Direct search based on probabilistic descent in reduced spaces

Lindon Roberts* Clément W. Royer†

January 16, 2023

Abstract

Derivative-free algorithms seek the minimum value of a given objective function without using any derivative information. The performance of these methods often worsen as the dimension increases, a phenomenon predicted by their worst-case complexity guarantees. Nevertheless, recent algorithmic proposals have shown that incorporating randomization into otherwise deterministic frameworks could alleviate this effect for direct-search methods. In particular, the best guarantees and practical performance were obtained for direct-search schemes using a random vector uniformly distributed on the sphere and its negative at every iteration. This approach effectively draws directions from a random one-dimensional subspace, yet the properties of such subspaces have not been exploited in direct search, unlike for other derivative-free schemes. Moreover, existing theory is by design limited to bounded directions, and thus does not fully account for the numerous possibilities for generating random directions (such as drawing from a Gaussian distribution).

In this paper, we study a generic direct-search algorithm in which the polling directions are defined using random subspaces. Complexity guarantees for such an approach are derived thanks to probabilistic properties related to both the subspaces and the directions used within these subspaces. Our analysis crucially extends previous deterministic and probabilistic arguments by relaxing the need for directions to be deterministically bounded in norm. As a result, our approach encompasses a wide range of new optimal polling strategies that can be characterized using our subspace and direction properties. By leveraging results on random subspace embeddings and sketching matrices, we show that better complexity bounds are obtained for randomized instances of our framework. A numerical investigation confirms the benefit of randomization, particularly when done in subspaces, when solving problems of moderately large dimension.

1 Introduction

This paper is concerned with solving the following unconstrained optimization problem:

$$\min_{\boldsymbol{x}\in\mathbb{R}^n} f(\boldsymbol{x}), \tag{1}$$

*Mathematical Sciences Institute, Building 145, Science Road, Australian National University, Canberra ACT 2601, Australia (lindon.roberts@anu.edu.au).

†LAMSADE, CNRS, Université Paris Dauphine-PSL, Place du Maréchal de Lattre de Tassigny, 75016 Paris, France (clement.royer@lamsade.dauphine.fr). Funding for this author's research was partially provided by CNRS INS2I under the grant GASCON and by Agence Nationale de la Recherche through program ANR-19-P3IA-0001 (PRAIRIE 3IA Institute).

where $f : \mathbb{R}^n \to \mathbb{R}$ is a continuously differentiable function. We suppose that the derivative of f is unavailable for algorithmic purposes, thereby precluding the use of standard nonlinear optimization techniques. Problems of this form commonly arise in complex engineering problems where an expensive simulation code is to be calibrated [1]. Tackling such problems is the goal of derivative-free optimization. Over the past decades, this field has given rise to well-established algorithms, that have been successfully applied to real-world engineering problems [1, 14, 28]. Derivative-free optimization methods can be broadly classified into two categories: model-based algorithms, that construct a model of the objective function to guide the optimization process into selecting a new point to evaluate, and direct-search methods, that proceed by exploring the space along suitably chosen directions.

A recent trend in nonlinear optimization, and in derivative-free optimization in particular, is to compare optimization methods based on their complexity guarantees. In the context of derivative-free optimization, we are interested in the number of function evaluations required by these methods to reach a point $\boldsymbol{x}$ such that $\|\nabla f(\boldsymbol{x})\| \leq \epsilon$, where $\epsilon > 0$ is a given tolerance. Recent results in worst-case complexity analyzes have established that this number is of order $n^2\epsilon^{-2}$ for standard deterministic derivative-free frameworks such as trust region [18] and direct search with sufficient decrease [34], as well as $n^2\epsilon^{-3/2}$ for cubic regularization based on finite-difference estimates [11]. In the case of direct search, it was shown that the factor n^2 could not be improved upon using deterministic algorithms [15].

Despite this negative result, a number of recent algorithmic proposals based on randomized techniques have been shown to reduce the dependency in the dimension from n^2 to n. A direct-search method based on random directions was endowed with a complexity bound of order $n\epsilon^{-2}$ in a probabilistic sense, provided the directions were chosen uniformly on the unit sphere [20]. In particular, choosing two opposite directions was identified as the best choice from a theoretical viewpoint (in that it maximizes the probability of having a good direction among two uniformly drawn), that also yielded the best results in numerical experiments [20, 22]. Similar results were obtained through a different analysis for zeroth-order methods that constructed a gradient estimate, typically based on Gaussian sampling [4, 29, 30]. Very recently, Kozak et al. [26] also proposed to use directional derivatives sampled within random orthogonal subspaces to approximate the gradient. In a related, subsequent work, Kozak et al. [27] described a variant of this approach using finite difference estimates based on orthogonal directions drawn from random subspaces. In all the aforementioned approaches, a linear dependence in the problem dimension could be identified, representing an improvement over the deterministic setting. Note, however, that the seemingly general analysis proposed in the direct-search setting [20] did not include direction choices such as Gaussian vectors.

The situation was different in the model-based framework. Although randomized frameworks were proposed based on the same reasoning as direct-search methods, the complexity analysis did not suggest any possible improvement in terms of the problem dimension [2, 21]. However, recent approaches focusing on constructing models in random, low-dimensional subspaces did manage to achieve such a theoretical improvement [12] by leveraging random embeddings using for instance Gaussian or random orthogonal matrices. Such techniques appeared as promising to develop scalable derivative-free optimization techniques, which was done by Cartis et al. [7, 12]. Their approach builds on a general framework for derivative-based optimization in random subspaces [9, 10, 33], and adapts it to a derivative-free, model-based setting. However, the connection between the subspace arguments in the model-based literature and the use of random one-dimensional subspaces in direct search was not investigated.

In this paper, we revisit the analysis of direct-search methods based on probabilistic properties in order to improve our understanding of their behavior. To this end, we propose a framework that relies on search directions chosen within subspaces of the variable space. We define probabilistic properties on these subspaces and directions, by combining elements from the direct-search literature and that of model-based methods based on random subspaces. Such properties are then combined to yield complexity bounds for our framework, in a way that departs from existing analyses [20]. In particular, our reasoning allows for unbounded directions, and thus encompasses popular practical choices such as Gaussian directions. This decoupling of the subspace and direction generation enables us to introduce a suite of new methods that all match the best known bound in terms of dependencies on n, and find a new interpretation of direct search based on opposite, random directions. As a result, our framework for direct search together with the model-based framework [12] provides a coherent understanding of the benefits of randomization for scalable derivative-free optimization of nonconvex functions, resolving the disconnect between randomized direct-search and randomized model-based techniques that was previously observed [20, 21]. Our experiments confirm that a randomized subspace approach can be quite beneficial in a direct-search setting.

The rest of this document is organized as follows. In Section 2, we recall the main features of direct-search methods, and the associated complexity guarantees. We then describe a new paradigm to generate polling directions based on subspaces in Section 3, for which we establish probabilistic complexity guarantees. Numerical experiments for our proposed techniques are given in Section 4.

Software All the algorithms discussed here are available in an open-source Python package available on Github.[1]

Notation In what follows, $\|\cdot\|$ will denote the Euclidean norm for vectors or the operator 2-norm for matrices. We use $\log(\cdot)$ to denote the reciprocal of the exponential function and $\log_a(\cdot)$ to denote the base-a logarithmic function. The vectors will be denoted by bold lowercase letters (e.g. $\boldsymbol{x}$) while the matrices will be denoted by bold uppercase letters (e.g. $\boldsymbol{S}$). Sets will be denoted by cursive uppercase letters (e.g. $\mathcal{D}$). The letters m, n, r will always denote integers greater than or equal to 1. Finally, $\boldsymbol{I}_r$ will denote the identity matrix in $\mathbb{R}^{r\times r}$.

2 Direct-search framework and complexity results

In this section, we recall the key components of a direct-search method based on sufficient decrease. We focus on the properties that guarantee decrease in the objective function and, as a result, convergence to approximate stationary points. Section 2.1 recalls the fundamentals of direct search based on deterministic properties, while Section 2.2 extends the analysis to the case of probabilistically descent directions, following the reasoning of Gratton et al. [20].

2.1 Direct search based on deterministic descent

Algorithm 1 presents a simplified direct-search framework based on sufficient decrease, for which complexity results can easily be established. At every iteration, the algorithm chooses a set of

[1] `https://github.com/lindonroberts/directsearch`

polling directions of fixed cardinality, and evaluates the objective at the corresponding points. If one of these trial points satisfies the decrease condition (2), this point becomes the new iterate and the stepsize parameter is (possibly) increased. Otherwise, the current point does not change, and the stepsize is decreased. Such an adaptive behavior of the stepsize parameter is instrumental to establishing global convergence of the algorithm, and has roots in convergence analyzes based on line-search techniques [25].

Algorithm 1: Direct-search framework based on sufficient decrease.

Inputs: $\boldsymbol{x}_0 \in \mathbb{R}^n$, $\alpha_{\max} > 0$, $\alpha_0 \in (0, \alpha_{\max}]$, $c > 0$, $0 < \gamma_{\text{dec}} < 1 < \gamma_{\text{inc}}$, $m \in \mathbb{N}$.
for $k = 0, 1, \ldots$ **do**
  Compute a polling set $\mathcal{D}_k \subset \mathbb{R}^n$ of m vectors.
  If there exists $\boldsymbol{d}_k \in \mathcal{D}_k$ such that

$$f(\boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k) < f(\boldsymbol{x}_k) - \frac{c}{2}\alpha_k^2 \|\boldsymbol{d}_k\|^2, \tag{2}$$

  set $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + \alpha_k \boldsymbol{d}_k$ and $\alpha_{k+1} := \min\{\gamma_{\text{inc}}\alpha_k, \alpha_{\max}\}$.
  Otherwise, set $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k$ and $\alpha_{k+1} := \gamma_{\text{dec}}\alpha_k$.
end

The choice of the polling sets is crucial for obtaining theoretical guarantees on the behavior of Algorithm 1. The standard requirements rely on the following concept of cosine measure, which we define for an arbitrary dimension r for later use in the paper.

Definition 2.1 *Given a set of vectors $\mathcal{D} \subset \mathbb{R}^r$ and a nonzero vector $\boldsymbol{v} \in \mathbb{R}^r$, the* cosine measure of $\mathcal{D}$ at $\boldsymbol{v}$ *is defined by*

$$\text{cm}\,(\mathcal{D}, \boldsymbol{v}) := \max_{\boldsymbol{d} \in \mathcal{D}} \frac{\boldsymbol{d}^{\mathrm{T}}\boldsymbol{v}}{\|\boldsymbol{d}\|\|\boldsymbol{v}\|}. \tag{3}$$

The cosine measure of $\mathcal{D}$ *is then given by*

$$\text{cm}\,(\mathcal{D}) := \min_{\substack{\boldsymbol{v} \in \mathbb{R}^r \\ \|\boldsymbol{v}\| \neq 0}} \text{cm}\,(\mathcal{D}, \boldsymbol{v})\,. \tag{4}$$

Any set $\mathcal{D}$ such that $\text{cm}\,(\mathcal{D}) > 0$ is called a positive spanning set (PSS) for $\mathbb{R}^r$, as its elements span $\mathbb{R}^r$ by nonnegative linear combinations [1, 14]. Using PSSs leads to complexity results for Algorithm 1 under the following standard assumptions on the objective function.

Assumption 2.1 *There exists $f_{\text{low}} \in \mathbb{R}$ such that $f(\boldsymbol{x}) \geq f_{\text{low}}$ for all $\boldsymbol{x} \in \mathbb{R}^n$.*

Assumption 2.2 *The function f is continuously differentiable, and its derivative is L-Lipschitz continuous with $L > 0$.*

The analysis of Algorithm 1 is based on two key arguments. First, one can use the sufficient decrease property (2) to guarantee that the step size converges to zero, regardless of the cosine measure of the polling sets. This is the purpose of the following lemma.

Lemma 2.1 *Let Assumption 2.1 hold, and consider the step size sequence $\{\alpha_k\}$ produced by Algorithm 1. Suppose that the directions in $\{\mathcal{D}_k\}_k$ are uniformly bounded in norm. Then, there exists $\beta > 0$ that does not depend on $\{\mathcal{D}_k\}$ such that*

$$\sum_{k=0}^{\infty} \alpha_k^2 \leq \beta < \infty. \tag{5}$$

As a result, $\lim_{k\to\infty} \alpha_k = 0$.

A key assumption to derive the result of Lemma 2.1 is that the polling directions are uniformly bounded in norm. Such a property is easy to satisfy in practice (e.g. by normalizing directions), and leads to global convergence [25] and complexity results [34] for deterministic direct search. It was also used in analyzing probabilistic direct search [20], thereby restricting the kind of directions that could be used. Relaxing this requirement in a probabilistic fashion leads to a number of complications that we will deal with in Section 3.

The second main ingredient in deriving complexity bounds for Algorithm 1 relates the quality of the polling sets to the stepsize. It certifies that the method will move to a new point if the stepsize is small compared to the gradient norm at the current point.

Lemma 2.2 *Consider the k-th iteration of Algorithm 1 under the assumption that* $\operatorname{cm}(\mathcal{D}_k, -\nabla f(\boldsymbol{x}_k)) \geq \kappa \in (0,1)$ *and* $\|\boldsymbol{d}_k\| \leq D_{\max}$ *for any* $\boldsymbol{d} \in \mathcal{D}_k$. *Then, if*

$$\alpha_k < \frac{2}{(L+c)D_{\max}} \kappa \|\nabla f(\boldsymbol{x}_k)\|, \tag{6}$$

the sufficient decrease condition is satisfied for one direction in $\mathcal{D}_k$, and thus $\boldsymbol{x}_{k+1} \neq \boldsymbol{x}_k$.

In practice, the gradient is unknown, thus the assumption $\operatorname{cm}(\mathcal{D}_k, -\nabla f(\boldsymbol{x}_k)) \geq \kappa$ is replaced by $\operatorname{cm}(\mathcal{D}_k) \geq \kappa$, which is equivalent to assuming that $\mathcal{D}_k$ is a PSS in $\mathbb{R}^n$.

