Optimization Letters (2024) 18:57–71
https://doi.org/10.1007/s11590-023-02042-4

ORIGINAL PAPER

A clustering heuristic to improve a derivative-free algorithm for nonsmooth optimization

Manlio Gaudioso[1] · **Giampaolo Liuzzi**[2] · **Stefano Lucidi**[2]

Received: 25 November 2022 / Accepted: 30 June 2023 / Published online: 13 July 2023

Abstract
In this paper we propose an heuristic to improve the performances of the recently proposed derivative-free method for nonsmooth optimization CS-DFN. The heuristic is based on a clustering-type technique to compute an estimate of Clarke's generalized gradient of the objective function, obtained via calculation of the (approximate) directional derivative along a certain set of directions. A search direction is then calculated by applying a nonsmooth Newton-type approach. As such, this direction (as it is shown by the numerical experiments) is a good descent direction for the objective function. We report some numerical results and comparison with the original CS-DFN method to show the utility of the proposed improvement on a set of well-known test problems.

Keywords Nonsmooth optimization · Derivative-free methods · CS-DFN

1 Introduction

We consider the following unconstrained minimization problem

$$\min_{x \in \Re^n} f(x). \tag{1}$$

✉ Giampaolo Liuzzi
liuzzi@diag.uniroma1.it

Manlio Gaudioso
manlio.gaudioso@unical.it

Stefano Lucidi
lucidi@diag.uniroma1.it

[1] Dipartimento di Ingegneria Informatica, Modellistica, Elettronica e Sistemistica, Universitá della Calabria, 87030 Rende, CS, Italy

[2] Dipartimento di Ingegneria Informatica Automatica e Gestionale, "Sapienza" Università di Roma, Via Ariosto 25, 00185 Rome, Italy

We assume that the objective function f, (though nonsmooth) is Lipschitz continuous and that first-order information is unavailable or impractical to obtain. We require the following assumption.

Assumption 1 The function $f(x)$ is coercive, i.e. every level set is compact.

Useless to say that there is plenty of problems with the above features, especially coming from the engineering context. In the literature, many approaches have been proposed to tackle the nonsmooth problem (1) in the derivative-free framework. They can be roughly subdivided into two main classes: direct-type algorithms and model-based algorithms.

- *Direct-type methods*. The algorithms belonging to this class make use of suitable sampling of the objective function. They occasionally can heuristically use modeling techniques, but the convergence theory hinges on the sampling technique. In this class of methods, we cite the mesh adaptive direct search algorithm implemented in the software package NOMAD [2, 10], the linesearch derivative-free algorithm CS-DFN proposed in [6] and the discrete gradient method [3].
- *Model-based methods*. This class comprises all those algorithms whose convergence is based on the strategy used to build the approximating models. Within this class we can surely cite the recent trust-region derivative-free method proposed in [11].

In the relatively recent paper [6], a method for optimization of nonsmooth black-box problems has been proposed, namely CS-DFN. CS-DFN is able to solve problems more general than problem (1) above since it can handle also nonlinear and bound constraints. It is based on a penalization approach, namely the nonlinear constraints are penalized by an exact penalization mechanism whereas (possible) bound constraints on the variables are handled explicitly.

In this paper, we propose an improvement of CS-DFN by incorporating into its main algorithmic scheme a clustering heuristic to compute efficient search directions. Starting from an approximation of the directional derivatives along a certain set of directions, we construct a polyhedral approximation of the subdifferential which in turn is used to calculate a search direction in the steepest descent fashion. Along such direction we implement a linesearch procedure with extrapolation just like the one adopted by CS-DFN to explore its directions.

To asses the potentialities of the proposed improvement, we carry out an experimentation and comparison of CS-DFN with and without the proposed heuristic. The results, in our opinion, clearly show the advantages of the improved method over the original one.

The paper is organized as follows. In Sect. 2 we extend to a nonsmooth setting the steepest descent direction and a kind of Newton-type directions. In Sect. 3 we propose an heuristic to compute possibly efficient directions in a derivative-free context. In section 4 we describe an improved version of the CS-DFN algorithm which is obtained by suitably employing the improved directions just described. In Sect. 5 we report the

results of a numerical comparison between CS-DFN and the proposed improved version on a set of well-known test problems. Finally, Sect. 6 is devoted to some discussion and conclusions.

