Greedy Brownian - Continuous Ant Colony Optimization (GB-CACO)

[1]Aldo Taranto
ORCID: 0000-0001-6763-4997
Aldo.Taranto@anu.edu.au
School of Computing
Australian National University
Canberra, 2601, ACT, Australia

[2]Ron Addie
ORCID: 0000-0002-6664-8462
Ron.Addie@unisq.edu.au
School of Mathematics, Physics and Computing
University of Southern Queensland
Toowoomba, 4350, QLD, Australia

November 14, 2025

***Abstract*—This work extends the Brownian-bridge mechanism in BB-CACO by replacing it with fractional Brownian motion (fBm), removing the return-to-nest constraint and freeing those steps for broader path exploration. We develop two non-gradient descent (NGD) variants—fBm-CACO and its greedy form GfBm-CACO—and show through extensive tests on highly non-convex landscapes that GfBm-CACO reliably reaches better minima than SGD, RMSProp, and Adam. While gradient methods converge quickly but often stall in local minima on multimodal surfaces, our NGD methods use more erratic but wider exploration to escape such traps and frequently locate global optima. We adopt the greedy variant as the primary algorithm, naming it GB-CACO, and outline its utility for Loss Landscape Surface (LLS) optimisation.**

***Index Terms*—Fractional Brownian motion (fBm), Machine learning (ML), Algorithms, Benchmarking.**

I. Introduction

Modern ML optimization is dominated by gradient-based methods such as GD, SGD, and Adam, as implemented in TensorFlow and scikit-learn. However, non-gradient descent (NGD) frameworks can outperform these methods under specific loss-landscape (LLS) conditions. NGD algorithms optimize by directly sampling and evaluating the LLS without computing derivatives, enabling effective convergence when gradients are costly, undefined, unstable, or uninformative.

Evidence shows that gradient-free methods can outperform gradient-based ones—both in accuracy and efficiency—particularly on: (1) high-dimensional spaces where gradient computation is expensive, (2) non-differentiable or discontinuous regions, (3) highly non-convex landscapes with many local minima, (4) dynamic or time-varying surfaces, and (5) plateau regions with near-zero gradients. NGD methods are especially advantageous on fractal or irregular LLSs where gradients provide poor guidance.

Although modern frameworks accelerate gradient computation via batching, GPUs/TPUs, and autodifferentiation, backpropagation still carries significant computational overhead. Eliminating the gradient entirely—optimizing solely by LLS height evaluations —offers a radically simpler alternative. This raises two questions: can such a "naive" strategy outperform SGD, Adam, and RMSProp, and does its advantage scale to high-dimensional spaces?

A basic NGD approach uses m samples of n-dimensional fractional Brownian motion (fBm) paths of length T, selecting the terminal point with lowest loss as the "best path." A more effective strategy generates fBm paths one time step at a time, selecting the best next step at each iteration. This greedy variant —GfBm-CACO —typically improves exploration and convergence over the full-path fBm-CACO.

The computational framework further enhances efficiency by decomposing loss functions into simpler components, allowing alternative evaluation paths that bypass expensive calculations without altering mathematical equivalence. This yields substantial gains where loss evaluation, not gradients, becomes the principal bottleneck. To illustrate these algorithmic approaches for fBm-CACO and GfBm-CACO, and comparing them to gradient based algorithms like SGD, we show Figure 1.

Figure 1 illustrates how fBm-CACO explores multiple stochastic paths and how the greedy approach iteratively selects optimal steps. Conceptually, SGD is easily trapped in local minima, fBm-CACO can approach the global minimum via broad exploration, and GfBm-CACO most effectively tracks the global optimum through stepwise guided sampling.

II. Literature Review

Optimization algorithms that don't use gradients are typically called "gradient-free" or "derivative-free" optimization algorithms [1]. They are particularly useful when the objective function is non-differentiable, when gradients are expensive to compute, or when the function is a "black box" where we only have access to function evaluations [2], as is the case with LLS. These algorithms fall into several categories,

- Evolutionary algorithms (Genetic algorithms [3], Differential evolution [4], CMA-ES [5]).
- Swarm intelligence methods (Particle Swarm Optimization [6], Ant Colony Optimization [7]).