The updating process on $\{\alpha_k\}_k$ together with the result of Lemma 2.2 guarantees convergence of Algorithm 1 provided the cosine measure sequence $\{\operatorname{cm}(\mathcal{D}_k)\}_k$ is uniformly bounded below by $\kappa \in (0,1)$. Under this assumption, it is known [34] that Algorithm 1 reaches an iterate $\boldsymbol{x}_k$ such that $\|\nabla f(\boldsymbol{x}_k)\| \leq \epsilon$ using at most

$$\mathcal{O}\left(m\, \kappa^{-2}\, \epsilon^{-2}\right) \tag{7}$$

function evaluations.

A classical choice in direct search consists in selecting $\mathcal{D}_k$ as the set of coordinate vectors and their negatives in $\mathbb{R}^n$. In that case, one has $m = 2n$, $\kappa = \frac{1}{\sqrt{n}}$, and the bound becomes

$$\mathcal{O}\left(n^2\, \epsilon^{-2}\right). \tag{8}$$

In a deterministic setting, the dependency in n^2 cannot be improved while using positive spanning sets without additional information [15].

2.2 Direct search based on probabilistic descent

As highlighted in the previous section, the use of positive spanning sets is instrumental for convergence of classical direct-search methods. However, this property also incurs a dependency of n^2 in the complexity bounds, due to the need to cover an n-dimensional space. Gratton et al. [20] recently established that randomly generated direction sets *that do not form a PSS* can still provide a good approximation of a particular vector in that space, and that this is sufficient to produce good directions. The following property was thus introduced.

Definition 2.2 *Given $p \in (0,1]$ and $\kappa \in (0,1)$, the polling set sequence $\{\mathcal{D}_k\}$ used in Algorithm 1 is called (p,κ)-descent if*

$$\begin{cases} \mathbb{P}\left(\mathrm{cm}\left(\mathcal{D}_0, -\nabla f(\boldsymbol{x}_0)\right) \geq \kappa\right) & \geq \quad p \\ \\ \mathbb{P}\left(\mathrm{cm}\left(\mathcal{D}_k, -\nabla f(\boldsymbol{x}_k)\right) \geq \kappa | \mathcal{F}_{k-1}\right) & \geq \quad p, \qquad \forall k \geq 1, \end{cases} \tag{9}$$

where $\mathcal{F}_{k-1}$ is the σ-algebra generated by the random sets $\mathcal{D}_0, \ldots, \mathcal{D}_{k-1}$.

The (p,κ)-descent property is enough to establish probabilistic complexity results for Algorithm 1. Indeed, the result of Lemma 2.1 holds for every realization of the method provided the directions are bounded, while that of Lemma 2.2 now depends on the occurrence of the random event $\{\mathrm{cm}\left(\mathcal{D}_k, -\nabla f(\boldsymbol{x}_k)\right) \geq \kappa\}$.

Using martingale-type arguments, it is then possible to show that a method employing a (p,κ)-descent sequence converges almost surely to a point with zero gradient. In addition, high probability complexity guarantees hold, in that the method reaches an iterate satisfying $\|\nabla f(\boldsymbol{x}_k)\| \leq \epsilon$ using at most

$$\mathcal{O}(m\kappa^{-2}\epsilon^{-2})$$

function evaluations *with probability at least* $1 - \mathcal{O}(-\exp(C\epsilon^{-2}))$. Assuming $\mathcal{D}_k$ is randomly generated using $m \geq 2$ directions uniformly distributed on the unit sphere, one obtains an high-probability evaluation complexity bound in

$$\mathcal{O}(n\epsilon^{-2}). \tag{10}$$

Using $m = 2$ emerged as a good practical alternative, with the use of two opposite directions allowing to maximize the probability of having a descent set [20, Appendix B]. Other proposals based on random directions [30, 4] relied on evaluations along random Gaussian opposite directions, leading to a same improvement in the complexity bound (note that the probabilistic descent analysis does not apply to Gaussian direction since those are not deterministically bounded in norm). For such approaches, using evaluations along a one-dimensional subspace stood out as an efficient and theoretically sound strategy, but the role of the subspace was not further investigated. Taking the subspace nature of those directions into account is the key goal of this paper, and the subject of the next section.

3 Probabilistic descent in reduced spaces

Building in the framework of Algorithm 1, we propose a method that operates in a reduced space by selecting both a subspace of $\mathbb{R}^n$ and a set of polling directions within that subspace. This two-step process allows to identify deterministic and probabilistic conditions under which the subspace (resp. the directions) are of suitable quality.

3.1 Algorithm and suitable properties

Algorithm 2 details our proposed method. At every iteration, we first generate directions in $\mathbb{R}^r$ with $r \leq n$. We then combine those directions with a matrix $\boldsymbol{P}_k \in \mathbb{R}^{r\times n}$ to obtain a polling set in $\mathbb{R}^n$ *that belongs to an r-dimensional subspace of* $\mathbb{R}^n$. This property is a key feature of our method, as it allows our method to operate in subspaces of lower dimension than that of the original problem: at iteration k, our polling set is $\{\boldsymbol{P}_k^{\mathrm{T}}\boldsymbol{d}|\boldsymbol{d} \in \mathcal{D}_k\}$.

Algorithm 2: Direct-search method in reduced spaces.

Inputs: $\boldsymbol{x}_0 \in \mathbb{R}^n$, $\alpha_{\max} > 0$, $\alpha_0 \in (0, \alpha_{\max}]$, $c >$, $0 < \gamma_{\text{dec}} < 1 < \gamma_{\text{inc}}$; $m \in \mathbb{N}$, $r \leq n$.
for $k = 0, 1, ...$ **do**
  Compute a matrix $\boldsymbol{P}_k \in \mathbb{R}^{r\times n}$.
  Compute a set $\mathcal{D}_k \subset \mathbb{R}^r$ of m vectors.
  If there exists $\boldsymbol{d}_k \in \mathcal{D}_k$ such that

$$f(\boldsymbol{x}_k + \alpha_k \boldsymbol{P}_k^{\mathrm{T}}\boldsymbol{d}_k) < f(\boldsymbol{x}_k) - \frac{c}{2}\alpha_k^2 \|\boldsymbol{P}_k^{\mathrm{T}}\boldsymbol{d}_k\|^2, \tag{11}$$

  set $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k + \alpha_k \boldsymbol{P}_k^T \boldsymbol{d}_k$ and $\alpha_{k+1} := \min\{\gamma_{\text{inc}}\alpha_k, \alpha_{\max}\}$.
  Otherwise, set $\boldsymbol{x}_{k+1} := \boldsymbol{x}_k$ and $\alpha_{k+1} := \gamma_{\text{dec}}\alpha_k$.
end

To assess the quality of the polling sets, we define separate properties for the matrix $\boldsymbol{P}_k$ and the set $\mathcal{D}_k$, starting with the former. The matrix $\boldsymbol{P}_k$ produces an r-dimensional subspace of $\mathbb{R}^n$ in which polling directions will be generated. When $r < n$, the use of $\boldsymbol{P}_k$ will prevent the polling set from providing good approximations to all of $\mathbb{R}^n$. Nevertheless, for optimization purposes, we are merely interested in approximating the negative gradient (and its norm). Consequently, we require the matrix $\boldsymbol{P}_k$ to capture a significant portion of gradient information. In addition, defining the polling set through application of $\boldsymbol{P}_k^{\mathrm{T}}$ should alter the directions in the reduced subspace in a controlled way, which we express through bounds on the singular values of the matrix. These considerations lead to the following definition, motivated by a similar concept in the model-based setting [12, 9, 10, 33].

Definition 3.1 *Let η, σ and $P_{\max}$ be positive quantities. For any realization of Algorithm 2 and any $k \in \mathbb{N}$, the matrix $\boldsymbol{P}_k$ is called* $(\eta, \sigma, P_{\max})$-well aligned *for f at $\boldsymbol{x}_k$ provided*

$$\|\boldsymbol{P}_k \nabla f(\boldsymbol{x}_k)\| \geq \eta \|\nabla f(\boldsymbol{x}_k)\|, \tag{12}$$

$$\|\boldsymbol{P}_k\| \leq P_{\max}, \tag{13}$$

$$\sigma_{\min}(\boldsymbol{P}_k) \geq \sigma, \tag{14}$$

where $\sigma_{\min}(\cdot)$ denotes the minimum nonzero singular value of the matrix $\boldsymbol{P}_k$.

Conditions (12)–(14) are satisfied with $\eta = \sigma = P_{\max} = 1$ when $r = n$ and $\boldsymbol{P}_k$ is the identity matrix, but may not hold when $r < n$, or when $\boldsymbol{P}_k$ is a random matrix. For this reason, we introduce a probabilistic counterpart to Definition 3.1.

Definition 3.2 *The sequence* $\{\boldsymbol{P}_k\}_k$ *generated by Algorithm 2 is called* $(\eta, \sigma, P_{\max}, q)$-well-aligned *for* $q \in (0, 1]$ *if*

$$\begin{array}{rcl} \mathbb{P}\left(\boldsymbol{P}_0 \text{ is } (\eta, \sigma, P_{\max})\text{-well aligned}\right) & \geq & q \\ \forall k \geq 1, \qquad \mathbb{P}\left(\boldsymbol{P}_k \text{ is } (\eta, \sigma, P_{\max})\text{-well aligned} \mid \mathcal{F}_{k-1}\right) & \geq & q, \end{array} \tag{15}$$

where $\mathcal{F}_{k-1}$ *is the* σ*-algebra generated by* $\boldsymbol{P}_0, \mathcal{D}_0, \ldots, \boldsymbol{P}_{k-1}, \mathcal{D}_{k-1}$.

Our requirement on $\mathcal{D}_k$ is given below, and is similar to that used in Section 2.1.

Definition 3.3 *Let* $\kappa \in (0, 1]$ *and* $D_{\max} > 1$. *For any realization of Algorithm 2 and any index* $k \in \mathbb{N}$, *the set* $\mathcal{D}_k$ *is called* $(\kappa, D_{\max})$*-descent for* f *and* $\boldsymbol{P}_k$ *at* $\boldsymbol{x}_k$ *if*

$$\operatorname{cm}\left(\mathcal{D}_k, -\boldsymbol{P}_k \nabla f(x_k)\right) = \max_{\boldsymbol{d} \in \mathcal{D}_k} \frac{-\boldsymbol{d}^{\mathrm{T}} \boldsymbol{P}_k \nabla f(\boldsymbol{x}_k)}{\|\boldsymbol{d}\| \|\boldsymbol{P}_k \nabla f(\boldsymbol{x}_k)\|} \geq \kappa \tag{16}$$

and

$$\forall \boldsymbol{d} \in \mathcal{D}_k, \quad D_{\max}^{-1} \leq \|\boldsymbol{d}\| \leq D_{\max}. \tag{17}$$

Examples of sets satisfying Definition 3.3 are positive spanning sets in $\mathbb{R}^r$ with unitary elements. As for the properties of $\boldsymbol{P}_k$, we provide a probabilistic counterpart of Definition 3.3 below.

Definition 3.4 *The sequence* $\{\mathcal{D}_k\}_k$ *generated by Algorithm 2 is called* $(\kappa, D_{\max}, p)$-descent *for* $p \in (0, 1]$ *if*

$$\begin{array}{rcl} \mathbb{P}\left(\mathcal{D}_0 \text{ is } (\kappa, D_{\max})\text{-descent} \middle| \mathcal{F}_{-1/2}\right) & \geq & p \\ \forall k \geq 1, \qquad \mathbb{P}\left(\mathcal{D}_k \text{ is } (\kappa, D_{\max})\text{-descent} \mid \mathcal{F}_{k-1/2}\right) & \geq & p, \end{array} \tag{18}$$

where $\mathcal{F}_{k-1/2}$ *is the* σ*-algebra generated by* $\boldsymbol{P}_0, \mathcal{D}_0, \ldots, \boldsymbol{P}_{k-1}, \mathcal{D}_{k-1}, \boldsymbol{P}_k$ *and* $\mathcal{F}_{-1/2}$ *is the* σ*-algebra generated by* $\boldsymbol{P}_0$.

Definition 3.4 departs from Definition 2.2 in that it allows for unbounded directions, as long as such directions occur with a small probability. This enables for instance the use of Gaussian vectors, a distribution that was not immediately covered by the analysis of Gratton et al. [20]. In Section 3.3 we give several possible choices for $\boldsymbol{P}_k$ and $\mathcal{D}_k$ that satisfy these requirements.

3.2 Complexity analysis

In this section, we leverage the probabilistic properties defined above to derive complexity results for Algorithm 2. Our analysis follows a reasoning previously developed for derivative-free methods based on probabilistic properties [20, 21], but involves two properties of this form at every iteration, respectively related to $\boldsymbol{P}_k$ and $\mathcal{D}_k$. Although these properties can be handled jointly, the analysis still departs from existing ones as we allow for directions that are unbounded in norm. At the same time, we point out that our approach still relies on exact function values, and therefore does not require Lyapunov functions similar to those used in stochastic optimization [31].

For the rest of this section, let $\mathcal{S}$ denote the index set of *successful iterations* (i.e. the k for which $\boldsymbol{x}_{k+1} \neq \boldsymbol{x}_k$) and $\mathcal{U}$ denote the index set of *unsuccessful iterations* (for which $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k$). The following lemma describes sufficient conditions under which an iteration must be successful, based on our properties of interest.

Lemma 3.1 *Let Assumption 2.2 hold, and consider the k-th iteration of a realization of Algorithm 2. Suppose further that $\mathcal{D}_k$ is $(\kappa, D_{\max})$-descent and that $\boldsymbol{P}_k$ is $(\eta, \sigma, P_{\max})$-well aligned. Finally, suppose that*

$$\alpha_k < \bar{\alpha}\|\nabla f(\boldsymbol{x}_k)\|, \quad \textit{where} \quad \bar{\alpha} := \frac{2\kappa\eta}{(L+c)P_{\max}^2 D_{\max}^3}. \tag{19}$$

Then, the k-th iteration is successful.