1.1 Definitions and notations

Definition 1 Given a point $x \in \Re^n$ and a direction $d \in \Re^n$, the Clarke directional derivative of f at x along d is defined as [4]

$$f^\circ(x;d) = \limsup_{y \to x, t \downarrow 0} \frac{f(y+td) - f(y)}{t}.$$

Moreover the Clarke generalized gradient (or subdifferential) $\partial_C f(x)$ is defined as

$$\partial_C f(x) = conv\{s : s \in \Re^n, \nabla f(x_k) \to s,\ x_k \to x,\ x_k \notin \Omega_f\},$$

Ω_f being the set (of zero measure) where f is not differentiable.

The following property holds:

$$f^\circ(x;d) = \max_{s \in \partial_C f(x)} s^\top d \tag{2}$$

Definition 2 A point $x^* \in \Re^n$ is Clarke stationary for Problem (1) when $f^\circ(x^*, d) \geq 0$, for all $d \in \Re^n$.

In the following, we denote by e_i, $i = 1, \ldots, n$, the i-th column of the canonical basis in $\Re^n$ and by e a vector of all ones of appropriate dimensions.

2 Descent type directions

In the context of nonsmooth optimization, efficient search directions can be computed by using the information provided by the subdifferential of the objective function. In the following subsections, we describe how such directions can be obtained.

2.1 Steepest descent direction g_k^S

In this subsection we recall a classic approach [14] to compute a generalization to nonsmooth functions of the steepest descent direction for continuously differentiable functions.

Let us consider the vector which minimizes the following "first order-type" model of the objective function.

$$\min\ f(x_k) + f^\circ(x_k; d) + \tfrac{1}{2}\|d\|^2 \tag{3}$$

Note that, in the case of continuously differentiable functions, we have that $f^\circ(x_k;d) = \nabla f(x_k)^T d_k$ and that the solution of Problem (3) is given by $d^* = -\nabla f(x_k)$.

For nonsmooth functions, standard results [14] lead to the following proposition.

Proposition 1 *Let d^S be the solution of Problem (3). Then*

(i) *The vector d^* is given by*

$$d^S = -g_k^S$$

where

$$g_k^S = \begin{array}{l} \operatorname{argmin} \|\xi\|^2 \\ s.t.\ \xi \in \partial f(x_k) \end{array} \tag{4}$$

(ii) *The vector d^S satisfies $f^\circ(x_k; -g_k^S) = -\|g_k^S\|^2$.*

(iii) *For any $\gamma \in (0,1)$ there exists a $\bar{\alpha} > 0$ such that*

$$f(x_k - \alpha g_k^S) \leq f(x_k) - \alpha\gamma\|g_k^S\|^2$$

with $\alpha \in (0, \bar{\alpha}]$

The above d_k^S direction is a first-order direction which (closely) resembles the steepest-descent direction for continuously differentiable case.

2.2 Newton-type direction d_k^N

In the nonsmooth case, obtaining a Newton-type direction is much more involved than in the differentiable case. In the latter case it suffices to pre-multiply the anti-gradient by the Hessian of the objective function. In the nonsmooth case instead of simply pre-multiplying direction g_k^S by any positive definite matrix, we resort to minimizing the following "second order-type" model.

$$\min\ f(x_k) + f^\circ(x_k;d) + \frac{1}{2} d^T B_k d \tag{5}$$

where B_k is a positive definite matrix. Let us call the solution of problem (5) d_k^N.

For problem (5) the following proposition can be proved.

Proposition 2 *Let B_k be positive definite and assume d_k^N be the solution of Problem (5). Then*

(i) *The vector d_k^N is given by*

$$d_k^N = -B_k^{-1} g_k^N$$

where

$$g_k^N = \begin{array}{l} \operatorname{argmin}\ \xi^T B_k^{-1} \xi \\ s.t.\ \xi \in \partial f(x_k) \end{array} \tag{6}$$

(ii) *The vector* d_k^N *satisfies* $f^\circ(x_k; d_k^N) = -(g_k^N)^T B_k^{-1} g_k^N = (g_k^N)^T d_k^N$.

(iii) *For any* $\gamma \in (0,1)$ *there exists a* $\bar{\alpha} > 0$ *such that*

$$f(x_k - \alpha B_k^{-1} g_k^N) \leq f(x_k) - \alpha\gamma (g_k^N)^T B_k^{-1} g_k^N$$

with $\alpha \in (0, \bar{\alpha}]$

Proof By repeating the similar arguments of proof of Theorem 5.2.8 in [14] we have that function $\phi(d) = f^\circ(x_k; d) + \frac{1}{2} d^T B_k d$ is strictly convex. Therefore Problem (5) has a unique minimizer d^* such that:

$$0 \in \partial f^\circ(x_k; d^*) + B_k d^*. \tag{7}$$

Recalling Lemma 5.2.7 of [14] we have:

$$\partial f^\circ(x_k; d^*) \subseteq \left\{ \xi \in \partial f(x_k) : \xi^T d^* = f^\circ(x_k; d^*) \right\} \tag{8}$$

