delta\right)^2\right)\|v_{l-1}\|^2.$$

Let $r = 1 - \left(\frac{1-c}{2L}\delta\right)^2$. Note that we have $\delta < \|v_l\| \leq L$ for all $l \in \mathbb{N}$, so $r \in \,]0,1[$. Additionally, r does not depend on l, so we have

$$\|v_l\|^2 < r\|v_{l-1}\|^2 < r^2\|v_{l-1}\|^2 < \cdots < r^{l-1}\|v_1\|^2 \leq r^{l-1}L^2.$$

In particular, there is some l such that $\|v_l\| \leq \delta$, which is a contradiction. □

Remark 3.2 The proof of Theorem 3.1 shows that for convergence of Algorithm 2, it would be sufficient to consider only a single $j \in I_l$ in step 5. Similarly, for the initial approximation W_1, a single element of $\partial_\varepsilon f_i(x)$ for any $i \in \{1,\dots,k\}$ would be enough. A modification of either step can potentially reduce the number of executions of step 5 (i.e., Algorithm 1) in Algorithm 2 in case the ε-subdifferentials of multiple objective functions are similar. Nonetheless, we will restrain the discussion in this article to Algorithm 2 as it is, since both modifications also introduce a bias toward certain objective functions, which we want to avoid.

To highlight the strengths of Algorithm 2, we will consider an example where standard gradient sampling approaches can fail to obtain a useful descent direction.

Example 3.3 For $a, b \in \mathbb{R}\backslash\{0\}$ consider the locally Lipschitz function

$$f : \mathbb{R}^2 \to \mathbb{R}^2, \quad x \mapsto \begin{pmatrix}(x_1-1)^2 + (x_2-1)^2 \\ |x_2 - a|x_1|| + bx_2\end{pmatrix}.$$

The set of nondifferentiable points is

$$\Omega_f = (\{0\} \times \mathbb{R}) \cup \{(\lambda, a|\lambda|)^\top : \lambda \in \mathbb{R}\},$$

separating $\mathbb{R}^2$ into four smooth areas (cf. Fig. 4a). For large $a > 0$, the two areas above the graph of $\lambda \mapsto a|\lambda|$ become small, making it difficult to compute the subdifferential close to $(0, 0)^\top$.

Let $a = 10, b = 0.5, \varepsilon = 10^{-3}$ and $x = (10^{-4}, 10^{-4})^\top$. In this case, $(0, 0)^\top$ is the minimal point of f_2 and

$$\begin{aligned} \partial_\varepsilon f_2(x) &= \operatorname{conv}\left\{ \begin{pmatrix} -a \\ b-1 \end{pmatrix}, \begin{pmatrix} a \\ b+1 \end{pmatrix}, \begin{pmatrix} a \\ b-1 \end{pmatrix}, \begin{pmatrix} -a \\ b+1 \end{pmatrix} \right\} \\ &= \operatorname{conv}\left\{ \begin{pmatrix} -10 \\ -0.5 \end{pmatrix}, \begin{pmatrix} 10 \\ 1.5 \end{pmatrix}, \begin{pmatrix} 10 \\ -0.5 \end{pmatrix}, \begin{pmatrix} -10 \\ 1.5 \end{pmatrix} \right\}. \end{aligned}$$

In particular, $0 \in \partial_\varepsilon f_2(x)$, so the descent direction from (5) with the exact ε-subdifferentials is zero. When applying Algorithm 2 in x, after two iterations we obtain

$$\tilde{v} = v_2 \approx (0.118, 1.185) \cdot 10^{-9},$$

i.e., $\|\tilde{v}\| \approx 1.191 \cdot 10^{-11}$. Thus, x is correctly identified as (potentially) Pareto optimal. The final approximation W_2 of $F_\varepsilon(x)$ is

$$W_2 = \left\{ \xi_1^1, \xi_1^2, \xi_2^2 \right\} = \left\{ \begin{pmatrix} -1.9998 \\ -1.9998 \end{pmatrix}, \begin{pmatrix} 10 \\ -0.5 \end{pmatrix}, \begin{pmatrix} -10 \\ 1.5 \end{pmatrix} \right\}.$$

The first two elements of W_2 are the gradients of f_1 and f_2 in x from the first iteration of Algorithm 2, and the last element is the gradient of f_2 in the point $x' = x + tv_1 = (0.038, 0.596)^\top \cdot 10^{-3} \in B_\varepsilon(x)$ from the second iteration. The result is visualized in Fig. 4.

Building on Algorithm 2, it is now straightforward to construct the descent method for locally Lipschitz continuous MOPs given in Algorithm 3. In step 4, the classical Armijo backtracking line search was used (cf. [4]) for the sake of simplicity. Note that it is well defined due to step 4 in Algorithm 2.