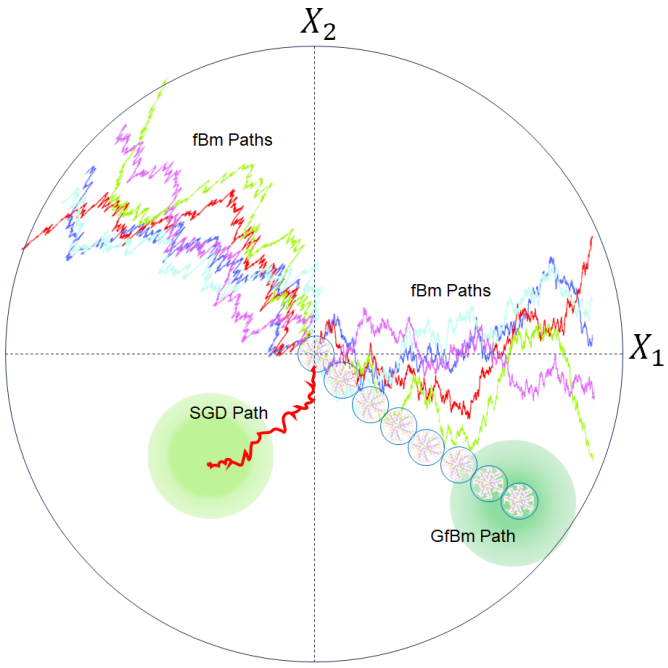


Fig. 1: Non-Gradient Descent Concepts

fBm-CACO: Numerous m instances of n-dimensional fBm paths are generated. In this example, 1-dimensional fBm paths were generated to illustrate the concept in its simplest framework, where there is one local minimum and one global minimum.
SGD is attracted to and is trapped in the local minimum due to its gradient seeking process.
fBm-CACO: These paths' terminal position at time $t = \mathcal{T}$ is the outer circle. Here, m paths were generated and the "best candidate path" has out performed SGD in getting closer to, but not finding the global minimum.
GfBm-CACO: In a similar manner, also not to scale, this approach just takes 1 step in all m directions. Each step forms a smaller circle at the terminal location, and the path with the lowest height forms the start of the next step. In this sytlised example, GfBm has found the global minimum.

- Direct search methods [8] (Nelder-Mead [9], Powell's method [10]).
- Model-based methods (Bayesian optimization [11]).
- Random search methods (Simulated annealing [12]).
- Zero-order methods (methods that approximate gradients using function evaluations [13]).

The "Curse of Dimensionality" in optimization theory, though it is more of a collection of mathematical principles than a single formal theorem, affects both non-gradient as well as gradient based algorithms [14]. However, this does not mean that the impact of this affects both approaches equally. As this research will demonstrate, the greater the size of the multi-dimensional surface, then the more non-convex that it becomes [15]. This adversely affects gradient based algorithms moreso than non-gradient based algorithms.

The specific aspect relating to increasing local minima, leading to less convexity, in higher dimensions is sometimes described by the "No Free Lunch Theorem" [16] and various results from statistical learning theory [17]. Furthermore, there is the so-called the "Theory of Rugged Landscapes" [18] or the "Complexity of Optimization Landscapes" [19]. Some key relevant mathematical principles include,

- As dimensionality increases, the volume of the space increases exponentially, creating more "room" for local minima to exist [20].
- In higher dimensions, random functions tend to have exponentially more stationary points (including local minima) [21].
- By considering a sphere inscribed within a cube and generalizing this construction to an n-dimensional hypersphere within a hypercube, distinct geometric patterns emerge. As dimensionality increases, the ratio of the hypersphere's volume to that of its enclosing hypercube approaches zero, indicating that in high-dimensional spaces, most of the volume is concentrated near the "corners" of the hypercube [22].

