amse
école d'économie d'aix-marseille
aix-marseille school of economics

Working Papers / Documents de travail

Hybrid Random Concentrated Optimization Without Convexity Assumption

Pierre Bertrand
Michel Broniatowski
Wolfgang Stummer

WP 2025 - Nr 24

Hybrid Random Concentrated Optimization Without Convexity Assumption

Pierre Bertrand[1], Michel Broniatowski[2], Wolfgang Stummer[3]
[1] Aix Marseille Univ, CNRS, AMSE, Marseille, France;
[2]LPSM, Sorbonne University, Paris, France;
[3] Maths Department, University of Erlangen-Nürnberg (FAU), Erlangen, Germany

December 16, 2025

Abstract

We propose a new random method to minimize deterministic continuous functions over subsets $\mathcal{S}$ of high-dimensional space $\mathbb{R}^K$ without assuming convexity. Our procedure alternates between a Global Search (GS) regime to identify candidates and a Concentrated Search (CS) regime to improve an eligible candidate in the constraint set $\mathcal{S}$. Beyond the alternation between those completely different regimes, the originality of our approach lies in leveraging high dimensionality. We demonstrate rigorous concentration properties under the CS regime. In parallel, we also show that GS reaches any point in $\mathcal{S}$ in finite time. Finally, we demonstrate the relevance of our new method by giving two concrete applications. The first deals with the reduction of the ℓ_1-norm of a LASSO solution. Secondly, we compress a neural network by pruning weights while maintaining performance; our approach achieves significant weight reduction with minimal performance loss, offering an effective solution for network optimization.

[1]

1 Introduction

In high-dimensional constrained optimization, finding the minimum of a function without strong assumptions such as convexity remains a major challenge. Many classical methods, including gradient-based approaches, rely on smoothness and convexity properties, which are not always satisfied in currently omnipresent real-world applications, particularly in machine learning and neural network compression. As far as the latter is concerned, it is well known that

[1]The project leading to this publication has received funding from the French government under the "France 2030" investment plan managed by the French National Research Agency (reference :ANR-17-EURE-0020) and from Excellence Initiative of Aix-Marseille University - A*MIDEX.

optimally accelerating network inference is an increasingly pressing issue due to the growing complexity of networks and their reuse as terminal networks. Various methods exist to speed up an existing network, such as training multiple networks (see e.g. [21], [21]) or distillation (see e.g. [22]); for general surveys on neural network compression, the reader is e.g. referred to the comprehensive works of [9], [20], [16], [19], [27], as well as [2].

Another omnipresent issue in machine learning, artificial intelligence and its adjacent research fields is the strive for model sparsity, which can e.g. be tackled by optimization techniques such as the widely used ℓ_1-norm-minimizing LASSO method (cf. [25]; see also e.g. [8], for a broader embedding).

Of course, there is a huge amount of other tasks in machine learning which are modeled as optimization problems, often in high-dimensional spaces (for a recent, very comprehensive and detailed overview, see e.g. the book of [24], [24]).

Inspired by the above-described issues, we develop a new *general* method for finding the minimum of a function f on a constraint set (search space) $\mathcal{S} \subset \mathbb{R}^K$ (for instance, $\mathcal{S}$ may be connected with the pruning of the K weights of a neural network). More detailed, our *random* method has two regimes: the first one is a Global Search (GS) and the second one is a Concentrated Search (CS). The GS phase enables broad exploration of the search space $\mathcal{S}$, ensuring that the algorithm does not get trapped in local minimizers. The CS phase refines promising candidates by exploiting local structure, improving convergence towards an optimal or near-optimal solution. The originality of our approach lies in leveraging high dimensionality: we demonstrate that CS benefits from concentration properties, while GS guarantees reaching any point in finite time.

In Section 2 we first introduce in detail the two above-mentioned regimes GS and CS, and discuss their alternating interplay as well as their algorithmic manifestation. Afterwards, in Section 3, we derive the underlying rigorous properties. In both regimes, the new point is located within a ball whose radius depends on the norm of the center. Proposition 1 states that, together with some additional control, the entire set $\mathcal{S}$ is reachable by GS through this generation scheme. Furthermore, concerning CS we prove in Lemma 1 that its realizations are highly concentrated within an arbitrarily small neighborhood of the current point. Moreover, in Proposition 2, we demonstrate that this neighborhood is well covered. As an interesting side effect, we discover that the higher the dimension K is, the more effective our properties become.
To illustrate the effectiveness of our proposed new method, we present two corresponding applications. First, in Section 4 we deal with the above-mentioned LASSO context, and reduce the "sparsity-quantifying" ℓ_1-norm of the LASSO minimizer while keeping "almost-equal" performance quality (the latter being manifested through an appropriate construction of the constraint set $\mathcal{S}$). These investigations also provide insight into the alternating interplay between the GS and CS regimes. Our second application concerns with the above-mentioned topic of neural network compression: for this, we define the function f in connec-

tion with the pruning percentage at a given threshold, and specify the constraint set $\mathcal{S}$ in connection with the performance score. Accordingly, our method takes as input a weights-vector provided by an initial training and optimizes its pruning rate. The main challenge remains maintaining performance (see e.g. Table 1 in [19]). Our method serves as a complementary approach in this context and has the advantage of providing convergence guarantees.

Both applications highlight the versatility and potential of our method for large-scale optimization problems. While some random search approaches are known to be inefficient in some contexts, we argue that — when properly integrated with concentrated search — random search can become a powerful tool for escaping local optima and for "navigating" on complicated landscapes. Moreover, our results suggest that weak regularity assumptions suffice to achieve meaningful improvements over some traditional methods.

2 Description of the method

In this section, we explain in detail our new hybrid random concentrated optimization approach. For this and the rest of the paper, we denote by x^k the k-th component of a vector $x \in \mathbb{R}^K$.

2.1 Objective and principle structure

Our objective is to solve the constrained minimization problem

$$\min_{s \in \mathcal{S}} f(s) \tag{1}$$

where f is a function on a set $\mathcal{S} \in \mathbb{R}^K$ of high-dimensional vectors (i.e. K is very large); notice that we tackle this problem "directly", i.e. no regularization (or similar) method is involved. Throughout this paper, we only need two "light" assumptions, namely that f is continuous (but not necessarily differentiable) and that the constraint set (search space) $\mathcal{S}$ is an open connected set (and therefore a path-connected set). In particular, f is allowed to be non-convex and the same holds for $\mathcal{S}$ (for the latter, see e.g. Remark 3 for a concrete situation in the context of neural network compression). This lack of convexity properties leads us to a $K-$dimensional random walk (on *continuous* space) approach rather than a gradient descent approach. For this, we iteratively generate realizations/simulations of random variables traversing the space $\mathbb{R}^K$, control their belonging to $\mathcal{S}$ and keep the outcome with the minimal f. In doing so, the *fundamental structure* of our new hybrid random optimization method consists of three major components: (i) a Global Search (GS) regime, (ii) a Concentrated Search (CS) regime, and (iii) a scheme of switching between these two regimes. Basically, the role of GS is "vast" candidate-solution-exploration of the entire space whereas the role of CS amounts to a "locally refined" candidate-solution-search which effectively takes into account high-dimensional concentration properties of the underlying random simulations.

For further verbal explanations of the underlying main structure of our new approach, let us call — as common — any element in $\mathcal{S}$ a *candidate solution*. For our approach, we start with a candidate solution $s_0 \in \mathcal{S}$, and also set the "control-variable" $c_0 := s_0$. At the beginning of the iterative step $m \to m+1$, we have two components at hand: the best-so-far (incumbent) candidate solution $s_m \in \mathcal{S}$ (which is the candidate solution obtained throughout the first m steps which has the lowest function value) and a "control-variable" value $c_m \in \mathcal{S}$. In case that the iterative step belongs to the GS regime, we simulate a random normal (Gaussian) vector a_m with mean c_m and a covariance matrix which also depends on c_m. Afterwards, a_m is deterministically transformed — through a "cut-off-filter" — to the vector α_m which satisfies componentwise

$$|\alpha_m^k - c_m^k| \leq C * |c_m^k| \quad \text{for all } k = 1, \ldots, K \tag{2}$$

for some constant $C > 0$ (and hence, the ℓ_1-distance/norm relationship $||\alpha_m - c_m||_1 \leq C * ||c_m||_1$ holds); then, from α_m the new updated vectors s_{m+1} and c_{m+1} are computed in a certain GS-dependent way.

In case that the iterative step belongs to the CS regime, we simulate a random normal (Gaussian) vector $a_m = \alpha_m$ with mean s_m and a covariance matrix which also depends on s_m and and which additionally takes into account the above-mentioned concentration properties. Eventually, from α_m the new updated vectors s_{m+1} and c_{m+1} are computed in a certain CS-dependent way. This iteration process stops when we have found candidate solutions for which f becomes sufficiently small or when the best-so-far function value f no longer decreases over an arbitrary number of steps.

In the following, we explain the full details of the above-mentioned components.