Proof. To find a contradiction, suppose that iteration k is unsuccessful. Then, by Assumption 2.2, for all $\boldsymbol{d} \in \mathcal{D}_k$ we have

$$\begin{aligned} -\frac{c}{2}\alpha_k^2\|\boldsymbol{P}_k^T\boldsymbol{d}\|^2 &\leq f(\boldsymbol{x}_k + \alpha_k \boldsymbol{P}_k^T\boldsymbol{d}) - f(\boldsymbol{x}_k), \\ &\leq \alpha_k \boldsymbol{d}^T\boldsymbol{P}_k\nabla f(\boldsymbol{x}_k) + \frac{L}{2}\alpha_k^2\|\boldsymbol{P}_k^T\boldsymbol{d}\|^2. \end{aligned}$$

Since $\mathcal{D}_k$ is a $(\kappa, D_{\max})$-descent set, there exists $\boldsymbol{d}_k \in \mathcal{D}_k$ such that

$$\frac{\boldsymbol{d}_k^T(-\boldsymbol{P}_k\nabla f(\boldsymbol{x}_k))}{\|\boldsymbol{d}_k\|\,\|\boldsymbol{P}_k\nabla f(\boldsymbol{x}_k)\|} = \operatorname{cm}(\mathcal{D}_k, -\boldsymbol{P}_k\nabla f(\boldsymbol{x}_k)) \geq \kappa. \tag{20}$$

Therefore, we obtain

$$\begin{aligned} -\frac{c}{2}\alpha_k^2\|\boldsymbol{P}_k^T\boldsymbol{d}_k\|^2 &\leq -\kappa\alpha_k\|\boldsymbol{d}_k\|\|\boldsymbol{P}_k\nabla f(\boldsymbol{x}_k)\| + \frac{L}{2}\alpha_k^2\|\boldsymbol{P}_k^{\mathrm{T}}\boldsymbol{d}_k\|^2 \\ \kappa\alpha_k\|\boldsymbol{d}_k\|\,\|\boldsymbol{P}_k\nabla f(\boldsymbol{x}_k)\| &\leq \frac{L+c}{2}\alpha_k^2\|\boldsymbol{P}_k^{\mathrm{T}}\boldsymbol{d}_k\|^2 \\ \kappa\|\boldsymbol{d}_k\|\,\|\boldsymbol{P}_k\nabla f(\boldsymbol{x}_k)\| &\leq \frac{L+c}{2}\alpha_k\|\boldsymbol{P}_k^{\mathrm{T}}\boldsymbol{d}_k\|^2 \end{aligned}$$

Using now the property (12) on $\boldsymbol{P}_k$ together with the bound (17) on $\|\boldsymbol{d}_k\|$ leads to

$$\kappa\|\boldsymbol{d}_k\|\,\|\boldsymbol{P}_k\nabla f(\boldsymbol{x}_k)\| \geq \kappa\eta D_{\max}^{-1}\|\nabla f(\boldsymbol{x}_k)\|$$

as well as

$$\|\boldsymbol{P}_k^{\mathrm{T}}\boldsymbol{d}_k\|^2 \leq \|\boldsymbol{P}_k^{\mathrm{T}}\|^2\|\boldsymbol{d}_k\|^2 \leq P_{\max}^2 D_{\max}^2.$$

Putting everything together, we arrive at

$$\kappa\eta D_{\max}^{-1}\|\nabla f(\boldsymbol{x}_k)\| \leq \frac{L+c}{2}P_{\max}^2 D_{\max}^2\alpha_k \quad \Leftrightarrow \quad \alpha_k \geq \bar{\alpha}\|\nabla f(\boldsymbol{x}_k)\|,$$

and this contradicts (19). □

We now introduce the following indicator variables:

$$Z_k := \mathbf{1}\left(\mathcal{D}_k \text{ } (\kappa, D_{\max})\text{-descent and } \boldsymbol{P}_k \text{ } (\eta, \sigma, P_{\max})\text{-well aligned}\right), \tag{21a}$$

$$V_k(\alpha) := \mathbf{1}\left(\alpha_k < \alpha\right) \quad \forall \alpha > 0, \tag{21b}$$

$$W_k := \mathbf{1}\left(k \in \mathcal{S}\right), \tag{21c}$$

whose realizations will be denoted by $z_k, v_k(\alpha), w_k$, respectively. Our goal is to bound the sum of z_k by a quantity involving the gradient norm. To this end, we study careful combinations of the indicator variables above, for which we can provide bounds that are independent of the iteration: this technique has proven useful in establishing complexity guarantees for derivative-free optimization schemes with probabilistic components [12, 13, 31, 20, 21].

Our first result bounds the sum of squared stepsizes for a certain subset of successful iterations. Unlike in analyses of direct search based solely on descent properties [20], where such a bound can be obtained for all iterations, here it only holds for those successful iterations that use both well-aligned subspace matrices and descent directions.

Lemma 3.2 *Let Assumption 2.1 hold. For any realization of Algorithm 2,*

$$\sum_{k=0}^{\infty} z_k w_k \alpha_k^2 \leq \beta := \frac{2D_{\max}^2(f(\boldsymbol{x}_0) - f_{\text{low}})}{c\sigma^2}. \tag{22}$$

Proof. It suffices to consider iterations for which $z_k w_k = 1$, i.e. successful iterations for which $\mathcal{D}_k$ is $(\kappa, D_{\max})$-descent and $\boldsymbol{P}_k$ is $(\eta, \sigma, P_{\max})$-well aligned. For such an iteration, there exists $\boldsymbol{d}_k \in \mathcal{D}_k$ such that $\boldsymbol{x}_{k+1} = \boldsymbol{x}_k + \alpha_k \boldsymbol{P}_k^{\mathrm{T}} \boldsymbol{d}_k$, and

$$f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1}) \geq \frac{c}{2}\alpha_k^2 \|\boldsymbol{P}_k^{\mathrm{T}} \boldsymbol{d}_k\|^2 \geq \frac{c}{2}\alpha_k^2 \sigma^2 \|\boldsymbol{d}_k\|^2 \geq \frac{c}{2}\sigma^2 D_{\max}^{-2} \alpha_k^2.$$

On the other hand, Assumption 2.1 guarantees that

$$f(\boldsymbol{x}_0) - f_{\text{low}} \geq \sum_{k=0}^{\infty} f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1}) \geq \sum_{k=0}^{\infty} z_k w_k \left(f(\boldsymbol{x}_k) - f(\boldsymbol{x}_{k+1})\right).$$

Thus, we obtain

$$\sum_{k=0}^{\infty} z_k w_k \alpha_k^2 \leq \frac{2D_{\max}^2(f(\boldsymbol{x}_0) - f_{\text{low}})}{c\sigma^2},$$

proving the desired result. □

The next two results are obtained by carefully examining the behavior of the stepsize sequence. This is another notable departure from the analysis of direct search based on probabilistic descent [20], that is due to our two probabilistic properties. We note that similar results have been derived in the context of randomized model-based methods [12].

Lemma 3.3 *Let Assumption 2.1 hold. Consider a realization of Algorithm 2 and an index k such that $\min_{0\leq\ell\leq k-1} \|\nabla f(\boldsymbol{x}_\ell)\| > 0$. Then,*

$$\sum_{j=0}^{k-1} v_j\left(\overline{\alpha}_k\right) w_j \;\leq\; \mu \sum_{j=0}^{k-1} v_j\left(\frac{\gamma_{\text{inc}}}{\gamma_{\text{dec}}}\overline{\alpha}_k\right)(1 - w_j) \tag{23}$$

with $\mu := \log_{\gamma_{\text{inc}}}(\gamma_{\text{dec}}^{-1})$.

$$\overline{\alpha}_k := \frac{\gamma_{\text{dec}}}{\gamma_{\text{inc}}} \min\left\{\gamma_{\text{inc}}^{-1}\alpha_0, \bar{\alpha} \min_{0\leq\ell\leq k-1} \|\nabla f(\boldsymbol{x}_\ell)\|\right\}, \tag{24}$$

and $\bar{\alpha}$ is defined as in Lemma 3.1.

Proof. If $v_j\left(\overline{\alpha}_k\right) w_j = 0$ for every $j \leq k-1$, the bound clearly holds. For the rest of the proof, we thus assume that there exist at least one index $j \in \{0, \ldots, k-1\}$ such that $v_j\left(\overline{\alpha}_k\right) w_j = 1$. Since $\frac{\gamma_{\text{inc}}}{\gamma_{\text{dec}}} > 1$, we also have $v_j\left(\frac{\gamma_{\text{inc}}}{\gamma_{\text{dec}}}\overline{\alpha}_k\right) w_j = 1$, thus there also exists at least one index satisfying this property.

Consider a sequence of iterates $j_1, \ldots, j_2$ such that $v_j\left(\frac{\gamma_{\text{inc}}}{\gamma_{\text{dec}}}\overline{\alpha}_k\right) = 1$ for every $j \in \{j_1, \ldots, j_2\}$, with $v_{j_2}\left(\overline{\alpha}_k\right) w_{j_2} = 1$ and either $j_1 = 0$ or $v_{j_1-1}\left(\frac{\gamma_{\text{inc}}}{\gamma_{\text{dec}}}\overline{\alpha}_k\right) = 0$ (note that such a sequence necessarily exists by assumption). Using the updating rules on the stepsize together with the bound $\overline{\alpha}_k < \gamma_{\text{inc}}^{-1}\alpha_0 \leq \gamma_{\text{inc}}^{-1}\alpha_{\max}$, we obtain for any $j = j_1, \ldots, j_2$ that

$$\begin{cases} \alpha_{j+1} & = \min\{\gamma_{\text{inc}}\alpha_j, \alpha_{\max}\} = \gamma_{\text{inc}}\alpha_j, & \text{if } v_j(\overline{\alpha}_k)w_j = 1, \\ \alpha_{j+1} & \geq \alpha_j, & \text{if } (1 - v_j(\overline{\alpha}_k))v_j\left(\frac{\gamma_{\text{inc}}}{\gamma_{\text{dec}}}\overline{\alpha}_k\right) w_j = 1, \\ \alpha_{j+1} & = \gamma_{\text{dec}}\alpha_j, & \text{if } v_j\left(\frac{\gamma_{\text{inc}}}{\gamma_{\text{dec}}}\overline{\alpha}_k\right)(1 - w_j) = 1. \end{cases} \tag{25}$$

Note that the rules (25) cover all possible cases. Applying rule (25) iteratively gives:

$$\alpha_{j_2+1} \quad \geq \quad \gamma_{\text{inc}}^{\sum_{j=j_1}^{j_2} v_j(\overline{\alpha}_k)w_j} \gamma_{\text{dec}}^{\sum_{j=j_1}^{j_2} v_j\left(\frac{\gamma_{\text{inc}}}{\gamma_{\text{dec}}}\overline{\alpha}_k\right)(1-w_j)} \alpha_{j_1}.$$

Moreover, since $v_{j_2}\left(\overline{\alpha}_k\right) w_{j_2} = 1$, we have

$$\alpha_{j_2+1} = \gamma_{\text{inc}}\alpha_{j_2} \leq \gamma_{\text{inc}}\overline{\alpha}_k,$$

leading to

$$\frac{\gamma_{\text{inc}}\overline{\alpha}_k}{\alpha_{j_1}} \geq \gamma_{\text{inc}}^{\sum_{j=j_1}^{j_2} v_j(\overline{\alpha}_k)w_j} \gamma_{\text{dec}}^{\sum_{j=j_1}^{j_2} v_j\left(\frac{\gamma_{\text{inc}}}{\gamma_{\text{dec}}}\overline{\alpha}_k\right)(1-w_j)}. \tag{26}$$

To bound the ratio $\frac{\gamma_{\text{inc}}\overline{\alpha}_k}{\alpha_{j_1}}$, we consider two cases. If $j_1 = 0$, then $\gamma_{\text{dec}}\overline{\alpha}_k \leq \alpha_0$ by definition of $\overline{\alpha}_k$, and thus $\gamma_{\text{dec}}\frac{\overline{\alpha}_k}{\alpha_{j_1}} \leq 1$. Otherwise, if $j_1 > 0$, we have by definition of j_1 that $v_{j_1-1}\left(\frac{\gamma_{\text{inc}}}{\gamma_{\text{dec}}}\overline{\alpha}_k\right) = 0$, that is, $\alpha_{j_1-1} \geq \frac{\gamma_{\text{inc}}}{\gamma_{\text{dec}}}\overline{\alpha}_k$. Per the updating rules on the step size, this implies

$$\alpha_{j_1} \geq \gamma_{\text{dec}}\alpha_{j_1-1} \geq \gamma_{\text{inc}}\overline{\alpha}_k,$$

hence $\gamma_{\text{inc}}\frac{\overline{\alpha}_k}{\alpha_{j_1}} \leq 1$ also holds in this case. Plugging this bound into (26) yields

$$1 \geq \gamma_{\text{inc}}^{\sum_{j=j_1}^{j_2} v_j(\overline{\alpha}_k)w_j} \gamma_{\text{dec}}^{\sum_{j=j_1}^{j_2} v_j\left(\frac{\gamma_{\text{inc}}}{\gamma_{\text{dec}}}\overline{\alpha}_k\right)(1-w_j)}.$$

Taking logarithms, we obtain that

$$0 \geq \log(\gamma_{\text{inc}}) \sum_{j=j_1}^{j_2} v_j(\overline{\alpha}_k)w_j + \log(\gamma_{\text{dec}}) \sum_{j=j_1}^{j_2} v_j\left(\frac{\gamma_{\text{inc}}}{\gamma_{\text{dec}}}\overline{\alpha}_k\right)(1 - w_j),$$

which after rearranging becomes

$$\sum_{j=j_1}^{j_2} v_j(\overline{\alpha}_k)w_j \leq \log_{\gamma_{\text{inc}}}(\gamma_{\text{dec}}^{-1}) \sum_{j=j_1}^{j_2} v_j\left(\frac{\gamma_{\text{inc}}}{\gamma_{\text{dec}}}\overline{\alpha}_k\right)(1 - w_j) = \mu \sum_{j=j_1}^{j_2} v_j\left(\frac{\gamma_{\text{inc}}}{\gamma_{\text{dec}}}\overline{\alpha}_k\right)(1 - w_j).$$

To conclude, we simply observe that

$$\{j \mid v_j(\overline{\alpha}_k)w_j = 1\} \subset \left\{j \;\middle|\; v_j\left(\tfrac{\gamma_{\text{inc}}}{\gamma_{\text{dec}}}\overline{\alpha}_k\right) = 1\right\},$$

or equivalently, that every $j \in \{0, \ldots, k-1\}$ for which $v_j(\overline{\alpha}_k)w_j = 1$ is in such a subsequence $j_1, \ldots, j_2$. As a result, we obtain

$$\sum_{j=0}^{k-1} v_j(\overline{\alpha}_k)w_j \;\leq\; \mu \sum_{j=0}^{k-1} v_j\left(\tfrac{\gamma_{\text{inc}}}{\gamma_{\text{dec}}}\overline{\alpha}_k\right)(1-w_j).$$

□

Lemma 3.4 *Let Assumption 2.1 hold. Consider a realization of Algorithm 2 and an index k such that $\min_{0\leq\ell\leq k-1} \|\nabla f(\boldsymbol{x}_\ell)\| > 0$. Then,*

$$\sum_{j=0}^{k-1} \left(1 - v_j\left(\overline{\alpha}_k\right)\right)\left(1-w_j\right) \;\leq\; \frac{1}{\mu}\sum_{j=0}^{k-1}\left(1 - v_j\left(\frac{\gamma_{\text{dec}}}{\gamma_{\text{inc}}}\overline{\alpha}_k\right)\right)w_j + \log_{\gamma_{\text{dec}}^{-1}}\left(\frac{\alpha_0}{\gamma_{\text{dec}}\overline{\alpha}_k}\right), \tag{27}$$

where μ and $\overline{\alpha}_k$ are defined as in Lemma 3.3.