Algorithm 3 Nonsmooth descent method

```
Require: Initial point $x_1 \in \mathbb{R}^n$, tolerances $\varepsilon, \delta > 0$, Armijo parameters $c \in ]0, 1[$, $t_0 > 0$.
1: Set $j = 1$.
2: Compute a descent direction $v_j$ via Algorithm 2.
3: If $\|v_j\| \leq \delta$ then stop.
4: Compute
       $$\bar{s} = \inf(\{s \in \mathbb{N} \cup \{0\} : f_i(x_j + 2^{-s} t_0 v_j) \leq f_i(x_j) - 2^{-s} t_0 c \|v_j\|^2 \ \forall i \in \{1, ..., k\}\})$$
   and set $\bar{t} = \max(\{2^{-\bar{s}} t_0, \frac{\varepsilon}{\|v_j\|}\})$.
5: Set $x_{j+1} = x_j + \bar{t} v_j$, $j = j + 1$ and go to step 2.
```

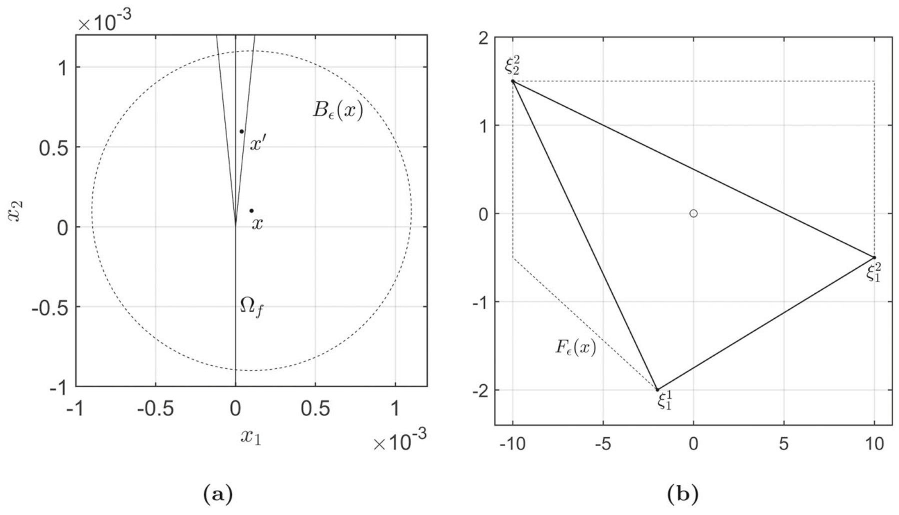

Fig. 4 **a** The set of nondifferentiable points Ω_f of f, the ball $B_\varepsilon(x)$ for the ε-subdifferential and the point in which subgradients were computed for Algorithm 2 in Example 3.3. **b** The approximation of $F_\varepsilon(x)$ in Algorithm 2

Since we introduced the two tolerances ε (for the ε-subdifferential) and δ (as a threshold for when we consider ε-subgradients to be zero), we cannot expect that Algorithm 3 computes points which satisfy the optimality condition (1). This is why we introduce the following definition, similar to the definition of ε-stationarity from [11].

Definition 3.2 Let $x \in \mathbb{R}^n$, $\varepsilon > 0$ and $\delta > 0$. Then, x is called (ε, δ)-*critical*, if

$$\min_{v \in -F_\varepsilon(x)} \|v\| \leq \delta.$$

It is easy to see that (ε, δ)-criticality is a necessary optimality condition for Pareto optimality, but a weaker one than (1). The following theorem shows that convergence in the sense of (ε, δ)-criticality is what we can expect from our descent method.

Theorem 3.2 *Let $(x_j)_j$ be the sequence generated by Algorithm 3. Then, either the sequence $(f_i(x_j))_j$ is unbounded below for each $i \in \{1, \dots, k\}$, or $(x_j)_j$ is finite with the last element being (ε, δ)-critical.*

Proof Assume that $(x_j)_j$ is infinite. Then, $\|v_j\| > \delta$ for all $j \in \mathbb{N}$. Let $j \in \mathbb{N}$. If $\bar{t} = 2^{-\bar{s}} t_0 \geq \frac{\varepsilon}{\|v_j\|}$ in step 4 of Algorithm 3, then

$$\begin{aligned} f_i(x_j + \bar{t} v_j) - f_i(x_j) &= f_i(x_j + 2^{-\bar{s}} t_0 v_j) - f_i(x_j) \\ &\leq -2^{-\bar{s}} t_0 c \|v_j\|^2 \leq -c\varepsilon \|v_j\| < -c\varepsilon\delta < 0 \end{aligned}$$

for all $i \in \{1, \dots, k\}$. If instead $\bar{t} = \frac{\varepsilon}{\|v_j\|}$ in step 4, then the same inequality holds due to Theorem 3.1. This implies that $(f_i(x_j))_j$ is unbounded below for each $i \in \{1, \dots, k\}$.

Now assume that $(x_j)_j$ is finite, with $\bar{x}$ and $\bar{v}$ being the last elements of $(x_j)_j$ and $(v_j)_j$, respectively. Since the algorithm stopped, we must have $\|\bar{v}\| \leq \delta$. From the application of Algorithm 2 in step 2, we know that there must be some $\overline{W} \subseteq F_\varepsilon(\bar{x})$ such that $\bar{v} = \operatorname*{argmin}_{v \in -\operatorname{conv}(\overline{W})} \|v\|^2$. This implies

$$\min_{v \in -F_\varepsilon(\bar{x})} \|v\| \leq \min_{v \in -\operatorname{conv}(\overline{W})} \|v\| = \|\bar{v}\| \leq \delta,$$

which completes the proof. □

Finally, before considering numerical examples in the following section, we will discuss the influence of the parameters of Algorithm 3 on its results and performance:

– The tolerance $\varepsilon > 0$ is the radius of the closed ball that we use for the ε-subdifferential (cf. Definition 3.1). On the one hand, by definition, we can only expect Algorithm 3 to compute the actual Pareto critical set up to a distance of ε, which is the motivation for choosing ε relatively small. On the other hand, ε controls how early the algorithm notices nondifferentiable points, so a small ε means that the algorithm might take a longer time until the nondifferentiability of the objective vector is detected. These two properties should be kept in mind when choosing ε in practice.
– The tolerance $\delta > 0$ is the threshold for when we consider the norm of the descent direction (i.e., a convex combination of ε-subgradients) to be zero. The result of Algorithm 3 will be closer to the Pareto critical set the smaller we choose δ. But since more iterations are needed, it also takes more time until the algorithm stops.
– The parameters $c \in]0, 1[$ and $t_0 > 0$ are used for the step length, where t_0 is the initial step length and c is the percentage of how much actual descent we want to have compared to the predicted descent based on the ε-subgradients. The closer c is chosen to 1, the steeper the descent of the computed descent direction will be and the closer the direction computed by Algorithm 2 will be to the theoretical descent direction from (5). Then again, this also increases the iterations required in Algorithm 2 and therefore the runtime of Algorithm 3.