The relations (7) and (8) imply that a vector g_k^N exists such that:

$$\begin{aligned} g_k^N &= -B_k d^*, \\ (g_k^N)^T d^* &= f^\circ(x_k; d^*). \end{aligned}$$

and, hence,

$$-(g_k^N)^T B_k^{-1} g_k^N = f^\circ(x_k; -B_k^{-1} g_k^N) \tag{9}$$

which proves point (ii) by setting $d_k^N = d^*$.

Now the definition of $f^\circ(x_k; -B_k^{-1} g_k^N)$ and (9) give:

$$f^\circ(x_k; -B_k^{-1} g_k^N) = \max_{\xi \in \partial f(x_k)} \xi^T (-B_k^{-1} g_k^N) = -(g_k^N)^T B_k^{-1} g_k^N,$$

and

$$(g_k^N)^T B_k^{-1} g_k^N \leq (g_k^N)^T B_k^{-1} \xi, \qquad \text{for all} \qquad \xi \in \partial f(x_k), \tag{10}$$

which implies

$$(g_k^N)^T B_k^{-1}(\xi - g_k^N) \geq 0, \qquad \text{for all} \qquad \xi \in \partial f(x_k), \tag{11}$$

Therefore, (11) shows that the vector g_k^N is the unique solution of Problem 6.

Finally point (iii) again follows from definition of $f^\circ(x_k; -B_k^{-1} g_k^N)$ and (9).◁ □

3 An heuristic approach to define efficient directions

At the base of the proposed heuristics is the hypothesis that non-smoothness of the objective function is due to its *finite* max structure. Such hypothesis appear realistic as a wide range of nonsmooth optimization problems, coming from practical applications, are of the min max type. The essence of our method is the attempt to construct an approximation of the subdifferential by estimating a certain set of subgradients (the *generators* in the following) starting from an estimate of the directional derivatives along a sufficiently large sets of directions.

The only assumptions about function f are Lipschitz continuity and Assumption 1, nevertheless, drawing inspiration from the paper [13] (see also [1]), given points $y_j \in \Re^n$, $j = \{1, 2, \ldots, p\}$, sufficiently close to x, we approximate $f(x)$ by using the following piece-wise quadratic model function,

$$f^{\square}(x) = \max_{j=1,\ldots,p} \{q_j(x)\}$$

with

$$q_j(x) = f(y_j) + g_j^\top (x - y_j) + \frac{1}{2}(x - y_j)^\top H_j (x - y_j)$$

where $g_j \in \partial f(y_j)$ and $H_j = H(y_j)$, $j = 1, \ldots, p$. We remark that, while we assume that the model structure of f is a max of a finite number of functions, the number p of such functions is unknown and has to be estimated via a trial–and–error calculation process.

We can write,

$$\partial f^{\square}(x) = \partial \max_{j=1,\ldots,p} \{q_j(x)\} \subseteq conv\{g_j + H_j(x - y_j), j = 1, \ldots, p\} = C(x).$$

Furthermore, by assuming that $f(x) \approx f^{\square}(x)$, we have

$$f^\circ(x;d) = \max_{s \in \partial f(x)} d^\top s \approx \max_{s \in \partial f^{\square}(x)} d^\top s \leq \max_{s \in C(x)} d^\top s = d^\top (g_{\bar{i}} + H_{\bar{i}}(x - y_{\bar{i}})). \tag{12}$$

In the actual case, $C(x)$ is the convex hull of a given number of generator vectors v_j, $j = 1, \ldots, p$. We can try and estimate those generators by using the quantities computed by the algorithm.

More in particular, let x_k be the current iterate of the algorithm. Assume that a certain set of directions $d_i \in \Re^n$ and $\alpha_i > 0$, $i = 1, \ldots, r$, along with their respective stepsizes, are available. They can be either directions where failure in the descent has previously occurred or even predefined ones, e.g. the unit vectors ($d_i = \pm e_i$). Define

$$s_i = \frac{f(x_k + \alpha_i d_i) - f(x_k)}{\alpha_i} \approx f^\circ(x_k; d_i). \tag{13}$$

By using (12), for $i = 1, \ldots, r$,

$$f^\circ(x_k; d_i) \approx d_i^\top v_{j_i}, \quad \textit{for some} \quad j_i \in \{1, 2, \ldots, p\}.$$