2.2 Global Search (GS) Regime

Recall that at the beginning of the iterative step $m \to m+1$, we have $s_m \in \mathcal{S}$ and $c_m \in \mathcal{S}$ at hand. As indicated above, the GS regime is used to explore the space when "far from the boundary of $\mathcal{S}$". Accordingly, for each component $k \in \{1, \ldots, K\}$ we draw independently a single random variable a_m^k which is normal (Gaussian) with mean (center) c_m^k and standard deviation $\sigma_m^k := \frac{|c_m^k|}{3}$, i.e.

$$a_m^k := W^k \sim \mathcal{N}(c_m^k, (\sigma_m^k)^2)\,;$$

this is deterministically transformed into

$$\alpha_m^k := \begin{cases} a_m^k, & \text{if } |a_m^k - c_m^k| \leq C * |c_m^k|\,, \\ c_m^k, & \text{otherwise,} \end{cases} \tag{3}$$

where $C > 2$ is a pregiven constant (cf. Proposition 1, $C = 2.1$ in our experiments). Clearly, (3) implies (2).

Notice that for $c_m^k \neq 0$, the condition in the first line of (3) is equivalent to $\left|\frac{a_m^k}{c_m^k} - 1\right| \leq C$, and that $\frac{a_m^k}{c_m^k}$ expresses the (dis)similarity of a_m^k from c_m^k.

This construction (3) ensures that the new vector $\alpha_m := (\alpha_m^1, \ldots, \alpha_m^K)$ remains relatively close to $c_m \in \mathcal{S}$ and thus "increases the chances" of being in $\mathcal{S}$.

Moreover, the condition in the first line of (3) is highly likely fulfilled for many k's, since (with $\sigma_m^k = \frac{|c_m^k|}{3}$):

$$\mathbb{P}\left(\left|\frac{a_m^k}{c_m^k} - 1\right| \leq 1\right) = \mathbb{P}\left(\left|\frac{a_m^k - c_m^k}{\sigma_m^k}\right| \leq 3\right) = 0.9973$$

according to the well-known $3\sigma-$rule of Gaussian distributions. This choice of variance implies that the larger $|c_m^k|$ is, the more scattered is a_m^k. Therefore, if $c_m \in \mathcal{S}$ is very close to the boundary $\partial\mathcal{S}$, then a_m may be in $\mathcal{S}$ and very close to $\partial\mathcal{S}$ with substantially small probability, since some of its components may fullfill the condition in the first line of (3) but may be nevertheless too large to keep the resulting vector α_m in $\mathcal{S}$.

In the current GS regime, the update of the control-variable value c_m is constructed by

$$c_{m+1} := \begin{cases} \alpha_m, & \text{if } \alpha_m \in \mathcal{S}\,, \\ c_m, & \text{otherwise;} \end{cases}$$

moreover, the best-so-far (incumbent) candidate solution $s_m \in \mathcal{S}$ is updated according to

$$s_{m+1} := \begin{cases} \alpha_m, & \text{if } \alpha_m \in \mathcal{S} \text{ and } f(\alpha_m) < f(s_m)\,, \\ s_m, & \text{otherwise.} \end{cases}$$

Clearly, by construction we have $s_{m+1} \in \mathcal{S}$ and $c_{m+1} \in \mathcal{S}$.

2.3 Concentrated Search (CS) Regime

Recall that at the beginning of the iterative step $m \to m+1$, we have $s_m \in \mathcal{S}$ and $c_m \in \mathcal{S}$ at hand. As indicated above, the CS regime aims at exploring *small neighborhoods* of the best-so-far candidate solution s_m by simulating a random vector a_m with a control on the domain of the *high-dimensional* space $\mathbb{R}^K$ which bears positive mass; a_m will indeed concentrate on such domain. The ingredients to be used for making such replicates are twofold:

- firstly, a_m should concentrate sharply around s_m. A way is to adopt the standpoint elaborated in [3, 4] which amounts to simulate sequences $a_m := (a_{m,n})_{n\geq 1}$ which for large n concentrate sharply and rapidly around s_m. The integer n can be considered as a tuning parameter. When specialized to the Gaussian setting, we may adopt the same construction with the benefit that the concentration of $(\alpha_{m,n})_{n\geq 1}$ around s_m can be calibrated in terms of the dimension K for any given value of n, which anyhow should be chosen as a multiple of K.

- secondly, the high-dimensionality should be taken into account in order to provide some insight on the special domain in $\mathbb{R}^K$ which bears the realizations of $a_{m,n}$ with lower-bounded probability for fixed n. Such specific concentration features are well known for Gaussian vectors in high-dimensional spaces; see e.g. [28].

Both requirements argue in favor of choosing normally (Gaussian) distributed vectors $a_{m,n}$. We formally construct an appropriate $a_m = a_{m,n}$ as follows. To start with, we choose n as a multiple of K and consider n as a "free parameter". Since the latter is large, and as we will observe in the applications, $n = K$ is sufficient to ensure concentration.

Similarly to the GS regime, in the CS regime we employ for a_m a Gaussian random vector whose (diagonal-form) covariance matrix also depends on its mean/center (here, s_m), however with additional concentration features. For this, we define the standard-deviations vector

$$\sigma_m := \left(\sigma_m^1, \ldots, \sigma_m^K\right)$$

whose k-th coordinate is $\sigma_m^k := |s_m^k|$. Furthermore, we group the indices $\{1, \ldots, n\}$ into K blocks of equal size $\frac{n}{K}$ (which is an integer, by assumption):

$$I_k := \left\{(k-1) * \frac{n}{K} + 1, \ldots, k * \frac{n}{K}\right\}, \qquad k = 1, \ldots, K. \tag{4}$$

To proceed, let $\{W_i^k : k \in \{1, \ldots, K\}, i \in I_k\}$ be a family of independent identically distributed random variables, with $\mathcal{N}(0,1)$ (i.e. a standard Gaussian) distribution. From this, we construct for all $k = 1, \ldots, K$, and all $i \in I_k$ the random variables

$$\tilde{W}_i^k := \sigma_m^k W_i^k + K s_m^k \tag{5}$$

and subsequently

$$a_m^k := \frac{1}{n} \sum_{i \in I_k} \tilde{W}_i^k \ .$$

Accordingly, the random vector

$$a_m := \left(a_m^1, \ldots, a_m^K\right)$$

has mean

$$\mathbb{E}(a_m) = s_m$$

so that a_m is centered as desired; moreover, a_m has covariance matrix of diagonal form, with the diagonal elements

$$\mathrm{Var}\left[a_m^k\right] = \frac{(s_m^k)^2}{nK}, \qquad k = 1, \ldots, K, \tag{6}$$

which indicates that the realization is much more concentrated than in the GS regime. Indeed, the divisor nK is — by construction — a multiple of K^2, with

K being large (e.g. we have $K = 404234$ in the neural-network-compression context of Subsection 5.2 below).
Skipping "control" (2) in the current CS regime, we directly have $\alpha_m = a_m$. Ultimately, in CS regime, the update of the control-variable value c_m is constructed by

$$c_{m+1} := c_m,$$

i.e. it remains unaltered; moreover, the best-so-far (incumbent) candidate solution $s_m \in \mathcal{S}$ is updated according to

$$s_{m+1} := \begin{cases} \alpha_m, & \text{if } \alpha_m \in \mathcal{S} \text{ and } f(\alpha_m) < f(s_m)\,, \\ s_m, & \text{otherwise.} \end{cases}$$

Clearly, by construction we have $s_{m+1} \in \mathcal{S}$ and $c_{m+1} \in \mathcal{S}$.

2.4 Alternating Between Regimes

The two regimes play distinct roles: the concentration of CS is much stronger than that of GS. We will have the opportunity in Section 3 to quantify the differences and demonstrate the role of each: ultimately GS finds a candidate close to the minimum of f, and CS refines it.

These roles naturally lead to starting with GS and then, when the current minimum no longer evolves, switching to CS to refine near the boundary. This approach works, and its application to network compression is detailed in Subsection 5.3.1.

However, the choice of regime change is crucial and results in a very localized search around the current center after the switch; escaping this region becomes almost impossible. To avoid this pitfall, we propose to permanently alternate between GS and CS at every step: beginning at $m = 0$, even steps use GS, and odd steps use CS. The results of this method, which we recommend as it removes the regime-switching parameter, are presented in Subsection 5.3.2.

In the alternating case, the GS regime follows its own control-variable c within $\mathcal{S}$. Updates of c are not considering the value of f. The CS regime simulates locally around the best-so-far candidate solution (current minimizer) s whose updates occur only if f decreases and the point is in $\mathcal{S}$. Besides, if GS finds a new best-so-far candidate solution, the control-variable s of CS is updated. Conversely, CS cannot update the GS control-variable c.

2.5 Remarks

- Both means (centers) c_m and s_m belong to $\mathcal{S}$. Moreover, the center s_m always contains the best-so-far minimizer.

- The constant C used in the "control" (2) is a parameter to be set. As long as $C > 2$, any point in $\mathcal{S}$ is reachable regardless of the initial center (mean) c_0 (cf. Proposition 1).

- The choice of the normal (Gaussian) distribution is neither critical in the GS regime nor in the CS regime. One could use log-concave distribution while preserving the conclusions of all the propositions in this paper. The code, in fact, provides the option to select between the normal and Laplace distributions.

- When K is large — and since n is assumed to be a multiple of K — the choice $n = 1 * K$ already ensures significant concentration around the center in the CS regime.