Examples for the choice of these parameters can be found in the following section. Furthermore, we will introduce a modification of Algorithm 3 with a dynamic tolerance ε.

4 Numerical Examples

In this section, we will consider several numerical examples. We will begin by discussing the typical behavior of Algorithm 3 and presenting a modification (Algorithm 4) which has some practical advantages. Afterward, we compare the performance of both algorithms to the *multiobjective proximal bundle method* [18]. In order to approximate the entire Pareto set of nonsmooth MOPs, we will then combine Algorithm 3 with the *subdivision algorithm* [29], and show how the resulting method can be used for the solution of sparse optimization problems.

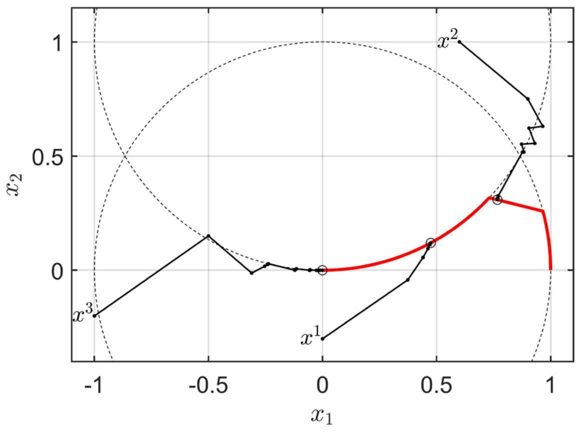

Fig. 5 Result of Algorithm 3 in three different starting points for the MOP (16). The Pareto set is shown in red, and the dashed lines show the set of nondifferentiable points Ω_f

4.1 Typical Behavior

In smooth areas, the behavior of Algorithm 3 is almost identical to the behavior of the multiobjective steepest descent method [4]. The only difference stems from the fact that, unlike the Clarke subdifferential, the ε-subdifferential does not reduce to the gradient when f is continuously differentiable. As a result, Algorithm 3 may behave differently in points $x \in \mathbb{R}^n$ where

$$\max\{\|\nabla f_i(x) - \nabla f_i(y)\| : y \in B_\varepsilon(x),\ i \in \{1, \dots, k\}\}$$

is large. (If f is twice differentiable, this can obviously be characterized in terms of second order derivatives.) Nevertheless, if ε is chosen small enough, this difference can be neglected. Thus, in the following, we will focus on the behavior with respect to the nonsmoothness of f.

To show the typical behavior of Algorithm 3, we consider the objective function $f : \mathbb{R}^2 \to \mathbb{R}^2$,

$$x \mapsto \begin{pmatrix} \max\{x_1^2 + (x_2 - 1)^2 + x_2 - 1, -x_1^2 - (x_2 - 1)^2 + x_2 + 1\} \\ -x_1 + 2(x_1^2 + x_2^2 - 1) + 1.75|x_1^2 + x_2^2 - 1| \end{pmatrix} \tag{16}$$

from [18] (combining *Crescent* from [30] and *Mifflin 2* from [31]). The set of nondifferentiable points is $\Omega_f = S^1 \cup (S^1 + (0, 1)^\top)$. We consider the starting points

$$x^1 = (0, -0.3)^\top, \quad x^2 = (0.6, 1.0)^\top, \quad x^3 = (-1, -0.2)^\top,$$

the tolerances $\varepsilon = 10^{-3}$, $\delta = 10^{-3}$ and the Armijo parameters $c = 0.25$, $t_0 = 1$. The results are shown in Fig. 5.

We will briefly go over the result for each starting point:

– For x^1, the sequence moves through the smooth area like the steepest descent method until a point is found with a distance less or equal ε to the set of nondifferentiable points Ω_f. In that point, more than one ε-subgradient is required to obtain a sufficient approximation of the ε-subdifferentials. Since this part of Ω_f is Pareto optimal, no acceptable descent direction (cf. (8)) is found and the algorithm stops (in a (ε, δ)-critical point).

– For x^2, the sequence starts zigzagging around the non-optimal part of Ω_f, since the points are too far away from Ω_f for the algorithm to notice the nondifferentiability. When a point is found with distance less or equal ε to Ω_f, a better descent direction is found, breaking the zigzagging motion.
– For x^3, the sequence has a similar zigzagging motion to the previous case. The difference is that this time, the sequence moves along Ω_f until a Pareto optimal point in Ω_f is found.

As described above, the zigzagging behavior when starting in x^2 is caused by the fact that ε was too small for the method to notice the nondifferentiability. To circumvent problems like this and quickly move through problematic areas, it is possible to apply Algorithm 3 consecutively with decreasing values of ε. The result is Algorithm 4. (A similar idea was implemented in [12].)