Proof. The proof follows the template of that of Lemma 3.3. The bound trivially holds if $\left(1 - v_j\left(\overline{\alpha}_k\right)\right)\left(1-w_j\right) = 0$ for every $j \leq k-1$. Therefore, we suppose that there exists at least one index $j \in \{0, \ldots, k-1\}$ such that $(1 - v_j(\overline{\alpha}_k))(1-w_j) = 1$. Since $\frac{\gamma_{\text{dec}}}{\gamma_{\text{inc}}} < 1$, we also have $\left(1 - v_j\left(\frac{\gamma_{\text{dec}}}{\gamma_{\text{inc}}}\overline{\alpha}_k\right)\right)(1-w_j) = 1$, thus there also exists at least one index satisfying this property.

Consider now a sequence of iterates $j_1, \ldots, j_2$ such that $\left(1 - v_j\left(\frac{\gamma_{\text{dec}}}{\gamma_{\text{inc}}}\overline{\alpha}_k\right)\right) = 1$ for every $j = j_1, \ldots, j_2$, with $\left(1 - v_{j_2}\left(\overline{\alpha}_k\right)\right)\left(1-w_{j_2}\right) = 1$ and either $j_1 = 0$ or $v_{j_1-1}\left(\frac{\gamma_{\text{dec}}}{\gamma_{\text{inc}}}\overline{\alpha}_k\right) = 1$: such a sequence necessarily exists by assumption. From the updating rules on the stepsize, we obtain the following possible cases:

$$\begin{cases} \alpha_{j+1} & = \gamma_{\text{dec}}\alpha_j, \quad \text{if } (1 - v_j(\overline{\alpha}_k))(1-w_j) = 1, \\ \alpha_{j+1} & \leq \alpha_j, \quad \text{if } v_j(\overline{\alpha}_k)\left(1 - v_j\left(\frac{\gamma_{\text{dec}}}{\gamma_{\text{inc}}}\overline{\alpha}_k\right)\right)(1-w_j) = 1, \\ \alpha_{j+1} & \leq \gamma_{\text{inc}}\alpha_j, \quad \text{if } \left(1 - v_j\left(\frac{\gamma_{\text{dec}}}{\gamma_{\text{inc}}}\overline{\alpha}_k\right)\right)w_j = 1, \end{cases} \tag{28}$$

for any $j = j_1, \ldots, j_2$. By applying rule (28) iteratively, we thus obtain

$$\alpha_{j_2+1} \leq \gamma_{\text{dec}}^{\sum_{j=j_1}^{j_2}(1-v_j(\overline{\alpha}_k))(1-w_j)}\gamma_{\text{inc}}^{\sum_{j=j_1}^{j_2}\left(1-v_j\left(\frac{\gamma_{\text{dec}}}{\gamma_{\text{inc}}}\overline{\alpha}_k\right)\right)w_j}\alpha_{j_1}.$$

In addition, using that $\left(1 - v_{j_2}\left(\overline{\alpha}_k\right)\right)\left(1-w_{j_2}\right) = 1$, we also have

$$\alpha_{j_2+1} = \gamma_{\text{dec}}\alpha_{j_2} \geq \gamma_{\text{dec}}\overline{\alpha}_k,$$

thus

$$\gamma_{\text{dec}}\frac{\overline{\alpha}_k}{\alpha_{j_1}} \leq \gamma_{\text{dec}}^{\sum_{j=j_1}^{j_2}(1-v_j(\overline{\alpha}_k))(1-w_j)}\gamma_{\text{inc}}^{\sum_{j=j_1}^{j_2}\left(1-v_j\left(\frac{\gamma_{\text{dec}}}{\gamma_{\text{inc}}}\overline{\alpha}_k\right)\right)w_j}. \tag{29}$$

Taking the logarithm, we get

$$\log\left(\gamma_{\text{dec}}\frac{\overline{\alpha}_k}{\alpha_{j_1}}\right) \leq -\log(\gamma_{\text{dec}}^{-1})\sum_{j=j_1}^{j_2}(1-v_j(\overline{\alpha}_k))(1-w_j) + \log(\gamma_{\text{inc}})\sum_{j=j_1}^{j_2}\left(1-v_j\left(\tfrac{\gamma_{\text{dec}}}{\gamma_{\text{inc}}}\overline{\alpha}_k\right)\right)w_j,$$

which after re-arranging gives

$$\begin{aligned}\sum_{j=j_1}^{j_2}(1-v_j(\overline{\alpha}_k))(1-w_j) &\leq \log_{\gamma_{\text{dec}}^{-1}}(\gamma_{\text{inc}})\sum_{j=j_1}^{j_2}\left(1-v_j\left(\tfrac{\gamma_{\text{dec}}}{\gamma_{\text{inc}}}\overline{\alpha}_k\right)\right)w_j - \log_{\gamma_{\text{dec}}^{-1}}\left(\gamma_{\text{dec}}\frac{\overline{\alpha}_k}{\alpha_{j_1}}\right)\\ &= \frac{1}{\mu}\sum_{j=j_1}^{j_2}\left(1-v_j\left(\tfrac{\gamma_{\text{dec}}}{\gamma_{\text{inc}}}\overline{\alpha}_k\right)\right)w_j + \log_{\gamma_{\text{dec}}^{-1}}\left(\frac{\alpha_{j_1}}{\gamma_{\text{dec}}\overline{\alpha}_k}\right)\end{aligned}$$

by definition of μ.

To bound the last term, we consider two cases. If $j_1 = 0$, then $\frac{\alpha_{j_1}}{\gamma_{\text{dec}}\overline{\alpha}_k} = \frac{\alpha_0}{\gamma_{\text{dec}}\overline{\alpha}_k}$. Otherwise, if $j_1 > 0$, we have by definition of j_1 that $v_{j_1-1}\left(\frac{\gamma_{\text{dec}}}{\gamma_{\text{inc}}}\overline{\alpha}_k\right) = 1$, that is, $\alpha_{j_1-1} < \frac{\gamma_{\text{dec}}}{\gamma_{\text{inc}}}\overline{\alpha}_k$. Per the updating rules on the step size, this implies

$$\alpha_{j_1} \leq \gamma_{\text{inc}}\alpha_{j_1-1} < \gamma_{\text{dec}}\overline{\alpha}_k,$$

hence $\frac{\alpha_{j_1}}{\gamma_{\text{dec}}\overline{\alpha}_k} \leq 1$ and thus $\log_{\gamma_{\text{dec}}^{-1}}\left(\frac{\alpha_{j_1}}{\gamma_{\text{dec}}\overline{\alpha}_k}\right) < 0$. Overall, we thus obtain that

$$\sum_{j=j_1}^{j_2}(1-v_j(\overline{\alpha}_k))(1-w_j) \leq \begin{cases} \frac{1}{\mu}\sum_{j=j_1}^{j_2}\left(1-v_j\left(\frac{\gamma_{\text{dec}}}{\gamma_{\text{inc}}}\overline{\alpha}_k\right)\right)w_j + \log_{\gamma_{\text{dec}}^{-1}}\left(\frac{\alpha_0}{\gamma_{\text{dec}}\overline{\alpha}_k}\right) & \text{if } j_1 = 0\\ \frac{1}{\mu}\sum_{j=j_1}^{j_2}\left(1-v_j\left(\frac{\gamma_{\text{dec}}}{\gamma_{\text{inc}}}\overline{\alpha}_k\right)\right)w_j & \text{otherwise.}\end{cases}$$

Finally, we note that

$$\{j \mid (1-v_j(\overline{\alpha}_k))(1-w_j) = 1\} \subset \left\{j \;\middle|\; 1-v_j\left(\tfrac{\gamma_{\text{dec}}}{\gamma_{\text{inc}}}\overline{\alpha}_k\right) = 1\right\},$$

on equivalently, that every $j \in \{0,\ldots,k-1\}$ for which $(1-v_j(\overline{\alpha}_k))(1-w_j) = 1$ is in such a subsequence $j_1,\ldots,j_2$. This allows us to conclude

$$\sum_{j=0}^{k-1}(1-v_j(\overline{\alpha}_k))(1-w_j) \leq \frac{1}{\mu}\sum_{j=0}^{k-1}\left(1-v_j\left(\tfrac{\gamma_{\text{dec}}}{\gamma_{\text{inc}}}\overline{\alpha}_k\right)\right)w_j + \log_{\gamma_{\text{dec}}^{-1}}\left(\frac{\alpha_0}{\gamma_{\text{dec}}\overline{\alpha}_k}\right).$$

□

The results of Lemmas 3.3 and 3.4 are sufficient to obtain a bound on the number of iterations for which the directions are generated from both a descent set and a well-aligned subspace matrix.

Proposition 3.1 *For any realization of Algorithm 2 and any positive integer k,*

$$\sum_{j=0}^{k-1} z_j \leq \frac{(1-p_0)C}{\min\{\gamma_{\text{inc}}^{-2}\alpha_0^2, \bar{\alpha}^2[\tilde{g}_k]^2\}} + (1-p_0)\log_{\gamma_{\text{dec}}^{-1}}\left(\frac{\gamma_{\text{inc}}\alpha_0}{\gamma_{\text{dec}}^2\min\{\gamma_{\text{inc}}^{-1}\alpha_0, \bar{\alpha}\tilde{g}_k\}}\right) + p_0 k, \tag{30}$$

where $\tilde{g}_k = \min_{0\leq j\leq k-1} \|\nabla f(\boldsymbol{x}_j)\|$,

$$C := \frac{(\mu\gamma_{\rm dec}^2 + \gamma_{\rm inc}^2)\gamma_{\rm inc}^2\beta}{\mu\gamma_{\rm dec}^4}, \tag{31}$$

and

$$p_0 := \max\left\{\frac{\ln(\gamma_{\rm dec})}{\ln(\gamma_{\rm inc}^{-1}\gamma_{\rm dec})}, \frac{\ln(\gamma_{\rm inc})}{\ln(\gamma_{\rm inc}\gamma_{\rm dec}^{-1})}\right\} = \max\left\{\frac{1}{1+\mu}, \frac{\mu}{1+\mu}\right\}. \tag{32}$$

Proof. For any $j = 0, \ldots, k-1$, we have

$$z_j = z_j v_j(\overline{\alpha}_k) w_j + z_j(1 - v_j(\overline{\alpha}_k))(1 - w_j) + z_j(1 - v_j(\overline{\alpha}_k))w_j. \tag{33}$$

Indeed, the bound clearly holds when $z_j = 0$. Assuming $z_j = 1$, it also holds when $v_j(\overline{\alpha}_k)w_j = 1$ or $(1 - v_j(\overline{\alpha}_k))(1 - w_j) = 1$, while Lemma 3.1 implies that we cannot have $z_j v_j(\overline{\alpha}_k)(1 - w_j) = 1$. This last property will be instrumental in the proof of our result.

Summing (33) over all $j = 0, \ldots, k-1$, we obtain:

$$\sum_{j=0}^{k-1} z_j = \sum_{j=0}^{k-1} z_j(1 - v_j(\overline{\alpha}_k))w_j + \sum_{j=0}^{k-1} z_j v_j(\overline{\alpha}_k) w_j + \sum_{j=0}^{k-1} z_j(1 - v_j(\overline{\alpha}_k))(1 - w_j). \tag{34}$$

We will provide separate bounds on the three sums on the right-hand side of (34).

Consider first an index such that $z_j\,(1 - v_j(\overline{\alpha}_k))\,w_j = 1$. By definition of $v_j(\overline{\alpha}_k)$, we obtain from $1 - v_j(\overline{\alpha}_k) = 1$ that

$$1 \leq \left(\frac{\alpha_j}{\overline{\alpha}_k}\right)^2 = \frac{\gamma_{\rm inc}^2\alpha_j^2}{\gamma_{\rm dec}^2 \min\{\gamma_{\rm inc}^{-2}\alpha_0^2, \bar{\alpha}^2 \min_{0\leq\ell\leq k-1} \|\nabla f(\boldsymbol{x}_\ell)\|^2\}}.$$

Thus,

$$z_j\,(1 - v_j(\overline{\alpha}_k))\,w_j \leq \frac{z_j w_j \gamma_{\rm inc}^2 \alpha_j^2}{\gamma_{\rm dec}^2 \min\{\gamma_{\rm inc}^{-2}\alpha_0^2, \bar{\alpha}^2[\tilde{g}_k]^2\}}$$

and summing over all indices up to $k-1$ gives

$$\sum_{j=0}^{k-1} z_j\,(1 - v_j(\overline{\alpha}_k))\,w_j \leq \frac{\gamma_{\rm inc}^2}{\gamma_{\rm dec}^2 \min\{\gamma_{\rm inc}^{-2}\alpha_0^2, \bar{\alpha}^2[\tilde{g}_k]^2\}} \sum_{j=0}^{k-1} z_j w_j \alpha_j^2 \leq \frac{\gamma_{\rm inc}^2\beta}{\gamma_{\rm dec}^2 \min\{\gamma_{\rm inc}^{-2}\alpha_0^2, \bar{\alpha}^2[\tilde{g}_k]^2\}}, \tag{35}$$

where the last inequality follows from Lemma 3.2.