2.6 Synthesis

In a similar spirit to *e.g.* [12] (who deal with simulated annealing in combinatorial/discrete optimization problems), for better transparency we synthesize our new *Controlled Iterated Concentrated Search (CICS)* procedure into eight different components, to end up with the (pseudo-)algorithm displayed in Algorithm 1.

The eight components in the above CICS are:

- the choice of the initial candidate solution INITIAL CANDIDATE SOLUTION s_0 (line 1 of the algorithm) given by the application;

- the choice of the underlying REGIME SWITCHING SCHEME (lines 2 and 8); the underlying label ℓ has values 1 or 2 (line 9); here described in Subsection 2.4;

- the STOPPING CRITERION which determines when the execution is finished (line 7); here is left free to the user;

- the GLOBAL SEARCH REGIME (label $\ell = 1$) which is used to generate new candidate solutions which are highly likely *substantially far* from the best-so-far (incumbent) candidate solution (line 10); here described in Subsection 2.2;

- the CONCENTRATED SEARCH REGIME (label $\ell = 2$) which is employed to generate new candidate solutions which are highly likely *close* to the best-so-far candidate solution (line 14); here described in Subsection 2.3;

- the INITIAL CONTROL, which determines the initial control parameter c_0 (line 6); here set to be equal to s_0;

- the CANDIDATE SOLUTION UPDATE which determines the exact formula of updating the best-so-far candidate solution in terms of the auxiliary random point α which is simulated in dependence of the current control parameter c (lines 11 and 15); here described at the end of both Subsections 2.2 and 2.3;

- the CONTROL UPDATE which determines the exact formula of updating the current control parameter c in terms of the above-mentioned auxiliary random point α (lines 12 and 16); here described at the end of both Subsections 2.2 and 2.3.

Algorithm 1: Component-based formulation of the CICS. The components are written in SMALLCAPS.

Given:
an open and arc-connected search space (constraint set) $\mathcal{S} \subset \mathbb{R}^K$ with high-dimension;
a continuous function $f : \mathcal{S} \mapsto \mathbb{R}$ to be minimized.

Objective:
to find an approximation of the optimal solution s^* and the corresponding optimal objective-function value $f(s^*)$.

Input:
an INITIAL CANDIDATE SOLUTION s_0,
a REGIME SWITCHING SCHEME.

Output:
the best candidate solution s^*_{appr} found during the search and the corresponding objective-function value $f(s^*_{appr})$.

$\hat{s} \leftarrow s_0$;
$s^*_{appr} \leftarrow s_0$;
$m \leftarrow 0$;
$c_0 \leftarrow$ initialize the control parameter according to INITIAL CONTROL;
while STOPPING CRITERION *is not met* **do**
  while REGIME SWITCHING SCHEME *is executed* **do**
    if *label* ℓ *in* REGIME SWITCHING SCHEME *is equal to 1* **then**
      simulate an auxiliary point α_m according to GLOBAL SEARCH REGIME;
      in terms of α_m, compute the best-so-far candidate solution s_{m+1} according to CANDIDATE SOLUTION UPDATE 1;
      in terms of α_m, compute the control parameter c_{m+1} according to CONTROL UPDATE 1;
    else
      simulate an auxiliary point α_m according to CONCENTRATED SEARCH REGIME;
      in terms of α_m, compute the best-so-far solution s_{m+1} according to CANDIDATE SOLUTION UPDATE 2;
      in terms of α_m, compute the control parameter c_{m+1} according to CONTROL UPDATE 2;
  end
  $s^*_{appr} \leftarrow \widehat{s}_{m+1}$
  $m \leftarrow m + 1$;
end
return s^*_{appr} *and* $f(s^*_{appr})$;

3 Properties of the Algorithm

This section justifies the relevance of the two regimes GS and CS by formally demonstrating that each achieves its objective:

- GS explores the entire space in finite time. Indeed, in Proposition 1 below (cf. Subsection 3.1) we will show that for any arbitrary initial control $c_0 \in \mathcal{S}$, the GS regime will always reach — in finite time and with strictly positive probability — any arbitrarily small neighborhood of the global minimizer (i.e. the overall-best candidate solution, which is not necessarily unique) of f.

- In contrast, in the below-mentioned Lemma 1 and Proposition 2 dedicated to the CS regime (cf. Subsection 3.2) we will show that:
  i) our CS-search for better (i.e. lower-f-valued) candidate solutions concentrates in a neighborhood of the best-so-far candidate solution s_m through the reduction of its variance caused by a large parameter n, and
  ii) that the realizations of our correspondingly employed simulations (of a_m) are most likely not at the center of this neighborhood, due to an argument based on the high dimension K of the underlying space.

In the following, we give the exact details.

3.1 GS Explores the Space

The two regimes run in parallel, and CS has no impact on GS since c_m cannot be updated by CS. In this section, we focus on the progression of the algorithm under GS, as if each step m were performed under GS.

proposition 1 (The Algorithm Explores All of $\mathcal{S}$)**.** *Let β be a point in $\mathcal{S}$. For any initial center $c_0 \in \mathcal{S}$ and any $\eta > 0$ small enough to ensure $B_\eta(\beta) \subset \mathcal{S}$, there exists $m_0 \in \mathbb{N}$ such that $\mathbb{P}(c_{m_0+1} \in B_\eta(\beta)) > 0$.*

Proof. Since $\mathcal{S}$ is arc-connected, there exists a path $\gamma : [0,1] \mapsto \mathcal{S}$ such that $\gamma(0) = c_0$ and $\gamma(1) = \beta$. We will construct a sequence of centers $c_0, \ldots, c_{m_0}$ under the GS regime such that the probability of transitioning from one to the next is nonzero, and such that the target ball $B_\eta(\beta)$ becomes reachable from c_{m_0}.

During the GS regime, any point can be reached in a single step with nonzero probability (albeit potentially small) by a_m. However, we have imposed a control mechanism (namely, (2)) for transitioning from a_m to α_m. We will show that despite this control, the ball around β remains reachable in a finite number of iterations.

The path begins at c_0, an arbitrary point in $\mathcal{S}$. First, we define

$$\underline{\delta_0} = \min_k |c_0^k|$$

and assume without loss of generality that $\eta < \underline{\delta_0}$.

For $\delta \in \mathbb{R}_+^K$ and $c \in \mathbb{R}^K$, we define the hyperrectangle

$$H_{C*\delta}(c) = \{x \in \mathbb{R}^K \mid |x^k - c^k| \leq C * \delta^k \text{ for all k}\}.$$

Additionally, we define

$$H_m = H_{C*\delta_m}(c_m) - int\,(H_{2*\delta_m}(c_m))$$

where

$$\delta_m^k = |c_m^k|,\ \forall 1 \leq k \leq K.$$

Since $C > 2$, H_m is nonempty. The purpose of this "annulus" H_m is to maintain $\min_k \delta_m^k \geq \underline{\delta_0}$ along the path $c_0, \ldots, c_{m_0}$; indeed along iterations, the absolute value of each coordinate is striclty increasing. This ensures that $\delta_{m+1}^k > \delta_m^k > \underline{\delta_0}$ for all k. This allows (2) to be less and less restrictive and therefore increases the speed along path γ. Let us now restrict realizations on a tubular neighborhood of γ included in $\mathcal{S}$. We define

$$t_m = \sup\{t \in [0,1] \mid \gamma(t) \in H_{C*\delta_m}(c_m)\}$$

and $\beta_m = \gamma(t_m)$.

Case 1: $t_m = 1$. In this case, $\beta_m = \beta \in H_{C*\delta_m}(c_m)$, and $B_\eta(\beta)$ becomes reachable by α_m with nonzero probability. Specifically, "$a_m \in (B_\eta(\beta) \cap H_{C*\delta_m}(c_m))$" is an event with nonzero probability. Furthermore, $a_m \in H_{C*\delta_m}(c_m)$ implies that the control (2) is satisfied and $\alpha_m = a_m$, which concludes the argument: $m = m_0$.

Case 2: $t_m < 1$. In this case, there exists $\epsilon \in [0, \eta]$ such that $B_\epsilon(\beta_m) \subset \mathcal{S}$ because $\mathcal{S}$ is open. Now, "$a_m \in (B_\epsilon(\beta_m) \cap H_m)$" is an event with nonzero probability. Moreover, $a_m \in H_m$ implies that the control (2) is satisfied and $\alpha_m = a_m$. Since $B_\epsilon(\beta_m) \subset \mathcal{S}$, we have $\alpha_m \in \mathcal{S}$, and thus

$$c_{m+1} = \alpha_m = a_m.$$

Additionally, by the definition of H_m, we have that $\delta_{m+1}^k > \delta_m^k > \underline{\delta_0}$ for all k. Next, we analyze the path between c_m and c_{m+1}. Since $c_{m+1} \in B_\epsilon(\beta_m)$ and $\epsilon < \eta < \underline{\delta_0} \leq \min_k \delta_{m+1}^k$, we have $\beta_m \in B_\epsilon(c_{m+1}) \subset H_{C*\delta_{m+1}}(c_{m+1})$. Furthermore, because of $C > 2$ we obtain

$$B_{\underline{\delta_0}}(\beta_m) \subset B_{\underline{\delta_0}+\epsilon}(c_{m+1}) \subset H_{2*\delta_{m+1}}(c_{m+1}) \subset H_{C*\delta_{m+1}}(c_{m+1}).$$

Thus, $\beta_m \in H_{C*\delta_{m+1}}(c_{m+1})$, leading to $t_{m+1} \geq t_m$. Now, either $t_{m+1} = 1$ or $||\beta_{m+1} - \beta_m||_1 \geq \underline{\delta_0}$; indeed, β_{m+1} belongs to the annular with center c_{m+1} and whose radius is greater than $\underline{\delta_0}$.