It is then possible to compute estimates of the generators v_j, $j = 1, \ldots, p$, as those which provide the best approximation of the s_i's, hence we solve the problem

$$\min_{\hat{v}_1, \ldots, \hat{v}_p} \sum_{i=1}^{r} \min_{j=1,\ldots,p} \{(d_i^\top \hat{v}_j - s_i)^2\}. \tag{14}$$

The above problem is a hard, nonsmooth nonconvex problem of the clustering type. It can be put however in DC (Difference of Convex) form as in [9]. Since it has to be solved many times during the proposed algorithm, we prefer to resort, in our implementation, to a greedy heuristic of the k-means-type [7, 12, 16]. It works as follows. An initial set of p tentative generators is defined. Then each couple (d_i, s_i) is assigned to the generator which ensures the best approximation of s_i. Once the couples have been clustered, the generators are updated in a least-square fashion and the procedure is repeated.

Algorithm 1 k-means-type Algorithm (p, G)

1: **Data.** $p \in \mathbb{N}$, $G = \{(d_i, s_i), i = 1, \ldots, r\}$, $d_i \in \Re^n$, $s_i \in \Re$, $h_{\max} > 0$.

2: Set $I = \{1, \ldots, r\}$, $\hat{v}_j^{(0)} = 0$, $j = 1, \ldots, p$.
3: **for** $h = 0, \ldots, h_{\max}$ **do**
4: **for** $j = 1, \ldots, p$ **do**
5: Compute

$$I_j^{(h)} = \{i \in I : (d_i^\top \hat{v}_j^{(h)} - s_i)^2 \le (d_i^\top \hat{v}_\ell^{(h)} - s_i)^2, \ \ell \ne j\}$$
$$\hat{v}_j^{(h+1)} = \arg\min_{v \in \Re^n} \sum_{i \in I_j^{(h)}} (v^\top d_i - s_i)^2$$
$$\phi_j^{(h+1)} = \sum_{i \in I_j^{(h)}} ((\hat{v}_j^{(h+1)})^\top d_i - s_i)^2$$

6: **end for**
7: **end for**
8: **Return** $\hat{v}_j^{(h_{\max}+1)}$ and $\phi_j^{(h_{\max}+1)}$ for $j = 1, \ldots, p$.

Then, we can compute an estimate of direction d_k^N by solving problem (6) where $\partial f(x_k)$ is approximated by $conv(\hat{v}_i, \ldots, \hat{v}_p)$. More precisely, we define the following algorithm that computes a search direction.

Algorithm 2 Direction computation (B, G)

Data. $\epsilon > 0$, $G = \{(d_i, s_i), i = 1, \ldots, r\}$, $d_i \in \Re^n$, $s_i \in \Re$, $B \in \mathcal{S}^{n \times n}$, $B \succ 0$.
Set $\hat{d}^N = 0$
for $p = 2, \ldots, n$ **do**
  Compute $\hat{v}_1, \ldots, \hat{v}_p$ and $\phi_1, \ldots, \phi_p$ by k-means-type Algorithm (p, G), i.e. Algorithm 1.
  if $\sum_{i=1}^{p} \phi_i < \epsilon$ **or** $p = n$ **then**
    Compute $\hat{d}^N$ by

$$\hat{d}^N = \underset{\xi \in conv(\hat{v}_1, \ldots, \hat{v}_p)}{\arg\min} \xi^\top B^{-1} \xi.$$

  end if
end for

In the following we give an example of how the heuristic works.

Example 1 Consider the (convex) nonsmooth function *maxl* [13], defined as

$$f(x) = \max_{1 \le i \le n} |x_i|.$$

Take point $\bar{x}$, $\bar{x}_i = 1$, $i = 1, \ldots, n$, where f exhibits a kink and it is $f(\bar{x}) = 1$. Observe that none among the $2n$ (signed) coordinate directions $\pm e_i$ is a descent one at $\bar{x}$ (it is in fact $f^\circ(\bar{x}; -e_i) = 0$ and $f^\circ(\bar{x}; e_i) = 1$, $i = 1, \ldots, n$). Calculation of the $2n$ ratios s_i as in (13), along the directions e_i and $-e_i$ leads to $s_i = 1$ and $s_i = 0$, respectively, for $i = 1, \ldots, n$. It is easy to verify that, letting $p = n$ in Algorithm 1, an optimal solution to problem (14) is $\hat{v}_j = e_j$, $j = 1, \ldots, n$. Finally, solving

$$\bar{d} = - \underset{v \in conv\{\hat{v}_j,\, j=1,\ldots,n\}}{\arg\min} \|v\|$$

we obtain $\bar{d} = -\dfrac{e}{n}$, which is indeed a descent direction at $\bar{x}$.

