A derivative-free method for structured optimization problems

Andrea Cristofari*, Francesco Rinaldi*

*Department of Mathematics "Tullio Levi-Civita"
University of Padua
Via Trieste, 63, 35121 Padua, Italy
E-mail: andrea.cristofari@unipd.it, rinaldi@math.unipd.it

Abstract. Structured optimization problems are ubiquitous in fields like data science and engineering. The goal in structured optimization is using a prescribed set of points, called atoms, to build up a solution that minimizes or maximizes a given function. In the present paper, we want to minimize a black-box function over the convex hull of a given set of atoms, a problem that can be used to model a number of real-world applications. We focus on problems whose solutions are sparse, i.e., solutions that can be obtained as a proper convex combination of just a few atoms in the set, and propose a suitable derivative-free inner approximation approach that nicely exploits the structure of the given problem. This enables us to properly handle the dimensionality issues usually connected with derivative-free algorithms, thus getting a method that scales well in terms of both the dimension of the problem and the number of atoms. We analyze global convergence to stationary points. Moreover, we show that, under suitable assumptions, the proposed algorithm identifies a specific subset of atoms with zero weight in the final solution after finitely many iterations. Finally, we report numerical results showing the effectiveness of the proposed method.

Keywords. Derivative-free optimization. Decomposition methods. Large-scale optimization.

MSC2000 subject classifications. 90C06. 90C30. 90C56.

1 Introduction

In this paper, we consider an optimization problem of the type

$$\min_{x \in \mathcal{M}} f(x), \tag{$P0$}$$

where $\mathcal{M}$ is the convex hull of a finite set of points $\mathcal{A} = \{a_1, \dots, a_m\} \subset \mathbb{R}^n$ called *atoms* (some of them might not be extreme points of $\mathcal{M}$) and $f : \mathbb{R}^n \to \mathbb{R}$ is a continuously differentiable function. We further assume that first-order information related to the objective function is unavailable or impractical to obtain (e.g., functions are expensive to evaluate or somewhat noisy). Since any point $x \in \mathcal{M}$ can be written as a convex combination of the atoms in $\mathcal{A}$, Problem $(P0)$ can be equivalently reformulated considering the simplicial representation of the feasible set:

$$\min_{w \in \Delta_{m-1}} f(Aw), \tag{$P1$}$$

where $A = \begin{bmatrix} a_1 & \dots & a_m \end{bmatrix} \in \mathbb{R}^{n \times m}$ and $\Delta_{m-1} = \{w \in \mathbb{R}^m : e^T w = 1,\ w \geq 0\}$, with e being the vector made of all ones. Thus, each variable w_i gives the weight of the i-th atom in the convex combination.

We are particularly interested in instances of Problem $(P1)$ that admit a sparse solution, i.e., instances whose solutions can be obtained as a proper convex combination of a small subset of atoms.

This occurs, e.g., when $m \gg n$ (as a consequence of Carathéodory's theorem [9]). We would like to notice that this is not the only case that gives sparse solutions. We can have polytopes with $\mathcal{O}(n)$ vertices that, thanks to their structure, can induce sparsity anyway. A classic example is the ℓ_1 ball [6].

This black-box structured optimization problem is somehow related to sparse *atomic decomposition* (see, e.g., [11, 21] and references therein). In such a context the atomic structure can be exploited when developing tailored solvers for the problem.

There exists a significant number of real-world applications that fits our mathematical model. Interesting examples include, among others, black-box adversarial attacks on deep neural networks with ℓ_1 or ℓ_∞ bounded perturbations (see, e.g., [8, 13, 27] and references therein), and reinforcement learning (see, e.g., [28, 41] and references therein) with constrained policies.

In principle, Problem $(P1)$ can be tackled by any linearly constrained derivative-free optimization algorithm. A large number of those methods are available in the literature. Nice overviews can be found in, e.g., [2, 16, 30, 33]. An important class of methods is represented by direct-search schemes (see, e.g., [30] for further details). Those approaches explore the objective function along suitably chosen sets of directions that somehow take into account the shape of the feasible region around the current iterate, and usually are given by the positive generators of an approximate tangent cone related to nearby active constraints [31, 34]. The chosen directions both guarantee feasibility and allow a decrease in the objective function value, when a sufficiently small stepsize is taken. Line search techniques can also be used to better explore the search directions [39]. Moreover, conditions for the active-set identification are described in [35].

Another approach for the linearly constrained setting is proposed in [24], where the authors introduce the notions of deterministic and probabilistic feasible descent (they basically consider the projection of the negative gradient on an approximate tangent cone identified by nearby active constraints). For the deterministic case, a complexity bound for direct search (with sufficient decrease) is given. They further prove global convergence with probability 1 when using direct search based on probabilistic feasible descent, and derive a complexity bound with high probability.

The use of global optimization strategies combined with direct-search approaches for linearly constrained problems has been investigated in [19, 47, 48].

Model-based approaches (see, e.g., [2, 16]) can also be used for solving linearly constrained derivative-free optimization problems. In [46], Powell described trust-region methods for quadratic models with linear constraints, which are used in the LINCOA software [43], developed by the same author for derivative-free linearly constrained optimization. Moreover, an extension of Powell's NEWUOA algorithm [44, 45] to the linearly constrained case has been developed in [25].

Since the derivative-free strategies listed above do not exploit the peculiar structure of Problem $(P1)$, they might get stuck when the problem dimensions increase.

Another way to deal with the original Problem $(P0)$ is by generating the facet-inducing halfspaces that describe the feasible set $\mathcal{M}$. In our case, the facet description could be obtained from the atom list by means of suitable facet enumeration strategies (see, e.g., [5]). This might obviously help in case $m \gg n$ and the polytope has a specific structure. We anyway need to keep in mind that there exists a number of problems where using the facet description is not a viable option. A first example is when $\mathcal{A}$, is linear with respect to the problem dimension, but $\mathcal{M}$ does not have a polynomial description in terms of facet-inducing halfspaces. Another interesting example is given by problems where the inner description is not available and we only have an oracle that generates our atoms. Furthermore, since we consider instances whose solutions can be obtained using a very small number of atoms (usually much smaller than the dimension n), it would be better to exploit the vertex description when devising a new method.

We hence propose a new algorithmic scheme that tries to take into account the features of the

considered problem, thus allowing us to solve large-scale instances. At each iteration, our approach performs three different steps:

(i) it approximately solves a reduced problem whose feasible set is an inner description of $\mathcal{M}$ (given by the convex hull of a suitably chosen subset of atoms);

(ii) it tries to refine the inner description of the feasible set by including new atoms;

(iii) it tries to remove atoms by proper rules in order to keep the dimensions of the reduced problem small.

More in detail, the approximate minimization of the reduced problem is carried out by means of a tailored algorithm that combines the use of a specific set of sparse directions containing positive generators of the tangent cone at the current iterate with a line search similar to those described in, e.g., [37, 38, 39]. Furthermore, the addition/removal of new atoms guarantees an improvement of the objective function whenever we approximately solve the reduced problem. Those key features enable us to prove the convergence of the method and, under suitable assumptions, the asymptotic finite identification of a specific subset of atoms with zero weight in the final solution. This identification result has relevant implications on the computational side. The algorithm indeed keeps the reduced problem small enough along the iterations when the final solution is sparse, thus guaranteeing a significant objective function reduction even with a small budget of function evaluations.

The proposed method is somehow related to inner approximation approaches (see, e.g., [7] and references therein) for convex optimization problems. Anyway, those methods cannot be directly applied to the class of problems considered here due to the following reasons:

- they require assumptions on the objective functions that might be hard to verify in a DFO context;
- they normally use first/second order information to carry out the (approximate) minimization of the reduced problem and to select new atoms to be included in the inner description (see, e.g., [26, 42]).

In our framework, we only require smoothness of the objective function and use zeroth order information (i.e., function evaluations) to approximately minimize the reduced problem and to select a new atom. To the best of our knowledge this is the first time that a complete theoretical and computational analysis of a derivative-free inner approximation approach is carried out.

2 A basic algorithm for minimization over the unit simplex

In our framework, we need an inner solver for approximately minimize the objective function over a subset of atoms. This motivates us to design a tailored approach for problems of the following form:

$$\min_{y \in \Delta_{\bar{m}-1}} \varphi(y), \tag{1}$$

where $\varphi\colon \mathbb{R}^{\bar{m}} \to \mathbb{R}$ is a continuously differentiable function. The scheme of the method, that we named `DF-SIMPLEX`, is reported in Algorithm 1. It combines the use of a suitable set of sparse directions containing positive generators of the tangent cone at the current iterate with a specific line search that guarantees feasibility.

We start by choosing a feasible point $y^0 \in \Delta_{\bar{m}-1}$ and some stepsizes $\hat{\alpha}_i^0,\ i = 1, \ldots, \bar{m}$ (note that we have a starting stepsize for each component y_i of the solution). At each iteration k, we select a variable index j_k such that $y_{j_k}^k$ is "sufficiently positive" (see line 3 in Algorithm 1) and define the directions $d_i^k = \pm(e_i - e_{j_k})$, for all indices $i \neq j_k$, where with $e_i \in \mathbb{R}^{\bar{m}}$ we denote from now on the ith

vector of the canonical basis, i.e., the vector made of all zeros except for the ith component that is equal to 1. Search directions of this form are related to those used in the 2-coordinate descent method proposed in [17], with the difference that here, unlike in [17], first-order information is not available, and then, both $e_i - e_{j_k}$ and $e_{j_k} - e_i$ must be explored for all $i \neq j_k$. Once these search directions are computed, for each of them we perform a line search to get a sufficient reduction in the objective function and we suitably update the values of the starting stepsizes $\hat{\alpha}_i^k$, $i = 1, \ldots, \bar{m}$. The line search procedure is reported in Algorithm 2. It is similar to those described in, e.g., [37, 38, 39]. Notice that, in Algorithm 2, we have $\bar{\alpha} = (z_i^k)_{j_k}$ at line 1 and $\bar{\alpha} = (z_i^k)_i$ at line 3.

Algorithm 1 `DF-SIMPLEX`

1 Choose a point $y^0 \in \Delta_{\bar{m}-1}$, $\tau \in (0, 1]$, $\theta \in (0, 1)$, $\gamma > 0$, $\delta \in (0, 1)$ and $\hat{\alpha}_1^0, \ldots, \hat{\alpha}_{\bar{m}}^0 > 0$
2 For $k = 0, 1, \ldots$
3 Choose j_k such that $y_{j_k}^k \geq \tau \max_{i=1,\ldots,\bar{m}} y_i^k$ and let $\alpha_{j_k}^k = 0$
4 Set $z_1^k = y^k$
5 For $i = 1, \ldots, \bar{m}$
6 If $(i \neq j_k)$ then
7 Set $\tilde{d} = e_i - e_{j_k}$
8 Compute α and d by `Line Search Procedure`$(z_i^k, \tilde{d}, \hat{\alpha}_i^k, \gamma, \delta)$
9 If $\alpha = 0$, then set $\hat{\alpha}_i^{k+1} = \theta \hat{\alpha}_i^k$
10 else set $\hat{\alpha}_i^{k+1} = \alpha$
11 else set $\alpha = 0$ and $d = 0$
12 End if
13 Set $\alpha_i^k = \alpha$, $d_i^k = d$ and $z_{i+1}^k = z_i^k + \alpha_i^k d_i^k$
14 End for
15 Let $\xi_i = \hat{\alpha}_i^{k+1}$, $i \in \{1, \ldots, \bar{m}\} \setminus \{j_k\}$, and $\xi_{j_k} = \hat{\alpha}_{j_k}^k$
16 Set $\hat{\alpha}_{j_k}^{k+1} = \min_{i=1,\ldots,\bar{m}} \xi_i$
17 Set $y^{k+1} = z_{\bar{m}+1}^k$
18 End for

Algorithm 2 `Line Search Procedure`$(z, d, \hat{\alpha}, \gamma, \delta)$

1 Compute the largest $\bar{\alpha}$ such that $z + \bar{\alpha} d \in \Delta_{\bar{m}-1}$ and set $\alpha = \min\{\bar{\alpha}, \hat{\alpha}\}$
2 If $\alpha > 0$ and $\varphi(z + \alpha d) \leq \varphi(z) - \gamma \alpha^2$, then go to line 6
3 Compute the largest $\bar{\alpha}$ such that $z - \bar{\alpha} d \in \Delta_{\bar{m}-1}$ and set $\alpha = \min\{\bar{\alpha}, \hat{\alpha}\}$
4 If $\alpha > 0$ and $\varphi(z - \alpha d) \leq \varphi(z) - \gamma \alpha^2$, then set $d = -d$ and go to line 6
5 Set $\alpha = 0$ and go to line 10
6 Let $\beta = \min\{\bar{\alpha}, \alpha/\delta\}$
7 While $(\alpha < \bar{\alpha}$ and $\varphi(z + \beta d) \leq \varphi(z) - \gamma \beta^2)$
8 Set $\alpha = \beta$ and $\beta = \min\{\bar{\alpha}, \alpha/\delta\}$
9 End while
10 Return α, d

It should be noticed that, in practice, shuffling the search directions used at each iteration k can improve performances. All the theoretical results that will be shown below can be easily adapted to that case.

2.1 Theoretical analysis

To analyze the theoretical properties of the algorithm, let us first recall a stationarity condition for problem (1).

Proposition 1. *A feasible point y^* of Problem* (1) *is stationary if and only if there exists $\lambda^* \in \mathbb{R}$*

such that, for all $i = 1, \dots, \bar{m}$,

$$\nabla_i \varphi(y^*) \begin{cases} \geq \lambda^*, & \text{if } y_i^* = 0, \\ = \lambda^*, & \text{if } y_i^* > 0. \end{cases} \tag{2}$$

We now show that the line search strategy embedded in `DF-SIMPLEX` always terminates in a finite number of steps.

Proposition 2. *`Line Search Procedure` has finite termination.*

Proof. We need to show that the while loop at lines 7–9 ends in a finite number of steps. Arguing by contradiction, assume that this is not true. Then, within the while loop we generate a divergent monotonically increasing sequence of feasible stepsizes α's, which contradicts the fact that $\Delta_{\bar{m}-1}$ is a bounded set. □

In the next proposition, we prove that the stepsizes α_i^k generated using our line search go to zero. This is a standard technical result that will be needed to show convergence of the algorithm.