In conclusion, we construct a sequence of points

$$\beta_0 = \gamma(0) = c_0, \ldots, \beta_m = \gamma(t_m), \ldots, \beta_{m_0+1} = \gamma(1) = \beta$$

along the path γ. To these points are associated a list of centers $c_0, \ldots, c_{m_0}$ reached by the algorithm under the GS regime and which respect $c_m \in B_\eta(\beta_m)$ for all m.

The finite integer m_0 exists because at each step, $t_{m+1} > t_m$ and $||\beta_{m+1} - \beta_m||_1 \geq \underline{\delta_0}$ or $t_{m+1} = 1$. We can even estimate: $m_0 \leq \frac{||\gamma||_1}{\underline{\delta_0}}$. □

3.2 CS Explores the Neighborhoods of Local Minima

This section analyzes the exploration properties under the CS regime, a method for exploring the local neighborhoods of successive best-so-far minimizers s_m of f obtained during previous iterations of the algorithm. Under the hypothesis of a theoretical minimizer near the boundary, CS refines the raw center s_m provided by GS; the small amplitude concentrated search is performed to refine the minimization.

For sake of clearness we simplify several notations: center s_m is quoted as μ; the standard-deviation vector $\sigma_m = (\sigma_m^1, \ldots, \sigma_m^K)$ is quoted as σ; $X_n(\sigma)$ denotes the realizations $a_{m,n}$ for which the relevant parameter here is n whereas m can be dropped.

We construct sequences of Gaussian vectors $X_n(\sigma) := \left(X_n^1(\sigma^1), \ldots, X_n^K(\sigma^K)\right)$ taking values in $\mathbb{R}^K$ with independent components such that $\mathbb{E}\left[X_n(\sigma)\right] = \mu = \left(\mu^1, \ldots, \mu^K\right)$, and whose co-variance matrix is $\sigma I_{\mathbb{R}^K} \sigma^T$. Here, σ is a strictly positive vector in $\mathbb{R}^K$.

Denote by $V_{\bar{\sigma}}$ (resp. $V_{\underline{\sigma}}$) $\in \mathbb{R}^K$ the vector for which all components are equal to $\bar{\sigma} = \max_k \sigma^k$ (resp. $\underline{\sigma} = \min_k \sigma^k$) and let

$$S_n^2 := \frac{(\bar{\sigma})^2}{nK}.$$

lemma 1 (The CS regime explores the neighborhood of its center)**.** *The random vector $X_n(\sigma)$ concentrates in an annulus around μ with probability lower-bounded by $\eta \geq 0$ independently of n.*

Formally, there exists a constant $\eta_K \in (0,1)$, such that for any $\eta \in (0, \eta_K)$ there exists $\theta \in (0,1)$ such that, for all n (multiple of K) it holds

$$\mathbb{P}\left(||X_n(\sigma) - \mu||_2 \in D_\eta\right) > \eta$$

where $D_\eta := \left(D_\eta^-, D_\eta^+\right)$ with $D_\eta^- = (1-\theta) S_n \sqrt{K}$ and $D_\eta^+ = (1+\theta) S_n \sqrt{K}$.

Proof. Let's begin by making the random vector isotropic. We observe that

$$\begin{aligned} & \mathbb{P}\left(||X_n(\sigma) - \mu||_2 \notin [(1-\theta) S_n \sqrt{K}, (1+\theta) S_n \sqrt{K}]\right) \\ \leq \quad & \mathbb{P}\left(||X_n(V_{\bar{\sigma}}) - \mu||_2 \notin [(1-\theta) S_n \sqrt{K}, (1+\theta) S_n \sqrt{K}]\right). \end{aligned}$$

Now, $\frac{X_n(V_{\bar{\sigma}}) - \mu}{S_n}$ satisfies the assumptions in [14], and therefore

$$\begin{aligned} \mathbb{P}\left(\left|||X_n(V_{\bar{\sigma}}) - \mu||_2 - S_n\sqrt{K}\right| \geq S_n \sqrt{K} t\right) \quad &= \quad \mathbb{P}\left(\left| \left|\left| \frac{X_n(V_{\bar{\sigma}}) - \mu}{S_n} \right|\right|_2 - \sqrt{K} \right| \geq t\sqrt{K}\right) \\ &\leq \quad C_1 \exp\left(-C_2 t K^{1/2}\right) \end{aligned}$$

for positive constants C_1 and C_2 independent of n and all positive t.

For a given $\eta \in (0,1)$, we define

$$\theta_\eta = \frac{\ln(C_1) - \ln(1-\eta)}{C_2 \sqrt{K}}. \tag{7}$$

As long as K is large enough to have $\frac{\ln(C_1)}{C_2\sqrt{K}} < 1$, there exists a η_K which satisfies the claim

$$\eta_K = \sup\{\eta \ / \ \eta > 0 \ \& \ \theta_\eta < 1\}; \tag{8}$$

furthermore,

$$\lim_{K \mapsto \infty} \eta_K = 1.$$

□

Lemma 1 directly applies to the vector a_m simulated under the CS regime, provided that the center s_m does not move, in which case σ_m is constant. This property applies at the end of the algorithm when GS can no longer improve the current minimum, and while CS has not yet improved the minimum and updated the center.

We now prove that given $|X_n(\sigma) - \mu||_2 \in D_\eta$, realizations of $X_n(\sigma)$ will indeed fill the annulus with center μ, inner radius D_η^- and outer radius D_η^+.

proposition 2 (CS regime efficiently fills the annulus D_η)**.** *Let $\underline{\epsilon} < \overline{\epsilon}$ be both in D_η. Then, conditionally on $||X_n(\sigma) - \mu||_2 \in D_\eta$, there exists $\gamma \in (0,1)$ such that*

$$\mathbb{P}\left(\, ||X_n(\sigma) - \mu||_2 \in (\underline{\epsilon}, \overline{\epsilon}) \ \mid \ ||X_n(\sigma) - \mu||_2 \in D_\eta\right) = \gamma \tag{9}$$

holds independently of n.

Proof. For any $\beta \in (0,1)$ denote by q_β the β-quantile of the standard normal distribution, hence $P\left(\mathcal{N}(0,1) \leq q_\beta\right) = \beta$.

We choose $\underline{\delta} < \overline{\delta}$, both in $(0,1)$ through

$$\underline{\epsilon}^2 = \left(1 + \frac{\sqrt{2}q_{\underline{\delta}}}{\sqrt{K}}\right)\frac{\underline{\sigma}^2}{n}$$

and

$$\overline{\epsilon}^2 = \left(1 + \frac{\sqrt{2}q_{\overline{\delta}}}{\sqrt{K}}\right)\frac{\overline{\sigma}^2}{n}.$$

Denoting

$$A := \frac{1}{\mathbb{P}\left(||\mathbf{X}_n(\sigma) - \mu||_2 \in D_\eta\right)}$$

it holds:

$$\begin{aligned}
& \mathbb{P}\left(\, ||X_n(\sigma) - \mu||_2 \in (\underline{\epsilon}, \overline{\epsilon}) \ \mid \ ||X_n(\sigma) - \mu||_2 \in D_\eta\right) \\
= \ & A.\mathbb{P}\left(||X_n(\sigma) - \mu||_2 \in (\underline{\epsilon}, \overline{\epsilon})\right) \\
= \ & A.\left[\mathbb{P}\left(||X_n(\sigma) - \mu||_2 \geq \underline{\epsilon}\right) - \mathbb{P}\left(||X_n(\sigma) - \mu||_2 \geq \overline{\epsilon}\right)\right] \\
\geq \ & A.\left[\mathbb{P}\left(\chi^2(K) \geq nK\underline{\epsilon}^2/\underline{\sigma}^2\right) - \mathbb{P}\left(\chi^2(K) \geq nK\overline{\epsilon}^2/\overline{\sigma}^2\right)\right] \\
\simeq \ & A.\left[\mathbb{P}\left(\mathcal{N}(0,1) \geq \frac{n\sqrt{K}\underline{\epsilon}^2/\underline{\sigma}^2 - \sqrt{K}}{\sqrt{2}}\right) - \mathbb{P}\left(\mathcal{N}(0,1) \geq \frac{n\sqrt{K}\overline{\epsilon}^2/\overline{\sigma}^2 - \sqrt{K}}{\sqrt{2}}\right)\right],
\end{aligned}$$

where we used the approximation of a Chi square random variable $\chi^2(K)$ with K degrees of freedom which asserts that $\left(\chi^2(K) - K\right)/\sqrt{2K}$ is close to a standard normal random variable $\mathcal{N}(0,1)$ for large K.

Eventually,

$$\mathbb{P}\left(\,||X_n(\sigma) - \mu||_2 \in (\underline{\epsilon}, \overline{\epsilon}) \;\middle|\; ||X_n(\sigma) - \mu||_2 \in D_\eta\right) \geq A\left[\overline{\delta} - \underline{\delta}\right].$$

With $\overline{\epsilon}$ and $\underline{\epsilon}$ in D_η, we prove that $\overline{\delta}$ and $\underline{\delta}$ exist with $\underline{\delta} < \overline{\delta}$ independently of n.