Algorithm 4 ε-decreasing nonsmooth descent method

```
Require: Initial point $x_1 \in \mathbb{R}^n$, tolerances $\delta > 0, \varepsilon_1 > ... > \varepsilon_K > 0$, Armijo parameters $c \in ]0, 1[, t_0 > 0$.

1: Set $y_1 = x_1$ and $i = 1$.
2: while $i \leq K$ do
3:     Apply Algorithm 3 with initial point $y_i$ and tolerance $\varepsilon = \varepsilon_i$. Let $y_{i+1}$ be the final element in the
       generated sequence.
4:     Set $i = i + 1$.
5: end while
```

Remark 4.1 In addition to a decreasing sequence of $(\varepsilon_i)_i$, it would be possible to choose a decreasing sequence of $(\delta_i)_i$ in Algorithm 4. A similar strategy was used in [12]. For $\varepsilon_i \to 0$ and $\delta_i \to 0$, convergence of the resulting sequence $(y_i)_i$ to critical points was reported. Since our methods can be seen as a generalization of the method from [12], this could be a way to obtain convergence to Pareto critical points also in our case, and the analysis of this is a promising direction for future work.

4.2 Comparison to the Multiobjective Proximal Bundle Method

We will now compare Algorithms 3 and 4 to the *multiobjective proximal bundle method* (MPB) by Mäkelä, Karmitsa and Wilppu from [18] (see also [20]). As test problems, we consider the 18 MOPs in Table 1, which are created on the basis of the scalar problems from [18]. Problems 1 to 15 are convex (and were also considered in [32]) and problems 16 to 18 are nonconvex. Since the expressions of the objective functions are relatively basic, we are able to differentiate all test problems by hand to obtain explicit formulas for the subgradients. For each test problem, we choose a 10×10 grid of 100 starting points in the corresponding area given in Table 1.

For the MPB, we use the Fortran implementation from [20] with the default parameters. For Algorithm 3, we use the parameters $\varepsilon = 10^{-3}$, $\delta = 10^{-3}$, $c = 0.25$ and

Table 1 Test problems (using objectives from [18])

Nr.	f_i	Area	Nr.	f_i	Area
1.	CB3, DEM	$[-3, 3]^2$	10.	QL, LQ	$[-3, 3]^2$
2.	CB3, QL	$[-3, 3]^2$	11.	QL, Mifflin 1	$[-3, 3]^2$
3.	CB3, LQ	$[0.5, 1.5]^2$	12.	QL, Wolfe	$[-3, 3]^2$
4.	CB3, Mifflin 1	$[-3, 3]^2$	13.	LQ, Mifflin 1	$[0.5, 1.5] \times [-0.5, 1]$
5.	CB3, Wolfe	$[-3, 3]^2$	14.	LQ, Wolfe	$[-3, 3]^2$
6.	DEM, QL	$[-3, 3]^2$	15.	Mifflin 1, Wolfe	$[-3, 3]^2$
7.	DEM, LQ	$[-3, 3]^2$	16.	Crescent, Mifflin 2	$[-0.5, 1.5]^2$
8.	DEM, Mifflin 1	$[-3, 3]^2$	17.	Mifflin 2, WF	$[-3, 3]^2$
9.	DEM, Wolfe	$[-3, 3]^2$	18.	Mifflin 2, SPIRAL	$[-3, 3]^2$

$t_0 = \max\{\|v_j\|^{-1}, 1\}$ (i.e., the initial step size t_0 is chosen depending on the norm of the descent direction v_j in the current iteration). For Algorithm 4, we additionally use $\varepsilon_1 = 10^{-1}$, $\varepsilon_2 = 10^{-2}$, $\varepsilon_3 = 10^{-3}$. By this choice of parameters, all three methods produce results of similar approximation quality in terms of their spread along and individual distance to the Pareto sets.

To compare the performance of the three methods, we count the number of evaluations of objectives f_i, their subgradients $\xi \in \partial f_i$ and the number of iterations (i.e., descent steps) needed. (This means that one call of f will account for k evaluations of objectives.) Since the MPB always evaluates all objectives and subgradients in a point, the value for the objectives and the subgradients are the same here. The results are shown in Table 2 and are discussed in the following.

- *Function Evaluations* When considering the number of function evaluations, we see that Algorithm 3 needs almost 12 times and Algorithm 4 needs almost 4 times as many evaluations as the MPB. In our methods, these evaluations are used to check whether a descent direction is acceptable (cf. (8)) and for the computation of the Armijo step length. One reason for the larger average amount is the fact that unlike the MPB, our methods are autonomous in the sense that they do not reuse information from previous iterations, so some information is potentially gathered multiple times. Additionally, the step length we use is fairly simple, so it might be possible to lower the number of evaluations by using a more sophisticated step length (see [33], Chapter 3.5, for an overview of techniques for the smooth, single-objective case). When comparing our methods to each other, we see that Algorithm 4 is a lot more efficient than Algorithm 3 when the number of evaluations is high and is slightly less efficient when the number of evaluations is low. The reason for this is that for some problems, some of the iterations of Algorithm 4 will be redundant, because the $(\varepsilon_{i-1}, \delta)$-critical point of the previous iteration is already (ε_i, δ)-critical.
- *Subgradient evaluations* For the subgradient evaluations, we see that the MPB is slightly superior to our methods on problems 3, 5 and 16, but inferior on the rest. Overall, Algorithm 3 needs 36.4% more and Algorithm 4 needs 20.3% fewer