These principles help explain why optimization in high dimensions typically encounters many more local minima, making global optimization increasingly difficult as dimensionality grows. This also provides research that supports the potential feasibility of non-gradient optimisation algorithms over traditional gradient based approaches such as SGD.

III. Methodology

The methodology can be formulated by providing the following update functions for fBm-CACO and GfBm-CACO. These are contrasted with our other NGD algorithm, BB-CACO [23].

A. fBm-CACO

fBm CACO update formula,

$$x_{t+1} = x_t + \alpha \cdot (B_t^H - B_{t-1}^H) \tag{1}$$

where,
- x_t: Current parameter/state.
- α: Step size parameter.
- $H \in (0, 1)$: Hurst parameter.
- $H = 0.5$: Standard Brownian motion.
- $H < 0.5$: Anti-persistent behavior.
- $H > 0.5$: Persistent/trending behavior.
- B_t: fBm process.
- $(B_t^H - B_{t-1}^H)$: Incremental change in fBm.

Characteristics:
- Generalizes standard Brownian motion.
- Captures long-range dependence.
- Allows for tunable memory and correlation.
- Handles non-linear and non-differentiable landscapes more robustly.
- Interpolates between deterministic gradient descent and pure stochastic exploration.

Interpretation:
- The update is scaled by the Hurst parameter H.
- When $H = 0.5$, reduces to standard Brownian motion.
- H controls the "memory" and persistence of the optimization trajectory.

B. Greedy fBm-CACO

Greedy fBm CACO update formula,

$$x_{t+1} = x_t +_{\Delta B_i^H} \left(f(x_t + \alpha \Delta B_i^H) \right) \tag{2}$$

where,
- $\{\Delta B_1^H, \Delta B_2^H, \cdots, \Delta B_n^H\} = n$ random fBM increments Δ.
- α: Step size parameter.
- $H \in (0, 1)$: Hurst parameter.
- B_t: fBm process.

IV. Implementation

To help implement and benchmark optimization algorithms, mathematical surfaces are often used to mimick the multi-dimensional complexity of LLSs [24]. In this paper, we will examine two classes of complex non-convex surfaces;

1) **Classical Non-convex Surfaces**
   1. Rastragin [25], 2. Ackley [26], 3. Rosenbrock [27], 4. Sphere/Convex [28], 5. Schwefel [29], 6. Griewank [30].
2) **Highly Non-convex Surfaces**
   1. Alpine N.2 [31], 2. Deceptive [32], 3. Michalewicz [33], 4. Lévy [34], 5. Step Rastrigin [35], 6. Zakharov [35].

These surfaces were implemented in the Python programming language as follows.

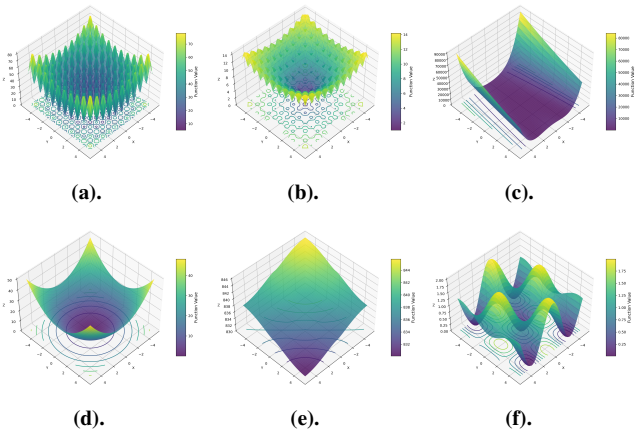
A. Classical Non-convex Surfaces

```
def rastrigin(x, y):
A = 10
return A * 2 + (x**2 - A * np.cos(2 * np.pi * x)) + (y**2 -
A * np.cos(2 * np.pi * y))
-----------------------------------------------------

def ackley(x, y):
a = 20
b = 0.2
c = 2 * np.pi
return -a * np.exp(-b * np.sqrt(0.5 * (x**2 + y**2)))
- np.exp(0.5 * (np.cos(c * x) + np.cos(c * y))) + a +
np.exp(1)
-----------------------------------------------------

def rosenbrock(x, y):
return (1 - x)**2 + 100 * (y - x**2)**2
-----------------------------------------------------

def sphere(x, y):
return x**2 + y**2
-----------------------------------------------------

def schwefel(x, y):
return 418.9829 * 2 - (x * np.sin(np.sqrt(abs(x))) + y *
np.sin(np.sqrt(abs(y))))
-----------------------------------------------------

def griewank(x, y):
return (x**2 / 4000 + y**2 / 4000) - (np.cos(x) * np.cos(y /
np.sqrt(2))) + 1
-----------------------------------------------------
```