4 The improved CS-DFN algorithm

This section is devoted to the definition of the improved version of algorithm CS-DFN which we call Fast-CS-DFN. The method is basically the CS-DFN Algorithm introduced in reference [6], a derivative-free linesearch-type algorithm for the minimization of black-box (possibly) nonsmooth functions. It works by performing derivative-free line searches along the coordinate directions and resorting to the use of a further search direction when the stepsizes used to explore the coordinate

directions are sufficiently small. The rationale behind this choice is connected with the observation that the coordinate directions might not be descent directions near a non-stationary point of non-smoothness. In such situations, a richer set of directions must be used to (at least asymptotically) be able to improve the non-stationary point. The convergence analysis of CS-DFN carried out in [6] hinges on the use of asymptotically dense sequences of search directions so that, at non-stationary points, for sufficiently large k a direction of descent is used.

The algorithm that we propose, namely Fast-CS-DFN, is a modification of CS-DFN. The relevant differences between the two methods are:

1. for the sake of simplicity, problem (1) is unconstrained; hence in Fast-CS-DFN no control to enforce feasibility with respect to the bound constraints is needed;
2. after the deployment of the direction d_k, Fast-CS-DFN makes use of Algorithm 2 to compute a direction that tries to exploits the information gathered during the optimization process to heuristically improve the last produced point.

The Fast-CS-DFN Algorithm is reported in Algorithm 3.

Algorithm 3 Algorithm Fast-CS-DFN

Input. $\theta \in (0,1)$, $\eta > 0$, $x_0 \in \Re^n$, $\tilde{\alpha}_0 > 0$, $\tilde{\alpha}_0^i > 0$, $d_0^i = e^i$, for $i = 1, \ldots, n$, $G_0 = \emptyset$, a sequence $\{d_k\}$ of search directions such that $\|d_k\| = 1$, for all k.
for $k = 0, 1, \ldots$ **do**
  Set $y_k^1 = x_k$, $G_k^1 = G_k$.
  for $i = 1, 2, \ldots, n$ **do**
    Compute α, d_{k+1}^i, G_k^{i+1} by the *Continuous Search*$(\tilde{\alpha}_k^i, y_k^i, d_k^i, G_k^i; \alpha, d_{k+1}^i)$.
    If $(\alpha = 0)$ **then** set $\alpha_k^i = 0$ and $\tilde{\alpha}_{k+1}^i = \theta \tilde{\alpha}_k^i$
    else set $\alpha_k^i = \alpha$ and $\tilde{\alpha}_{k+1}^i = \alpha$.
    Set $y_k^{i+1} = y_k^i + \alpha_k^i d_{k+1}^i$.
  end for
  if $\left(\max_{i=1,\ldots,n}\{\alpha_k^i, \tilde{\alpha}_k^i\} \leq \eta\right)$ **then**
    Compute α_k, $\tilde{d}_k$ and G_k^{n+2} by the *Continuous Search*$(\tilde{\alpha}_k, y_k^{n+1}, d_k, G_k^{n+1}; \alpha_k, \tilde{d}_k)$.
    If $(\alpha_k = 0)$ **then** $\tilde{\alpha}_{k+1} = \theta \tilde{\alpha}_k$ and $y_k^{n+2} = y_k^{n+1}$
    else $\tilde{\alpha}_{k+1} = \alpha_k$ and $y_k^{n+2} = y_k^{n+1} + \alpha_k \tilde{d}_k$.
    Build a symmetric matrix B_k
    Compute $\hat{d}_k^N$ by *Direction computation*(B_k, G_k^{n+2}), i.e. Algorithm 2.
    Compute $\check{\alpha}_k$, $\check{d}_k$ and $\bar{G}$ by the *Continuous Search*$(\tilde{\alpha}_k, y_k^{n+2}, \hat{d}_k, G_k^{n+2}; \check{\alpha}_k, \check{d}_k)$.
    Set $x_{k+1} = y_k^{n+2} + \check{\alpha}_k \check{d}_k$
    If $\bar{G} = \emptyset$ **then** set $G_{k+1} = \emptyset$ **else** set $G_{k+1} = G_k^{n+2}$
  else
    set $\tilde{\alpha}_{k+1} = \tilde{\alpha}_k$, $\alpha_k = \check{\alpha}_k = 0$ and $y_k^{n+2} = y_k^{n+1}$, $G_k^{n+2} = G_k^{n+1}$.
    Set $x_{k+1} = y_k^{n+2}$, $G_{k+1} = G_k^{n+2}$
  end if
end for
Output. The sequences $\{x_k\}$, $\{\alpha_k\}$, $\{\tilde{\alpha}_k\}$, $\{\alpha_k^i\}$ and $\{\tilde{\alpha}_k^i\}$, for $i = 1, \ldots, n$.