Proposition 3. *Let* $\{y^k\}$ *be a sequence of points produced by* `DF-SIMPLEX`*. Then,*

$$\lim_{k \to \infty} \alpha_i^k = 0, \quad i = 1, \dots, \bar{m}.$$

Proof. For every fixed $i \in \{1, \dots, \bar{m}\}$, we partition the iterations into two subsets K' and K'' such that

$$\alpha_i^k = 0 \Leftrightarrow k \in K' \quad \text{and} \quad \alpha_i^k \neq 0 \Leftrightarrow k \in K''.$$

If K'' is a finite set, necessarily $\alpha_i^k = 0$ for all sufficiently large k and the result trivially holds. If K'' is an infinite set, to obtain the desired result we need to show that

$$\lim_{\substack{k \to \infty \\ k \in K''}} \alpha_i^k = 0. \tag{3}$$

By instructions of the algorithm, for all $k \in K''$ we have that

$$\varphi(y^{k+1}) \leq \varphi(z_{i+1}^k) \leq \varphi(z_i^k) - \gamma(\alpha_i^k)^2 \leq \varphi(y^k) - \gamma(\alpha_i^k)^2.$$

Combining these inequalities with the fact that $\Delta_{\bar{m}-1}$ is a bounded set and φ is continuous, it follows that $\{\varphi(y^k)\}$ converges and, since $\varphi(y^k) - \varphi(y^{k+1}) \geq \gamma(\alpha_i^k)^2$ for all $k \in K''$, we get (3). □

By taking into account Proposition 3, it is easy to get the following corollary, related to the sequences of intermediate points $\{z_i^k\}$, $i = 1, \dots, \bar{m}$.

Corollary 1. *Let* $\{y^k\}$ *be a sequence of points produced by* `DF-SIMPLEX`*. Then,*

$$\lim_{k \to \infty} \|y^k - z_i^k\| = 0, \quad i = 1, \dots, \bar{m}.$$

We now give the proof of another important result for the global convergence analysis. More specifically, we show that starting stepsizes $\hat{\alpha}_i^k$ considered in the algorithm go to zero as well.

Proposition 4. *Let* $\{y^k\}$ *be a sequence of points produced by* `DF-SIMPLEX`*. Then,*

$$\lim_{k \to \infty} \hat{\alpha}_i^k = 0, \quad i = 1, \dots, \bar{m}.$$

Proof. For every fixed $i \in \{1, \dots, \bar{m}\}$, we partition the iterations into three subsets K_1, K_2 and K_3 such that

$$\alpha_i^k \neq 0 \Leftrightarrow k \in K_1, \quad \alpha_i^k = 0, i \neq j_k \Leftrightarrow k \in K_2 \quad \text{and} \quad i = j_k \Leftrightarrow k \in K_3. \tag{4}$$

From the instructions of the algorithm, we have that

$$\hat{\alpha}_i^{k+1} = \alpha_i^k \geq \hat{\alpha}_i^k, \quad \forall\, k \in K_1, \tag{5}$$

$$\hat{\alpha}_i^{k+1} = \theta \hat{\alpha}_i^k < \hat{\alpha}_i^k, \quad \forall\, k \in K_2, \tag{6}$$

$$\hat{\alpha}_i^{k+1} = \min\{\hat{\alpha}_h^{k+1}, \hat{\alpha}_i^k\} \leq \hat{\alpha}_i^k, \quad h \in \{1, \dots, \bar{m}\} \setminus \{j_k\}, \quad \forall\, k \in K_3. \tag{7}$$

If K_1 is an infinite subset, using (5) and Proposition 3 we obtain

$$\lim_{\substack{k \to \infty \\ k \in K_1}} \hat{\alpha}_i^{k+1} = 0, \tag{8}$$

which, combined with (6) and (7), yields to the desired result. Therefore, in the rest of the proof we assume K_1 to be a finite set.

First, consider the case where K_3 is a finite set, that is, there exists $\bar{k}$ such that $k \in K_2$ for all $k \geq \bar{k}$. For each $k \in K_2$, define l_k as the largest iteration index such that $l_k < k$ and $l_k \in K_1$ (if it does not exist, we let $l_k = 0$). Also define q_k as the number of iterations belonging to K_3 between l_k and k. Therefore, there are $k - l_k - q_k$ iterations belonging to K_2 between l_k and k. From (6)–(7), it follows that

$$\hat{\alpha}_i^{k+1} \leq \theta^{k - l_k - q_k}\, \hat{\alpha}_i^{l_k + 1}.$$

Using the fact that both l_k and q_k are bounded from above (since both K_1 and K_3 are finite sets), we have that $\lim_{\substack{k \to \infty \\ k \in K_2}} \theta^{k - l_k - q_k} = 0$. Therefore, $\lim_{\substack{k \to \infty \\ k \in K_2}} \hat{\alpha}_i^{k+1} = \lim_{k \to \infty} \hat{\alpha}_i^{k+1} = 0$ and the desired result is obtained.

Now, we consider the case where K_3 is an infinite set and we distinguish two subcases. If K_2 is an infinite set, from (6) and (7) we have that $\lim_{\substack{k \to \infty \\ k \in K_2 \cup K_3}} \hat{\alpha}_i^{k+1} = \lim_{k \to \infty} \hat{\alpha}_i^{k+1} = 0$ and the desired result is obtained. Else (i.e., if K_2 is a finite set), there exists $\tilde{k}$ such that $k \in K_3$ for all $k \geq \tilde{k}$ and, picking any index $t \in \{1, \dots, \bar{m}\} \setminus \{i\}$, we can partition the iterations into three subsets Q_1, Q_2 and Q_3 such that

$$\alpha_t^k \neq 0 \Leftrightarrow k \in Q_1, \quad \alpha_t^k = 0, t \neq j_k \Leftrightarrow k \in Q_2 \quad \text{and} \quad t = j_k \Leftrightarrow k \in Q_3.$$

Since $i \in K_3$ for all $k \geq \tilde{k}$, we have that Q_3 is a finite set and, with the same arguments given above for the case where K_3 is a finite set, we obtain that $\lim_{k \to \infty} \hat{\alpha}_t^k = 0$. Using the fact that, from the instructions of the algorithm,

$$\hat{\alpha}_i^{k+1} \leq \min_{h \in \{1, \dots, \bar{m}\} \setminus \{i\}} \hat{\alpha}_h^{k+1}, \quad \forall\, k \in K_3,$$

the desired result is obtained. □

Now, we can state the main convergence result related to `DF-SIMPLEX`. In particular, we show that every limit point of the sequence $\{y^k\}$ generated by the proposed method is stationary for Problem (1).

Theorem 1. *Let $\{y^k\}$ be a sequence of points produced by `DF-SIMPLEX`. Then, every limit point y^* is stationary for Problem* (1).

Proof. Let us consider a subsequence such that

$$\lim_{k \to \infty,\, k \in K} y^k = y^*,$$

with $K \subseteq \{1, 2, \ldots\}$. Since the set of indices $\{1, \ldots, \bar{m}\}$ is finite, it is possible to consider a further subsequence, still denoted by $\{y^k\}_K$ without loss of generality, such that $j_k = \hat{\jmath}$ for all $k \in K$.

We first show that a real number $\rho > 0$ and an iteration $\bar{k} \in K$ exist such that

$$(z_i^k)_{\hat{\jmath}} \geq \rho, \quad \forall\, k \geq \bar{k},\, k \in K, \quad i = 1, \ldots, \bar{m}. \tag{9}$$

Let $\bar{h}$ be any index such that $y_{\bar{h}}^* > 0$ and let ρ be a positive real number such that $y_{\bar{h}}^* \geq (4/\tau)\rho$. For all sufficiently large $k \in K$ we have that $y_{\bar{h}}^k \geq (2/\tau)\rho$ and, recalling how we choose the index j_k (see line 3 of Algorithm 1), for all sufficiently large $k \in K$ we obtain

$$y_{\hat{\jmath}}^k \geq \tau \max_{i=1,\ldots,\bar{m}} y_i^k \geq \tau y_{\bar{h}}^k \geq 2\rho.$$

Using Corollary 1, it follows that

$$\lim_{k\to\infty,\, k\in K} z_i^k = y^*, \quad i = 1, \ldots, \bar{m}, \tag{10}$$

implying that (9) holds and $y_{\hat{\jmath}}^* > 0$.

From (2) we have that y^* is a stationary point if and only if a $\lambda^* \in \mathbb{R}$ exists such that

$$\nabla_i \varphi(y^*) \begin{cases} \geq \lambda^*, & \text{if } y_i^* = 0, \\ = \lambda^*, & \text{if } y_i^* > 0, \end{cases}$$

for all $i = 1, \ldots, \bar{m}$. Since we have just proved that $y_{\hat{\jmath}}^* > 0$, in our case we have that y^* is a stationary point if and only if

$$\nabla_i \varphi(y^*) \begin{cases} \geq \nabla_{\hat{\jmath}} \varphi(y^*), & \text{if } y_i^* = 0, \\ = \nabla_{\hat{\jmath}} \varphi(y^*), & \text{if } y_i^* > 0, \end{cases}$$

for all $i = 1, \ldots, \bar{m}$.

So, assuming by contradiction that y^* is not a stationary point, an index t must exist such that one of the following two cases holds.

(i) $y_t^* = 0$ and $\nabla_t \varphi(y^*) < \nabla_{\hat{\jmath}} \varphi(y^*)$. By the mean value theorem, we can write

$$\varphi(z_t^k - \hat{\alpha}_t^k (e_t - e_{\hat{\jmath}})) - \varphi(z_t^k) = -\hat{\alpha}_t^k \nabla \varphi(u_t^k)^T (e_t - e_{\hat{\jmath}}),$$

where $u_t^k = z_t^k - \omega_t^k \hat{\alpha}_t^k (e_t - e_{\hat{\jmath}})$ and $\omega_t^k \in (0, 1)$. Using Proposition 4 and (10), we have that

$$\lim_{k\to\infty,\, k\in K} \nabla \varphi(u_t^k)^T (e_t - e_{\hat{\jmath}}) = \nabla \varphi(y^*)^T (e_t - e_{\hat{\jmath}}) = \nabla_t \varphi(y^*) - \nabla_{\hat{\jmath}} \varphi(y^*) < 0.$$

It follows that, for all sufficiently large $k \in K$,

$$\varphi(z_t^k - \hat{\alpha}_t^k (e_t - e_{\hat{\jmath}})) > \varphi(z_t^k). \tag{11}$$

Now, using Proposition 3 we have that, for all sufficiently large $k \in K$,

$$z_t^k + \hat{\alpha}_t^k (e_t - e_{\hat{\jmath}}) \in \Delta_{\bar{m}-1}. \tag{12}$$

Taking into account (12) and the instructions of the algorithm, for all sufficiently large $k \in K$ either $\alpha_t^k = 0$ and $\varphi(z_t^k + \hat{\alpha}_t^k (e_t - e_{\hat{\jmath}})) > \varphi(z_t^k) - \gamma(\hat{\alpha}_t^k)^2$, or $\alpha_t^k \neq 0$. In the latter case, combining (11) and (12) we have that, for all sufficiently large $k \in K$, the algorithm does not move along the direction $e_{\hat{\jmath}} - e_t$, and then, $d_t^k = e_t - e_{\hat{\jmath}}$. Using Proposition 4 we also get that,

for all sufficiently large $k \in K$, $z_t^k + \frac{\alpha_t^k}{\delta}(e_t - e_{\hat{\jmath}}) \in \Delta_{\bar{m}-1}$. Therefore, taking into account the `Line Search Procedure` we have that

$$\varphi\Big(z_t^k + \frac{\alpha_t^k}{\delta}(e_t - e_{\hat{\jmath}})\Big) > \varphi(z_t^k) - \gamma\Big(\frac{\alpha_t^k}{\delta}\Big)^2,$$

for all sufficiently large $k \in K$. Using the mean value theorem in the two above inequalities, we have that either

$$\nabla\varphi(\nu_t^k)^T(e_t - e_{\hat{\jmath}}) > -\gamma\hat{\alpha}_t^k \quad \text{or} \quad \nabla\varphi(s_t^k)^T(e_t - e_{\hat{\jmath}}) > -\gamma\frac{\alpha_t^k}{\delta},$$

where $\nu_t^k = z_t^k + \pi_t^k\hat{\alpha}_t^k(e_t - e_{\hat{\jmath}})$, with $\pi_t^k \in (0,1)$ and $s_t^k = z_t^k + \eta_t^k[\alpha_t^k/\delta](e_t - e_{\hat{\jmath}})$, with $\eta_t^k \in (0,1)$. Using Proposition 3, Proposition 4 and the continuity of $\nabla\varphi$, we can take the limits for $k \to \infty$, $k \in K$, and we obtain $\nabla\varphi(y^*)^T(e_t - e_{\hat{\jmath}}) \geq 0$, contradicting the fact that $\nabla_t\varphi(y^*) < \nabla_{\hat{\jmath}}\varphi(y^*)$.

(ii) $y_t^* > 0$ and $\nabla_t\varphi(y^*) \neq \nabla_{\hat{\jmath}}\varphi(y^*)$. First note that, since $y_{\hat{\jmath}}^* > 0$, necessarily $y_t^* < 1$ and, consequently, for all sufficiently large $k \in K$ both the directions $\pm(e_t - e_{\hat{\jmath}})$ are feasible at z_t^k.

Now, assume that $\nabla_t\varphi(y^*) < \nabla_{\hat{\jmath}}\varphi(y^*)$. Reasoning as in case (i), we obtain $\nabla\varphi(y^*)^T(e_t - e_{\hat{\jmath}}) \geq 0$, thus getting a contradiction. Then, necessarily $\nabla_t\varphi(y^*) > \nabla_{\hat{\jmath}}\varphi(y^*)$ but, repeating again the same reasoning as in case (i) with minor modifications, we obtain $\nabla\varphi(y^*)^T(e_t - e_{\hat{\jmath}}) \leq 0$, getting a new contradiction and thus proving the desired result.