We now bound the second term on the right-hand side of (34). Thanks to Lemma 3.3, we

have

$$\begin{aligned}
\sum_{j=0}^{k-1} z_j v_j(\overline{\alpha}_k) w_j &\leq \sum_{j=0}^{k-1} v_j(\overline{\alpha}_k) w_j \\
&\leq \mu \sum_{j=0}^{k-1} v_j\left(\frac{\gamma_{\text{inc}}}{\gamma_{\text{dec}}}\overline{\alpha}_k\right)(1-w_j) \\
&= \mu\left[\sum_{j=0}^{k-1} z_j v_j\left(\frac{\gamma_{\text{inc}}}{\gamma_{\text{dec}}}\overline{\alpha}_k\right)(1-w_j) + \sum_{j=0}^{k-1}(1-z_j) v_j\left(\frac{\gamma_{\text{inc}}}{\gamma_{\text{dec}}}\overline{\alpha}_k\right)(1-w_j)\right] \\
&\leq \mu\left[\sum_{j=0}^{k-1} z_j v_j\left(\frac{\gamma_{\text{inc}}}{\gamma_{\text{dec}}}\overline{\alpha}_k\right)(1-w_j) + \sum_{j=0}^{k-1}(1-z_j)(1-w_j)\right]
\end{aligned}$$

Now, for any $j = 0, \ldots, k-1$, we have

$$z_j v_j\left(\frac{\gamma_{\text{inc}}}{\gamma_{\text{dec}}}\overline{\alpha}_k\right)(1-w_j) \leq z_j v_j(\bar{\alpha}\tilde{g}_k)(1-w_j) = 0,$$

hence the first sum in the above expression is always zero. Thus,

$$\sum_{j=0}^{k-1} z_j v_j(\overline{\alpha}_k) w_j \leq \mu \sum_{j=0}^{k-1}(1-z_j)(1-w_j). \tag{36}$$

We finally consider the last sum in the right-hand side of (34). Applying Lemma 3.4, we get

$$\begin{aligned}
\sum_{j=0}^{k-1} z_j(1-v_j(\overline{\alpha}_k))(1-w_j) &\leq \sum_{j=0}^{k-1}(1-v_j(\overline{\alpha}_k))(1-w_j) \\
&\leq \frac{1}{\mu}\sum_{j=0}^{k-1}\left(1-v_j\left(\frac{\gamma_{\text{dec}}}{\gamma_{\text{inc}}}\overline{\alpha}_k\right)\right) w_j + \log_{\gamma_{\text{dec}}^{-1}}\left(\frac{\alpha_0}{\gamma_{\text{dec}}\overline{\alpha}_k}\right) \\
&= \frac{1}{\mu}\sum_{j=0}^{k-1} z_j\left(1-v_j\left(\frac{\gamma_{\text{dec}}}{\gamma_{\text{inc}}}\overline{\alpha}_k\right)\right) w_j \\
&\quad + \frac{1}{\mu}\sum_{j=0}^{k-1}(1-z_j)\left(1-v_j\left(\frac{\gamma_{\text{dec}}}{\gamma_{\text{inc}}}\overline{\alpha}_k\right)\right) w_j + \log_{\gamma_{\text{dec}}^{-1}}\left(\frac{\alpha_0}{\gamma_{\text{dec}}\overline{\alpha}_k}\right) \\
&\leq \frac{1}{\mu}\sum_{j=0}^{k-1} z_j\left(1-v_j\left(\frac{\gamma_{\text{dec}}}{\gamma_{\text{inc}}}\overline{\alpha}_k\right)\right) w_j \\
&\quad + \frac{1}{\mu}\sum_{j=0}^{k-1}(1-z_j) w_j + \log_{\gamma_{\text{dec}}^{-1}}\left(\frac{\alpha_0}{\gamma_{\text{dec}}\overline{\alpha}_k}\right).
\end{aligned}$$

By the same reasoning used to obtain (35), we can bound the first sum in the last expression as follows:

$$\sum_{j=0}^{k-1} z_j\left(1-v_j\left(\frac{\gamma_{\text{dec}}}{\gamma_{\text{inc}}}\overline{\alpha}_k\right)\right) w_j \leq \frac{\gamma_{\text{inc}}^4 \beta}{\gamma_{\text{dec}}^4 \min\{\gamma_{\text{inc}}^{-2}\alpha_0^2, \bar{\alpha}^2\|\tilde{\boldsymbol{g}}_k\|^2\}}.$$

Therefore, the following bound holds:

$$\sum_{j=0}^{k-1} z_j(1-v_j(\overline{\alpha}_k))(1-w_j) \leq \frac{\gamma_{\text{inc}}^4\beta}{\mu\gamma_{\text{dec}}^4 \min\{\gamma_{\text{inc}}^{-2}\alpha_0^2, \bar{\alpha}^2\|\tilde{\boldsymbol{g}}_k\|^2\}} + \frac{1}{\mu}\sum_{j=0}^{k-1}(1-z_j)w_j + \log_{\gamma_{\text{dec}}^{-1}}\left(\frac{\alpha_0}{\gamma_{\text{dec}}\overline{\alpha}_k}\right). \tag{37}$$

To conclude the proof, we combine (34), (35), (36) and (37) as follows:

$$\begin{aligned}
\sum_{j=0}^{k-1} z_j &\leq \sum_{j=0}^{k-1} z_j(1-v_j(\overline{\alpha}_k))w_j + \sum_{j=0}^{k-1} z_j v_j(\overline{\alpha}_k)w_j + \sum_{j=0}^{k-1} z_j(1-v_j(\overline{\alpha}_k))(1-w_j) \\
&\leq \frac{\gamma_{\text{inc}}^2\beta}{\gamma_{\text{dec}}^2 \min\{\gamma_{\text{inc}}^{-2}\alpha_0^2, \bar{\alpha}^2[\tilde{g}_k]^2\}} + \mu\sum_{j=0}^{k-1}(1-z_j)(1-w_j) + \frac{\gamma_{\text{inc}}^4\beta}{\mu\gamma_{\text{dec}}^4 \min\{\gamma_{\text{inc}}^{-2}\alpha_0^2, \bar{\alpha}^2[\tilde{g}_k]^2\}} \\
&\quad + \frac{1}{\mu}\sum_{j=0}^{k-1}(1-z_j)w_j + \log_{\gamma_{\text{dec}}^{-1}}\left(\frac{\alpha_0}{\gamma_{\text{dec}}\overline{\alpha}_k}\right) \\
&= \frac{(\mu\gamma_{\text{dec}}^2 + \gamma_{\text{inc}}^2)\gamma_{\text{inc}}^2\beta}{\mu\gamma_{\text{dec}}^4 \min\{\gamma_{\text{inc}}^{-2}\alpha_0^2, \bar{\alpha}^2[\tilde{g}_k]^2\}} + \mu\sum_{j=0}^{k-1}(1-z_j)(1-w_j) \\
&\quad + \frac{1}{\mu}\sum_{j=0}^{k-1}(1-z_j)w_j + \log_{\gamma_{\text{dec}}^{-1}}\left(\frac{\alpha_0}{\gamma_{\text{dec}}\overline{\alpha}_k}\right) \\
&\leq \frac{(\mu\gamma_{\text{dec}}^2 + \gamma_{\text{inc}}^2)\gamma_{\text{inc}}^2\beta}{\mu\gamma_{\text{dec}}^4 \min\{\gamma_{\text{inc}}^{-2}\alpha_0^2, \bar{\alpha}^2[\tilde{g}_k]^2\}} + \max\left\{\mu, \frac{1}{\mu}\right\}\sum_{j=0}^{k-1}(1-z_j) + \log_{\gamma_{\text{dec}}^{-1}}\left(\frac{\alpha_0}{\gamma_{\text{dec}}\overline{\alpha}_k}\right) \\
&= \frac{C}{\min\{\gamma_{\text{inc}}^{-2}\alpha_0^2, \bar{\alpha}^2[\tilde{g}_k]^2\}} + \max\left\{\mu, \frac{1}{\mu}\right\}\sum_{j=0}^{k-1}(1-z_j) + \log_{\gamma_{\text{dec}}^{-1}}\left(\frac{\alpha_0}{\gamma_{\text{dec}}\overline{\alpha}_k}\right),
\end{aligned}$$

where the last line simply uses the formula (31) for C. Re-arranging the terms gives

$$\sum_{j=0}^{k-1} z_j \leq \frac{\max\left\{\mu, \frac{1}{\mu}\right\}}{1+\max\left\{\mu, \frac{1}{\mu}\right\}}k + \frac{1}{1+\max\left\{\mu, \frac{1}{\mu}\right\}}\left[\frac{C}{\min\{\gamma_{\text{inc}}^{-2}\alpha_0^2, \bar{\alpha}^2[\tilde{g}_k]^2\}} + \log_{\gamma_{\text{dec}}^{-1}}\left(\frac{\alpha_0}{\gamma_{\text{dec}}\overline{\alpha}_k}\right)\right].$$

By using the formula (24) for $\overline{\alpha}_k$ and observing that

$$\frac{\max\left\{\mu, \frac{1}{\mu}\right\}}{1+\max\left\{\mu, \frac{1}{\mu}\right\}} = \max\left\{\frac{\mu}{1+\mu}, \frac{1}{1+\mu}\right\} = p_0$$

we arrive at the desired result. □

Note that Proposition 3.1 yields an alternate definition of p_0 than that identified by Gratton et al [22], which was $\frac{\log(\gamma_{\text{dec}})}{\log(\gamma_{\text{inc}}^{-1}\gamma_{\text{dec}})}$. However, we point out that both terms in (32) are equal whenever $\gamma_{\text{inc}} = \gamma_{\text{dec}}^{-1}$, corresponding to $p_0 = \frac{1}{2}$: this particular setting has been widely used in analyzing derivative-free methods based on probabilistic properties [13].

In addition to the result of Proposition 3.1, we also have the following concentration inequality on the sum of the z_k variables.

Lemma 3.5 *Consider the sequences $\{\mathcal{D}_k\}_k$ and $\{\boldsymbol{P}_k\}_k$ generated by Algorithm 2, and suppose that the sequences are $(\kappa, D_{\max}, p)$-descent and $(\eta, \sigma, P_{\max}, q)$-well-aligned, respectively.*

Let $\pi_k(\lambda) := \mathbb{P}\left(\sum_{j=0}^{k-1} Z_j \leq \lambda k\right)$ for any $\lambda \in (0, pq)$. Then,

$$\pi_k(\lambda) \leq \exp\left[-\frac{(pq-\lambda)^2}{2pq}k\right]. \tag{38}$$

The reasoning behind this result is the same as Lemma 4.5 of Gratton et al. [20], except that the variable Z_k involves two random events. To connect it with our probabilistic properties of interest, one must use

$$\begin{aligned}\mathbb{P}\left(Z_k = 1 | Z_0, \ldots, Z_{k-1}\right) &= \mathbb{P}\left(\mathcal{D}_k \text{ is } (\kappa, D_{\max})\text{-descent } \& \ \boldsymbol{P}_k \text{ is } (\eta, \sigma, P_{\max}) \text{ well-aligned} | \mathcal{F}_{k-1}\right) \\ &= \mathbb{P}\left(\mathcal{D}_k \text{ is } (\kappa, D_{\max})\text{-descent} | \mathcal{F}_{k-1/2}\right) \\ &\quad \times \mathbb{P}\left(\boldsymbol{P}_k \text{ is } (\eta, \sigma, P_{\max}) \text{ well-aligned} | \mathcal{F}_{k-1}\right),\end{aligned}$$

which holds because the events on $\mathcal{D}_k$ and $\boldsymbol{P}_k$ are conditionally independent.

Combining the results of Proposition 3.1 and Lemma 3.5 lead to our main high-probability complexity result. The proof of this result follows from that of Theorem 4.6 in Gratton et al. [20], and merely differs by the presence of the additional logarithmic term in (30).

Theorem 3.1 *Suppose that the sequences $\{\mathcal{D}_k\}_k$ and $\{\boldsymbol{P}_k\}_k$ generated by Algorithm 2 are $(\kappa, D_{\max}, p)$-descent and $(\eta, \sigma, P_{\max}, q)$-well-aligned, respectively. Suppose further that $pq > p_0$, where p_0 is defined as in Proposition 3.1, and consider an index k and a tolerance $\epsilon > 0$ such that*

$$k \geq \frac{2}{pq - p_0}\left[\frac{(1-p_0)C}{\bar{\alpha}^2}\epsilon^{-2} + (1-p_0)\log_{\gamma_{\text{dec}}}\left(\frac{\gamma_{\text{dec}}^2 \bar{\alpha}}{\gamma_{\text{inc}}\alpha_0}\epsilon\right)\right] \tag{39}$$

with C defined as in Proposition 3.1, and

$$\epsilon \leq \min\left\{1, \frac{\gamma_{\text{inc}}^{-1}\alpha_0}{\bar{\alpha}}\right\}. \tag{40}$$

Then,

$$\mathbb{P}\left(\tilde{G}_k \leq \epsilon\right) \geq 1 - \exp\left[-\frac{(pq-p_0)^2}{8pq}k\right], \tag{41}$$

where $\tilde{G}_k$ is the random variable associated with $\tilde{g}_k$.

When $r = n$ and $\boldsymbol{P}_k = \boldsymbol{I}_n$, our result is of the same order as that of Gratton et al [20, Theorem 4.6]. The additional logarithmic term in (39) can be viewed as an additional cost incurred by the generalization of the reasoning to handle unbounded directions and random subspaces.