From $\underline{\epsilon} > D_\eta^-$ defined in Lemma 1, it holds

$$\begin{aligned} & \left(1 + \frac{\sqrt{2}q_{\underline{\delta}}}{\sqrt{K}}\right)\frac{\underline{\sigma}^2}{n} \geq (1-\theta)^2\,\frac{\underline{\sigma}^2}{n} \\ \Leftrightarrow \quad & q_{\underline{\delta}} \geq \left[(1-\theta)^2 - 1\right]\frac{\sqrt{K}}{\sqrt{2}}. \end{aligned} \tag{10}$$

Similarly, since $\overline{\epsilon} < D_\eta^+$ defined in Lemma 1 it also holds

$$\begin{aligned} & \left(1 + \frac{\sqrt{2}q_{\overline{\delta}}}{\sqrt{K}}\right)\frac{\overline{\sigma}^2}{n} \leq (1+\theta)^2\,\frac{\underline{\sigma}^2}{n} \\ \Leftrightarrow \quad & q_{\overline{\delta}} \leq \left[(1+\theta)^2\,\frac{\underline{\sigma}^2}{\overline{\sigma}^2} - 1\right]\frac{\sqrt{K}}{\sqrt{2}}. \end{aligned} \tag{11}$$

We check that $\overline{\delta} > \underline{\delta}$, equivalently that $q_{\overline{\delta}} > q_{\underline{\delta}}$ independently upon n. Using inequalities (10) and (11) this happens to hold as long as θ is close enough to 1, meaning η small enough. Choosing first η such that θ_η unites inequalities (10) and (11) yields the corresponding $\underline{\delta}$ and $\overline{\delta}$ and indicates that realizations of $X_n(\sigma)$ fill the annulus D_η. □

To sum up, Proposition 2 tells that under the CS regime, when n is large enough (remember that it is a multiple of K), the realizations are concentrated very close to the center, thus locally improving the current minimum. Moreover, Lemma 1 indicates that CS realizations will have a non-zero probability of being in an annulus of radius $S_n\sqrt{K}$, thus sufficiently far from the center to improve the current solution.

3.3 Overall behavior

If the minimum is achieved inside $\mathcal{S}$, it can be reached by a gradient descent approach since this corresponds to optimizing over an open subset of $\mathbb{R}^K$. However, in many applications (notably our two applications in sections 4 and 5) the minimum of f will be near the boundary of $\mathcal{S}$. This observation leads us to aim at minimizing the perturbation of the center near the boundary; this amounts to first generate a center near the boundary and secondly be aware that we are near the boundary. This first step is fulfilled by the coarse regime called GS. In

parallel, a second regime CS refines the best candidate found so far. CS operates in parallel with GS without evaluating proximity to the boundary; henceforth CS avoids to answer whether the current center is close to the border of $\mathcal{S}$.

Regime GS's ability to reach a point close to the frontier is guaranteed by Proposition 1, which holds as soon as $C > 2$; we have chosen $C = 2.1$. With such a tuning, making use of the specific sampling procedure of the random variables a_m^k developed in Section 2, our algorithm does not get trapped in local minimizer of the function f. Also Condition (2) has a specific role whenever the proxy c_m of some minimizer of f is close to the boundary of $\mathcal{S}$, which is the usual case in constrained optimization. Indeed in such case assuming that for all k the standard deviation of a_m^k is proportional to $|c_m^k|$, whenever (2) fails for at least one coordinate k, the vector a_m belongs to $\mathcal{S}$ with very small probability, which allows for limiting the simulation burden of the algorithm.

In both applications below, the set $\mathcal{S}$ is defined by the level set of a function g. The starting point is a solution s_0 provided by an external algorithm assumed to be efficient but not optimal. We define $\mathcal{S}$ as the set of vectors w for which $g(w) \leq g(s_0) + \epsilon$. By its very definition the constraint defining the set $\mathcal{S}$ places the initial solution s_0 close to $\partial\mathcal{S}$. Assuming g and f are antagonistic, the optimal solution necessarily lies on $\partial\mathcal{S}$. By construction of $\mathcal{S}$, our method tolerates a slight increase in g in order to improve s_0 by decreasing f.

4 Improvement of the LASSO

To justify the relevance of our proposed method, we first give an application to the omnipresent LASSO context (cf. [25]; see also e.g. [8], for a broader embedding), where one aims to solve the minimization problem

$$\min_{\theta \in \mathbb{R}^K} \sum_{i=1}^{M} \left(y_i - \sum_{k=1}^{K} z_i^k \cdot \theta^k \right)^2 + \lambda \cdot ||\theta||_1, \tag{12}$$

with data observations y_i $(i = 1, \ldots, M)$, deterministic explanatory variables z_i^k (with $z_i^1 := 1$) and ℓ_1-norm-regularization (penalization) parameter $\lambda \geq 0$. The corresponding (say, Scikit-learn software-delivered solution of the) not necessarily unique minimizer is denoted by $\widehat{\theta}$. In the following, our goal is to find points θ with "almost-equal" performance as $\widehat{\theta}$, but whose ℓ_1-norm $||\theta||_1$ is smaller than $||\widehat{\theta}||_1$.

4.1 Rewriting the problem

As a key for our corresponding investigations, we use the well-known fact (see e.g. Chapter 9 of [24], and the more general work of [17]) that the LASSO task (12) is "equivalent" to finding the minimizer for the *basis pursuit denoising* problem (cf. [10]; see also e.g. [6]; [18]; [5]; [7]; [13]; [29]; [26]; [11]; [23]; [15]; [1])

$$\min_{\theta \in \mathcal{S}} ||\theta||_1 \tag{13}$$
$$\text{with} \quad \mathcal{S} := \Big\{ \theta \in \mathbb{R}^K : \sum_{i=1}^{M} \Big(y_i - \sum_{k=1}^{K} z_i^k \cdot \theta^k \Big)^2 \leq \varepsilon \Big\}$$

for chosen fitting-quality (tolerance) parameter $\varepsilon > 0$. Notice that — as e.g. indicated in [17] — problems of type (13) are more "difficult" to solve than problems of type (12). Moreover, the relationship between λ and ε is generally not explicit, and thus the transfer between the corresponding problem solutions is generally also not explicit. Hence, tackling (13) for the search of θ with low ℓ_1-norm (and thus "low sparsity") makes sense even if one has already solved (12). Indeed, one can start to solve the problem (13) with algorithms which use as an initial control parameter (starting point) the above-mentioned LASSO solution $\widehat{\theta}$.

As far as practical implementation is concerned, some LASSO-solvers like Scikit-learn (which we employ below) deliver as output $||\widehat{\theta}||_1$ and $r(\widehat{\theta})$ which employs — as performance score — the so-called *coefficient of determination*

$$r(\theta) := 1 - \frac{RSSQ(\theta)}{\sum_{i=1}^{M} \Big(y_i - \overline{y} \Big)^2}$$

with residual sums of squares $RSSQ(\theta) := \sum_{i=1}^{M} \Big(y_i - \sum_{k=1}^{K} z_i^k \cdot \theta^k \Big)^2$ and data mean $\overline{y} := \frac{1}{M} \cdot \sum_{i=1}^{M} y_i$. Accordingly, we rewrite

$$\mathcal{S} = \Big\{ \theta \in \mathbb{R}^K : r(\theta) \geq r_0 \Big\}, \tag{14}$$

through the correspondence $\varepsilon = (1 - r_0) \cdot \sum_{i=1}^{M} \Big(y_i - \overline{y} \Big)^2$ for some pre-chosen minimum performance score $r_0 \in (0, 1)$ which is typically close to 1. Our goal is to fix some $r_0 \approx r(\widehat{\theta})$ and to look for solutions of (13) with constraint set (14), which have smaller ℓ_1 norm than $\widehat{\theta}$.

remark 1 (Norm vs. performance in the Lasso problem)**.** *Diminishing the norm $||\theta||_1$ of a candidate can be performed by removing ϵ to each of its components; it will reduce the performance $r(\theta)$ continuously. This indicates that the optimal solution belongs to ∂S. With the previous construction of S (since $r_0 \approx r(\widehat{\theta})$), the starting point $\hat{\theta}$ is near ∂S. Therefore, $\hat{\theta}$ is nearly-optimal solution within constraint set S. This implies that the objective of our algorithm is, by design, to improve an existing solution given at start.*

4.2 Data set and implementation

We have carried out the goal formulated at the end of the previous Subsection 4.1 by employing our above-described *hybrid random concentrated optimization method*, on a concrete problem of dimension $K = 5001$ which we randomly

generated using the Python Scikit-learn code displayed in Figure 1. This and the code used for the rest of Section 4 as well as Section 5 is publicly available at `https://github.com/p052/GS_CS_algo.git`.