□

2.2 Choice of the stopping condition

Now, we describe the stopping condition employed in `DF-SIMPLEX`. As we will see in the next section, this is a key tool for the theoretical analysis of the general inner approximation scheme that embeds `DF-SIMPLEX` as solver of the reduced problem. Moreover, under the assumption that ∇f is Lipschitz continuous, we will show that the stationarity error of the solution returned by `DF-SIMPLEX` is upper bounded by a term that depends on the tolerance chosen in the stopping criterion (see Theorem 2 below).

Given a tolerance $\epsilon > 0$, a standard choice in direct search methods is to terminate the algorithm when a suitable steplength control parameter falls below ϵ. In our case, this means that $\hat{\alpha}_i^k \leq \epsilon$, $i = 1, \ldots, \bar{m}$. Additionally, we prevent each $\hat{\alpha}_i^k$ to become smaller than ϵ. In particular, at line 9 of Algorithm 1 instead of setting $\hat{\alpha}_i^{k+1} = \theta\hat{\alpha}_i^k$ we use the following rule:

$$\hat{\alpha}_i^{k+1} = \max\{\theta\hat{\alpha}_i^k, \epsilon\}. \tag{13}$$

We see that, if $\epsilon = 0$, we have exactly the rule reported in the scheme of Algorithm 1. In order to stop the algorithm, we also require that no progress is made along any feasible direction, that is $\alpha_i^k = 0$ for all $i \neq j_k$.

Summarizing, given $\epsilon > 0$, we use (13) to update each $\hat{\alpha}_i^{k+1}$ at line 9 of Algorithm 1 and we terminate the algorithm at the first iteration k such that

$$\hat{\alpha}_i^k = \epsilon, \ \forall\, i \in \{1, \ldots, \bar{m}\}, \qquad \text{and} \qquad \alpha_i^k = 0, \ \forall\, i \neq j_k. \tag{14}$$

In the next proposition it is shown that this stopping condition is well defined.

Proposition 5. *Given $\epsilon > 0$, the stopping condition* (14) *is satisfied by* `DF-SIMPLEX` *after a finite number of iterations.*

Proof. First note that, in view of (13), we have that

$$\hat{\alpha}_i^k \geq \epsilon, \quad \forall\, k \geq 0, \quad \forall\, i \in \{1, \ldots, \bar{m}\}.$$

Now we show that an iteration $\bar{k}$ exists such that

$$\hat{\alpha}_i^k = \epsilon, \quad \forall\, k \geq \bar{k}, \quad \forall\, i \in \{1, \ldots, \bar{m}\}. \tag{15}$$

Proceeding by contradiction, assume that this is not true. Then, an infinite subsequence $\{y^k\}_{K \subseteq \{0,1,\ldots\}}$ and an index $i \in \{1, \ldots, \bar{m}\}$ exist such that

$$\hat{\alpha}_i^k > \epsilon, \; \forall\, k \in K. \tag{16}$$

Using the same arguments given in the proof of Proposition 3, we have that

$$\lim_{k \to \infty} \alpha_i^k = 0. \tag{17}$$

Then, to obtain the desired contradiction with (16) we can reason similarly as in the proof of Proposition 4, with minor changes that are now described. Define K_1, K_2 and K_3 as in (4). The following relations hold:

$$\hat{\alpha}_i^{k+1} = \alpha_i^k \geq \hat{\alpha}_i^k \geq \epsilon, \quad \forall\, k \in K_1, \tag{18}$$

$$\epsilon \leq \hat{\alpha}_i^{k+1} = \max\{\theta \hat{\alpha}_i^k, \epsilon\} \leq \hat{\alpha}_i^k, \quad \forall\, k \in K_2, \tag{19}$$

$$\epsilon \leq \hat{\alpha}_i^{k+1} \leq \hat{\alpha}_i^k, \quad \forall\, k \in K_3. \tag{20}$$

From (18) and (17), we see that K_1 cannot be an infinite set. So, we only have to consider the cases where K_1 is finite. If K_3 is also a finite set (and then K_2 is an infinite set), we can define l_k ad q_k as in the proof of Proposition 4 and for all $k \in K_2$ we obtain $\epsilon \leq \hat{\alpha}_i^{k+1} \leq \max\{\theta^{k-l_k-q_k} \hat{\alpha}_i^{l_k+1}, \epsilon\}$. It follows that $\hat{\alpha}_i^k = \epsilon$ for all sufficiently large iterations. If K_3 is a infinite set, we distinguish two subcases. If K_2 is also an infinite set, from (19) and (20) again we have $\hat{\alpha}_i^k = \epsilon$ for all sufficiently large iterations. Else (i.e., if K_2 is a finite set), we can reason as in the last part of the proof of Proposition 4, defining in the same way the index t and the three subsets Q_1, Q_2 and Q_3, obtaining that Q_3 is a finite set and, with the same arguments given above for the case where K_3 is a finite set, $\alpha_t^k = \epsilon$ for all sufficiently large iterations. Using the fact that $\epsilon \leq \hat{\alpha}_i^{k+1} \leq \min_{h \in \{1,\ldots,\bar{m}\} \setminus \{i\}} \hat{\alpha}_h^{k+1}$ for all $k \in K_3$, also in this case we obtain that $\hat{\alpha}_i^k = \epsilon$ for all sufficiently large iterations. So, (15) holds.

Finally, to conclude the proof now we show that, for all sufficiently large iterations, $\alpha_i^k = 0$ for all $i \neq j_k$. Proceeding by contradiction, assume that this is not true. Then, an infinite subsequence $\{y^k\}_{K \subseteq \{0,1,\ldots\}}$ and an index $i \in \{1, \ldots, \bar{m}\}$ exist such that $\alpha_i^k > 0$, $\forall\, k \in K$. From the instructions of the algorithm we have that $\hat{\alpha}_i^{k+1} = \alpha_i^k \geq \hat{\alpha}_i^k \geq \epsilon, \quad \forall\, k \in K$. Since, using again the same arguments given in the proof of Proposition 3, we have that $\lim_{k \to \infty} \alpha_i^k = 0$, we thus obtain a contradiction. □

2.3 Additional stationarity results

Using the stopping condition (14) with a given tolerance $\epsilon > 0$, we want to show that, when ∇f is Lipschitz continuous, the solution $\bar{y}$ returned by `DF-SIMPLEX` satisfies the following condition:

$$\max_{y \in \Delta_{\bar{m}-1}} -\nabla \varphi(\bar{y})^T (y - \bar{y}) \leq C\epsilon, \tag{21}$$

for a suitable constant $C > 0$. Note that $\bar{y}$ is stationary if and only if

$$\max_{y \in \Delta_{\bar{m}-1}} -\nabla \varphi(\bar{y})^T (y - \bar{y}) = 0,$$

thus the quantity given in (21) provides a measure for the stationarity error at $\bar{y}$.

The desired error bound can be obtained by suitably adapting standard results of direct-search methods for linearly constrained problems (see [31, 34]). In order to carry out the analysis, we first need to recall a few definitions and to point out some geometric properties of the search directions used in DF-SIMPLEX.

To this extent, it is convenient to consider a reformulation of Problem (1) as an inequality constrained problem of the following form:

$$\begin{aligned} &\min_{y} \varphi(y) \\ &s.t. \quad c_i^T y \leq b_i, \quad i = 1, \ldots, \bar{m}+2, \end{aligned} \tag{22}$$

where $c_1 = e$, $c_2 = -e$, $c_{i+2} = -e_i$, $i = 1, \ldots, \bar{m}$, and $b_1 = 1$, $b_2 = -1$, $b_{i+2} = 0$, $i = 1, \ldots, \bar{m}$.

Let us recall the definition of *active constraints*, *tangent cone* and *normal cone* for the above problem.

Definition 1. *Let y be a feasible point of Problem (22). We say that a constraint c_i is active at y if $c_i^T y = b_i$. We also indicate with $Z(y)$ the index set of active constraints at y, that is, $Z(y) = \{i \colon c_i^T y = b_i\}$.*

Definition 2. *Let y be a feasible point of Problem (22). We indicate with $N(y)$ the normal cone at y, defined as the cone generated by the active constraints at y:*

$$N(y) = \{v \in \mathbb{R}^{\bar{m}} \colon v = \sum_{i \in Z(y)} \lambda_i c_i,\ \lambda_i \geq 0,\ i \in Z(y)\}.$$

We also indicate with $T(y)$ the tangent cone at y, defined as the polar of $N(y)$:

$$T(y) = \{v \in \mathbb{R}^{\bar{m}} \colon v^T d \leq 0,\ \forall\ d \in N(y)\}.$$

It is easy to see that the tangent cone $T(y)$ at a feasible point y of Problem (22) can be equivalently described as follows:

$$T(y) = \{v \in \mathbb{R}^{\bar{m}} \colon e^T v = 0,\ v_i \geq 0,\ i \colon y_i = 0\}. \tag{23}$$

Now, for every iteration k of DF-SIMPLEX, let D^k be the set of all the search directions in $\{\pm(e_i - e_{j_k}),\ i = 1, \ldots, \bar{m},\ i \neq j_k\}$ that are feasible at y^k (where a search direction d is said to be *feasible* at y^k if there exists $\bar{\alpha} > 0$ such that $y + \alpha d \in \Delta_{\bar{m}-1}$ for all $\alpha \in (0, \bar{\alpha}]$). The next remark describes an important property of the set D^k.

Remark 1. *For every iteration k of DF-SIMPLEX, D^k is a set of generators for the tangent cone $T(y^k)$.*

From now on, given a vector v and a convex cone $\mathcal{C}$, we define $v_{\mathcal{C}}$ as the projection of v onto $\mathcal{C}$. Thus, $v_{T(y)}$ is the projection of v onto $T(y)$ and $v_{N(y)}$ is the projection of v onto $N(y)$.

Before stating the desired result, we also need the following lemma to show that, for any vector $v \in \mathbb{R}^{\bar{m}}$ and for any iteration k, a direction $d \in D^k$ exists such that the inner product $v^T d$ is lower bounded by $\|v_{T(y^k)}\|$ up to some constant.

Lemma 1. *For every iteration k of DF-SIMPLEX, we have that*

$$\max_{d \in D^k} v^T d \geq \frac{\|v_{T(y^k)}\|}{2(\bar{m}-1)}, \quad \forall\ v \in \mathbb{R}^{\bar{m}}.$$

Proof. We first observe that any vector $\sigma \in T(y^k)$ can be expressed as a non-negative linear combination of the vectors in D^k with coefficients $|\sigma_i| \leq \|\sigma\|$, that is

$$\sigma = \sum_{\substack{i \neq j_k \\ i:\ \sigma_i \neq 0}} \text{sign}(\sigma_i)(e_i - e_{j_k})|\sigma_i|. \tag{24}$$

Now, pick any vector $v \in \mathbb{R}^{\bar{m}}$ and, for the sake of simplicity, define $u_1, \ldots, u_{|D^k|}$ the directions in D^k. It follows that there exist non-negative coefficients $\lambda_1, \ldots, \lambda_{|D^k|}$, with $0 \leq \lambda_i \leq \|v_{T(y^k)}\|$, $i = 1, \ldots, |D^k|$, such that $v_{T(y^k)} = \sum_{i=1}^{|D^k|} \lambda_i u_i$, and then, $v^T v_{T(y^k)} = \sum_{i=1}^{|D^k|} \lambda_i v^T u_i$. Therefore, an index $i \in \{1, \ldots, |D^k|\}$ exists such that

$$\lambda_i v^T u_i \geq \frac{1}{|D^k|} v^T v_{T(y^k)} \geq \frac{1}{2(\bar{m}-1)} v^T v_{T(y^k)} \geq \frac{1}{2(\bar{m}-1)} \|v_{T(y^k)}\|^2,$$

where the last inequality follows from the property of the projection. Since we have $0 \leq \lambda_i \leq \|v_{T(y^k)}\|$, the result is obtained. □

We are finally ready to provide a bound on the stationarity error for the solution returned by `DF-SIMPLEX`.

Theorem 2. *Assume that $\nabla\varphi$ is Lipschitz continuous with constant L and the stopping condition* (14) *is used with a given tolerance $\epsilon > 0$. Then, the solution $\bar{y}$ returned by* `DF-SIMPLEX` *is such that*

$$\max_{y \in \Delta_{\bar{m}-1}} -\nabla\varphi(\bar{y})^T(y - \bar{y}) \leq C\epsilon,$$

where $C = 2\sqrt{2}(\bar{m}-1)(2L+\gamma)$.