B. Highly Non-convex Surfaces

```
def alpine_n2(x):
"""Alpine N.2 function - highly multimodal with many sharp
peaks
Product of sines creates exponentially many local optima
"""
return -np.prod(np.sqrt(np.abs(x)) * np.sin(x))
-----------------------------------------------------

def deceptive(x):
"""Custom deceptive function - misleading gradients
Designed to trap gradient-based methods
"""
Create deceptive landscape with false gradient information
n = len(x)
g = 0
for i in range(n):
if np.abs(x[i]) < 0.8:
g += (x[i] + 0.5)**2  Local attractor away from optimum
else:
g += 0.1 * x[i]**2  True basin near optimum

Add high-frequency oscillations
g += 0.5 * np.sum(np.sin(20 * np.pi * x)**2)
return g
-----------------------------------------------------

def michalewicz(x, m=10):
"""Michalewicz function - steep ridges and drops
Very challenging for gradient descent, many local optima
"""
i = np.arange(1, len(x) + 1)
return -np.sum(np.sin(x) * np.sin(i * x**2 / np.pi)**(2*m))
-----------------------------------------------------

def levy(x):
"""Levy function - highly multimodal with global structure
Gradient-based methods struggle due to many local minima
"""
w = 1 + (x - 1) / 4
term1 = np.sin(np.pi * w[0])**2
term3 = (w[-1] - 1)**2 * (1 + np.sin(2 * np.pi * w[-1])**2)
wi = w[:-1]
sum_term = np.sum((wi - 1)**2 * (1 + 10 * np.sin(np.pi * wi
+ 1)**2))
return term1 + sum_term + term3
-----------------------------------------------------

def step_rastrigin(x):
"""Step Rastrigin - discontinuous with flat regions
Gradients vanish in flat regions, making gradient descent
fail
"""
x_stepped = np.floor(2 * x) / 2  Create steps
return 10 * len(x) + np.sum(x_stepped**2 - 10 * np.cos(2 *
np.pi * x_stepped))
-----------------------------------------------------

def zakharov(x):
"""Zakharov function - plate-shaped with strong coupling
between dimensions
Sensitive to coordinate transformations
"""
i = np.arange(1, len(x) + 1)
sum1 = np.sum(x**2)
sum2 = np.sum(0.5 * i * x)
return sum1 + sum2**2 + sum2**4
-----------------------------------------------------
```

These n-dimensional functions have been implemented in python to plot only 3-dimensional slices. For example, the high-dimensional LLS used in ChatGPT-3 has about 175 billion parameters, creating at least a 175 billion-dimensional surface embedded in a 175 billion-dimensional space, with loss function $\mathcal{L} : \mathbb{R}^{150\text{B}} \to \mathbb{R}$. For these proxies for LLS, taking any two dimensions together with the loss $\mathcal{L}$ 'height' dimension can create countless 3-D slices, and some of these surfaces have been plotted in Figure 2.



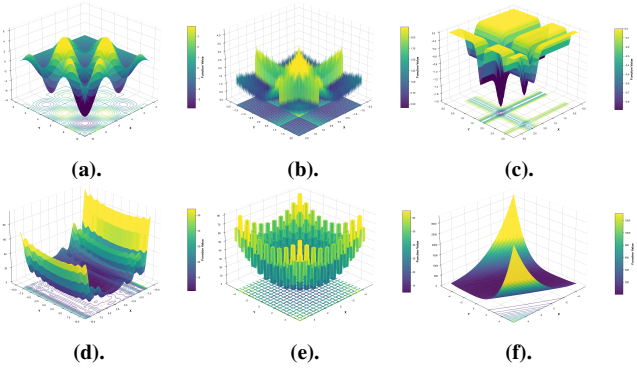
(a). (b). (c).