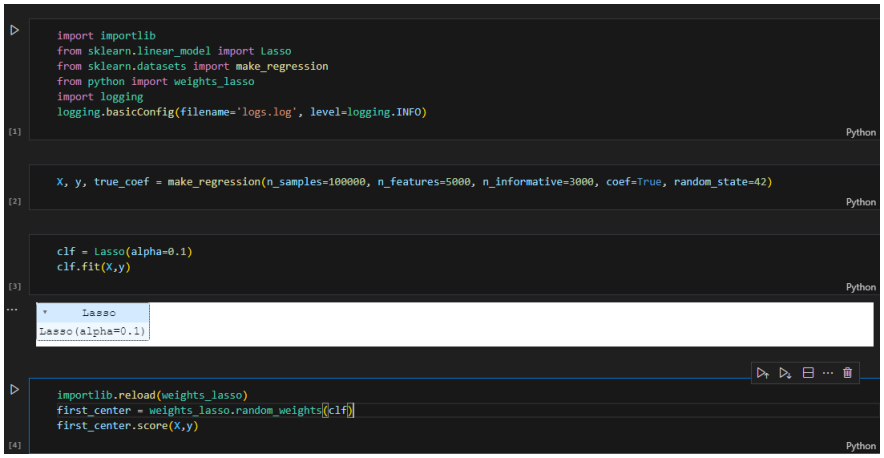

Figure 1: Generation of a problem and application of the LASSO

For the outcoming LASSO solution $\widehat{\theta}$ we obtained $||\widehat{\theta}||_1 \approx 147285.47$ as well as the performance score of slightly higher than $r_0 = 0.999$ of about $r(\widehat{\theta}) = 0.999566$.

As already indicated above, for our algorithm we take as starting point $c_0 := \widehat{\theta}$ and we aim to iteratively reduce $||c_0||_1$ while staying within $\mathcal{S}$, that is, maintaining a performance of at least $r_0 := 0.999$. For this, we apply our method by alternating between GS and CS regimes and stop arbitrarily after 10000 steps. We begin with step $m = 0$. Then, at each step m, if m is even, we use the GS regime, and if m is odd, we use the CS regime.

In the GS regime, there are no additional parameters: we generate a_m and update c_{m+1} and s_{m+1} if necessary. In the CS regime, the proved assertions in Section 3.2 guarantee concentration around the center s_m. These assertions are true for any fixed n.

Recall that n is a multiple of K, which in the current concrete application is equal to 5001. We tested two variants: the first one by taking $\frac{n}{K} = 1$, and the second with $\frac{n}{K}$ uniformly chosen between 1 and m. The latter variant ensures an asymptotic concentration since n increases with m.

However, while the provided code includes both variants, the results presented here use the first one. The fact that we observe concentration with $\frac{n}{K} = 1$ empirically demonstrates the validity of the lemmas for fixed n. The asymptotic actually is in K and not in $\frac{n}{K}$.

4.3 Results

Over 10000 steps, the reduction in the norm and the corresponding performance of successive minima s_m are shown in Figure 2.

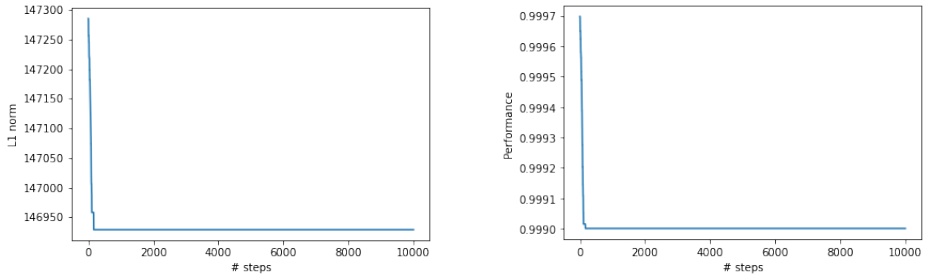

Figure 2: Evolution of the L_1 norm and of the performance over 10000 steps

As can be seen, our method works and allows for a marginal improvement of the vector $\widehat{\theta} = c_0$ provided by the LASSO. Since this is a "good" vector and thus near the boundary of $\mathcal{S}$, we expect the GS regime to be less effective and that only the CS regime will reduce the norm. This is exactly what happens, and we will discuss this further at the end of this subsection. For now, let us focus on the first 400 steps since no further improvement occurs afterward.

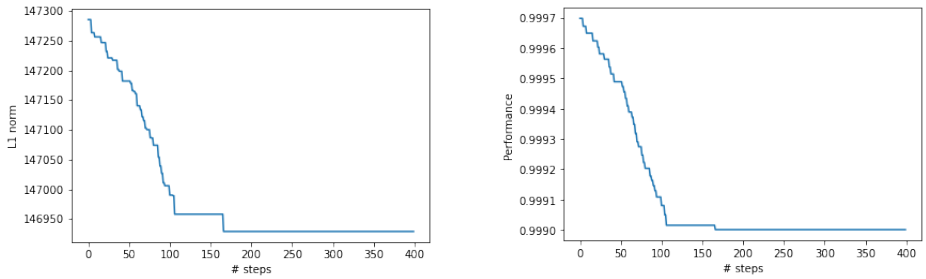

Figure 3: Evolution of the L_1 norm and of the performance over 400 steps

We see that each reduction in the norm is accompanied by a reduction in performance. We are close to the boundary, and there are no surprises such as "better performance and lower norm", which could occur in the case where (substantially) $s_m \in \mathring{\mathcal{S}}$. Here, we push the constraint to its limit. We observe a steady decrease until around step 100, then a single reduction around step 160; the vector does not change afterwards until step 10000. It is unlikely that we will do better. Altogether, the ℓ_1-norm decreases from 147285.47 to 146929.35. An improvement occurs but is minor, at 0.2%, as the solution found by the LASSO is nearly perfect. However, this quality indicates that we are near the boundary of $\mathcal{S}$, and it allows us to verify that the alternation between the regimes behaves as expected: GS never updates s_m and CS seldom does. We can also clearly

see that only the CS regime leads to a reduction, and this happens frequently at the beginning. Indeed, out of 400 steps (as well as 10000 steps), 31 steps updated the current minimum, and all of them were odd steps therefore using the CS regime: 3, 7, 15, 21, 23, 29, 35, 37, 41, 51, 53, 55, 57, 59, 63, 65, 67, 69, 71, 75, 77, 79, 85, 87, 89, 91, 93, 99, 103, 105, 165. Summing up, the current application demonstrates the effectiveness of the CS regime in marginally improving a solution near the boundary of $\mathcal{S}$.

5 Improving Pruning Efficiency

As another proof for the relevance of our proposed method, we give a second application, to the compression of neural networks.

5.1 Problem specification

The optimal acceleration of neural network inference is an urgent contemporary issue, due to the growing complexity of networks and their reuse as terminal networks. Various different acceleration methods exist, such as training multiple networks (see e.g. [21], [21]) or distillation (see e.g. [22]); for general surveys on neural network compression, the reader is e.g. referred to the comprehensive works of [9], [20], [16], [19], [27], as well as [2]; one major challenge is maintaining performance (see e.g. Table 1 in [19]).

In the following, we show how our newly developed, general hybrid random concentrated optimization method can be used for compressing some (here, classification-concerning) neural networks by pruning, to get significant weight reduction with minimal performance loss; for the latter, our procedure serves as a complementary approach and has the advantage of providing convergence guarantees (cf. Section 3).

In order to achieve these goals, we define the function f — over weights-vectors w — in connection with the pruning percentage at a given threshold, and specify the constraint set $\mathcal{S}$ in connection with the corresponding performance score. Accordingly, we take as initial input a weight-vector provided by an initial training. More detailed, for $K-$dimensional weights vectors $w \in \mathbb{R}^K$ and a desired pregiven threshold $t > 0$ (which serves as a hyper-parameter), we define $T_t(w)$ as the pruning rate of w at threshold t, that is

$$T_t(w) := \frac{1}{K} \cdot \sum_{k=1}^{K} \mathbf{1}_{|w^k|<t} \; ; \tag{15}$$

consistently, $T_0(w) := 0$ means *no pruning*. Similarly, for weights vectors $w \in \mathbb{R}^K$ we define $P_t(w)$ as the *performance score — in terms of percentage of correctly classified instances — after pruning at* t, where all weights below this threshold t are set to 0; consistently, $P_0(w)$ quantifies the performance score when no pruning is done. With this at hand, for fixed pre-chosen threshold $t > 0$ and fixed pre-chosen minimal performance score $\eta \in (0, 1)$ (where η is

typically larger than 0.85) our objective is then to find the maximizer(s) of the optimization problem

$$\begin{aligned} &\max_{w \in \mathcal{S}} T_t(w) \\ &\text{with} \quad \mathcal{S} := \mathcal{S}_{\eta,t} := \left\{ w \in \mathbb{R}^K \,:\, P_t(w) > \eta \right\}, \end{aligned} \tag{16}$$

i.e. find the (not necessarily unique) weights vector which delivers the highest pruning rate (at t) while maintaining a performance score of higher than η. Clearly, by choosing $f(w) := -T_t(w)$ this can be converted into our context of *minimization* of f on the constraint set $\mathcal{S}$.

5.2 Data set and implementation

We have carried out the goal formulated at the end of the previous Subsection 5.1 by employing our above-described *hybrid random concentrated optimization method*, on a concrete neural-network-based classification problem for Fashion MNIST. The corresponding code used for the rest of the current Section 5 is publicly available at
`https://github.com/p052/GS_CS_algo.git`.

As a *first preliminary step*, on Fashion MNIST we trained — in a simple and classic way — a network with an employed network architecture described in Figure 4.