Proof. Let k be the last iteration of `DF-SIMPLEX`, so that $y^k = \bar{y}$. In view of Lemma 1, used with $v = -\nabla\varphi(\bar{y})$, we have that a $d \in D^k$ exists such that

$$-\nabla\varphi(\bar{y})^T d \geq \frac{\left\|[-\nabla\varphi(\bar{y})]_{T(\bar{y})}\right\|}{2(\bar{m}-1)}. \tag{25}$$

Since in (14) we require $\alpha_i^k = 0$ for all $i \neq j_k$ (i.e., no progress is made along any feasible direction), from the instructions of the algorithm and the `Line Search Procedure` we have that

$$\varphi(\bar{y} + \alpha d) > \varphi(\bar{y}) - \gamma\alpha^2, \tag{26}$$

with

$$0 < \alpha \leq \epsilon, \tag{27}$$

where the last inequalities for α follow from the fact that each $\hat{\alpha}_i^k$ is required to be equal to ϵ in (14). By the mean value theorem, $\varphi(\bar{y}+\alpha d) - \varphi(\bar{y}) = \alpha\nabla\varphi(\bar{y}+\eta\alpha d)^T d$, for some $\eta \in (0,1)$. Thus, from (26), we obtain $\alpha\nabla\varphi(\bar{y}+\eta\alpha d)^T d + \gamma\alpha^2 > 0$. Dividing both terms by α, we get $\nabla\varphi(\bar{y}+\eta\alpha d)^T d + \gamma\alpha > 0$. Now, we subtract $\nabla\varphi(\bar{y})^T d$ to both terms of the above inequality, obtaining

$$[\nabla\varphi(\bar{y}+\eta\alpha d) - \nabla\varphi(\bar{y})]^T d + \gamma\alpha > -\nabla\varphi(\bar{y})^T d.$$

Using the fact that $\nabla\varphi$ is Lipschitz continuous, we have $[\nabla\varphi(\bar{y}+\eta\alpha d) - \nabla\varphi(\bar{y})]^T d \leq L\eta\alpha\|d\|^2 \leq 2L\alpha$, where the last inequality follows from the fact that $\eta \in (0,1)$ and $\|d\| = \sqrt{2}$. Then, $2L\alpha + \gamma\alpha >$

$-\nabla\varphi(\bar{y})^T d$. Combining this inequality with (25) and (27), we get $\big\|[-\nabla\varphi(\bar{y})]_{T(\bar{y})}\big\| < 2\epsilon(\bar{m}-1)(2L+\gamma)$. To conclude the proof, we thus have to show that

$$\max_{y\in\Delta_{\bar{m}-1}} -\nabla\varphi(\bar{y})^T(y-\bar{y}) \leq \sqrt{2}\big\|[-\nabla\varphi(\bar{y})]_{T(\bar{y})}\big\|. \tag{28}$$

Since, by polar decomposition, every vector $v\in\mathbb{R}^{\bar{m}}$ can be written as $v = v_{T(\bar{y})} + v_{N(\bar{y})}$ (see, e.g., [49]) we have $-\nabla\varphi(\bar{y}) = [-\nabla\varphi(\bar{y})]_{T(\bar{y})} + [-\nabla\varphi(\bar{y})]_{N(\bar{y})}$. Therefore, for any $y\in\Delta_{\bar{m}-1}$ we can write

$$-\nabla\varphi(\bar{y})^T(y-\bar{y}) = [-\nabla\varphi(\bar{y})]^T_{T(\bar{y})}(y-\bar{y}) + [-\nabla\varphi(\bar{y})]^T_{N(\bar{y})}(y-\bar{y}). \tag{29}$$

In order to upper bound the right-hand side term of the above inequality, we first write

$$[-\nabla\varphi(\bar{y})]^T_{T(\bar{y})}(y-\bar{y}) \leq \big\|[-\nabla\varphi(\bar{y})]_{T(\bar{y})}\big\|\|y-\bar{y}\| \leq \sqrt{2}\big\|[-\nabla\varphi(\bar{y})]_{T(\bar{y})}\big\|, \tag{30}$$

where the last inequality follows from the fact that both y and $\bar{y}$ belong to $\Delta_{\bar{m}-1}$. Moreover, we have that $y-\bar{y}\in T(\bar{y})$. Therefore, from the definition of the tangent cone, we also have that

$$[-\nabla\varphi(\bar{y})]^T_{N(\bar{y})}(y-\bar{y}) \leq 0. \tag{31}$$

From (29), (30) and (31), we conclude that

$$-\nabla\varphi(\bar{y})^T(y-\bar{y}) \leq \sqrt{2}\big\|[-\nabla\varphi(\bar{y})]_{T(\bar{y})}\big\|, \quad \forall\, y\in\Delta_{\bar{m}-1},$$

that is (28) holds and the result is obtained. □

3 Optimize, Refine & Drop (ORD) Algorithm

In principle, we might use barycentric coordinates to represent the feasible set of Problem $(P0)$, thus obtaining a new problem of the form given in $(P1)$ that might be solved by `DF-SIMPLEX` (or any other solver for linearly constrained optimization). Unfortunately, since the number of variables in Problem $(P1)$ is the same as the number of atoms in $\mathcal{A}$, when $|\mathcal{A}|$ increases (keep in mind that this is often the case in our context), it gets hard to obtain a reasonable solution within the given budget of function evaluations. We further notice that in our context good points usually lie in small dimensional faces of the feasible set (i.e., only a small number of atoms is needed to assemble those points). This is the reason why we propose an inner approximation scheme to tackle the problem.

Algorithm 3 `Optimize, Refine & Drop (ORD) Algorithm`

Choose $\{\epsilon^k\} \searrow 0$, $\mathcal{A}^0 \subseteq \mathcal{A}$, $a_{i_0} \in \mathcal{A}^0$, set $x^0 = a_{i_0}$, $y^0 = e_{i_0} \in \mathbb{R}^{|\mathcal{A}^0|}$, $\hat{\mu}^0 \in (0,1)$, $\gamma > 0$ and $\theta \in (0,1)$
For $k = 0, 1, \ldots$

Optimize Phase

Let A^k be the matrix with the atoms in $\mathcal{A}^k$ as columns (so that $x^k = A^k y^k$)
Run `DF-SIMPLEX` from y^k to compute an approximate solution $\bar{y}^k$ of Problem (32) with tolerance ϵ^k
Set $\bar{x}^k = A^k \bar{y}^k$

Refine Phase

If there exists an index $i_k \in \{1, \ldots, m\}$ and a scalar $\mu^k \in [\hat{\mu}^k, 1]$ such that

$$f(\bar{x}^k + \mu^k(a_{i_k} - \bar{x}^k)) \leq f(\bar{x}^k) - \gamma(\mu^k)^2, \quad a_{i_k} \in \mathcal{A} \setminus \mathcal{A}^k,$$

then set $x^{k+1} = \bar{x}^k + \mu^k(a_{i_k} - \bar{x}^k)$, $\mathcal{R}^k = \{a_{i_k}\}$ and $\hat{\mu}^{k+1} = \hat{\mu}^k$
Else set $x^{k+1} = \bar{x}^k$, $\mathcal{R}^k = \emptyset$ and $\hat{\mu}^{k+1} = \theta\hat{\mu}^k$

Drop Phase

Choose a subset $\mathcal{D}^k \subseteq \{a \in \mathcal{A}^k \text{ such that } a = A^k e_h \text{ and } \bar{y}^k_h = 0\}$

Let $\mathcal{A}^{k+1} = \mathcal{A}^k \cup \mathcal{R}^k \setminus \mathcal{D}^k$, and set $y^{k+1} \in \Delta_{|\mathcal{A}^{k+1}|-1}$ such that $x^{k+1} = \sum_{a_i \in \mathcal{A}^{k+1}} a_i y_i^{k+1}$

End for

At a given iteration k, our method considers a reduced problem by approximating the set $\mathcal{M}$ with the convex hull of a set $\mathcal{A}^k \subseteq \mathcal{A}$, and tries to suitably improve this description by including/removing atoms according to some given rule. We can now describe in depth the three main phases that characterize our approach.

Let A^k be the matrix whose columns are the atoms in $\mathcal{A}^k$. First, in the *Optimize Phase*, we use `DF-SIMPLEX` to compute an approximate solution of the following reduced problem:

$$\min_{y \in \Delta_{|\mathcal{A}^k|-1}} \varphi^k(y), \tag{32}$$

where $\varphi^k(y) = f(A^k y)$. In particular, we run `DF-SIMPLEX` on Problem (32) until a given tolerance ϵ^k is reached, according to the stopping condition discussed in Subsection 2.2.

In the second phase, the so-called *Refine Phase*, we try to get a better inner description of $\mathcal{M}$ by choosing an atom $a_{i_k} \in \mathcal{A} \setminus \mathcal{A}^k$, with $i_k \in \{1, \ldots, m\}$, that guarantees improvement of the objective value (we use $\mathcal{R}^k$ to indicate the set that, if non-empty, is a singleton composed by the atom to be added to $\mathcal{A}^k$). In practice, we randomly pick the atoms in $\mathcal{A} \setminus \mathcal{A}^k$, with no repetition, and we stop when we find one satisfying a sufficient decrease condition.

Finally, in the last phase (*Drop Phase*), we get rid of some atoms in $\mathcal{A}^k$ thanks to a simple selection rule (we will use the notation $\mathcal{D}^k$ to indicate the set of atoms to be removed from $\mathcal{A}^k$). This tool enables us to keep the dimension of the reduced problem small enough along the iterations.

The detailed scheme is reported in Algorithm 3. We would like to notice that the parameters γ and θ can be different from those used in `DF-SIMPLEX`.

We first introduce suitable optimality conditions for $(P0)$ that will be exploited in the theoretical analysis of our algorithmic framework.

Proposition 6. *A feasible point x^* of Problem $(P0)$ is stationary if and only if*

$$\nabla f(x^*)^T(a - x^*) \geq 0, \quad \forall\ a \in \mathcal{A}.$$

Now, we prove that the stepsize used to define the sufficient decrease in the atom selection of the second phase (see line 6 of Algorithm 3) goes to zero. This result will be needed in the global convergence analysis of the method.

Proposition 7. *Let $\{x^k\}$ be a sequence of points produced by Algorithm 3. Then,*

$$\lim_{k\to\infty} \hat{\mu}^k = 0.$$

Proof. We partition the iterations into two subsets K_1 and K_2 such that

$$\hat{\mu}^{k+1} = \hat{\mu}^k \Leftrightarrow k \in K_1 \quad \text{and} \quad \hat{\mu}^{k+1} = \theta\hat{\mu}^k \Leftrightarrow k \in K_2, \tag{33}$$

that is, the iterations in K_1 are those where the test at line 6 of Algorithm 3 is satisfied, while the iterations in K_2 are those where that test is not satisfied. From line 6 of Algorithm 3, for all $k \in K_1$ we have that

$$f(x^{k+1}) = f(\bar{x}^k + \mu^k(a_{i_k} - \bar{x}^k)) \leq f(\bar{x}^k) - \gamma(\mu^k)^2 \leq f(x^k) - \gamma(\mu^k)^2,$$

where $f(\bar{x}^k) \leq f(x^k)$ in the last inequality follows from the fact that $\bar{x}^k = A^k\bar{y}^k$ and $\bar{y}^k$ is obtained from `DF-SIMPLEX` with a starting point y^k satisfying $x^k = A^k y^k$. Therefore, if K_1 is infinite, using the fact that f is continuous and the feasible set is bounded it follows that $\{f(x^k)\}$ converges and

$$\lim_{\substack{k\to\infty \\ k\in K_1}} \mu^k = 0. \tag{34}$$

Since $\hat{\mu}^k \leq \mu^k$ for all $k \in K_1$, it follows that $\{\hat{\mu}^k\}_{K_1} \to 0$. Taking into account that $\hat{\mu}^{k+1} = \theta\hat{\mu}^k$ for all $k \in K_2$, we obtain that the desired holds if K_1 is infinite.

If K_1 is finite, there exists $\bar{k}$ such that $k \in K_2$ for all $k \geq \bar{k}$. For each $k \in K_2$, define l_k as the largest iteration index such that $l_k < k$ and $l_k \in K_1$ (if it does not exist, we let $l_k = 0$). Therefore, there are $k - l_k$ iterations belonging to K_2 between l_k and k, implying that $\hat{\mu}^{k+1} \leq \theta^{k-l_k}\hat{\mu}^{l_k+1}$. Using the fact that l_k is bounded from above (since K_1 is finite), we have that $\lim_{\substack{k\to\infty \\ k\in K_2}} \theta^{k-l_k} = 0$. Therefore, $\lim_{\substack{k\to\infty \\ k\in K_2}} \hat{\mu}^{k+1} = 0$ and the desired result is obtained. □

We thus get the following useful corollary.

Corollary 2. *Let $\{x^k\}$ be a sequence of points produced by Algorithm 3. Then,*

$$\lim_{k\to\infty} \|x^{k+1} - \bar{x}^k\| = 0.$$

Proof. As in the proof of Proposition 7, let us define K_1 and K_2 satisfying (33). If K_1 is a finite set, from the instructions of the algorithm we have that an iteration $\tilde{k}$ exists such that $x^{k+1} = \bar{x}^k$ for all $k \geq \tilde{k}$ and the desired result is obtained. If K_1 is an infinite set, by the same arguments used in the proof of Proposition 7 we get (34), that is,

$$\lim_{\substack{k\to\infty \\ k\in K_1}} \|x^{k+1} - \bar{x}^k\| = 0,$$

and the desired result is obtained since $x^{k+1} = \bar{x}^k$ for all $k \in K_2$. □

In the next theorem, we prove global convergence of the proposed algorithm.

Theorem 3. *Let $\{x^k\}$ be a sequence of points produced by Algorithm 3. Then, (at least) one limit point x^* exists such that x^* is stationary for Problem $(P0)$.*

Proof. Using Proposition 7, the fact that the feasible set of every reduced Problem (32) is bounded and the fact that $\mathcal{A}$ is a finite set, there exists an infinite subset of iterations $K \subseteq \{0, 1, \ldots\}$ such that

$$\mathcal{A}^k = \bar{\mathcal{A}}, \quad \forall\ k \in K; \quad \lim_{\substack{k \to \infty \\ k \in K}} \bar{y}^k = y^*; \quad \mu^{k+1} < \mu^k,\ \forall\ k \in K.$$

Since $\mathcal{A}^k$ is constant for all $k \in K$, also the matrix A^k and the function φ^k are the same for all $k \in K$, and let us denote them by $\bar{A}$ and $\bar{\varphi}$, respectively. Hence, we also have

$$\lim_{\substack{k \to \infty \\ k \in K}} x^k = \bar{A} y^* = x^*.$$

Taking into account Proposition 6, to obtain the desired result we have to show that

$$\nabla f(x^*)^T (a - x^*) \geq 0, \quad \forall\ a \in \bar{\mathcal{A}}, \tag{35a}$$

$$\nabla f(x^*)^T (a - x^*) \geq 0, \quad \forall\ a \in \mathcal{A} \setminus \bar{\mathcal{A}}. \tag{35b}$$

To prove (35a), for all iterations $k \in K$ consider the points $\bar{y}^k$, which are returned by `DF-SIMPLEX` when the stopping condition (14) is satisfied. Since the set of directions used in `DF-SIMPLEX` is finite, without loss of generality we can assume that, for all $k \in K$, the set of feasible directions at $\bar{y}^k$ used in the last iteration of `DF-SIMPLEX` is the same for all $k \in K$. Let us denote this set of directions by D. Since the stopping condition (14) requires that no progress is made along any direction, from the instructions of `DF-SIMPLEX` we have that, at any iteration $k \in K$,

$$\bar{\varphi}(\bar{y}^k + \alpha d) > \bar{\varphi}(\bar{y}^k) - \gamma \alpha^2, \quad \forall\ d \in D,$$

with $0 < \alpha \leq \epsilon^k$. By the mean value theorem, $\bar{\varphi}(\bar{y}^k + \alpha d) - \bar{\varphi}(\bar{y}^k) = \alpha \nabla \bar{\varphi}(\bar{y}^k + \eta^k \alpha d)^T d$, for some $\eta^k \in (0, 1)$. Then, for any $k \in K$,

$$\nabla \bar{\varphi}(\bar{y}^k + \eta^k \alpha d)^T d \geq -\gamma \alpha \geq -\gamma \epsilon^k, \quad \forall\ d \in D.$$