Some comments about Algorithm Fast-CS-DFN are in order.

1. Fast-CS-DFN except for steps 14–18 and for the mechanism used to produce G_{k+1} starting from G_k, exactly is the CS-DFN method as described in [6];
2. The new direction $\hat{d}_k^N$ is used when the stepsizes α_k^i and $\tilde{\alpha}_k^i$, $i = 1, \ldots, n$, are sufficiently small and after the deployment of the direction d_k;
3. The computation of the new direction $\hat{d}_k^N$ performed at step 15 hinges (a) on the matrix B_k and (b) on the set of couples G_k^{n+2}.

   (a) To build B_k, we maintain a set of points Y_k which is managed in just the same way as described in [6];
   (b) As for the set G_k^{n+2}, it stores information on the consecutive failures encountered up to the current point, i.e. in the deployment of the coordinate directions and the direction d_k. The direction d_k is the one (possibly) used at step 11 of Algorithm Fast-CS-DFN. Note also that set G_k^{n+1} is emptied every time a non-null step is computed by the algorithm along any direction;

4. The asymptotic convergence properties of Fast-CS-DFN are analogous to that of CS-DFN. The theoretical analysis follows quite easily from the results proved for CS-DFN in [6]. As it can be noted, the new iterate x_{k+1} defined by Algorithm Fast-CS-DFN is such that $f(x_{k+1}) \leq f(y_k^{n+2})$. In fact, when step 17 is executed $x_{k+1} = y_k^{n+2} + \check{\alpha}\check{d}_k$ and $f(x_{k+1}) \leq f(y_k^{n+2})$. When step 21 is executed, then $x_{k+1} = y_k^{n+2}$ and $f(x_{k+1}) = f(y_k^{n+2})$.

Algorithm 4 Continuous Search $(\tilde{\alpha}, y, p, G; \alpha, p^+)$

1: **Data**: $\gamma > 0$, $\delta \in (0, 1)$
2: Set $\alpha = \tilde{\alpha}$.
3: **If** $f(y + \alpha p) \leq f(y) - \gamma\alpha^2$ **then** set $p^+ = \quad p$ and go to Step 7.
4: **If** $f(y - \alpha p) \leq f(y) - \gamma\alpha^2$ **then** set $p^+ = -p$ and go to Step 7.
5: $G^+ \leftarrow G \cup \{(p, (f(y + \alpha p) - f(y))/\alpha), (-p, (f(y - \alpha p) - f(y))/\alpha)\}$
6: Set $\alpha = 0$, **return** α, $p^+ = p$ and G^+
7: Let $G^+ \leftarrow \emptyset$ and $\beta = \alpha/\delta$.
8: **If** $f(y + \beta p^+) > f(y) - \gamma\beta^2$ **return** α, p^+ and G^+
9: Set $\alpha = \beta$ and go to Step 7.

5 Numerical results

The proposed Fast-CS-DFN algorithm has been implemented in Python 3.9 and compared with CS-DFN [6] (available through the DFL library http://www.iasi.cnr.it/~liuzzi/dfl as package `FASTDFN`). In the implementation of Fast-CS-DFN

we adopted all the choices of CS-DFN and we set $h_{\max} = 10$ in Algorithm 1 and $\epsilon = 1$ in Algorithm 2. The comparison has been carried out on a set of 47 nonsmooth problems. In the following subsections we briefly describe the test problems collection, the metrics adopted in the comparison and, finally, the obtained results.

5.1 Test problems collection

In Table 1 description of the test problems is reported. In particular, each table entry gives the problem name, the number n of variables and the reference where the problem definition can be found.