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 28, 28, 64)	320
max_pooling2d (MaxPooling2D)	(None, 14, 14, 64)	0
dropout (Dropout)	(None, 14, 14, 64)	0
conv2d_1 (Conv2D)	(None, 14, 14, 32)	8,224
max_pooling2d_1 (MaxPooling2D)	(None, 7, 7, 32)	0
dropout_1 (Dropout)	(None, 7, 7, 32)	0
flatten (Flatten)	(None, 1568)	0
dense (Dense)	(None, 256)	401,664
dropout_2 (Dropout)	(None, 256)	0
dense_1 (Dense)	(None, 10)	2,570

Total params: 412,778 (1.57 MB)
Trainable params: 412,778 (1.57 MB)
Non-trainable params: 0 (0.00 B)

Figure 4: Network architecture

This network is (as common in image classification) divided into two parts: the first one being *convolutional*, and the second one being *dense*. As can be seen from Figure 4, the first part has (relatively seen) only a *few* parameters/weights,

namely 8544 (concatenated into a vector w_{con}); on the other hand, the second part contains the vast majority of the parameters/weights, namely 404234 (concatenated into a vector w). All activation functions, *except* for the last one — which is a softmax leading to the probability of each of the ten classes — are ReLU.

After training, we obtained as output a concrete vector $\widehat{w}_{tot} := (\widehat{w}_{con}, \widehat{w})$ of total 412778 weights which lead to a performance score of $P_0(\widehat{w}_{tot}) = 0.91$ on the test sample, i.e. 91% of the images were correctly classified. We saved these weights $\widehat{w}_{tot}$, and froze its convolutional part $\widehat{w}_{con}$ for the rest of our analyses. With its dense part $\widehat{w}$, in contrast, we employed $s_0 := c_0 := \widehat{w}$ as starting point of our above-described new algorithm to solve the minimization problem (with dimension $K := 404234$)

$$\begin{array}{l} \min\limits_{w \in \mathcal{S}} \left(-T_t(w)\right) \\ \text{with} \quad \mathcal{S} := \mathcal{S}_{\eta_0,t} := \left\{ w \in \mathbb{R}^K \,:\, P_t((\widehat{w}_{con}, w)) > \eta \right\}; \end{array} \tag{17}$$

notice that the required condition $s_0 \in \mathcal{S}$ is satisfied here for reasonable choices of $t > 0$ and $\eta \in (0, 1)$. For instance, for very small threshold t one gets $P_t((\widehat{w}_{con}, \widehat{w})) \approx P_0((\widehat{w}_{con}, \widehat{w}))$, so that if η is chosen to be close to and lower than $P_0((\widehat{w}_{con}, \widehat{w}))$, then $P_t((\widehat{w}_{con}, s_0)) = P_t((\widehat{w}_{con}, \widehat{w})) > \eta$. Accordingly, in the current concrete application we employed $t = 0.01$ and $\eta = 0.87$.

After this initialization, as a *second step* we employed our hybrid random concentrated optimization method (cf. Section 2) for solving the minimization problem (17), with two different regime alternations (cf. Subsections 5.3.1 and 5.3.2). As a side effect, one can see that our method can be comfortably used even if the constraint set $\mathcal{S}$ is given only "implicitly" and thus its detailed shape may also be known only "implicitly" (after all, the crucial performance scores can be straightforwardly computed after each simulation step).

remark 2 (Pruning vs. performance)**.** *Let us note here that (similarly to Remark 1) the pruning rate can be improved as long as performance is strictly greater than the limit defining $\mathcal{S}$.*

remark 3. *[non-convexity of $\mathcal{S}$] The dependency of weights between layers is significant, leading to the constraint set $\mathcal{S}$ being non-convex: if two weights vectors w_1 and w_2 are both in $\mathcal{S}$, then it generally does not follow that their average $\frac{w_1+w_2}{2}$ is also in $\mathcal{S}$. Indeed, the average between two weights vectors that each provide good performance score is expected to lead to, at best, average performance score. For instance, we obtained by training twice (empirically but reproducibly, see the above-mentioned code link) two different weights vectors with performance scores $P_0((\widehat{w}_{con}, w_1)) \approx 0.90 > 0.87 = \eta$ and $P_0((\widehat{w}_{con}, w_2)) \approx 0.90 > 0.87 = \eta$ such that $P_0((\widehat{w}_{con}, \frac{w_1+w_2}{2})) \approx 0.70 < 0.87 = \eta$. Additionally, we can show that T_t is not concave ($f = -T_t$ not convex). If w_1 and w_2 are*

equal except on component 1 *with* $w_1^1 = 2t + 1$ *and* $w_2^1 = 0$ *we would have:*

$$T_t(\frac{w_1 + w_2}{2}) = T_t(w_1) < \frac{T_t(w_1) + T_t(w_2)}{2}$$

since $T_t(w_2) = T_t(w_1) + \frac{1}{K}$.

5.3 Results

5.3.1 GS followed by CS

As indicated in the general presentation of Subsection 2.4, as a first regime-alternation variant of our method we iteratively generated new weights (solution candidates) according to GS and then — after the *incumbent candidate minimum* was considered to be stable because the step size became too large — we switched to CS iterations. In this first algorithm, the application of the GS regime leads to the behaviour displayed in Figure 5. One can see that in the left-hand display the pruning rate increases sharply, rising from 10% to about 30%. The performance score in the right-hand display, however, drops quickly from 91% to 88% which is very close to the pregiven minimum threshold $\eta = 87\%$; this indicates that the outcoming incumbent candidate solution is near the boundary $\partial\mathcal{S}$ of $\mathcal{S}$. Unfortunately, the GS regime soon reaches its limit as the last 58 iterations result in no increase in $T_t(\cdot)$.

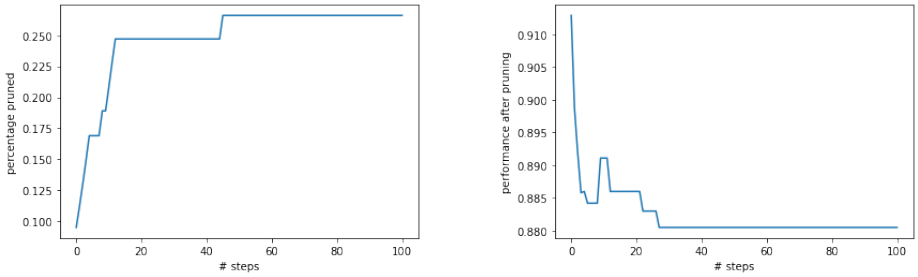

Figure 5: for GS only: Evolution of $T_t(\cdot)$ (on the left) and $P_t((\widehat{w}_{con}, \cdot))$ (on the right) over the first 100 iterations

Being stably near the boundary $\partial\mathcal{S}$ after $m = 100$ GS iterations, we then switched regimes in the second phase by applying *only* CS, the center being the last incumbent candidate solution found by GS. The corresponding overall evolution (from step $m = 0$ to step $m = 1000$) of the pruning rate and the performance score is shown in the Figure 6: the first phase represented by the GS regime (cf. also Figure 5) leads to a weights vector close to the boundary, whereas the second phase, in the CS regime, allows for a much slower but significant increase in the pruning rate $T_t(\cdot)$, reaching about 40% pruning while maintaining substantially the same performance score. A clear difference in pruning-rate update is observed between the two regimes: GS makes substantial upward steps, while CS increases slowly but frequently.

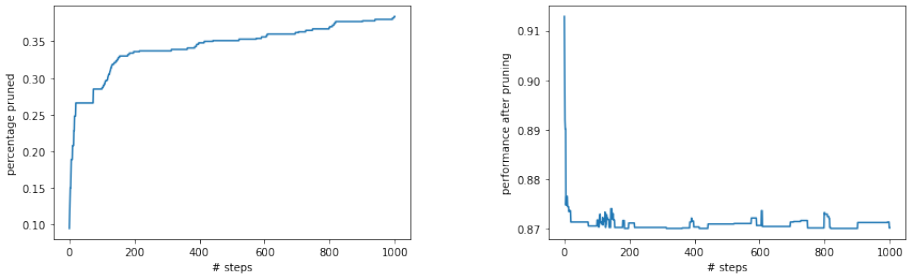

Figure 6: Evolution of $T_t(\cdot)$ (on the left) and $P_t((\widehat{w}_{con}, \cdot))$ (on the right) over 1000 iterations, with GS for steps $0 - 100$ and CS for steps $101 - 1000$.

5.3.2 Permanent alternation between GS and CS

On the same dataset, the same hyperparameter-settings $t = 0.01$, $\eta = 0.87$ and the same starting point $s_0 := c_0 := \widehat{w}$ as in Subsection 5.3.1, we also applied the *second* regime-change variant of our method (cf. Subsection 2.4): iteratively generating new weights (solution candidates) by *permanently* alternating between the regimes GS (even steps) and CS (odd steps). This procedure offers several advantages: Proposition 1 applies to the GS regime; Lemma 1 and Proposition 2 apply to the CS regime; and both regimes continue in parallel throughout the process, eliminating one "parameter": the timing of the regime-switching step.