Using the fact that $\eta^k \in (0, 1)$, $\alpha \leq \epsilon^k$ and $\epsilon^k \to 0$, we have that

$$\lim_{\substack{k \to \infty \\ k \in K}} (\bar{y}^k + \eta^k \alpha d) = y^*, \quad \forall\ d \in D.$$

Therefore, from the continuity of $\nabla \bar{\varphi}$ it follows that

$$\nabla \bar{\varphi}(y^*)^T d \geq 0, \quad \forall\ d \in D. \tag{36}$$

Now consider any point $y \in \Delta_{|\bar{\mathcal{A}}|-1}$. Reasoning as in the last part of the proof of Theorem 2, we have that $y - y^* \in T(y^*)$. Moreover, it is easy to verify that the set $D^* = \{d \in D \text{ such that } d \text{ is feasible at } y^*\}$ is a set of generators for $T(y^*)$. Therefore, denoting by $d_1, \ldots, d_{|D^*|}$ the directions that form the set D^*, we have that $y - y^* = \sum_{i=1}^{|D^*|} \lambda_i d_i$, with $\lambda_i \geq 0$, $i = 1, \ldots, |D|$. Taking into account (36), it follows that

$$\nabla \bar{\varphi}(y^*)^T (y - y^*) = \sum_{i=1}^{|D^*|} \lambda_i \nabla \bar{\varphi}(y^*)^T d_i \geq 0, \quad \forall\ y \in \Delta_{|\bar{\mathcal{A}}|-1}.$$

Then, for all $y \in \Delta_{|\bar{\mathcal{A}}|-1}$ we have that

$$\begin{aligned} 0 \leq \nabla \bar{\varphi}(y^*)^T (y - y^*) &= [\bar{A}^T \nabla f(\bar{A} y^*)]^T (y - y^*) = \nabla f(\bar{A} y^*)^T [\bar{A}(y - \bar{y}^*)] \\ &= \nabla f(x^*)^T (\bar{A} y - x^*). \end{aligned}$$

Since $\text{conv}(\bar{A}) = \{x \in \mathbb{R}^n : x = \bar{A}y, y \in \Delta_{|\mathcal{A}|-1}\}$, we obtain that

$$\nabla f(x^*)^T(x - x^*) \geq 0, \quad \forall\, x \in \text{conv}(\bar{A}),$$

implying that (35a) holds.

To prove (35b), note that, from the instructions of the algorithm, we have that $\mu^{k+1} < \mu^k$ only when the test at line 6 is not satisfied. Hence, for all $k \in K$,

$$f(\bar{x}^k + \mu^k(a - \bar{x}^k)) > f(\bar{x}^k) - \gamma(\mu^k)^2, \quad \forall\, a \in \mathcal{A} \setminus \bar{\mathcal{A}}.$$

By the mean value theorem, for any $a \in \mathcal{A} \setminus \bar{\mathcal{A}}$ we can write

$$f(\bar{x}^k + \mu^k(a - \bar{x}^k)) - f(\bar{x}^k) = \mu^k \nabla f(\bar{x}^k + \eta^k \mu^k(a - \bar{x}^k))^T(a - \bar{x}^k),$$

for some $\eta^k \in (0, 1)$. Therefore,

$$\nabla f(\bar{x}^k + \eta^k \mu^k(a - \bar{x}^k))^T(a - \bar{x}^k) > -\gamma\mu^k, \quad \forall\, k \in K.$$

From Proposition 7 and the fact that $\eta^k \in (0, 1)$, we have that

$$\lim_{\substack{k\to\infty \\ k\in K}} (\bar{x}^k + \eta^k \mu^k(a - \bar{x}^k)) = x^*.$$

Therefore, taking into account that $\mu^k \to 0$ and that ∇f is continuous, we obtain

$$0 \leq \lim_{\substack{k\to\infty \\ k\in K}} \nabla f(\bar{x}^k + \eta^k \mu^k(a - \bar{x}^k))^T(a - \bar{x}^k) = \nabla f(x^*)^T(a - x^*).$$

Since the above relation holds for all $a \in \mathcal{A} \setminus \bar{\mathcal{A}}$, we finally get (35b). □

4 Identification property of ORD

In our problem, every feasible point is expressed as a (not necessarily unique) convex combination of the atoms $a_i \in \mathcal{A}$. In this section we show that, under suitable assumptions, some atoms that are not needed to express the optimal solution are identified and discarded by ORD in a finite number of iterations. Loosely speaking, from a certain iteration we are guaranteed that the set $\mathcal{A}^k$ does not contain "useless" atoms. Before showing this property, we report a useful intermediate result.

Proposition 8. *Let x^* be a stationary point of Problem $(P0)$ and let $w^* \in \Delta_{m-1}$ be any vector such that $x^* = Aw^*$. Then, for every atom $a_i \in \mathcal{A}$ such that $\nabla f(x^*)^T(a_i - x^*) > 0$, we have that $w_i^* = 0$.*

Proof. Consider the reformulation of Problem $(P0)$ in (1), with $\varphi(w) = f(Aw)$. Let w^* be any feasible point of Problem (1) that $x^* = Aw^*$. Since x^* is stationary for Problem $(P0)$ and $\text{conv}(\mathcal{A}) = \{x \in \mathbb{R}^n : x = Aw, w \in \Delta_{m-1}\}$, we have that

$$\nabla f(x^*)^T(Aw - x^*) \geq 0, \quad \forall\, w \in \Delta_{m-1}.$$

Moreover, for all $w \in \Delta_{m-1}$ we can write

$$\nabla f(x^*)^T(Aw - x^*) = [A^T \nabla f(Aw^*)]^T(w - w^*) = \nabla\varphi(w^*)^T(w - w^*).$$

It follows that $\nabla\varphi(w^*)^T(w-w^*) \geq 0$ for all $w \in \Delta_{m-1}$, that is, w^* is stationary for Problem (1) and satisfies the following KKT conditions with multipliers $\lambda^* \in \mathbb{R}$ and $v^* \in \mathbb{R}^m$:

$$\nabla\varphi(w^*) - \lambda^* e - v^* = 0, \tag{37a}$$
$$e^T w^* = 1, \tag{37b}$$
$$(v^*)^T w^* = 0, \tag{37c}$$
$$w^* \geq 0, \tag{37d}$$
$$v^* \geq 0. \tag{37e}$$

From (37a) we can write

$$v^* = \nabla\varphi(w^*) - \lambda^* e, \tag{38}$$

and then, by (37c) we get that $0 = (v^*)^T w^* = (\nabla\varphi(w^*) - \lambda^* e)^T w^*$. Using (37b) we obtain that $\lambda^* = \nabla\varphi(w^*)^T w^*$, which, combined with (38), yields to

$$v^* = \nabla\varphi(w^*) - (\nabla\varphi(w^*)^T w^*)e$$

So, for all $h = 1, \ldots, m$ we have that

$$\begin{aligned} v_h^* &= \nabla\varphi(w^*)^T(e_h - w^*) = [A^T\nabla f(Aw^*)]^T(e_h - w^*) = \nabla f(x^*)^T(Ae_h - Aw^*) \\ &= \nabla f(x^*)^T(a_h - x^*). \end{aligned}$$

Therefore, if $\nabla f(x^*)^T(a_i - x^*) > 0$ for an atom $a_i \in \mathcal{A}$, this means that $v_i^* > 0$ and (37c), (37d) and (37e) yield to $w_i^* = 0$, thus proving the desired result. □

In the next theorem, we assume that $x^k \to x^*$ (this is pretty standard in the analysis of active-set identification properties) and show that, for k sufficiently large, the atoms satisfying the condition of Proposition 8 are not included in $\mathcal{A}^k$. To obtain such a result, we set $\mathcal{D}^k$ as follows:

$$\mathcal{D}^k = \{a \in \mathcal{A}^k \text{ such that } a = A^k e_h \text{ and } \bar{y}_h^k = 0\}. \tag{39}$$

Theorem 4. *Let $\{x^k\}$ be a sequence of points produced by Algorithm 3, where $\mathcal{D}^k$ is computed as in (39). Assume that $\lim_{k\to\infty} x^k = x^*$. Then, an iteration $\bar{k}$ exists such that, for all $k \geq \bar{k}$,*

$$\nabla f(x^*)^T(a - x^*) > 0,\ a \in \mathcal{A} \ \Rightarrow\ a \notin \mathcal{A}^k.$$

Proof. Let $a \in \mathcal{A}$ be an atom such that

$$\nabla f(x^*)^T(a - x^*) > 0. \tag{40}$$

First, we want to show that

$$a \notin \mathcal{R}^k, \quad \forall \text{ sufficiently large } k. \tag{41}$$

Arguing by contradiction, assume that (41) is not true. Then, an infinite subset of iterations $K \subseteq \{0, 1 \ldots\}$ exists such that $a \in \mathcal{R}^k$ for all $k \in K$. From the instructions of the algorithm, we have that

$$f(\bar{x}^k + \mu^k(a - \bar{x}^k)) \leq f(\bar{x}^k) - \gamma(\mu^k)^2, \quad \forall\ k \in K.$$

By the mean value theorem, we can write

$$f(\bar{x}^k + \mu^k(a - \bar{x}^k)) - f(\bar{x}^k) = \mu^k \nabla f(\bar{x}^k + \eta^k\mu^k(a - \bar{x}^k))^T(a - \bar{x}^k),$$

for some $\eta^k \in (0,1)$, and then

$$\nabla f(\bar{x}^k + \eta^k \mu^k (a - \bar{x}^k))^T (a - \bar{x}^k) \leq -\gamma \mu^k, \quad \forall\, k \in K.$$

From Corollary 2 and the fact that $\|\bar{x}^k - x^*\| \leq \|\bar{x}^k - x^{k+1}\| + \|x^{k+1} - x^*\|$, it follows that $\{\bar{x}^k\} \to x^*$. Taking also into account that $\eta^k \in (0,1)$ and $\{\mu^k\} \to 0$ (from Proposition 7), we have that

$$\lim_{\substack{k\to\infty \\ k\in K}} (\bar{x}^k + \eta^k \mu^k (a - \bar{x}^k)) = x^*.$$

Therefore, using the continuity of ∇f we obtain

$$0 \geq \lim_{\substack{k\to\infty \\ k\in K}} \nabla f(\bar{x}^k + \eta^k \mu^k (a - \bar{x}^k))^T (a - \bar{x}^k) = \nabla f(x^*)^T (a - x^*),$$

which contradicts (40). Thus, (41) holds.

Now, to prove the desired result we proceed by contradiction. Namely, we assume that an infinite subset of iterations $K \subseteq \{0, 1 \ldots\}$ exists such that $a \in \mathcal{A}^k$ for all $k \in K$. In view of (41), an iteration $\hat{k} \in K$ must exist such that

$$a \in \mathcal{A}^k \setminus \mathcal{D}^k, \quad \forall\, k \geq \hat{k},\, k \in K. \tag{42}$$

Using the fact that $\mathcal{A}$ is a finite set and the feasible set of every restricted Problem (32) is compact, without loss of generality we can assume that $\mathcal{A}^k$ is constant for all $k \in K$ and that $\{\bar{y}^k\}$ converges to y^* (passing to a further subsequence if necessary). Namely,

$$\mathcal{A}^k = \bar{\mathcal{A}}, \quad \forall\, k \in K, \tag{43a}$$

$$\lim_{\substack{k\to\infty \\ k\in K}} \bar{y}^k = y^*. \tag{43b}$$

Since $\mathcal{A}^k$ is constant for all $k \in K$, also the matrix A^k and the function φ^k are the same for all $k \in K$, and let us denote them by $\bar{A}$ and $\bar{\varphi}$, respectively. From the previous relations, and taking into account Proposition 7, we also have

$$x^* = \lim_{\substack{k\to\infty \\ k\in K}} x^k = \lim_{\substack{k\to\infty \\ k\in K}} \bar{x}^k = \lim_{\substack{k\to\infty \\ k\in K}} \bar{A}\bar{y}^k = \bar{A}y^*.$$

Moreover, let us denote by $\hat{\imath}$ the column index of the matrix $\bar{A}$ the corresponds to the atom a, that is, $\bar{A}e_{\hat{\imath}} = a$.