The result of Theorem 3.1 can be stated in a number of alternate ways, depending on the quantity of interest (gradient norm, number of iterations) and the type of result that is sought (high probability, fixed probability guarantee, in expectation). We provide below two results that are of particular interest to our approach. The first one is a high-probability complexity bound on the number of function evaluations required to reach an approximate stationary point.

Corollary 3.1 *Under the assumptions of Theorem 3.1, let N_ϵ be the number of function evaluations required by Algorithm 2 to reach a point $\boldsymbol{x}_k$ such that $\|\nabla f(\boldsymbol{x}_k)\| \leq \epsilon$, where $\epsilon > 0$ satisfies* (40)*. Then,*

$$\mathbb{P}\left(N_\epsilon \leq \left\lceil \frac{2m}{pq - p_0}\phi(\epsilon)\right\rceil\right) \geq 1 - \exp\left[-\frac{(pq - p_0)}{4pq}\phi(\epsilon)\right] \tag{42}$$

where $\phi(\epsilon) = \frac{(1-p_0)C}{\bar{\alpha}^2}\epsilon^{-2} + (1 - p_0)\log_{\gamma_{\text{dec}}}\left(\frac{\gamma_{\text{dec}}^2\bar{\alpha}}{\gamma_{\text{inc}}\alpha_0}\epsilon\right)$.

The proof of Corollary 3.1 follows from that of Theorem 4.8 in Gratton et al. [20]. Its result shows that Algorithm 2 has a high-probability complexity bound in

$$\frac{2m}{pq - p_0}\phi(\epsilon) = \mathcal{O}\left(m\frac{1}{pq - p_0}\eta^{-2}\sigma^{-2}P_{\max}^4 D_{\max}^8\kappa^{-2}\epsilon^{-2}\right) \tag{43}$$

The second corollary of our main result illustrates how our analysis leads to bounds in expectation: this result can be obtained following the argument developed for derivative-free algorithms based on probabilistic properties [21, Theorem 2.14].

Corollary 3.2 *Under the assumptions of Theorem 3.1,*

$$\mathbb{E}\left[N_\epsilon\right] \leq \frac{2m}{pq - p_0}\phi(\epsilon) + \frac{1}{1 - \exp\left(-\frac{(pq-p_0)^2}{8pq}\right)}, \tag{44}$$

where N_ϵ and $\phi(\epsilon)$ are defined as in Corollary 3.1.

As $\phi(\epsilon) = \mathcal{O}(\epsilon^{-2})$ for $\epsilon < 1$, we obtain a complexity bound that matches that of other probabilistic techniques in terms of dependencies on ϵ [21]. In addition, the dependencies on m, κ and ϵ match that obtained for direct search based on deterministic descent (7). In the next two sections, we will establish evaluation complexity bounds for several choices of subspaces and polling sets.

3.3 Examples of direction generation techniques

The above analysis of Algorithm 2 requires that our poll directions $\{\mathcal{D}_k\}_k$ are $(\kappa, D_{\max}, p)$-descent and that our subspace matrices $\{\boldsymbol{P}_k\}_k$ are $(\eta, \sigma, P_{\max}, q)$-well-aligned. We now discuss several approaches to satisfying these requirements, both deterministic and probabilistic.

We begin by noting that our framework encompasses direct search based on deterministic and probabilistic descent. Indeed, if we take $r = n$ and $\boldsymbol{P}_k = \boldsymbol{I}_n$ for every k, $\boldsymbol{P}_k$ is $(\eta, \sigma, P_{\max}, q)$-well-aligned with $\eta = \sigma = P_{\max} = q = 1$, and Algorithm 2 reduces to Algorithm 1. We then recover classical, deterministic direct search by choosing all $\mathcal{D}_k$ to be the same PSS: the sequence $\{\mathcal{D}_k\}_k$ is then $(\kappa, D_{\max}, p)$-descent with $p = 1$, and $m \geq n + 1$. Table 1 shows the values of m, κ and $D_{\max}$ for three popular approaches: the coordinate vectors and their negatives, a set of $n + 1$ vectors with uniform angles [14, Chapter 2.1], or the coordinate vectors and the vector with all entries -1.[2] For probabilistic descent, we form $\mathcal{D}_k$ by generating m independent vectors uniformly distributed on the unit sphere: in that case, $\mathcal{D}_k$ is $(\kappa, D_{\max}, p)$-descent with $D_{\max} = 1$, $\kappa = \tau/\sqrt{n}$ and $p \geq 1 - \left(\frac{1}{2} + \frac{\tau}{\sqrt{2\pi}}\right)^m$ for any $\tau \in [0, \sqrt{n}]$ [20, Appendix B]. Choosing $\tau = 1$, we can

Method	m	κ	$D_{\max}$	Success prob. p
$[\mathbf{I}_r, -\mathbf{I}_r]$	$2r$	$r^{-1/2}$	1	1
Uniform angle PSS	$r+1$	r^{-1}	1	1
$[\mathbf{I}_r, -\boldsymbol{e}_r]$	$r+1$	$(r^2+2(r-1)\sqrt{r})^{-1/2}$	$\sqrt{r}$	1
Random unit vectors	$\mathcal{O}(1)$	$r^{-1/2}$	1	$1-(1/2+1/\sqrt{2\pi})^m$.

Table 1. Summary of methods for generating a direction set $\mathcal{D}_k$ in $\mathbb{R}^r$.

make the lower bound on p sufficiently large so that it exceeds p_0 by taking $m = \mathcal{O}(1)$. These choices are summarized in Table 1.

We now explain how to generate $\boldsymbol{P}_k$ matrices that are probabilistically well-aligned (the previous paragraph gives an example of a deterministically well-aligned matrix). Consider first the requirements (12) and (13). Three possible approaches can be used:

- If $\boldsymbol{P}_k$ has entries which are i.i.d. $\mathcal{N}(0, 1/r)$ and $r = \Omega((1-\eta)^{-2}|\log(1-q)|)$, then $\boldsymbol{P}_k$ satisfies (12) and (13) [5, Theorem 2.13] with $P_{\max} = \Theta(\sqrt{n/r})$ with high probability [3, Corollary 3.11].

- If $\boldsymbol{P}_k$ is an s-hashing matrix[3] with $s = \Theta((1-\eta)^{-1}|\log(1-q)|)$ and $r = \Omega((1-\eta)^{-2}|\log(1-q)|)$, then $\boldsymbol{P}_k$ satisfies (12) and (13) for $P_{\max} = \sqrt{n}$, where the value of $P_{\max}$ comes from $\|\boldsymbol{P}_k\|_2 \leq \|\boldsymbol{P}_k\|_F = \sqrt{n}$ [24].

- If $\boldsymbol{P}_k = \sqrt{n/r}\mathbf{I}_{r\times n}\boldsymbol{Q}^T$ where $\mathbf{I}_{r\times n}$ denotes the first r rows of the $n \times n$ identity matrix $\boldsymbol{I}_n$, and $\boldsymbol{Q} \in \mathbb{R}^{n\times n}$ is the orthogonal factor in the QR decomposition $\boldsymbol{Z} = \boldsymbol{Q}\boldsymbol{R} \in \mathbb{R}^{n\times n}$ of a matrix $\boldsymbol{Z}$ with i.i.d. standard normal entries such that the diagonal entries of $\boldsymbol{R}$ are positive, then $\|\boldsymbol{P}_k\|_2 \leq \sqrt{n/r}$, so $\boldsymbol{P}_k$ satisfies (13). Moreover, for a given r and η, $\boldsymbol{P}_k$ satisfies (12) with probability $q = 1 - I_{\eta r/d}(r/2, (n-r)/2)$, where $I_p(\alpha, \beta)$ is the regularized incomplete Beta function [26, Lemma 1]. Although this does not give a closed form for a suitable choice of r, numerical evidence suggests that r may be chosen independently of n [26, Figure 1].

We now focus on condition (14), that corresponds to bounding the smallest singular value of $\boldsymbol{P}_k$ away from zero. For Gaussian matrices with entries in $\mathcal{N}(0, 1/r)$ as above, the estimate $\sigma_{\min}(\boldsymbol{P}_k^T) \sim \sqrt{n/r} - 1$ holds with high probability [32, Theorem 1.1]. In the case of orthogonal subsampling (the third case above), we automatically have $\sigma_{\min}(\boldsymbol{P}_k^T) = \sqrt{n/r}$. We are unaware of theoretical results showing that $\sigma_{\min}(\boldsymbol{P}_k) = \Theta(\sqrt{n/r})$ for hashing matrices, we present numerical evidence that this estimate holds when n is sufficiently large compared to r in Appendix A.

Table 2 summarizes our results for the three random choices of $\boldsymbol{P}_k$ described above: for such choices, since r can be chosen as small compared to the ambient dimension n, we may choose $\mathcal{D}_k$ to be a standard PSS in $\mathbb{R}^r$, such as the columns of $[\mathbf{I}_r \ -\mathbf{I}_r]$. As result, our poll step tests the points $\boldsymbol{x}_k \pm \alpha_k \boldsymbol{p}_{k,i}$, where $\boldsymbol{p}_{k,i}$ is the i-th row of $\boldsymbol{P}_k$. In that case, we obtain directions that are very similar to those used in Gratton et al. [20]. In fact, using $r = 1$ and orthogonal $\boldsymbol{P}_k$ corresponds to using $\{\pm\sqrt{n}\ \boldsymbol{v}\}$, where $\boldsymbol{v} \in \mathbb{R}^n$ is a randomly drawn unit vector. Up to a scaling constant, this recovers the method described in Gratton et al. [20, p. 1535]. Our framework

[2]We thank Warren Hare and Gabriel Jarry-Bolduc for checking the value of κ in that latter case [23].

[3]i.e. every column of $\boldsymbol{P}_k$ has exactly s nonzero entries at randomly selected locations, each taking value $\pm 1/\sqrt{s}$ with independent probability 1/2.

Method	r	Success prob. q	$P_{\max}$	σ	Comments
Identity	n	1	1	1	—
Gaussian	$\mathcal{O}(1)$	$1 - e^{-\frac{(1-\eta)^2}{Ar}}$	$\Theta(\sqrt{n/r})$	$\Theta(\sqrt{n/r})$	$P_{\max}$, σ valid with high prob.
s-Hashing	$\mathcal{O}(1)$	$1 - e^{-\frac{(1-\eta)^2}{Ar}}$	$\sqrt{n}$	$\Theta(\sqrt{n/r})$	$s = \Theta(\frac{\lvert\log(1-q)\rvert}{1-\eta})$, σ from App. A
Orthogonal	$\mathcal{O}(1)$	$1 - I_{\eta r/d}(\frac{r}{2}, \frac{n-r}{2})$	$\sqrt{n/r}$	$\sqrt{n/r}$	r estimated from [26, Figure 1].

Table 2. Summary of methods for generating $\boldsymbol{P}_k$. The value A is used to represent a universal constant.

Columns of $\mathcal{D}_k \setminus \boldsymbol{P}_k$ choice	Identity	Gaussian	s-Hashing	Orthogonal
$[\mathbf{I}, -\mathbf{I}]$	n^2	n	n	n
Uniform angle PSS	n^3	n	n	n
$[\mathbf{I}, -\boldsymbol{e}]$	n^7	n	n	n
Random unit vectors	n	n	n	n

Table 3. Summary of evaluation complexity dependency on n for different choices of $\boldsymbol{P}_k$ and $\mathcal{D}_k$ in Algorithm 2.

is however more general: for instance, using a Gaussian matrix $\boldsymbol{P}_k$ with $r = 1$ leads to the directions $\{\pm\boldsymbol{v}\}$, where $\boldsymbol{v} \in \mathbb{R}^n$ is a vector with i.i.d. standard Gaussian components.

Remark 3.1 *Compared to traditional direct search, the use of random subspaces defined by $\boldsymbol{P}_k$ increases the per-iteration linear algebra cost of Algorithm 2: however, this cost depends on the ensemble from which $\boldsymbol{P}_k$ is generated. For instance, a Gaussian $\boldsymbol{P}_k$ would have a per-iteration cost of $\mathcal{O}(nr)$ to generate the matrix, and $\mathcal{O}(mnr)$ to construct each $\boldsymbol{P}_k^T\boldsymbol{d}$, noting that $m = \mathcal{O}(r)$ for most methods considered above. By contrast, an orthogonal $\boldsymbol{P}_k$ requires $\mathcal{O}(nr^2)$ to generate $\boldsymbol{P}_k$ via QR factorization, and again $\mathcal{O}(mnr)$ to construct $\boldsymbol{P}_k^T\boldsymbol{d}$. Finally, using hashing to generate $\boldsymbol{P}_k$ would take approximately $\mathcal{O}(ns)$ work, depending on the specific method used to select the nonzero locations, and only $\mathcal{O}(m \operatorname{nnz}(\boldsymbol{P}_k)) = \mathcal{O}(mns)$ to construct $\boldsymbol{P}_k^T\boldsymbol{d}$ using sparse multiplication. Note that this cost is lower when $\mathcal{D}_k$ contains sparse vectors, such as the columns of $[\mathbf{I}_r \ -\mathbf{I}_r]$.*

3.4 Evaluation complexity results

As illustrated in Section 3.3, the framework given by Algorithm 2 may be implemented in various ways. We now compare several instances of this method using the evaluation complexity bound (43) derived in Section 3.2. We pay particular attention to the dependency on n that arises as we instantiate the method, since our reduced space approach aims at reducing this dependency.

Table 3 highlights the dependency on n for several variants of the method. Note that for classical direct search ($\boldsymbol{P}_k = \mathbf{I}_n$, $\mathcal{D}_k$ deterministic), there is a substantial difference in complexity depending on the choice of $\mathcal{D}_k$. On the contrary, using a randomized subspace choice $\boldsymbol{P}_k$ always leads to an evaluation complexity in $\mathcal{O}(n\epsilon^{-2})$, independently of the choice of $\mathcal{D}_k$. Our result recovers the $\mathcal{O}(n\epsilon^{-2})$ complexity from direct search based on probabilistic descent [20], and shows how the same result may be achieved with substantially greater flexibility. It also gives further insight into the choice $\boldsymbol{P}_k = \mathbf{I}_n$ and $\mathcal{D}_k = \{\pm\boldsymbol{v}\}$ where $\boldsymbol{v}$ is a random unit vector, briefly discussed in Gratton et al. [20, p. 1535]. As explained above, this choice may be viewed as taking $\boldsymbol{P}_k$ to be orthogonal with $r = 1$ and $\mathcal{D}_k$ as the columns of $[\mathbf{I}_n, -\mathbf{I}_n]$.