5.2 Metrics

To compare our derivative-free algorithms we resort to the use of the well-known performance and data profiles (proposed in [5] and [15], respectively). In particular, let $\mathcal{P}$ be a set of problems and $\mathcal{S}$ a set of solvers used to tackle problems in $\mathcal{P}$. Let $\tau > 0$ be a required precision level and denote by t_{ps} the performance index, that is the number of function evaluations required by solver $s \in \mathcal{S}$ to solve problem $p \in \mathcal{P}$. Problem p is claimed to be solved when a point x has been obtained such that the following criterion is satisfied

$$f(x) \leq f_L + \tau(f(x_0) - f_L)$$

where $f(x_0)$ is the initial function value and f_L denotes the best function value found by any solver on problem p itself. Then, the performance ratio r_{ps} is

$$r_{ps} = \frac{t_{ps}}{\min_{i \in \mathcal{S}}\{t_{pi}\}}.$$

Finally, the performance and data profiles of solver s are so defined

$$\rho_s(\alpha) = \frac{1}{|\mathcal{P}|}|\{p \in P : \ r_{ps} \leq \alpha\}|, \quad d_s(\kappa) = \frac{1}{|\mathcal{P}|}|\{p \in P : \ t_{ps}/(n_p + 1) \leq \kappa\}|$$

where n_p is the number of variables of problem p. Particularly, the performance profile $\rho_s(\alpha)$ tells us the fraction of problems that solver s solves with a number of function evaluation which is at most α times the number of function evaluations required by the best performing solver on that problem. On the other hand, the data profile $d_s(\kappa)$ indicates the fraction of problems solved by s with a number of function evaluations which is at most equal to $\kappa(n_p + 1)$, that is the number of function evaluations required to compute κ simplex gradients.

When using performance and data profiles for bench-marking derivative-free algorithms, it is quite usual to consider (at least) three different levels of precision (low, medium and high) corresponding to $\tau = 10^{-1}, 10^{-3}, 10^{-5}$, respectively.

Table 1 Description of the test problems

Problem name	n	Origin
cb2	2	[13]
crescent	2	[8]
demymalo	2	[13]
davidon2	4	[13]
kowalik	4	[13]
lukgamma	4	[13]
oet5	4	[13]
oet6	4	[13]
polak6	4	[13]
colville1	5	[13]
hs78	5	[13]
lukexp	5	[13]
pbc1	5	[13]
shor	5	[13]
elattar	6	[13]
evd61	6	[13]
transformer	6	[13]
wong1	7	[13]
lukfilter	9	[13]
gill	10	[13]
maxquad	10	[13]
polak2	10	[13]
wong2	10	[13]
osborne2	11	[13]
polak3	11	[13]
steiner2	12	[13]
shelldual	15	[13]
watson	20	[13]
wild1	20	[15]
wild2	20	[15]
wild3	20	[15]
wild11	20	[15]
wild15	20	[15]
wild16	20	[15]
wild19	20	[15]
wild20	20	[15]
wild21	20	[15]
wong3	20	[13]
cb3	20, 30, 40	[8]
l1hilb	20, 30, 40	[8]
maxq	20, 30, 40	[8]

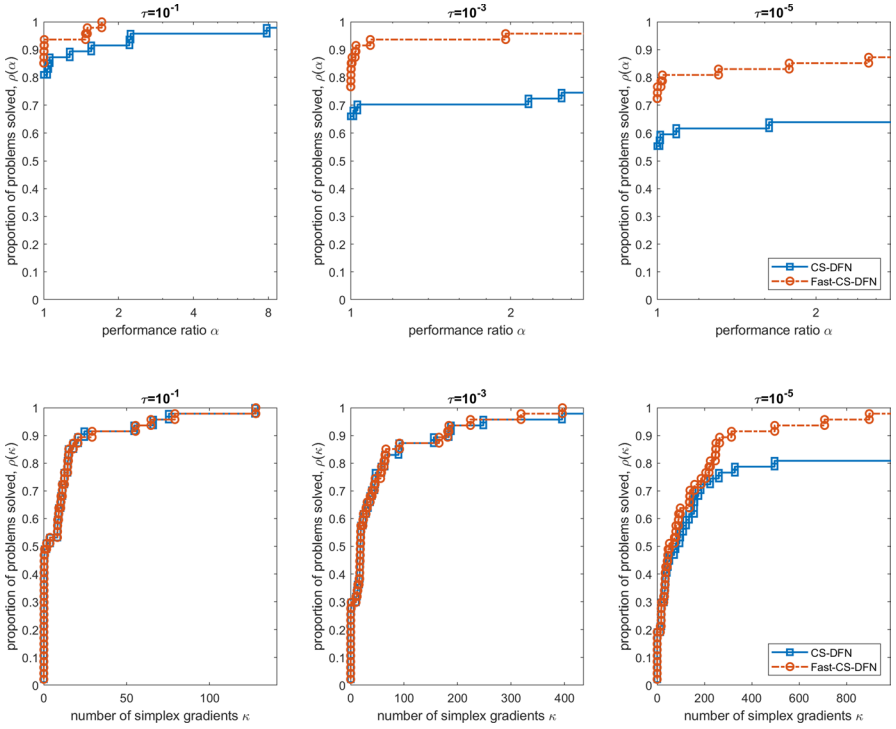

Fig. 1 Comparison of Fast-CS-DFN and CS-DFN

5.3 Results

Figure 1 reports the results of the comparison by means of performance and data profiles between Fast-CS-DFN and CS-DFN.