From (42) and (39), necessarily $\bar{y}^k_{\hat{\imath}} > 0$ for all $k \geq \hat{k}$, $k \in K$. Since the set of directions used in `DF-SIMPLEX` is finite, for all $k \in K$ we can assume that the directions used in the last iteration of `DF-SIMPLEX` are the same, having the form $\pm(e_h - e_{\hat{\jmath}})$, $h = 1, \ldots, |\bar{\mathcal{A}}|$, $h \neq \hat{\jmath}$, for some $\hat{\jmath} \in \{1, \ldots, |\bar{\mathcal{A}}|\}$, with $\bar{y}^k_{\hat{\jmath}} > 0$ for all $k \in K$. In particular, recalling the rule for computing the search directions in `DF-SIMPLEX` and that the stopping condition (14) requires that no progress is made along any direction, we have that

$$\bar{y}^k_{\hat{\jmath}} \geq \tau/|\bar{\mathcal{A}}|, \quad \forall\, k \in K. \tag{44}$$

Moreover, $e_{\hat{\jmath}} - e_{\hat{\imath}}$ is a feasible direction at $\bar{y}^k$ for all $k \geq \hat{k}$, since $\bar{y}^k_{\hat{\imath}} > 0$. So, using again the fact that the stopping condition (14) requires that no progress is made along any direction, from the instructions of `DF-SIMPLEX` we have that

$$\bar{\varphi}(\bar{y}^k + \alpha(e_{\hat{\jmath}} - e_{\hat{\imath}})) > \bar{\varphi}(\bar{y}^k) - \gamma\alpha^2, \quad k \geq \hat{k},\, k \in K,$$

with $0 < \alpha \leq \epsilon^k$. By the mean value theorem, we can write

$$\bar{\varphi}(\bar{y}^k + \alpha(e_{\hat{\jmath}} - e_{\hat{\imath}})) - \bar{\varphi}(\bar{y}^k) = \alpha \nabla \bar{\varphi}(\bar{y}^k + \eta^k \alpha(e_{\hat{\jmath}} - e_{\hat{\imath}}))^T (e_{\hat{\jmath}} - e_{\hat{\imath}}),$$

for some $\eta^k \in (0,1)$. Then

$$\nabla\bar{\varphi}(\bar{y}^k + \eta^k\alpha(e_{\hat{\jmath}} - e_{\hat{\imath}}))^T(e_{\hat{\jmath}} - e_{\hat{\imath}}) \geq -\gamma\alpha, \quad k \geq \hat{k},\, k \in K.$$

Since $\eta^k \in (0,1)$, $\alpha \leq \epsilon^k$ and $\{\epsilon^k\} \to 0$, we have that

$$\lim_{\substack{k\to\infty \\ k\in K}} (\bar{y}^k + \eta^k\alpha(e_{\hat{\jmath}} - e_{\hat{\imath}})) = y^*.$$

Therefore, from the continuity of $\nabla\bar{\varphi}$ and using again the fact that $\{\epsilon^k\} \to 0$, we obtain that

$$0 \leq \nabla\bar{\varphi}(y^*)^T(e_{\hat{\jmath}} - e_{\hat{\imath}}) = [\bar{A}^T\nabla f(\bar{A}y^*)]^T(e_{\hat{\jmath}} - e_{\hat{\imath}}) = \nabla f(x^*)^T(\bar{A}e_{\hat{\jmath}} - \bar{A}e_{\hat{\imath}}).$$

Let us denote by $\tilde{a}$ the atom that corresponds to the $\hat{\jmath}$-th column of $\bar{A}$, that is, $\bar{A}e_{\hat{\jmath}} = \tilde{a}$ (also recall that $\bar{A}e_{\hat{\imath}} = a$). Then

$$0 \leq \nabla f(x^*)^T(\tilde{a} - a) = \nabla f(x^*)^T(x^* - a) + \nabla f(x^*)^T(\tilde{a} - x^*) \tag{45}$$

Now, consider the vector $w^* \in \Delta_{m-1}$, obtained from y^* by adding the zero components corresponding to the atoms in $\mathcal{A} \setminus \bar{\mathcal{A}}$, so that $Aw^* = \bar{A}y^* = x^*$. We can assume, without loss of generality, that $\tilde{a}$ is also the $\hat{\jmath}$-th column of the full matrix A. Using (44), we can hence write $w^*_{\hat{\jmath}} > 0$. So, from Proposition 8 and stationarity of x^*, we have that $\nabla f(x^*)^T(\tilde{a} - x^*) = 0$. Using this equality in (45), we get $\nabla f(x^*)^T(a - x^*) \leq 0$, thus contradicting (40). □

4.1 Enhancing the *Drop Phase* by gradient estimates

Removing from A^k all the atoms with zero weight might be a too "aggressive" strategy (i.e., some of the atoms removed at the first iterations might be useful in the subsequent iterations). Then, we can define a more sophisticated rule to build $\mathcal{D}^k$ by using approximations of $\nabla\varphi^k(\bar{y}^k)$. In particular, at every iteration k we can set

$$\mathcal{D}^k = \{a \in \mathcal{A}^k \text{ such that } a = A^k e_h,\ \bar{y}^k_h = 0 \text{ and } (g^k)^T(e_h - \bar{y}^k) \geq 0\}, \tag{46}$$

where the vector g^k is an approximation of $\nabla\varphi^k(\bar{y}^k)$ satisfying

$$\|\nabla\varphi^k(\bar{y}^k) - g^k\| \leq r^k, \tag{47}$$

with $\{r^k\}$ being a sequence of positive scalars converging to zero (we will discuss later how to compute g^k efficiently such that (47) holds).

The rationale behind this choice lies in the fact that

$$\nabla\varphi^k(\bar{y}^k)^T(e_h - \bar{y}^k) = [(A^k)^T\nabla f(\bar{x}^k)]^T(e_h - \bar{y}^k) = \nabla f(\bar{x}^k)^T(a - \bar{x}^k),$$

and then a good approximation of $\nabla\varphi^k(\bar{y}^k)$ can help us to predict, in a neighborhood of x^*, the atoms $a \in \mathcal{A}$ such that $\nabla f(x^*)^T(a - x^*) > 0$. We now show that this choice of $\mathcal{D}^k$ ensures the same theoretical properties seen above for (39).

Theorem 5. *Let $\{x^k\}$ be a sequence of points produced by Algorithm 3, where $\mathcal{D}^k$ is computed as in (46). Assume that $\lim_{k\to\infty} x^k \to x^*$. Then, an iteration $\bar{k}$ exists such that, for all $k \geq \bar{k}$,*

$$\nabla f(x^*)^T(a - x^*) > 0,\ a \in \mathcal{A} \ \Rightarrow\ a \notin \mathcal{A}^k.$$

Proof. The first part of the proof is identical to the one given for Theorem 4. Namely, we assume that $a \in \mathcal{A}$ is an atom such that (40) holds and we obtain (41). To prove the desired result, we then proceed by contradiction, assuming that an infinite subset of iterations $K \subseteq \{0, 1 \ldots\}$ exists such that $a \in \mathcal{A}^k$ for all $k \in K$. In view of (41), an iteration $\hat{k} \in K$ must exist such that (42) holds. Now, assuming without loss of generality that $\{y^k\}$ satisfies (43), and using the same definitions of subsequences, matrices and indices given in the proof of Theorem 4, from (46) we have that two possible cases (that will be shown to lead to a contradiction) can occur for $k \geq \hat{k}$, $k \in K$: either (i) $\bar{y}_{\hat{i}}^k > 0$, or (ii) $\bar{y}_{\hat{i}}^k = 0$ and $(g^k)^T(e_{\hat{i}} - \bar{y}^k) < 0$. Since, by the same arguments used in the proof of Theorem 4, the first case cannot occur infinite times, necessarily $\bar{y}_{\hat{i}}^k = 0$ and $(g^k)^T(e_{\hat{i}} - \bar{y}^k) < 0$ for all sufficiently large $k \in K$. Taking into account (47), for all $k \in K$ we can write

$$\begin{aligned} |\nabla\bar{\varphi}(\bar{y}^k)^T(e_{\hat{i}} - \bar{y}^k) - (g^k)^T(e_{\hat{i}} - \bar{y}^k)| &= |(\nabla\bar{\varphi}(\bar{y}^k) - g^k)^T(e_{\hat{i}} - \bar{y}^k)| \\ &\leq \|\nabla\bar{\varphi}(\bar{y}^k) - g^k\|\|e_{\hat{i}} - \bar{y}^k\| \leq \sqrt{2}r^k. \end{aligned}$$

Therefore, for all $k \in K$ we have that

$$\begin{aligned} (g^k)^T(e_{\hat{i}} - \bar{y}^k) &\geq \sqrt{2}r^k + \nabla\bar{\varphi}(\bar{y}^k)^T(e_{\hat{i}} - \bar{y}^k) = \sqrt{2}r^k + [\bar{A}^T\nabla f(\bar{A}\bar{y}^k)]^T(e_{\hat{i}} - \bar{y}^k) \\ &= \sqrt{2}r^k + \nabla f(\bar{x}^k)(\bar{A}e_{\hat{i}} - \bar{A}\bar{y}^k) = \sqrt{2}r^k + \nabla f(\bar{x}^k)(a - \bar{x}^k). \end{aligned}$$

From the continuity of ∇f and the fact that $\{r^k\} \to 0$, taking the limits we obtain

$$\liminf_{\substack{k\to\infty \\ k\in K}} (g^k)^T(e_{\hat{i}} - \bar{y}^k) \geq \nabla f(x^*)^T(a - \bar{x}^*) > 0,$$

leading to a contradiction with the fact that $(g^k)^T(e_{\hat{i}} - \bar{y}^k) < 0$ for all $k \in K$. □

Now, we describe how to compute g^k in such a way that condition (47) is satisfied. Since point $\bar{y}^k$ is obtained in the *Optimize Phase* by running `DF-SIMPLEX` with a tolerance ϵ^k, we can simply use the sample points produced in the last iteration of `DF-SIMPLEX` plus one additional sample point not belonging to $\Delta_{|\mathcal{A}^k|-1}$, that is $\bar{y}^k - \epsilon_k \frac{\sqrt{2}}{|\mathcal{A}^k|}e$, to perform a simplex gradient computation in $\mathbb{R}^{|\mathcal{A}^k|}$ (see, e.g., [29] for definition of simplex gradient). The last sample point is needed to have a poised sample set. More in detail, let $\bar{y}^k, s_1^k, \ldots, s_r^k$ be all the available sample points, with $r \geq |\mathcal{A}^k|$, and let us denote $Y^k = \{\bar{y}^k, s_1^k, \ldots, s_r^k\}$. Moreover, let

$$S^k = \begin{bmatrix} s_1^k - \bar{y}^k & \ldots & s_r^k - \bar{y}^k \end{bmatrix}, \qquad b^k = \begin{bmatrix} \varphi^k(s_1^k) - \varphi^k(\bar{y}^k) & \ldots & \varphi^k(s_r^k) - \varphi^k(\bar{y}^k) \end{bmatrix}^T.$$

We compute g^k as the least-squares solution of $(S^k)^T g = b^k$. Under the assumption that ∇f is Lipschitz continuous with constant L, if the sample set Y^k is poised (i.e., if the columns of $(S^k)^T$ are linearly independent) from Theorem 3.1 in [18] it follows that $\|\nabla\varphi^k(\bar{y}^k) - g^k\| \leq \left(|\mathcal{A}^k|^{1/2}\frac{L}{2}\left\|(\Sigma^k)^{-1}\right\|\right)\nu^k$, where ν^k is the radius of the smallest ball centered at $\bar{y}^k$ enclosing the points $s_1^k, \ldots, s_r^k$, and Σ^k is obtained from the reduced singular value decomposition of S^T/ν^k, that is, $S^T/\nu^k = U^k\Sigma^k(V^k)^T$, for proper matrices U^k and V^k.

In our case, $\nu^k = \sqrt{2}\epsilon^k$ for all sufficiently large k (it follows from the stopping condition used in `DF-SIMPLEX` combined with the fact that $\{\epsilon^k\} \to 0$ and the fact that all the directions have norm equal to $\sqrt{2}$). Clearly, $\nu^k \to 0$ as $\epsilon^k \to 0$. Moreover, it is easy to see that Y^k is poised (it follows from the fact that `DF-SIMPLEX` uses directions of the form $\pm(e_i - e_{j_k})$ and we also considered an additional sample point along the direction $-e$). Using the notion of Λ-poisedness as given in [14, 15], it is also easy to see that $\left\|(\Sigma^k)^{-1}\right\|$ is upper bounded by a constant Λ for all sufficiently large iterations.[1]

[1]We can identify $\tilde{Y}^k \subseteq Y^k$, with $|\tilde{Y}^k| = |\mathcal{A}^k|$, such that $\tilde{Y}^k$ is $\tilde{\Lambda}$-poised in the ball centered at $\bar{y}^k$ with radius ν^k, and this implies that Y^k is Λ-poised in the same ball with $\Lambda = |\mathcal{A}^k|^{1/2}\tilde{\Lambda}$ (see [16], pag. 63), which, in turn, implies that Y^k is poised and, from Theorem 2.9 in [15], that $\left\|(\Sigma^k)^{-1}\right\| \leq |\mathcal{A}^k|^{1/2}\tilde{\Lambda} \leq m\tilde{\Lambda}$.

5 Numerical experiments

In this section, we analyze in depth the practical performances of the `ORD` algorithm. We carried out all our tests in MATLAB R2020b on an Intel(R) Core(TM) i7-9700 with 16 GB RAM memory, and used data and performance profiles [40] when comparing the method with other algorithms. Specifically, let S be a set of algorithms and P a set of problems. For each $s \in S$ and $p \in P$, let $t_{p,s}$ be the number of function evaluations required by algorithm s on problem p to satisfy the condition

$$f(x_k) \leq f_L + \tau(f(x_0) - f_L) \tag{48}$$

where $0 < \tau < 1$ and f_L is the best objective function value achieved by any solver on problem p. Then, data and performance profiles of solver s are respectively defined as follows:

$$\begin{aligned} d_s(\kappa) &= \frac{1}{|P|} \left| \{p \in P : t_{p,s} \leq \kappa(n_p + 1)\} \right|, \\ \rho_s(\iota) &= \frac{1}{|P|} \left| \left\{ p \in P : \frac{t_{p,s}}{\min\{t_{p,s'} : s' \in S\}} \leq \iota \right\} \right|, \end{aligned}$$

where n_p is the dimension of problem p.

5.1 Preliminary results

We first chose the following 25 objective functions from the literature (see, e.g., [1, 23]): Arwhead, Cosine, Cube, Diagonal 8, Extended Beale, Extended Cliff, Extended Denschnb, Extended Denschnf, Extended Freudenstein & Roth, Extended Hiebert, Extended Himmelblau, Extended Maratos, Extended Penalty, Extended PSC1, Extended Rosenbrock, Extended Trigonometric, Extended White & Holst, Fletchcr, Genhumps, Mccormk, Power, Quartc, Sine, Staircase 1, Staircase 2. Then, we built the test problems by randomly generating the atoms with a uniform distribution in $[0, 10]^n$. We would like to highlight that there was no relevant redundancy in the generated atoms. In cases where the atoms in $\mathcal{A}$ are highly redundant, it is possible to remove useless atoms by solving a sequence of linear programs. This redundancy test might anyway have a significant computational cost (especially when both the dimension of the problem and the number of the atoms are large).

In the first experiment, we compared `ORD` with the following algorithms:

- `DF-SIMPLEX`, the solver proposed in Section 2 for minimization over the unit simplex;
- `LINCOA` [43], a trust-region based solver for linearly constrained problems[2];
- `NOMAD` (v3.9.1) [3, 4], a solver for non-linearly constrained problems implementing the Mesh Adaptive Direct Search algorithm (MADS);
- `PSWARM` [47], a global optimization solver for linearly constrained problems combining pattern search and particle swarm;
- `SDPEN` [36], a solver for non-linearly constrained problems based on a sequential penalty approach.

When running our tests on `DF-SIMPLEX` and `LINCOA`, we used formulation $(P1)$ to represent the problems (note that $\bar{m} = m$ for `DF-SIMPLEX` in this case). Since `PSWARM` and `SDPEN` only handle inequality constraints, they were run by suitably rewriting $(P1)$ as an inequality constrained problem.