Table 2 Performance of MPB, Algorithms 3 and 4 for the test problems in Table 1 for 100 starting points

	$\# f_i$			$\#\partial f_i$			# Iter		
Nr.	MPB	Alg.3	Alg.4	MPB	Alg.3	Alg.4	MPB	Alg.3	Alg.4
1.	**1780**	6924	7801	1780	**1102**	1751	761	**492**	695
2.	**2522**	14,688	12,263	2522	**1906**	2351	1151	**842**	914
3.	**880**	5625	6447	**880**	921	1534	**340**	448	662
4.	**4416**	103,826	17,664	4416	11,774	**3415**	1832	4644	**1242**
5.	**2956**	30,457	16,877	**2956**	3479	3037	1377	1616	**1161**
6.	**1640**	8357	8684	1640	**1209**	1802	706	**552**	736
7.	**1702**	8736	8483	1702	**1307**	1832	723	**595**	739
8.	**4226**	8283	8620	4226	**1318**	1914	1204	**582**	759
9.	**1828**	8201	8794	1828	**1194**	1805	793	**536**	732
10.	**1782**	6799	7201	1782	**1101**	1722	684	**543**	733
11.	**4426**	52,096	17,594	4426	6311	**3189**	1964	2442	**1206**
12.	**2482**	15,146	12,446	2482	**1992**	2401	1140	**967**	1010
13.	**2662**	36,570	9513	2662	4958	**2247**	1221	1692	**787**
14.	**4264**	95,303	12,227	4264	9524	**2571**	1774	4379	**921**
15.	**3594**	85,936	15,669	3594	9329	**3124**	1444	3963	**1125**
16.	**2206**	20,372	11,094	**2206**	2596	2400	**884**	1194	947
17.	**2388**	7920	5852	2388	**1272**	1556	868	**626**	706
18.	**11,430**	166,707	31,528	11,430	16,676	**6902**	2789	8291	**2412**
Avg.	**3177**	37,886	12,153	3177	4332	**2531**	1203	1911	**972**
%	100	1192.5	382.5	100	136.4	79.7	100	158.9	80.8

In each row, the best performance for each criterion is written in bold
Shown is the number of function evaluations ($\# f_i$), subgradient evaluations ($\#\partial f_i$) and iterations (# Iter). The last two rows contain the rounded average over the respective column and the percentage compared to the MPB result

evaluations than the MPB. Regarding the comparison of Algorithms 3 and 4, we observe the same pattern as for the function evaluations.

– *Iterations* For the number of iterations, besides problem 5, we see the exact same pattern as for the number of subgradient evaluations, with Algorithm 3 needing 58.9% more and Algorithm 4 needing 19.2% fewer iterations than the MPB. Note that the MPB can perform *null steps*, which are iterations where only the bundle is enriched, while the current point in the descent sequence stays the same.

For our set of test problems, this leads us to the overall conclusion that in terms of function evaluations, the MPB seems to be superior to our methods, while in terms of subgradient evaluations, our methods seem to be (almost always) more efficient. Furthermore, as also discussed in Chapter 1 of [11] for the single-objective case, the implementation of bundle methods like the MPB (for the general nonconvex case) is somewhat complicated. The only publicly available implementation that we are aware of is the Fortran implementation from [20]. Compared to this, our method can be implemented relatively quickly.

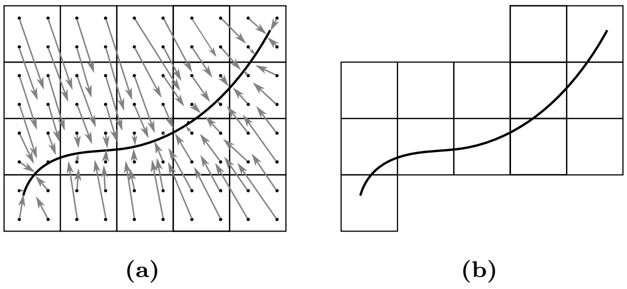

Fig. 6 Subdivision algorithm. **a** Applying g to a set of sample points. **b** Selection step, where boxes with an empty intersection with the image of g are removed

4.3 Globalization Using a Subdivision Algorithm

Note that so far, we have method (Algorithm 3) where we can put in some initial point from $\mathbb{R}^n$ and obtain a single (ε, δ)-critical point (close to an actual Pareto optimal point) as a result. To compute an approximation of the entire Pareto set, the straightforward approach to extend our method would be to just take a large set of well-spread initial points and apply our method to each of them. But since there is no guarantee that this actually yields a good pointwise discretization of the Pareto set, we instead combine our method with the *subdivision algorithm* which was developed for smooth problems in [29]. As this only requires minor adjustments for the nonsmooth case, we will only sketch the method here and refer to [29] for the details.

The idea is to interpret the nonsmooth descent method as a discrete dynamical system

$$x_{j+1} = g(x_j), \quad j = 0, 1, 2, \dots, \quad x_0 \in \mathbb{R}^n, \tag{17}$$

where $g : \mathbb{R}^n \to \mathbb{R}^n$ is the map that applies one iteration of Algorithm 3 to a point in $\mathbb{R}^n$. (For the sake of brevity, we have omitted the rest of the input of the algorithm here.) Since no information is carried over between iterations of the algorithm, the trajectory (i.e., the sequence) generated by the system (17) is the same as the one generated by Algorithm 3. In particular, this means that the Pareto set of the MOP is contained in the set of fixed points of the system (17). Thus, the subdivision algorithm (which was originally designed to compute attractors of dynamical systems) can be used to compute (a superset of) the Pareto set.