As we can see, the new algorithm Fast-CS-DFN is always more robust, namely it is able to solve the largest portion of problems within a given amount of computational effort. More in particular, from the performance profiles, we can also say that the new method is invariably more efficient than the original one since the profile curves always have higher values for $\alpha = 1$.

6 Conclusions

In the paper, we propose a strategy to compute (possibly) good descent directions that can be further heuristically exploited within derivative-free algorithms for nonsmooth optimization. In fact, we show that the use of the proposed direction within the CS-DFN algorithm [6] improves the performances of the method. Numerical

results on a set of nonsmooth optimization problems from the literature show the efficiency of the proposed direction computation strategy.

As a final remark, we point out that the proposed strategy could be embedded in virtually any optimization algorithm as an heuristic to try and produce improving points.

Acknowledgements We are really indebted with the anonymous Reviewer for their useful comments and suggestions which greatly helped us improving the manuscript.

Funding Open access funding provided by Università degli Studi di Roma La Sapienza within the CRUI-CARE Agreement.

Data availability The data sets generated during and/or analyzed during the current study are available in the DFL repository, http://www.iasi.cnr.it/~liuzzi/dfl as package `FASTDFN`.

References

1. Astorino, A., Frangioni, A., Gaudioso, M., Gorgone, E.: Piecewise quadratic approximations in convex numerical optimization. SIAM J. Optim. **21**(4), 1418–1438 (2011)
2. Audet, C., Le Digabel, S., Rochon Montplaisir, V., Tribes, C.: Nomad version 4: Nonlinear optimization with the mads algorithm. arxiv:2104.1167 (2021)
3. Bagirov, A.M., Karasözen, B., Sezer, M.: Discrete gradient method: derivative-free method for nonsmooth optimization. J. Optim. Theory Appl. **137**(2), 317–334 (2008)
4. Clarke, F.H.: Optimization and nonsmooth analysis. SIAM (1990)
5. Dolan, E.D., Morè, J.J.: Benchmarking optimization software with performance profiles. Math. Program. **91**(2), 201–213 (2002)
6. Fasano, G., Liuzzi, G., Lucidi, S., Rinaldi, F.: A linesearch-based derivative-free approach for nonsmooth constrained optimization. SIAM J. Optim. **24**(3), 959–992 (2014)
7. Forgy, E.W.: Cluster analysis of multivariate data: efficiency versus interpretability of classifications. Biometrics **21**(3), 768–769 (1965)
8. Karmitsa, N.: Test problems for large-scale nonsmooth minimization. Reports of the Department of Mathematical Information Technology. Series B, Scientific computing, 4/2007 (2007)
9. Khalaf, W., Astorino, A., d'Alessandro, P., Gaudioso, M.: A dc optimization-based clustering technique for edge detection. Optim. Lett. **11**, 627–640 (2017)
10. Le Digabel, S.: Algorithm 909: Nomad: nonlinear optimization with the mads algorithm. ACM Trans. Math. Softw. **37**(4), 44:1-44:15 (2011)
11. Liuzzi, G., Lucidi, S., Rinaldi, F., Vicente, L.N.: Trust-region methods for the derivative-free optimization of nonsmooth black-box functions. SIAM J. Optim. **29**(4), 3012–3035 (2019)
12. Lloyd, S.: Least squares quantization in pcm. IEEE Trans. Inf. Theory **28**(2), 129–137 (1982)
13. Lukšan, L., Vleček, J.: A bundle-Newton method for nonsmooth unconstrained minimization. Math. Program. **83**, 373–391 (1998)
14. Mäkelā, M.M., Neittaanmāki, P.: Nonsmooth optimization: analysis and algorithms with applications to optimal control. World Scientific Press (1992)

15. Morè, J.J., Wild, S.M.: Benchmarking derivative-free optimization algorithms. SIAM J. Optim. **20**(1), 172–191 (2009)
16. Selim, S.Z., Ismail, M.A.: K-means-type algorithms: a generalized convergence theorem and characterization of local optimality. IEEE Trans. Pattern Anal. Mach. Intell. **1**, 81–87 (1984)