[2]We would like to thank Tom M. Ragonneau and Zaikun Zhang for kindly sharing their MATLAB interface for the `LINCOA` software.

Namely, we used the substitution $y_1 = 1 - \sum_{i=2}^{n} y_i$ to eliminate the variable y_1, so that the new problem only has the constraints $\sum_{i=2}^{n} y_i \leq 1$ and $y_i \geq 0$, $i = 2, \ldots, n$.

We considered two different versions for NOMAD. The first one, referred to as NOMAD 1, uses the same formulation as the one used for PSWARM and SDPEN. The second one, referred to as NOMAD 2, considers the formulation ($P0$) and works in the original space $\mathbb{R}^n$ using a non-quantifiable black box constraint that only indicates if x belongs to $\mathcal{M}$ or not (this is carried out by solving a linear program).

We are interested in analyzing the performances of the algorithms for different ratios m/n, with m the number of atoms and n the number of variables. Notice that this might affect the sparsity of the final solution (i.e., the number of atoms needed to assemble x^*). In particular, from Carathéodory's theorem [9] we expect that the larger the ratio m/n, the sparser the solution. ORD should hence be more efficient than the competitors for larger values of m/n.

So, we fix $n = 10$ and set $m \in \{n, 5n, 10n, 20n\}$. In ORD we stopped the algorithm at the first iteration k that fails the test at line 6 of Algorithm 3 and such that

$$\hat{\mu}^k \leq \frac{10^{-4}}{\max_{a_i \in \mathcal{A} \setminus \mathcal{A}^k} \|a_i - \bar{x}^k\|}.$$

In DF-SIMPLEX we used the stopping condition described in Subsection 2.2, with $\epsilon = 10^{-4}$. In all the other algorithms, the parameters were set to their default values. Moreover, we used a budget of $100(n+1)$ function evaluations for every algorithm and we set the starting point as a randomly chosen vertex of Δ_{n-1}.

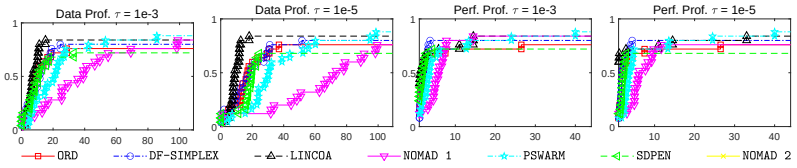
We report, in Figure 1, the data and performance profiles related to the experiment. Taking a look at the plots, we see that ORD clearly outperforms the competitors as the ratio m/n increases (and we get a sparser solution). More specifically, the average sparsity levels (i.e., the average percentage of atoms with zero weight) of the solutions found by ORD are 62.00% for $m = n$, 87.92% for $m = 5n$, 92.68% for $m = 10n$ and 96.08% for $m = 20n$.

In the second experiment, we considered the largest ratio m/n, obtained with $m = 20n$, and set the value of n to 20 and 50. For these new experiments, we decided to only run NOMAD 2. There are two main reasons why we did that. First, NOMAD 2 works in the original n-dimensional space, while NOMAD 1 works in an m-dimensional space (20 times larger than n in these experiments). Second, the maximum number of variables that NOMAD can handle is 1000, hence there is no way to run NOMAD 1 on the largest problems anyway.

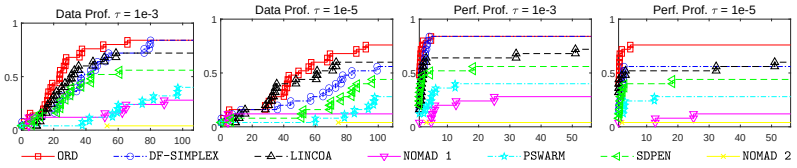
In Figure 2 we report the data and performance profiles related to the new experiment, only including the four solvers that got the best performances, that is, ORD, DF-SIMPLEX, LINCOA and SDPEN. We see that ORD clearly outperforms the other solvers. We would also like to notice that the average running time for ORD and DF-SIMPLEX, both written in MATLAB, is smaller than 0.1 seconds for $n = 20$ and smaller than 1 second for $n = 50$. It is the same order of magnitude as SDPEN, but is much smaller than LINCOA, which on average took about 50 seconds for $n = 20$ and about 650 seconds for $n = 50$.

In the final experiment, the aim was to analyze the behavior of the ORD algorithm on relatively large-scale instances. We thus considered once again the largest ratio $m/n = 20$ and set the value of n to 100, 200 and 500. Taking into account the previous results, we only compared ORD with DF-SIMPLEX in this case.

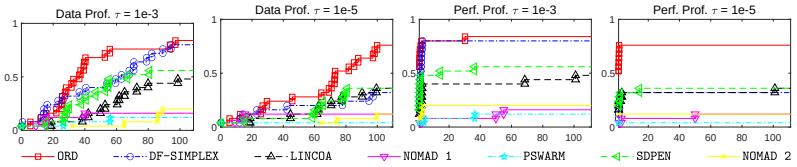
The data and performance profiles related to the comparisons, reported in Figure 3, confirm once again the effectiveness of ORD. In this case, we observed an increased difference between the two considered algorithms in the CPU time required to solve the problem: ORD on average took about 3 seconds for $n = 100$, about 30 seconds for $n = 200$ and less than 490 seconds for $n = 500$, while DF-SIMPLEX took less than 1 second for each problem with $n \in \{100, 200\}$ and on average less than 3 seconds for the problems with $n = 500$. This difference is mainly due to the computation of the simplex gradient that ORD performs in the *Drop Phase*. Anyway, ORD never exceeded 490 seconds for solving a problem.



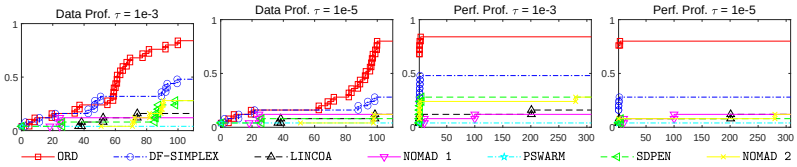
(a) $n = 10$, $m = 10$



(b) $n = 10$, $m = 50$

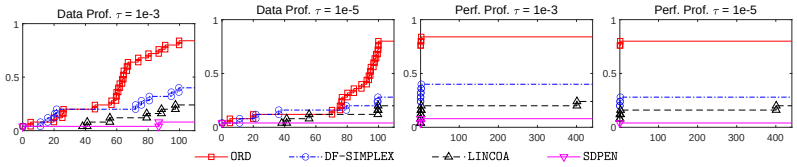


(c) $n = 10$, $m = 100$

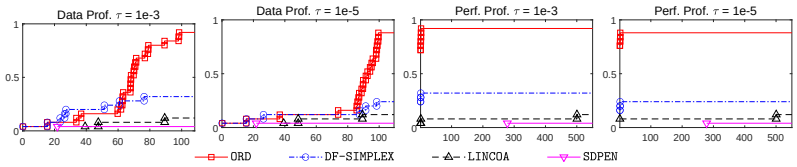


(d) $n = 10$, $m = 200$

Figure 1: Comparisons among `ORD`, `DF-SIMPLEX`, `LINCOA`, `NOMAD 1`, `PSWARM`, `SDPEN` and `NOMAD 2` for different ratios m/n.

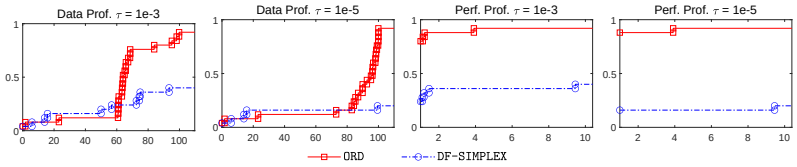


(a) $n = 20$, $m = 400$

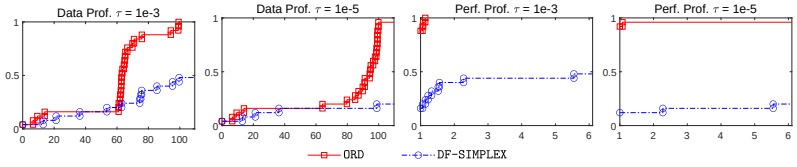


(b) $n = 50$, $m = 1,000$

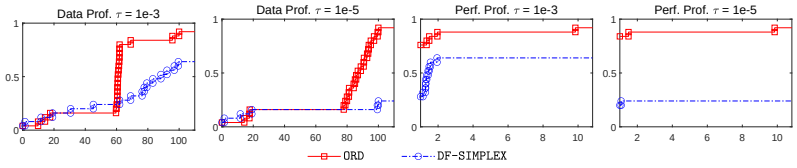
Figure 2: Comparisons among `ORD`, `DF-SIMPLEX`, `LINCOA` and `SDPEN` for different values of n and $m = 20n$.



(a) $n = 100$, $m = 2,000$



(b) $n = 200$, $m = 4,000$



(c) $n = 500$, $m = 10,000$

Figure 3: Comparisons between `ORD` and `DF-SIMPLEX` on large-scale instances.

The numerical experiments demonstrate that the methods exploiting the structure of the feasible region (i.e., `ORD`, `DF-SIMPLEX` and `LINCOA`) outperform the others. This is not surprising as the latter methods are designed to tackle more general optimization problems.

5.2 Black-box adversarial machine learning

Adversarial examples are maliciously perturbed inputs designed to mislead a machine learning model at test time. In many fields, such as sign identification for autonomous driving, the vulnerability of a model to such examples might have relevant security implications. An *adversarial attack* hence consists in taking a correctly classified data point x_0 and slightly modifying it to create a new data point that leads a given model to misclassification (see, e.g., [10, 13, 22] for further details).

We now consider a classifier that takes a vector $x \in \mathbb{R}^n$ as an input and outputs $F(x) \in \mathbb{R}^p$, where $[F(x)]_i \in [0, 1]$ represents the *confidence score* for class $i = 1, \ldots, p$, i.e., the predicted probability that x belongs to that class, and $\sum_{i=1}^{p} [F(x)]_i = 1$.

In many real-world applications, the internal configuration of such a classifier is unknown, and one can only access its input and output, i.e., one can only compute $F(x)$. In this case, we can perform a so-called *black-box adversarial attack* on the model [12, 13].

We formulate our problem as a *maximum allowable attack* [12, 22], namely,

$$\begin{aligned} &\min\ f(x_0 + x) \\ &s.t.\quad \|x\|_p \leq \varepsilon, \end{aligned} \tag{49}$$

where f is a suitably chosen attack loss function, x_0 is a correctly classified data point, x is the additive noise/perturbation, $\varepsilon > 0$ denotes the magnitude of the attack, and $p \geq 1$. We set $p = 1$ in the formulation (49), thus getting a *maximum allowable ℓ_1-norm attack*. It is easy to see that $\mathcal{M} = \{x \in \mathbb{R}^n : \|x\|_1 \leq \varepsilon\} = conv(\mathcal{A})$, with $\mathcal{A} = \{\pm \varepsilon e_i, i = 1, \ldots, n\}$, i.e., $\mathcal{M}$ is a polytope with $2n$ vertices (and then, $m = 2n$). This makes the problem fitting our model $(P0)$, and also gets sparsity

in the final solution. We focus on *untargeted attacks*, i.e., we aim to move a data point away from its current class, and use the loss function proposed in [13]:

$$f(z) = \max\{\log[F(z)]_{t_0} - \max_{i \neq t_0} \log[F(z)]_i, -\chi\}, \tag{50}$$

where t_0 is the original class, χ is a non-negative parameter and $\log 0$ is defined as $-\infty$. The rationale behind the use of this loss function is that, when $\log[F(z)]_{t_0} - \max_{i \neq t_0} \log[F(z)]_i \leq 0$, the sample z is not classified as the original label t_0, thus obtaining the desired misclassification. Moreover, the parameter χ can ensure a gap between $\log[F(z)]_{t_0}$ and $\max_{i \neq t_0} \log[F(z)]_i$.

In our experiments related to adversarial attacks, we set $\chi = 0$ for the loss function (50), as in [10, 13], and chose the parameter ε in problem (49) by means of a parameter selection, using up to 20 different values. We obtained ε values in the range $[0.0012n, 0.5059n]$. We thus solved (49) using `ORD` (our best solver in the preliminary experiments), `LINCOA` and `SDPEN` (the best competitors in the preliminary experiments). Note that an attack is successful only when the objective value is equal to χ, i.e., equal to 0 in our case. Therefore, in all the algorithms we inhibited any other stopping criterion (that we are allowed to control) and set the maximum number of objective function evaluations equal to $100(n+1)$. Moreover, we set the target objective value equal to 0 for both `ORD` and `LINCOA` (this option is not available for `SDPEN`). It is important to notice that for all the successful attacks we found solutions with a number of non-zero entries smaller than 3%.

5.2.1 Adversarial attacks on binary logistic regression models

First, we performed untargeted black-box attacks on binary logistic regression models. We used all the datasets from the LIBSVM web page (`https://www.csie.ntu.edu.tw/~cjlin/libsvm/`) with a number of features between 100 and $2,000$ and a number of training samples less than $50,000$. Here is the complete list: a1a, a2a, a3a, a4a, a5a, a6a, a7a, a8a, a9a, colon-cancer, madelon, mushrooms, w1a, w2a, w3a, w4a, w5a, w6a, w7a and w8a.

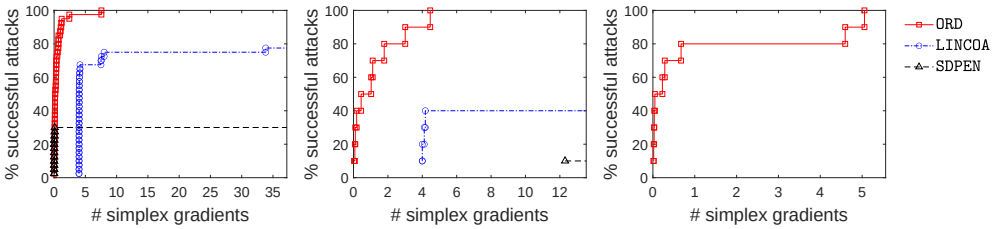
We used the training set to build an ℓ_2-regularized logistic regression model by means of the LIBLINEAR software [20] (a built-in cross validation was used to choose the regularization parameter) for all the 20 datasets. Then, we randomly selected, for each class and each dataset, a correctly classified test sample x_0 and used it in problem (49), thus getting 40 adversarial attacks. A built-in LIBLINEAR function was used to compute the probability estimates $[F(x)]_1$ and $[F(x)]_2$ in the loss function (50).