The subdivision algorithm starts with a large hypercube (or *box*) in $\mathbb{R}^n$ that contains the Pareto set and mainly consists of two steps:

1. *Subdivision* Divide each box in the current set of boxes into smaller boxes.
2. *Selection* Compute the image of the union of the current set of boxes under g and remove all boxes that have an empty intersection with this image. Go to step 1.

In practice, we realize step 1 by evenly dividing each box into 2^n smaller boxes and step 2 by using the image of a set of sample points. The algorithm is visualized in Fig. 6.

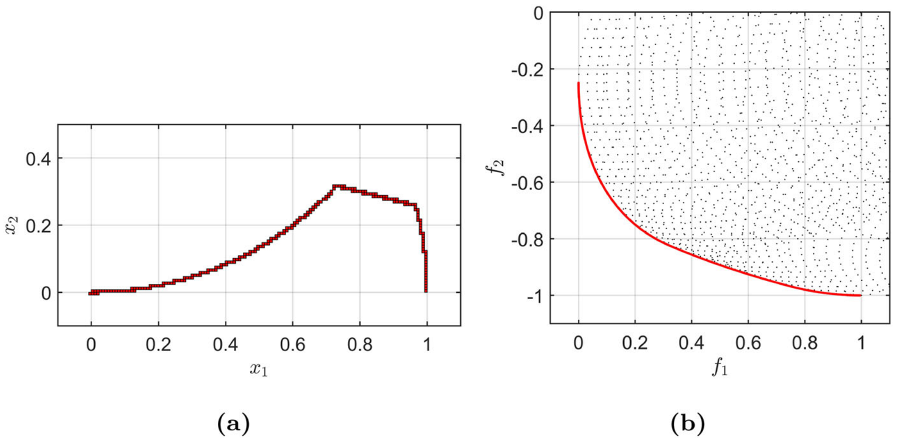

Fig. 7 **a** Result of the subdivision algorithm applied to problem 16 from Table 1. **b** Corresponding approximation of the Pareto front (red) and a pointwise discretization of the image of f (black)

Unfortunately, the convergence results of the subdivision algorithm only apply if g is a diffeomorphism. If the objective function f is smooth, then the descent direction is at least continuous (cf. [4]) and the resulting dynamical system g, while not being a diffeomorphism, still behaves well enough for the subdivision algorithm to work. If f is nonsmooth, then our descent direction is inherently discontinuous close to the nonsmooth points. Thus, the subdivision algorithm applied to (17) will (usually) fail to work. In practice, we were able to solve this issue by applying multiple iterations of Algorithm 3 in g at once instead of just one. Roughly speaking, this smoothes g by pushing the influence of the discontinuity further away from the Pareto set and was sufficient for convergence (in our tests).

Figure 7 shows the result of the subdivision algorithm for problem 16 from Table 1. Here, we used 15 iterations of Algorithm 3 in g, $[-0.5, 1.5]^2$ as the starting box and applied 8 iterations of the subdivision algorithm. For the approximation of the Pareto front (i.e., the image of the Pareto set), we evaluated f in all points of the image of g of the last selection step in the subdivision algorithm. We see that the algorithm produced a tight approximation of the Pareto set.

4.4 Application to Sparse Optimization

The globalized version of our method from Sect. 4.3 can be used to solve sparse (single-objective) optimization problems with an ℓ_1 penalty term, such as compressed sensing [22] or the sparse identification of nonlinear dynamics [23]. Abstractly, these problems can be denoted as

$$\min_{x \in \mathbb{R}^n} G(x) + \lambda \|x\|_1, \tag{18}$$

where $G : \mathbb{R}^n \to \mathbb{R}$ is a locally Lipschitz continuous function and $\lambda \geq 0$ is some weighting parameter. But as an alternative to the classical problem (18), sparse opti-

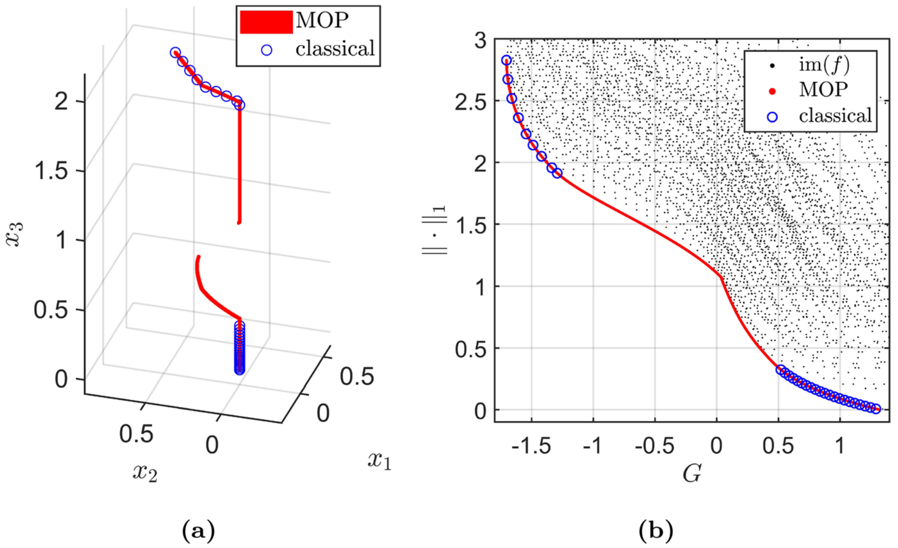

Fig. 8 **a** Result of the subdivision algorithm applied to the MOP (19) (red) and multiple solutions of the classical problem (18) (blue) for Example 4.1. **b** Corresponding approximation of the Pareto front (red), the image of the classical solutions under f (blue) and a pointwise discretization of the image of f (black)

mization can also be understood as solving the MOP

$$\min_{x\in\mathbb{R}^n} f(x) \quad \text{with} \quad f(x) = \begin{pmatrix} G(x) \\ \|x\|_1 \end{pmatrix}. \tag{19}$$