(d). (e). (f).

Fig. 2: Classical LLS Algorithm Benchmark Surfaces

The most popular LLS algorithm benchmark surfaces have the following key characteristics,
(a) Rastrigin Surface: Characterized by its multimodal landscape with numerous evenly distributed local minima, making it challenging for optimization algorithms.
(b) Ackley Surface: Features a nearly flat outer region with a steep central basin, requiring careful navigation through its sharp transitions.
(c) Rosenbrock Surface: A narrow, curved valley with a single global minimum, making convergence dependent on precise movement along the valley.
(d) Sphere/Convex Surface: The simplest, convex shape with a single global minimum at the center, ideal for testing basic optimization performance.
(e) Schwefel Surface: Highly multimodal with steep ridges and deep troughs, requiring algorithms to avoid deceptive local minima.
(f) Griewank Surface: Combines a complex landscape of small ripples with a single global minimum, blending linear and non-linear features.

Figure 2 shows the most popular LLS benchmark surfaces. The more complex benchmark surfaces are shown in Figure 3.



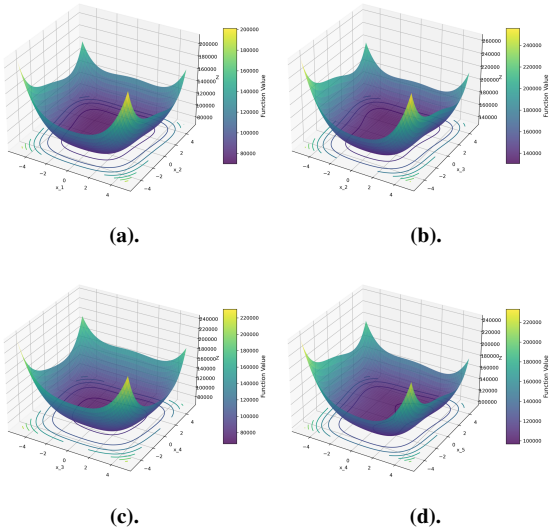
(a). (b). (c).

(d). (e). (f).

Fig. 3: Complex Non-convex LLS Algorithm Benchmark Surfaces

(a). Alpine N.2 Surface: a multimodal optimisation landscape defined by steep sinusoidal curves that create many regularly spaced local minima.
(b). Deceptive Surface: a landscape engineered so that local gradients intentionally lead search algorithms away from the global optimum.
(c). Michalewicz Surface: a highly multimodal surface with sharp, narrow valleys that make locating the global optimum extremely difficult.
(d). Levy Surface: a smooth but complex landscape featuring broad plateaus and deep basins with a global minimum near $(1, 1, \cdots)$.
(e). Step Rastrigin Surface: a discretised, staircase-like variant of the Rastrigin function that increases ruggedness and local traps.
(f). Zakharov Surface: a smooth, bowl-shaped benchmark combining linear and quadratic terms with a gently curved valley towards the optimum.

Figure 3 shows that these surfaces are more complex and non-convex. Unless the LLS is fully symetrical, as in $f(x_1, \cdots, x_n) = x_1^2 + \cdots + x_n^2$, then it does not matter which two x_i, x_j for $i \neq j$ plus a 'height' loss dimension, or even three x_i, x_j, x_k for $i \neq j \neq k$ dimensions are chosen to form 3-dimensional slices. Most of the time, there will be subtle differences between slices, as shown as in Figure 4.



(a). (b).

(c). (d).

Fig. 4: Four Different Yet Similar Rosenbrock Surface 3D Slices

Rosenbrock surface,
(a). Selected dimensions: X_1, X_2.
(b). Selected dimensions: X_2, X_3.
(c). Selected dimensions: X_3, X_4.
(d). Selected dimensions: X_4, X_5.