In Table 1a, we report, for each solver, the percentage of successful attacks and the average CPU time (in seconds). We further report, in Figure 4a, the percentage of successful attacks versus the required number of simplex gradients. We see that `ORD` solves all the problems within a few function evaluations, while `LINCOA` and `SDPEN` only solve 77.50% and 30.00% of the problems, respectively. Moreover the CPU time for `ORD` is always less than 1 second and, on average, is smaller than `LINCOA` and `SDPEN` of 4 and 2 orders of magnitude, respectively.

5.2.2 Adversarial attacks on deep neural networks

In the second experiment, we considered images of handwritten digits from the MATLAB Digits Dataset. This dataset has $10,000$ 28-by-28 grayscale images of all digits, divided into 10 classes of 1000 samples each. The dataset was randomly split using a ratio $90:10$ into training and testing set. The training set was then used to build a deep neural network with the same architecture as the one described in the examples related to deep learning networks for classification available in MATLAB (see `https://it.mathworks.com/help/deeplearning/ug/create-simple-deep-learning-network-for-classification.html` for further details).

We performed untargeted attacks on this deep neural network using `ORD`, `LINCOA` and `SDPEN` (notice that $n = 784$ and $m = 1568$ in this case). For each class, we randomly selected a correctly classified



(a) Binary logistic regression models (b) MATLAB Digits Dataset (c) Cifar-10 Dataset

Figure 4: Adversarial attacks on binary logistic regression models (a), on the MATLAB Digits Dataset (b), and on the Cifar-10 Dataset (c): percentage of successful attacks vs number of simplex gradients.

sample x_0 from the validation set and used it in the definition of problem (49). Note that each pixel must be a number in the interval $[0, 255]$. We hence scaled each variable in the range $[0, 1]$. In this case, our formulation (49) has a further set of constraints, that is $x_0 + x \in [0, 1]^n$. In order to get rid of those box constraints, we followed the approach described in [10, 13], and used a transformation of the form $x_i = (1 + \tanh \zeta_i)/2 - (x_0)_i$, with $\zeta \in \mathbb{R}^n$.

In Table 1b, we report the percentage of successful attacks and the average CPU time for each solver. We further report, in Figure 4b, the percentage of successful attacks versus the required number of simplex gradients. We see that `ORD` gets a 100% success rate with an average CPU time of around 1.5 seconds and a small amount of simplex gradients, while `LINCOA` and `SDPEN` have a success rate lower than 50% and a much larger CPU time.

In Figure 5a, we can see the images obtained with all the attacks applied by `ORD`. We notice that the new images are overall very similar to the original ones, differing, on average, in less than 0.7% of the pixels.

Table 1: Adversarial attacks: performance comparison of the DFO methods

(a) Binary logistic regression models

Alg.	Success rate	Avg time (s)
`ORD`	100.00%	0.04
`LINCOA`	77.50%	421.78
`SDPEN`	30.00%	6.80

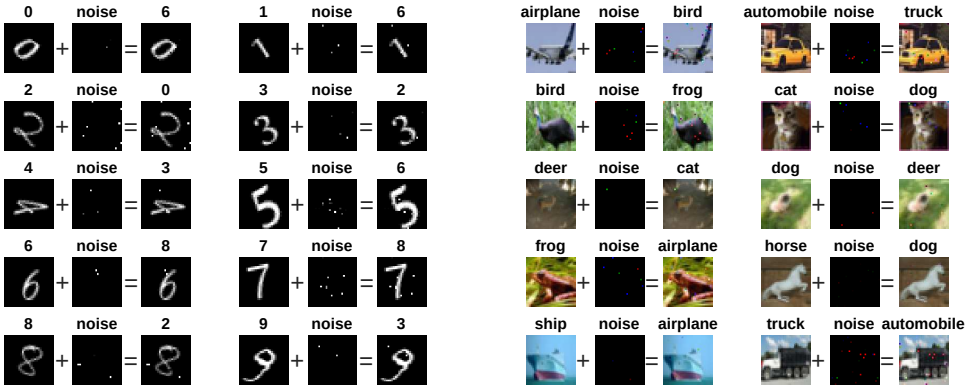
(b) MATLAB Digits Dataset

Alg.	Success rate	Avg time (s)
`ORD`	100.00%	1.48
`LINCOA`	40.00%	1125.93
`SDPEN`	10.00%	129.69

Finally, we considered the Cifar-10 dataset [32] and the trained network described in the MATLAB examples related to the training of residual networks for image classification (for details see `https://it.mathworks.com/help/deeplearning/ug/train-residual-network-for-image-classification.html`). The dataset contains $50,000$ samples in the training set and $10,000$ samples in the validation set, where each image is 32-by-32 with 3 color channels (thus getting $n = 3072$ and $m = 6144$).

We performed untargeted adversarial attacks on this deep neural network only using `ORD` (due to the large dimension of the problems, `LINCOA` and `SDPEN` do not give good results in terms of success rate and/or CPU time). We used the same procedure as the one used for the attacks on the MATLAB Digits Dataset.

We report, in Figure 4c, the percentage of successful attacks versus the required number of simplex gradients. We can see that `ORD` achieves 100% success rate (with an average CPU time of slightly



(a) MATLAB Digits Dataset (b) Cifar-10 Dataset

Figure 5: Adversarial attacks applied by `ORD` on the MATLAB Digits Dataset (a) and on the Cifar-10 Dataset (b). In each triple, we have on the left the original image with the correct label, on the right the new image with the misclassified label and in the middle the additive noise.

more than 30 seconds) using a few simplex gradients. In Figure 5b, we can see the images obtained with all the attacks applied by `ORD`. They are quite similar to the original ones, differing in about 0.2% of the pixels on average.

Acknowledgments

The authors would like to thank the two anonymous reviewers for their comments and suggestions that helped to improve the paper.

References

[1] N. Andrei. An unconstrained optimization test functions collection. *Adv. Model. Optim*, 10(1): 147–161, 2008.

[2] C. Audet and W. Hare. *Derivative-free and blackbox optimization*. Springer, 2017.

[3] C. Audet, S. L. Digabel, C. Tribes, and V. R. Montplaisir. The NOMAD project. Software available at URL `https://www.gerad.ca/nomad`.

[4] C. Audet, S. Le Digabel, and C. Tribes. NOMAD user guide. Technical Report G-2009-37, Les cahiers du GERAD, 2009. URL `https://www.gerad.ca/nomad/Downloads/user_guide.pdf`.

[5] D. Avis, D. Bremner, and R. Seidel. How good are convex hull algorithms? *Computational Geometry*, 7(5-6):265–301, 1997.

[6] F. Bach, R. Jenatton, J. Mairal, G. Obozinski, et al. Convex optimization with sparsity-inducing norms. *Optimization for Machine Learning*, 5:19–53, 2011.

[7] D. P. Bertsekas. *Convex optimization algorithms*. Athena Scientific, 2015.

[8] W. Brendel, J. Rauber, and M. Bethge. Decision-based adversarial attacks: Reliable attacks against black-box machine learning models. *In International Conference on Learning Representations*, 2018.

[9] C. Carathéodory. Über den variabilitätsbereich der koeffizienten von potenzreihen, die gegebene werte nicht annehmen. *Math. Ann.*, 64(1):95–115, 1907.

[10] N. Carlini and D. Wagner. Towards evaluating the robustness of neural networks. In *2017 IEEE symposium on security and privacy (sp)*, pages 39–57. IEEE, 2017.

[11] V. Chandrasekaran, B. Recht, P. A. Parrilo, and A. S. Willsky. The convex geometry of linear inverse problems. *Found. Comput. Math.*, 12(6):805–849, 2012.

[12] J. Chen, D. Zhou, J. Yi, and Q. Gu. A Frank-Wolfe Framework for Efficient and Effective Adversarial Attacks. *arXiv*, pages arXiv–1811, 2018.

[13] P.-Y. Chen, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh. ZOO: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models. In *Proceedings of the 10th ACM Workshop on Artificial Intelligence and Security*, pages 15–26, 2017.

[14] A. R. Conn, K. Scheinberg, and L. N. Vicente. Geometry of interpolation sets in derivative free optimization. *Math. Program.*, 111(1-2):141–172, 2008.

[15] A. R. Conn, K. Scheinberg, and L. N. Vicente. Geometry of sample sets in derivative-free optimization: polynomial regression and underdetermined interpolation. *IMA J. Numer. Anal.*, 28(4):721–748, 2008.

[16] A. R. Conn, K. Scheinberg, and L. N. Vicente. *Introduction to derivative-free optimization*, volume 8. Siam, 2009.

[17] A. Cristofari. An almost cyclic 2-coordinate descent method for singly linearly constrained problems. *Comput. Optim. Appl.*, 73(2):411–452, 2019.

[18] A. L. Custódio and L. N. Vicente. Using sampling and simplex derivatives in pattern search methods. *SIAM J. Optim.*, 18(2):537–555, 2007.

[19] Y. Diouane, S. Gratton, and L. N. Vicente. Globally convergent evolution strategies for constrained optimization. *Comput. Optim. Appl.*, 62(2):323–346, 2015.

[20] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin. LIBLINEAR: A library for large linear classification. *J. Mach. Learn. Res.*, 9(Aug):1871–1874, 2008.

[21] Z. Fan, H. Jeong, Y. Sun, and M. P. Friedlander. Atomic Decomposition via Polar Alignment: The Geometry of Structured Optimization. *Found. Trends Optim.*, 3(4):280–366, 2020.

[22] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[23] N. I. Gould, D. Orban, and P. L. Toint. CUTEst: a constrained and unconstrained testing environment with safe threads for mathematical optimization. *Comput. Optim. Appl.*, 60(3): 545–557, 2015.

[24] S. Gratton, C. W. Royer, L. N. Vicente, and Z. Zhang. Direct search based on probabilistic feasible descent for bound and linearly constrained problems. *Comput. Optim. Appl.*, 72(3): 525–559, 2019.

[25] E. A. Gumma, M. Hashim, and M. M. Ali. A derivative-free algorithm for linearly constrained optimization problems. *Comput. Optim. Appl.*, 57(3):599–621, 2014.

[26] D. W. Hearn, S. Lawphongpanich, and J. A. Ventura. Restricted simplicial decomposition: Computation and extensions. In *Computation Mathematical Programming*, pages 99–118. Springer, 1987.

[27] A. Ilyas, L. Engstrom, A. Athalye, and J. Lin. Black-box Adversarial Attacks with Limited Queries and Information. In *Int. Conf. on Machine Learning*, pages 2137–2146, 2018.

[28] L. P. Kaelbling, M. L. Littman, and A. W. Moore. Reinforcement learning: A survey. *J. Artificial Intelligence Res.*, 4:237–285, 1996.

[29] C. T. Kelley. *Iterative methods for optimization*. SIAM, 1999.

[30] T. G. Kolda, R. M. Lewis, and V. Torczon. Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Rev.*, 45(3):385–482, 2003.

[31] T. G. Kolda, R. M. Lewis, and V. Torczon. Stationarity results for generating set search for linearly constrained optimization. *SIAM J. Optim.*, 17(4):943–968, 2007.

[32] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. *Technical*

report, 2009.

[33] J. Larson, M. Menickelly, and S. M. Wild. Derivative-free optimization methods. *Acta Numer.*, 28:287–404, 2019.

[34] R. M. Lewis and V. Torczon. Pattern search methods for linearly constrained minimization. *SIAM J. Optim.*, 10(3):917–941, 2000.

[35] R. M. Lewis and V. Torczon. Active set identification for linearly constrained minimization without explicit derivatives. *SIAM J. Optim.*, 20(3):1378–1405, 2010.

[36] G. Liuzzi, S. Lucidi, and M. Sciandrone. Sequential penalty derivative-free methods for nonlinear constrained optimization. *SIAM J. Optim.*, 20(5):2614–2635, 2010.

[37] S. Lucidi and M. Sciandrone. A derivative-free algorithm for bound constrained optimization. *Comput. Optim. Appl.*, 21(2):119–142, 2002.

[38] S. Lucidi and M. Sciandrone. On the global convergence of derivative-free methods for unconstrained optimization. *SIAM J. Optim.*, 13(1):97–116, 2002.

[39] S. Lucidi, M. Sciandrone, and P. Tseng. Objective-derivative-free methods for constrained optimization. *Math. Program.*, 92(1):37–59, 2002.

[40] J. J. Moré and S. M. Wild. Benchmarking derivative-free optimization algorithms. *SIAM J. Optim.*, 20(1):172–191, 2009.

[41] A. Y. Ng and M. Jordan. PEGASUS: a policy search method for large MDPs and POMDPs. In *Proceedings of the Sixteenth conference on Uncertainty in artificial intelligence*, pages 406–415, 2000.

[42] M. Patriksson. *Nonlinear programming and variational inequality problems: a unified approach*, volume 23. Springer Science & Business Media, 2013.

[43] M. J. Powell. LINCOA. `https://en.wikipedia.org/wiki/LINCOA`. Accessed 11 May 2020.

[44] M. J. Powell. The NEWUOA software for unconstrained optimization without derivatives. In *Large-scale nonlinear optimization*, pages 255–297. Springer, 2006.

[45] M. J. Powell. Developments of NEWUOA for minimization without derivatives. *IMA J. Numer. Anal.*, 28(4):649–664, 2008.

[46] M. J. Powell. On fast trust region methods for quadratic models with linear constraints. *Math. Program. Comput.*, 7(3):237–267, 2015.

[47] A. I. F. Vaz and L. N. Vicente. A particle swarm pattern search method for bound constrained global optimization. *J. Global Optim.*, 39(2):197–219, 2007.

[48] A. I. F. Vaz and L. N. Vicente. PSwarm: a hybrid solver for linearly constrained global derivative-free optimization. *Optim. Methods Softw.*, 24(4-5):669–685, 2009.

[49] E. H. Zarantonello. Projections on Convex Sets in Hilbert Space and Spectral Theory. In *Contributions to nonlinear functional analysis*, pages 237–424. Elsevier, 1971.