Figure 7 shows the corresponding overall evolution (from step $m = 0$ to step $m = 1000$) of the pruning rate (on the left-hand side) and the performance score (on the right-hand side), in case of the above-described permanent regime-alternation. Notice that Figure 7 shows a similar behaviour to Figure 6: the evolution of the pruning rate and the performance score is mainly essentially comparable. Ultimately, the "first-GS-then-CS" regime-switching approach of Subsubsection 5.3.1 found a network where 38.4% of the dense-part-concerning weights can be pruned (recall that the convolutional-part-concerning weights remained unchanged/fixed), achieving an overall performance score of 87%, while the permanent-alternation approach of the current subsection resulted in a network where 35.7% of the dense-part-concerning weights can be pruned, leading to an identical overall performance score of 87%.

As a consequence, the above concrete results support the permanently alternating approach presented in Subsection 2.4 and the current subsection, since the performance score is close to the other regime-change variant (recall that the method is based on random generations) and the permanent alternation frees us from a technical parameter and a complex question: identifying proximity to the boundary.

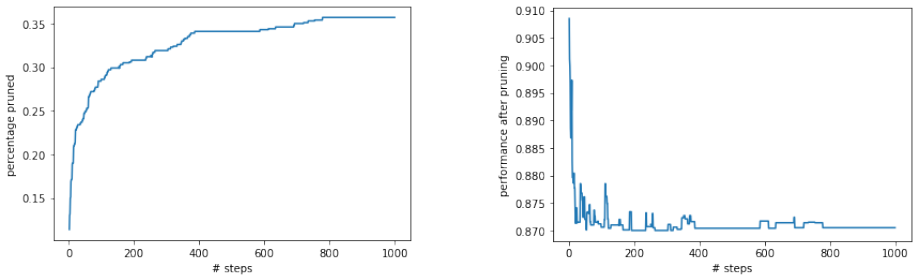

Figure 7: Evolution of $T_t(\cdot)$ (on the left) and $P_t((\widehat{w}_{con}, \cdot))$ (on the right) over 1000 iterations, in the case of permanent alternation

6 Conclusion

In this paper, we develop a new random method to minimize deterministic continuous functions over constraint sets $\mathcal{S}$ in high-dimensional space $\mathbb{R}^K$, where no convexity assumptions are required. Our approach employs both a Global Search (GS) regime as well as a Concentrated Search (CS) regime, with different possible strategies to switch between them. Basically, the role of GS is "vast" candidate-solution-exploration of the entire space whereas the role of CS amounts to a "locally refined" candidate-solution-search which effectively takes into account high-dimensional concentration properties of the underlying random simulations (which, amongst other things, turns out to be particularly useful in finding improved candidate solutions near the boundary of $\mathcal{S}$); these concentration properties are rigorously stated and proved. In parallel, we also show that GS reaches any point in $\mathcal{S}$ in finite time (in case that $\mathcal{S}$ satisfies the "light" assumption of path-connectedness).

As demonstrations of the effectiveness of our new method, we work out two concrete applications: the first one optimizes the reduction of the "sparsity-quantifying" ℓ_1-norm of the LASSO minimizer while keeping "almost-equal" performance quality, whereas the second one deals with the compression of a MNIST-fashion-classification concerning neural network through optimization of the corresponding pruning percentage at a given threshold while maintaining (at least) a required substantially high performance-score. In the course of this, we additionally emphasize the advantage that our approach (in contrast to some other methods) provides the above-mentioned convergence guarantees.

In future work, it would be interesting to extend our approach to other neural network architectures and to more complex application fields, particularly to computer vision or natural language processing tasks. Moreover, the additional integration of other compression techniques such as weight quantization or sparsified neural networks could further enhance efficiency gains. All these issues seem to principally tractable.

Summing up things, our new method provides a framework for generic high-

dimensional constrained optimization even under light assumptions on the objective function f and the constraint set $\mathcal{S}$.

Acknowledgements

W. Stummer is grateful to the Sorbonne Université Paris for its multiple partial financial support and especially to the LPSM for its multiple great hospitality. M. Broniatowski thanks very much the FAU Erlangen-Nürnberg for its partial financial support and hospitality.

References

[1] Bertsimas, D., Johnson, N.A.G.: Compressed sensing: a discrete optimization approach. Machine Learning **113**, 6725–6764 (2024)

[2] Bhalgaonkar, S., Munot, M., anuse, A.: Model compression of deep neural network architectures for visual pattern recognition: Current status and future directions. Computers and Electrical Engineering **116**, 109180 (2024)

[3] Broniatowski, M., Stummer, W.: A precise bare simulation approach to the minimization of some distances. I. Foundations. IEEE Transactions on Information Theory **69**(5), 3062–3120 (May 2023)

[4] Broniatowski, M., Stummer, W.: A precise bare simulation approach to the minimization of some distances. II. Further foundations. arxiv preprint arXiv:2402.08478 (2024)

[5] Candès, E.J.: The restricted isometry property and its implications for compressed sensing. Comptes Rendus de l'Academie des Sciences Paris, Series I **346**, 589–592 (2008)

[6] Candès, E.J., Romberg, J.K., Tao, T.: Stable signal recovery from incomplete and inaccurate measurements. Communications on Pure and Applied Mathematics **LIX**, 1207–1223 (2006)

[7] Candès, E.J., Wakin, M.B., Boyd, S.P.: Enhancing sparsity by reweighted ℓ_1 minimization. Journal of Fourier Analysis and Applications **14**, 877–905 (2008)

[8] Chen, T., Chen, X., Chen, W., Wang, Z.: Learning to optimize: a primer and a benchmark. Journal of Machine Learning Research **23**(189), 1–59 (2022)

[9] Choudhary, T., Mishra, V., Goswami, A., Sarangapani, J.: A comprehensive survey on model compression and acceleration. Artificial Intelligence Review **53**, 5113–5155 (2020)

[10] Donoho, D.L., Elad, M., Temlyakov, V.N.: Stable recovery of sparse overcomplete representations in the presence of noise. IEEE Transactions on Information Theory **52**(1), 6–18 (January 2006)

[11] Edgar, M.P., Gibson, G.M., Padgett, M.J.: Principles and prospects for single-pixel imaging. Nature Photonics **13**, 13–20 (2019)

[12] Franzin, A., Stützle, T.: Revisiting simulated annealing: A component-based analysis. Computers & Operations Research **104**, 191–206 (2019)

[13] Goldstein, T., Osher, S.: The split Bregman method for L1-regularized problems. SIAM Journal on Imaging Sciences **2**(2), 323–343 (2009)

[14] Guédon, O., Milman, E.: Interpolating thin-shell and sharp large-deviation estimates for isotropic log-concave measures. Geometric And Functional Analysis **21**(5), 1043–1068 (2011)

[15] Gupta, S., Singh, A., Sharma, A., Tripathy, R.K.: Exploiting tunable Q-factor wavelet transform domain sparsity to denoise wrist PPG signals. IEEE Transactions on Instrumentation and Measurement **72**, 4008012 (2023)

[16] Hoefler, T., Alistarh, D., Ben-Nun, T., Dryden, N., Peste, A.: Sparsity in Deep Learning: Pruning and growth for efficient inference and training in neural networks. Journal of Machine Learning Research **22**(241), 1–124 (2021)

[17] Lorenz, D., Worliczek, N.: Necessary conditions for variational regularization. Inverse Problems **29**, 075016 (2013)

[18] Lustig, M., Donoho, D., Pauly, J.M.: Sparse MRI: the application of compressed sensing for rapid MR imaging. Magnetic Resonance in Medicine **58**, 1182–1195 (2007)

[19] Marinó, G.C., Petrini, A., Malchiodi, D., Frasca, M.: Deep neural networks compression: A comparative survey and choice recommendations. Neurocomputing **520**, 152–170 (2023)

[20] O' Neill, J.: An survey of neural network compression. arxiv preprint arXiv:2006.03669v2 (2020)

[21] OpenAI: GPT-4 Technical Report. arxiv preprint arXiv:2303.08774v6 (2024)

[22] Sanh, V., Debut, L., Chaumond, J., Wolf, T.: DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arxiv preprint arXiv:1910.01108v4 (2020)

[23] Tardivel, P.J., Bogdan, M.: On the sign recovery by least absolute shrinkage and selection operator, thresholded least absolute shrinkage and selection operator, and thresholded basis pursuit denoising. Scandinavian Journal of Statistics **49**, 1636–1668 (2022)

[24] Theodoridis, S.: Machine Learning: A Bayesian and Optimization Perspective. Academic Press, London UK, 2nd edn. (2020)

[25] Tibshirani, R.: Regression shrinkage and selection via the lasso. Journal of the Royal Statistical Society Series B **58**(1), 267–288 (1996)

[26] Tran-Dinh, Q., Kyrillidis, A., Cevher, V.: Composite self-concordant minimization. Journal of Machine Learning Research **16**(12), 347–416 (2015)

[27] Vysogorets, A., Kempe, J.: Connectivity matters: neural network pruning through the less of effective sparsity. Journal of Machine Learning Research **24**(99), 1–23 (2023)

[28] Wegner, S.A.: Mathematical Introduction to Data Science. Springer-Verlag, Berlin (2024)

[29] Zhang, Y., Peterson, B.S., Ji, G., Dong, Z.: Energy preserved sampling for compressed sensing MRI. Computational and Mathematical Methethods in Medicine **2014**, 546814 (2014)