Figure 4 shows four different 3-dimensional slices of the Rosenbrock surface, in which each slice is very similar to the previous slice. A more dramatic example of a high-dimensional surface with larger differences between various 3-dimensional slices is shown in Figure **??** in the Appendices.

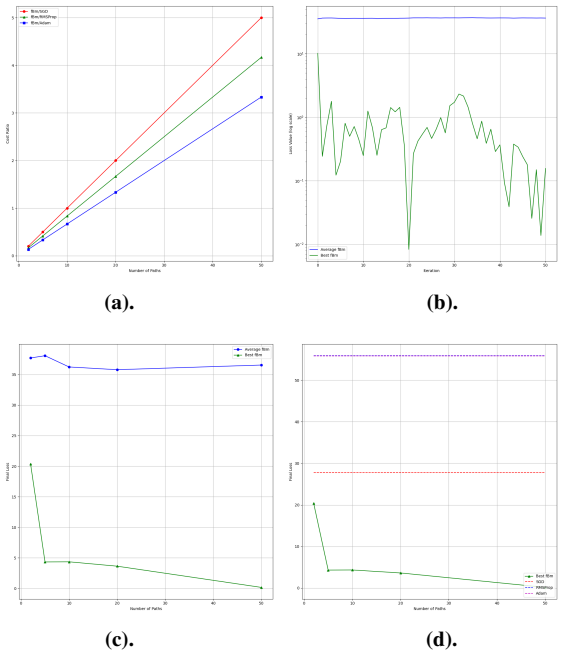
V. Results

A. Experiment 1 - Rastragin Surface Results

The following results compare fBm-CACO against GfBm-CACO, and also in relation to various gradient based algorithms, namely SGD, RMSProp and Adam. First, we establish the dynamics of how many fBm and GfBm paths can be made before the cost of computing the additional paths outweigh their benefit over calculating the gradients.

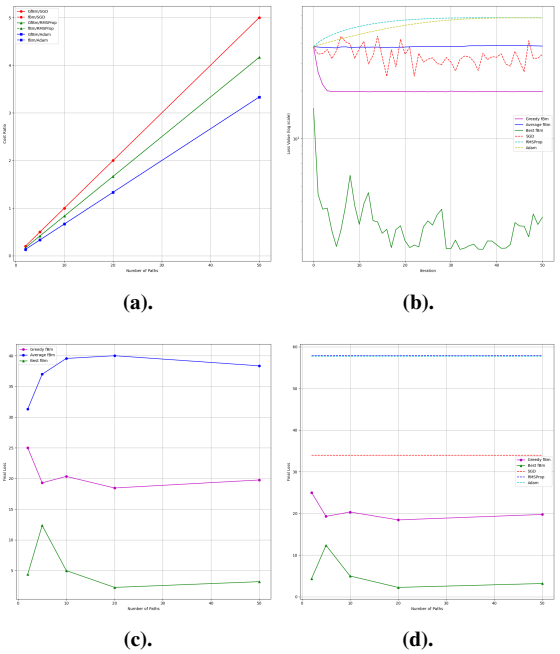
1) fBm-CACO Path Count Analysis: The results of comparing various fBm paths against SGD, RMSProp and Adam on the Rastragin surface, as shown in Figure 5.

Figure 5 shows that SGD, RMSProp and Adam perform better than fBM-CACO paths 'on average', but there is usually a fBM-CACO path with the lowest loss ('best' path) that outperforms the gradient based methods. If m fBm paths take less compute resources to evaluate than gradient based algorithms, and at least one of the m path(s) is more efficient than gradient based algorithms, then the fBm-CACO approach



(a). **(b).** **(c).** **(d).**

Fig. 5: fBm-CACO Path Count Analysis on Rastragin Surface
fBm outperforms SGD, RMSProp and Adam over the Rastragin surface.
(a). fBm: Final Loss vs Number of Paths.
(b). fBm: Cost Ratio vs Number of Paths.
(c). fBm: Loss Value vs Iteration.
(d). fBm vs Traditional Methods: Final Loss Comparison.