We conclude this section by commenting on the connections between our results and that obtained for model-based techniques. Unlike for direct search, in the model-based setting, using models that are of probabilistically good quality (or probabilistically fully linear [14]) is not known to improve the dependence on the dimension [21]. However, such an improvement can be obtained by using deterministically fully linear models in randomly drawn subspaces [12]. This result, together with the complexity analysis of this section, suggests that using random subspace exploration is a more general approach method for producing the complexity improvement observed in both direct-search and model-based techniques.

4 Numerical experiments

In this section, we investigate the practical performance of Algorithm 2 by comparing the following direct-search methods:

- Algorithm 1 with deterministic descent, setting $\mathcal{D}_k$ as either the columns of $[\mathbf{I}_n \ -\mathbf{I}_n]$ or $\mathcal{D}_k = [\mathbf{I}_n \ -\boldsymbol{e}_n]$, where $\boldsymbol{e}_n \in \mathbb{R}^n$ is the vector of ones;
- Algorithm 1 with probabilistic descent, using $\mathcal{D}_k = \{\pm \boldsymbol{v}\}$ where $\boldsymbol{v} \in \mathbb{R}^n$ is drawn from the uniform distribution on the unit sphere $\|\boldsymbol{v}\| = 1$;
- The Stochastic Three-Points (STP) algorithm from Bergou et al.[4], that relies on opposite directions uniform on the sphere, with step size schedule $\alpha_k = \alpha_0/(k+1)$;
- Algorithm 2 with $\boldsymbol{P}_k$ taken either to be a Gaussian matrix with $r \in \{1, 2, 3, 4, 5\}$, a hashing matrix, or a subsampled orthogonal matrix with $r \in \{1, 5\}$. The poll directions were chosen to be columns of $[\mathbf{I}_r \ -\mathbf{I}_r]$.

All methods used $\alpha_0 = 1$ and terminated if $\alpha_k < 10^{-6}$. The direct search methods used $\gamma_{\text{inc}} = 2$, $\gamma_{\text{dec}} = 0.5$, $\alpha_{\max} = 1000$, and used opportunistic polling[4] with the sufficient decrease condition

$$f(\boldsymbol{x}_k + \alpha_k \boldsymbol{P}_k^T \boldsymbol{d}_k) < f(\boldsymbol{x}_k) - \min\left(10^{-5}, 10^{-5}\alpha_k^2 \|\boldsymbol{P}_k^T \boldsymbol{d}_k\|^2\right), \tag{45}$$

where $\boldsymbol{P}_k = \boldsymbol{I}$ for standard and probabilistic direct search. All methods used in our experiments have been implemented within a Python package that has been made publicly available.[5]

4.1 Robust Regression

Our first set of experiments considers a robust linear regression problem with Gaussian data and a significant fraction of outliers, described by Carmon et al. [6]. Specifically, we perform linear regression using a smoothed biweight loss function:

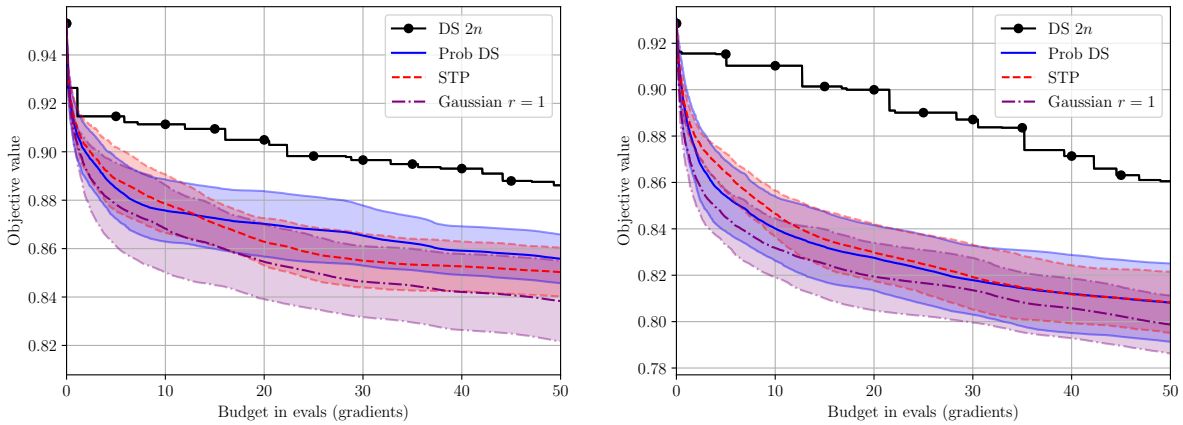
$$f(\boldsymbol{x}) = \frac{1}{m}\sum_{i=1}^{m} \phi(\boldsymbol{a}_i^T \boldsymbol{x} - b_i), \qquad \text{where} \qquad \phi(\theta) := \frac{\theta^2}{1+\theta^2}. \tag{46}$$

The vectors $\boldsymbol{a}_i \sim N(0, \mathbf{I}_n)$ are i.i.d. Gaussian vectors and the values b_i are the coordinates of $\boldsymbol{b} = \boldsymbol{A}\boldsymbol{z} + 3\boldsymbol{u}_1 + \boldsymbol{u}_2$, where $\boldsymbol{A}$ has columns $\boldsymbol{a}_i$, $\boldsymbol{z} \sim N(0, 4\mathbf{I}_m)$, $\boldsymbol{u}_1 \sim N(0, \mathbf{I}_m)$ and the entries of

[4] i.e. Do not check any other poll directions as soon as a $\boldsymbol{d}_k$ satisfying (45) is found.

[5] `https://github.com/lindonroberts/directsearch`

$\boldsymbol{u}_2$ are i.i.d. Bernoulli distributed with $p = 0.3$. We choose $n = 100$, $m = 200$, start all solvers from the origin and run each solver for a maximum of $50(n + 1)$ objective evaluations. For this initial study, we only compare direct search with $2n$ poll directions, probabilistic direct search, STP and Algorithm 2 with Gaussian $\boldsymbol{P}_k$ and $r = 1$. All randomized methods are run 10 times for a given problem.



(a) First problem instance

(b) Second problem instance

Figure 1. Objective value versus number of objective evaluations (in units of $n + 1$) for the nonconvex regression problem (46). For randomized methods, we show the average value (± 1 standard deviation) achieved over 10 instances.

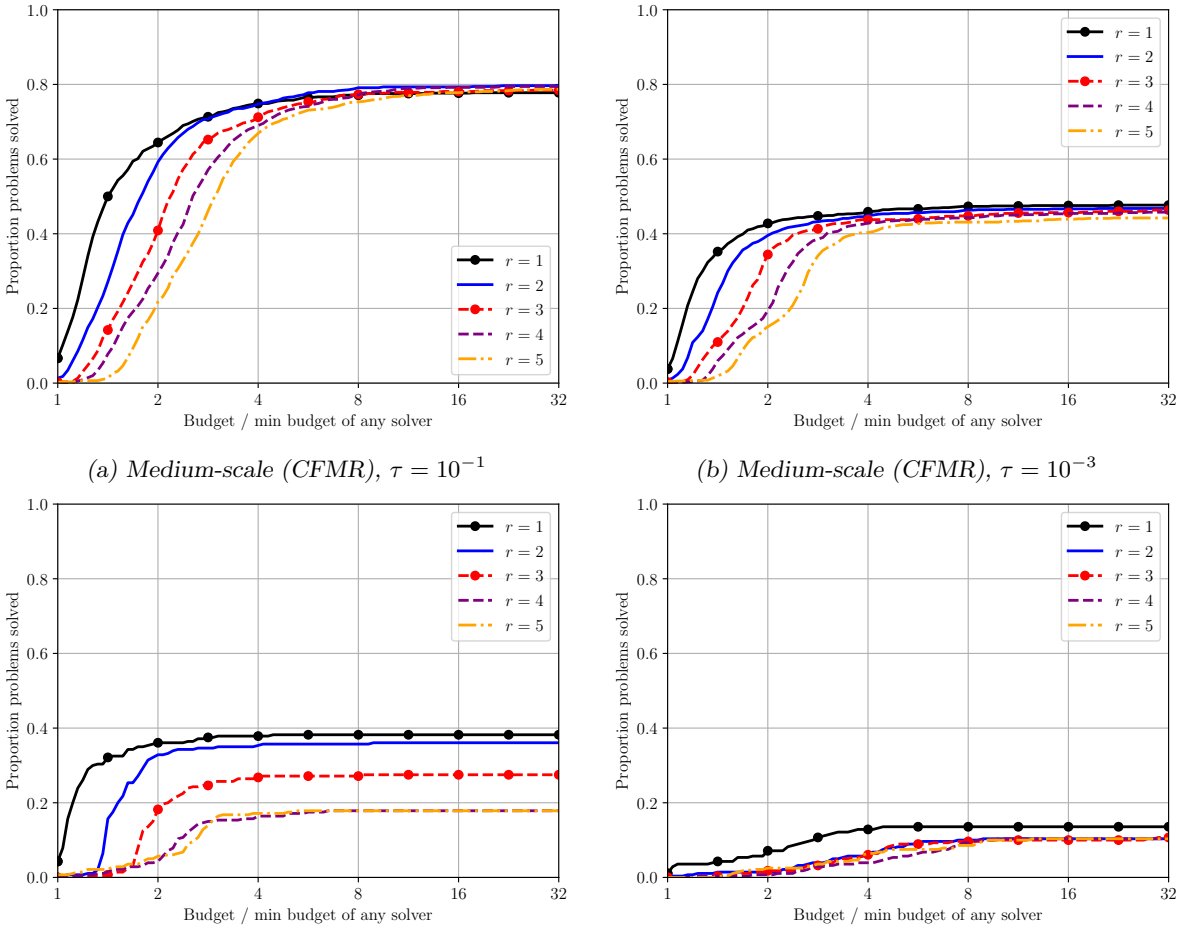
In Figure 1, we plot the average objective value reached (over the 10 runs per solver with ± 1 standard deviation error bounds) versus the number of objective evaluations normalized in terms of simplex gradients (i.e. units of $n + 1$ evaluations). We show these results for two instances of problem (46), with different choices of $\boldsymbol{a}_i$ and $\boldsymbol{b}$. All three randomized variants based on Algorithm 2 substantially outperform the traditional (deterministic) direct search method. Moreover, using randomized subspaces (effectively using Gaussian poll directions) leads to better results on average than using unit vectors as poll directions, as done in probabilistic direct search and STP.

4.2 CUTEst Collections

Our second set of experiments is based on two collections of problems from the CUTEst optimization test set [19]:

- The *CFMR* collection [8, Section 7] of 90 medium-scale problems ($25 \leq n \leq 120$, with $n \approx 100$ for 67 problems, approximately 75%), ignoring bound constraints if any;
- The *CR-large* collection [12, Appendix B] of 28 large-scale problems ($1000 \leq n \leq 5000$).

All solvers were run for at most $200(n + 1)$ objective evaluations for the CFMR collection, and at most $10(n + 1)$ evaluations for the CR-large collection. As above, all randomized methods were run 10 times on every problem.



(a) Medium-scale (CFMR), $\tau = 10^{-1}$

(b) Medium-scale (CFMR), $\tau = 10^{-3}$

(c) Large-scale (CR-large), $\tau = 10^{-1}$

(d) Large-scale (CR-large), $\tau = 10^{-3}$

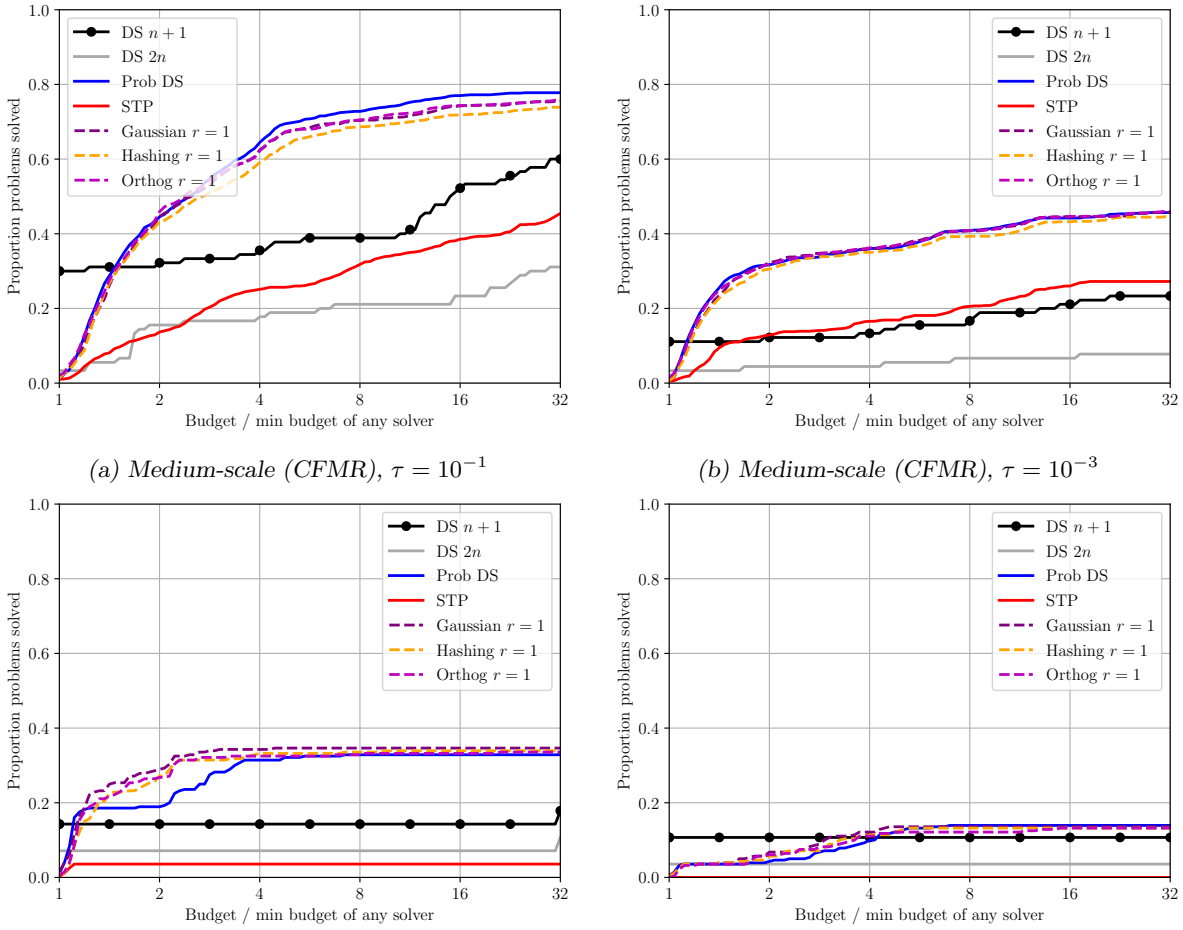
Figure 2. Comparison of r values for Gaussian sketching

In this case, for each problem P and solver instance $\mathcal{S}$, we measure the number of objective evaluations $N_{P,\mathcal{S}}$ required to achieve

$$f(\boldsymbol{x}_k) \leq f^* + \tau(f(\boldsymbol{x}_0) - f^*), \tag{47}$$

where $\tau \in (0, 1)$ measures the required accuracy and f^* is the true minimum objective value for each problem. If this accuracy level is never achieved for a particular solver instance (within the desired budget or before termination), we take the convention $N_{P,\mathcal{S}} = \infty$. We then compare solver performance using performance profiles [16], which, for a given solver $\mathcal{S}$, plots the fraction of test problems for which $N_{P,\mathcal{S}}$ is within some factor of $\min_{\mathcal{S}} N_{P,\mathcal{S}}$ (averaged over all solver instances).