Note that due to the ℓ_1-norm, this MOP is intrinsically nonsmooth (even if G is smooth). It is easy to see that (18) is a scalarization of the MOP (19), known as the *weighting method* [3]. It is well known that every solution of the weighting method is a Pareto optimal point of the corresponding MOP, but the opposite only holds if G is convex. So in the general, nonconvex case, solving problem (18) with varying λ will only yield a subset of the solution to the MOP (19). Furthermore, (19) does not depend on any parameter, so in contrast to (18) it only has to be solved once.

Example 4.1 Consider the nonconvex function

$$G : \mathbb{R}^3 \to \mathbb{R}, \quad x \mapsto \left(x_1 - \frac{1}{4}\right)^2 + \left(x_2 - \frac{1}{2}\right)^2 + (x_3 - 1)^4 - \frac{1}{2}\left(x_3 - \frac{1}{4}\right)^3.$$

Figure 8 shows the result of the subdivision algorithm applied to the MOP (19), both in variable and image space. Additionally, the solution of (18) (computed via the function *patternsearch* in MATLAB) is shown for 30 equidistant $\lambda \in [0, 4]$. As expected, all solutions of (18) are elements of the Pareto set. But since G is nonconvex, there is a large part of the Pareto set which cannot be obtained via (18).

5 Conclusions

In this article, we have developed a new descent method for locally Lipschitz continuous multiobjective optimization problems, which is based on the efficient approximation of the Clarke subdifferentials of the objective functions by the Goldstein ε-subdifferentials as in [12]. To avoid the computation of entire subdifferentials, we presented a method to obtain a finite set of ε-subgradients which is sufficient to compute a descent direction. The idea is to start with a rough approximation by only a few subgradients and then systematically enrich the approximation with new subgradients until a direction of sufficient descent is found. Together with an Armijo step length, we obtain a descent method that converges to points which satisfy a necessary condition for Pareto optimality. On a test set with 18 problems, a comparison showed that the multiobjective proximal bundle method from [18] is superior in terms of objective function evaluations, but our method requires less subgradient evaluations and iterations. Combination with the subdivision algorithm from [29] resulted in a method to compute entire Pareto sets of nonsmooth MOP. Finally, we showed that our method can be useful for solving sparse optimization problems, which occur in applications such as compressed sensing [22] or the sparse identification of nonlinear dynamics [23].

For future work, we believe that the extension to constrained MOPs can be achieved in a straightforward manner by adding constraints to the problem (7) to ensure that the descent direction maintains the feasibility of the descent sequence (similar to [6] for smooth problems). Additionally, in [34], the classical gradient sampling method for scalar nonsmooth optimization was generalized by allowing variable norms in the direction finding problem, increasing its efficiency. We expect that a similar generalization can be performed for problem (7), which potentially yields a similar increase in efficiency. Additional potential for increased performance lies in more advanced step length schemes as well as descent directions with memory (for instance, conjugate gradient like). On a related note, it would make sense to compare the performance of our method not only to the MPB, but also to proximal point methods for MOPs [16] and evolutionary approaches [1,2]. Furthermore, it might be possible to extend our method to infinite-dimensional nonsmooth MOPs [35,36]. Finally, for nonsmooth MOPs with large numbers of objectives (which are sometimes referred to as *many-objective optimization problems*), we believe that considering subsets of objectives is a very promising and efficient approach (cf. [37] for smooth problems). However, theoretical advances are required for locally Lipschitz continuous problems.

Acknowledgements This research was funded by the DFG Priority Programme 1962 "Non-smooth and Complementarity-based Distributed Parameter Systems."

Funding Open Access funding enabled and organized by Projekt DEAL.