(a). **(b).** **(c).** **(d).**

Fig. 6: GfBm-CACO Path Count Analysis on Rastragin Surface
GfBm-CACO outperforms fBM, SGD, RMSProp and Adam over the Rastragin surface.
(a). Greedy fBm: Final Loss vs Number of Paths.
(b). Greedy fBm: Cost Ratio vs Number of Paths.
(c). Greedy fBm: Loss Value vs Iteration.
(d). Greedy fBm vs Traditional Methods: Final Loss Comparison.

does have merit. Since each fBm path has a cost, it was found that about 10-20 paths was the optimal number of paths required to be generated per dimension. If too few paths are generated, then gradient based algorithms will more likely have a better outcome due to the more directed descent approach than from too few fBm paths. At the other extreme, if too many poor performing paths are generated, then gradient based algorithms will more likely have a better outcome due to the higher cost of evaluating numerous poor fBm paths. Further supporting results are shown in Table I.

TABLE I: Path Comparison Metrics Between fBm-CACO and SGD

Paths	SGD Final	Avg fBm	Best fBm	Cost Ratio	fBm Better?
1	34.6621	36.7311	6.2827	0.10	Yes
5	33.7260	38.2747	13.7518	0.50	Yes
10	33.5081	37.1082	8.9317	1.00	Yes
20	28.6644	37.2469	6.7931	2.00	Yes
50	36.1910	39.5867	1.0959	5.00	Yes
100	30.5853	35.6577	2.9564	10.00	Yes

Table I shows that the crossover point or ideal number is approximately 20 fBm paths. Beyond this point, the computational cost of fBm exceeds that of SGD, but usually finds at least one better solution.

2) GfBm-CACO Path Count Analysis: The optimality findings, of between 10-20 paths was also confirmed by the corresponding GfBm-CACO results on the Rastragin surface, as shown in Figure 6 (and the corresponding Table II).

Figure 6 highlights that an optimal range of 10 to 20 paths allows GfBm to effectively explore rough paths. While these paths may, on average, exhibit poorer performance compared to gradient-based algorithms, GfBm consistently identifies paths with lower heights than those found by gradient-based methods. This ultimately enables GfBm to achieve superior solutions at reduced computational costs, because it can cut unfavourable directions early on, which help it 'gravitate' towards the global minimum. The advantages of GfBm are particularly evident in highly complex, non-convex LLS. Furthermore, our analysis reveals that GfBm-CACO significantly outperforms fBm-CACO due to its more sophisticated search approach. This is further supported by Table II.

TABLE II: Comparison Metrics Between GfBm-CACO and Gradient Based Algorithms

Paths	SGD Loss	RMSP Loss	Adam Loss	Best fBm	fBm SGD	fBm RMSP	fBm Adam
1	31.81	57.25	57.25	9.67	0.10	0.08	0.07
5	35.02	63.88	63.94	9.65	0.50	0.42	0.33
10	28.31	57.02	57.04	6.14	1.00	0.83	0.67
20	35.74	58.07	58.02	6.15	2.00	1.67	1.33
50	29.08	55.05	55.08	4.27	5.00	4.17	3.33
100	34.24	55.47	55.46	2.03	10.00	8.33	6.67

Table II shows further evidence that the crossover point for GfBm paths (where GfBm cost exceeds optimizer but still finds better solutions) for SGD, RMSProp and Adam is about 20 paths. At 10 paths, costs are roughly equal, where SGD gradient calculations are approximately 10 times more expensive than generating fBm steps. The optimization quality

versus cost tradeoffs are,

- 1-10 fBm paths: Computationally cheaper than SGD but still find better minima.
- 20-50 fBm paths: Computationally more expensive but find significantly better minima.
- 100+ fBm paths: Dramatically more expensive with diminishing returns.

This metric of about 20 paths is only a yard stick specific to the Rastragin surface, and so an additional experiment was devised across multiple benchmark surfaces to verify the hypothesis that GfBm-CACO outperforms fBm-CACO and gradient based algorithms.