Figure 2 compares Algorithm 2 with Gaussian $\boldsymbol{P}_k$ for different values of r, for both CFMR and CR-large test sets and accuracy levels $\tau \in \{10^{-1}, 10^{-3}\}$. In the case of the medium-scale CFMR problems, we see that smaller r values give the best performance, but all values are able to solve essentially the same number of problems. In the case of large-scale CR-large problems, again smaller r values give the best performance, but we find that larger r values



(a) Medium-scale (CFMR), $\tau = 10^{-1}$

(b) Medium-scale (CFMR), $\tau = 10^{-3}$

(c) Large-scale (CR-large), $\tau = 10^{-1}$

(d) Large-scale (CR-large), $\tau = 10^{-3}$

Figure 3. Comparison of different solvers

are less robust in terms of the total number of problems solved (within the given evaluation budget). Motivated by these results, in Figure 3 we compare Algorithm 2 (with Gaussian and hashing $\boldsymbol{P}_k$ and $r = 1$) with the other direct-search variants. For both problem sets and accuracy levels, the best-performing methods are probabilistic direct search (Algorithm 1 with uniformly random directions) and the two Algorithm 2 variants, which for larger performance ratios strongly outperform deterministic direct search and STP. Comparing Algorithm 2 with probabilistic direct search, we see broadly a very similar level of performance, reflecting the similarity of the two methods (both randomized with at most two objective evaluations per iteration at $\boldsymbol{x}_k \pm \alpha_k \boldsymbol{v}$ for a random vector $\boldsymbol{v}$). However, for low-accuracy solutions to the larger test problems CR-large, we see that allowing non-unit poll directions in Algorithm 2 gives a notable performance improvement over probabilistic direct search.

Overall, we find that Algorithm 2 performs similarly well to probabilistic direct search in many instances, although there appears to be a reasonable fraction of problems (in CUTEst and the nonconvex regression problem) for which our new approach gives superior performance.

5 Conclusion

We have proposed a general direct-search framework equipped with complexity guarantees that allows for polling in low-dimensional subspaces. We identified properties of both the subspaces and the polling directions that enable the derivation of complexity guarantees, which we expressed in a probabilistic form. Both our theoretical analysis and our numerical experiments suggest that using one-dimensional subspaces is an overall efficient approach that allows for applying direct-search methods to larger dimensional problems than usually considered in the literature. Interestingly, the use of random embeddings does not remove all dependencies on the original dimension of the problem, but comes with the computational advantage of drawing only two directions per iteration.

Our approach is based on characterizing properties of the polling directions at the iteration level, while information from previous iterations is typically used in practice to improve the polling set. On the other hand, removing our conditional independence yields a number of mathematical challenges, that would have to be overcome in order to obtain a complexity result. Similarly, extending this method to handle noisy function values would require combining our analysis with existing frameworks for stochastic optimization, some of which have strong similarity to direct search [17, 28]. Both avenues of research represent natural perspectives for future work.

References

[1] C. Audet and W. Hare. *Derivative-Free and Blackbox Optimization*. Springer Series in Operations Research and Financial Engineering. Springer International Publishing, 2017.

[2] A. S. Bandeira, K. Scheinberg, and L. N. Vicente. Convergence of trust-region methods based on probabilistic models. *SIAM J. Optim.*, 24:1238–1264, 2014.

[3] A. S. Bandeira and R. van Handel. Sharp nonasymptotic bounds on the norm of random matrices with independent entries. *The Annals of Probability*, 44(4), 2016.

[4] E. Bergou, E. Gorbunov, and P. Richtárik. Stochastic three points method for unconstrained smooth minimization. *SIAM J. Optim.*, 30:2726–2749, 2020.

[5] S. Boucheron, G. Lugosi, and P. Massart. *Concentration Inequalities: A Nonasymptotic Theory of Independence*. Oxford University Press, Oxford, 2013.

[6] Y. Carmon, J. C Duchi, O. Hinder, and A. Sidford. "Convex until proven guilty": Dimension-free acceleration of gradient descent on non-convex functions. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70, Sydney, 2017. PMLR.

[7] C. Cartis, T. Ferguson, and L. Roberts. Scalable derivative-free optimization for nonlinear least-squares problems. In *ICML Workshop on Beyond First Order Methods in ML Systems*, 2020.

[8] C. Cartis, J. Fiala, B. Marteau, and L. Roberts. Improving the flexibility and robustness of model-based derivative-free optimization solvers. *ACM Trans. Math. Software*, 45:32:1–32:41, 2019.

[9] C. Cartis, J. Fowkes, and Z. Shao. A randomised subspace Gauss-Newton method for nonlinear least-squares. In *Workshop on "Beyond first-order methods in ML systems" at the 37th International Conference on Machine Learning*, Vienna, Austria, 2020.

[10] C. Cartis, J. Fowkes, and Z. Shao. Randomised subspace methods for non-convex optimization, with applications to nonlinear least-squares. Technical report, University of Oxford, 2022.

[11] C. Cartis, N. I. M. Gould, and Ph. L. Toint. On the oracle complexity of first-order and derivative-free algorithms for smooth nonconvex minimization. *SIAM J. Optim.*, 22:66–86, 2012.

[12] C. Cartis and L. Roberts. Scalable subspace methods for derivative-free nonlinear least-squares optimization. arXiv:2102.12016, 2021.

[13] C. Cartis and K. Scheinberg. Global convergence rate analysis of unconstrained optimization methods based on probabilistic models. *Math. Program.*, 169:337–375, 2018.

[14] A. R. Conn, K. Scheinberg, and L. N. Vicente. *Introduction to Derivative-Free Optimization*. MPS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics, Philadelphia, 2009.

[15] M. Dodangeh, L. N. Vicente, and Z. Zhang. On the optimal order of worst case complexity of direct search. *Optim. Lett.*, 10:699–708, 2016.

[16] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Math. Program.*, 91(2):201–213, 2002.

[17] J. C. Duchi, M. I. Jordan, M. J. Wainwright, and A. Wibisono. Optimal rates for zero-order convex optimization: the power of two function evaluations. *IEEE Trans. Inform. Theory*, 61:2788–2806, 2015.

[18] R. Garmanjani, D. Júdice, and L. N. Vicente. Trust-region methods without using derivatives: Worst case complexity and the non-smooth case. *SIAM J. Optim.*, 26:1987–2011, 2016.

[19] N. I. M. Gould, D. Orban, and Ph. L. Toint. CUTEst: a constrained and unconstrained testing environment with safe threads. *Comput. Optim. Appl.*, 60:545–557, 2015.

[20] S. Gratton, C. W. Royer, L. N. Vicente, and Z. Zhang. Direct search based on probabilistic descent. *SIAM J. Optim.*, 25:1515–1541, 2015.

[21] S. Gratton, C. W. Royer, L. N. Vicente, and Z. Zhang. Complexity and global rates of trust-region methods based on probabilistic models. *IMA J. Numer. Anal.*, 38:1579–1597, 2018.

[22] 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.*, 72:525–559, 2019.

[23] W. Hare and G. Jarry-Bolduc. A deterministic algorithm to compute the cosine measure of a finite positive spanning set. *Optim. Lett.*, 14:1305–1316, 2020.

[24] Daniel M. Kane and Jelani Nelson. Sparser Johnson-Lindenstrauss transforms. *Journal of the ACM*, 61(1):1–23, 2014.

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

[26] D. Kozak, S. Becker, A. Doostan, and L. Tenorio. A stochastic subspace approach to gradient-free optimization in high dimensions. *Comput. Optim. Appl.*, 79:339–368, 2021.

[27] D. Kozak, C. Molinari, L. Rosasco, L. Tenorio, and S. Villa. Zeroth order optimization with orthogonal random directions. arXiv:2107.03941v1, 2021.

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

[29] Yu. Nesterov. Random gradient-free minimization of convex functions. Technical Report 2011/1, CORE, Université Catholique de Louvain, 2011.

[30] Yu. Nesterov and V. Spokoiny. Random gradient-free minimization of convex functions. *Found. Comput. Math.*, 17:527–566, 2017.

[31] C. Paquette and K. Scheinberg. A stochastic line search method with convergence rate analysis. *SIAM J. Optim.*, 30:349–376, 2020.

[32] M. Rudelson and R. Vershynin. Smallest singular value of a random rectangular matrix. *Communications on Pure and Applied Mathematics*, 62(12):1707–1739, 2009.

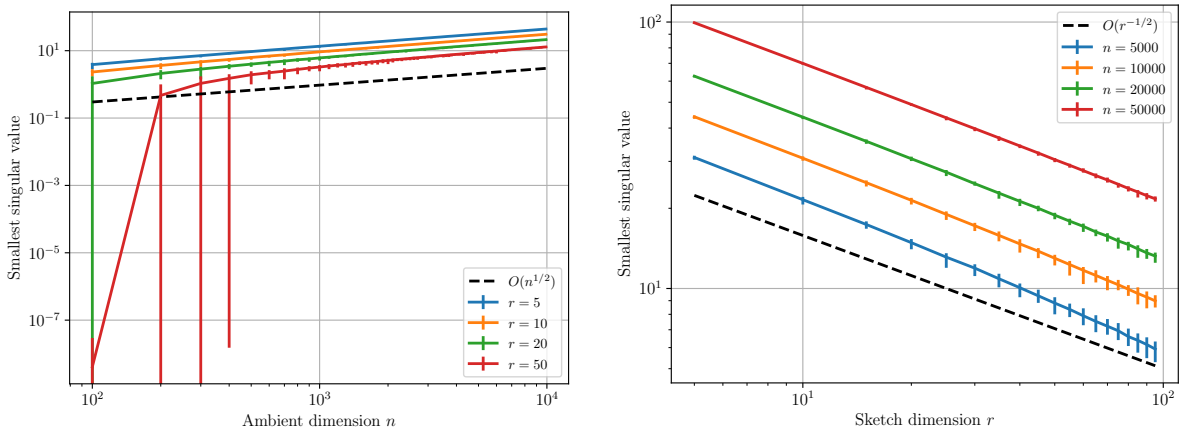
[33] Z. Shao. *On Random Embeddings and their Applications to Optimization.* PhD thesis, University of Oxford, 2022.

[34] L. N. Vicente. Worst case complexity of direct search. *EURO J. Comput. Optim.*, 1:143–153, 2013.

A Smallest Singular Value of Hashing Matrices

This appendix provides numerical evidence for the bound $\sigma = \Theta(\sqrt{n/r})$ when $\boldsymbol{P}_k$ is a hashing matrix (see the description in Section 3.3) and $\sigma > 0$ is the parameter used in Definitions 3.1 and 3.2. To this end, we generated 100 independent hashing matrices with $s = 1$ for different combinations of n and r (with $n \gg r$) and calculated their smallest singular value. Note that for hashing matrices, we have $\sigma \leq \sigma_{\min}(\boldsymbol{P}_k) = \sigma_{\min}(\boldsymbol{P}_k^T) \leq P_{\max}$, and that $P_{\max} \leq \sqrt{n}$.

In Figure 4, we plot the mean, maximum and minimum values for each of the 100 independent trials. When $n \gg r$, we observe that there is very little uncertainty in $\sigma_{\min}(\boldsymbol{P}_k)$, and the relationship $\sigma_{\min}(\boldsymbol{P}_k) = \mathcal{O}(\sqrt{n})$ holds, as shown by Figure 4a. Similarly, we see little uncertainty in $\sigma_{\min}(\boldsymbol{P}_k)$ as r varies (see Figure 4b), while the relationship $\sigma_{\min}(\boldsymbol{P}_k) = \mathcal{O}(1/\sqrt{r})$ holds. Altogether, these results suggest that the value $\sigma = \Theta(\sqrt{n/r})$ leads to satisfaction of (14) with high probability when $\boldsymbol{P}_k$ is a hashing matrix.



(a) $\sigma_{\min}(\boldsymbol{P}_k)$ *as a function of* n.

(b) $\sigma_{\min}(\boldsymbol{P}_k)$ *as a function of* r.

Figure 4. Computed values of $\sigma_{\min}(\boldsymbol{P}_k)$ *(minimum nonzero singular value) for hashing with* $s = 1$*, as a function of* n *and* r.