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

References

1. Deb, K.: Multi-objective Optimization using Evolutionary Algorithms, vol. 16. Wiley, Hoboken (2001)
2. 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)
3. Miettinen, K.: Nonlinear Multiobjective Optimization. Springer, New York (1998)
4. Fliege, J., Svaiter, B.F.: Steepest descent methods for multicriteria optimization. Math. Methods Oper. Res. **51**(3), 479–494 (2000)
5. Schäffler, S., Schultz, R., Weinzierl, K.: Stochastic method for the solution of unconstrained vector optimization problems. J. Optim. Theory Appl. **114**(1), 209–222 (2002)
6. Gebken, B., Peitz, S., Dellnitz, M.: A descent method for equality and inequality constrained multiobjective optimization problems. In: Trujillo, L., Schütze, O., Maldonado, Y., Valle, P. (eds.) Numerical and Evolutionary Optimization—NEO 2017, pp. 29–61. Springer, Cham (2019)
7. Fliege, J., Graa, L., Svaiter, B.F.: Newton's method for multiobjective optimization. SIAM J. Optim. **20**, 602–626 (2008)
8. Wang, J., Hu, Y., Yu, C.K.W., Li, C., Yang, X.: Extended Newton methods for multiobjective optimization: majorizing function technique and convergence analysis. SIAM J. Optim. **29**(3), 2388–2421 (2019). https://doi.org/10.1137/18m1191737
9. Shor, N.: Minimization Methods for Non-differentiable Function. Springer, Berlin (1985)
10. Kiwiel, K.C.: Proximity control in Bundle methods for convex nondifferentiable minimization. Math. Program. **46**, 105–122 (1990)
11. Burke, J., Lewis, A., Overton, M.: A robust gradient sampling algorithm for nonsmooth, nonconvex optimization. SIAM J. Optim. **15**, 751–779 (2005)
12. Mahdavi-Amiri, N., Yousefpour, R.: An effective nonsmooth optimization algorithm for locally Lipschitz functions. J. Optim. Theory Appl. **155**(1), 180–195 (2012)
13. Attouch, H., Garrigos, G., Goudou, X.: A dynamic gradient approach to Pareto optimization with nonsmooth convex objective functions. J. Math. Anal. Appl. **422**(1), 741–771 (2015)
14. Bello-Cruz, Y.: A subgradient method for vector optimization problems. SIAM J. Optim. **23**, 2169–2182 (2013)
15. Cruz Neto, J., Silva, G., Ferreira, O., Lopes, J.: A subgradient method for multiobjective optimization. Comput. Optim. Appl. **54**, 461–472 (2013)
16. Bonnel, H., Iusem, A.N., Svaiter, B.F.: Proximal Methods in Vector Optimization. SIAM J. Optim. **15**(4), 953–970 (2005). https://doi.org/10.1137/s1052623403429093
17. Grad, S.M.: A survey on proximal point type algorithms for solving vector optimization problems. In: Bauschke, H.H., Burachik, R.S., Luke, D.R. (eds.) Splitting Algorithms, Modern Operator Theory, and Applications, pp. 269–308. Springer, Berlin (2019). https://doi.org/10.1007/978-3-030-25939-6_11
18. Mäkelä, M.M., Karmitsa, N., Wilppu, O.: Multiobjective proximal bundle method for nonsmooth optimization. TUCS technical report No 1120, Turku Centre for Computer Science, Turku (2014)
19. Kiwiel, K.C.: A descent method for nonsmooth convex multiobjective minimization. Large Scale Syst. **8**(2), 119–129 (1985)
20. Mäkelä, M.M.: Multiobjective proximal bundle method for nonconvex nonsmooth optimization: fortran subroutine MPBNGC 2.0. Rep. Depart. Math. Inf. Technol. Ser. B. Sci. Comput. B **13**, 2003 (2003)
21. Goldstein, A.: Optimization of Lipschitz continuous functions. Math. Program. **13**, 14–22 (1977)
22. Chen, S.S., Donoho, D.L., Saunders, M.A.: Atomic decomposition by basis pursuit. SIAM Rev. **43**(1), 129–159 (2001)
23. Brunton, S.L., Proctor, J.L., Kutz, J.N.: Discovering governing equations from data by sparse identification of nonlinear dynamical systems. In: Proceedings of the National Academy of Sciences, pp. 3932–3937 (2016)

24. Clarke, F.: Optimization and Nonsmooth Analysis. Society for Industrial and Applied Mathematics (1983)
25. Mäkelä, M.M., Eronen, V.P., Karmitsa, N.: On Nonsmooth Multiobjective Optimality Conditions with Generalized Convexities, pp. 333–357. Springer, New York (2014)
26. Cheney, W., Goldstein, A.A.: Proximity maps for convex sets. Proc. Am. Math. Soc. **10**(3), 448–450 (1959)
27. Helou, E.S., Santos, S.A., Simes, L.E.A.: On the differentiability check in gradient sampling methods. Optim. Methods Softw. **31**(5), 983–1007 (2016)
28. Kiwiel, K.C.: A nonderivative version of the gradient sampling algorithm for nonsmooth nonconvex optimization. SIAM J. Optim. **20**(4), 1983–1994 (2010)
29. Dellnitz, M., Schütze, O., Hestermeyer, T.: Covering Pareto sets by multilevel subdivision techniques. J. Optim. Theory Appl. **124**(1), 113–136 (2005)
30. Kiwiel, K.C.: Methods of Descent for Nondifferentiable Optimization. Springer, Berlin (1985)
31. Mäkelä, M.M., Neittaanmäki, P.: Nonsmooth Optimization: Analysis and Algorithms with Applications to Optimal Control. World Scientific, Cambridge (1992)
32. Montonen, O., Karmitsa, N., Mäkelä, M.M.: Multiple subgradient descent bundle method for convex nonsmooth multiobjective optimization. Optimization **67**(1), 139–158 (2018)
33. Nocedal, J., Wright, S.: Numerical Optimization. Springer, New York (2006). https://doi.org/10.1007/978-0-387-40065-5
34. Curtis, F.E., Que, X.: An adaptive gradient sampling algorithm for non-smooth optimization. Optim. Methods Softw. **28**(6), 1302–1324 (2013)
35. Mordukhovich, B.: Multiobjective optimization problems with equilibrium constraints. Optim. Methods Softw. **117**, 331–354 (2008)
36. Christof, C., Müller, G.: Multiobjective Optimal Control of a Non-smooth Semilinear Elliptic Partial Differential Equation. European Series in Applied and Industrial Mathematics (ESAIM): Control, Optimisation and Calculus of Variations (2020)
37. Gebken, B., Peitz, S., Dellnitz, M.: On the hierarchical structure of Pareto critical sets. J. Global Optim. **73**(4), 891–913 (2019)