B. Experiment 2 - Complex Surface Results

Having established in Experiment 1 that both fBm-CACO and GfBm-CACO are viable optimisers on the Rastrigin surface, Experiment 2 now focuses on confirming which of the two NGD variants exhibits superior performance, and under what landscape conditions these NGD methods outperform classical gradient-based approaches. Here we use AdamW, which is Adam with decoupled weight decay. Instead of applying L2 regularization by adding the weight-decay term directly to the gradient (as Adam does), AdamW applies weight decay as a separate step, independent of the gradient update. To do so, we begin by evaluating them on a suite of classical benchmark surfaces, the results of which are shown in Figure 7.

Figure 7 highlights how the algorithms behave on the first group of multimodal benchmark surfaces, allowing us to compare their sensitivity to increasing dimensionality. To broaden this comparison to functions with fundamentally different curvature and basin structure, we next examine convergence on the Rosenbrock and Sphere surfaces in Figure 8.

Figure 8 demonstrates how the methods perform on smoother, bowl-shaped surfaces, where gradient information is typically more informative. To determine whether these trends persist on more intricate or deceptive landscapes, we next evaluate convergence on the Step Rastrigin, Deceptive, and Zakharov surfaces, as shown in Figure 9.

Figure 9 completes the visual convergence assessment by covering a final set of structured and irregular surfaces. While these plots reveal qualitative differences in convergence dynamics, we now finally in a strong position to summarise the quantitative outcomes through averaged final-loss comparisons, as shown in Table III.

Table III provides a high-level overview of mean final losses across surface functions and dimensions sampled. To more precisely contrast the optimisers' relative performance, we present a second table containing the same results in a unified, function-by-function layout, as shown in Table IV.

Table IV reorganises the results to facilitate direct comparison between methods. To further characterise performance variability and statistical significance, the next table reports the mean and standard deviation for each configuration, as shown in Table V.

Table V incorporates variability measures and significance testing, offering a more detailed view of algorithmic stability. For completeness, an ultra-compact summary that presents the final-loss landscape in a condensed, heatmap-style layout is presented in Table VI.

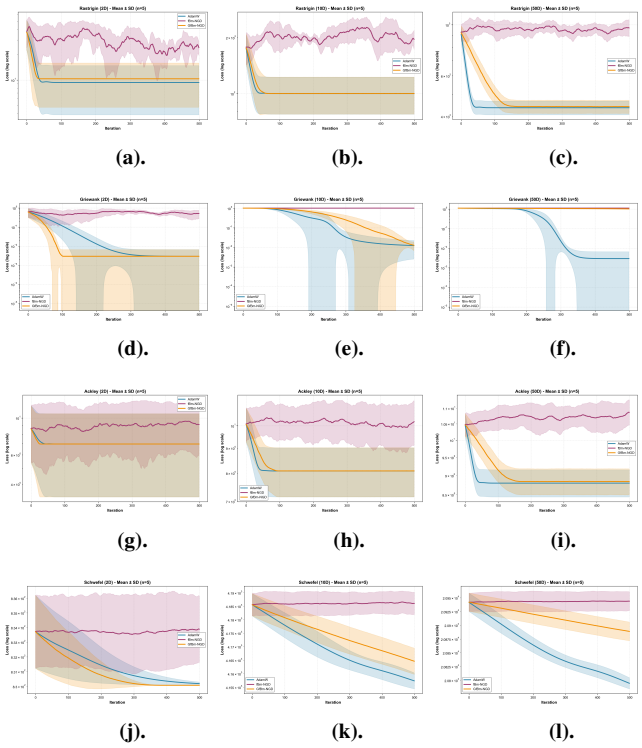


Fig. 7: Convergence Comparison I
(a). Rastragin 2D. (b). Rastragin 10D. (c). Rastragin 50D.
(d). Griewank 2D. (e). Griewank 10D. (f). Griewank 50D.
(g). Ackley 2D. (h). Ackley 10D. (i). Ackley 50D.
(j). Schwefel 2D. (k). Schwefel 10D. (l). Schwefel 50D.

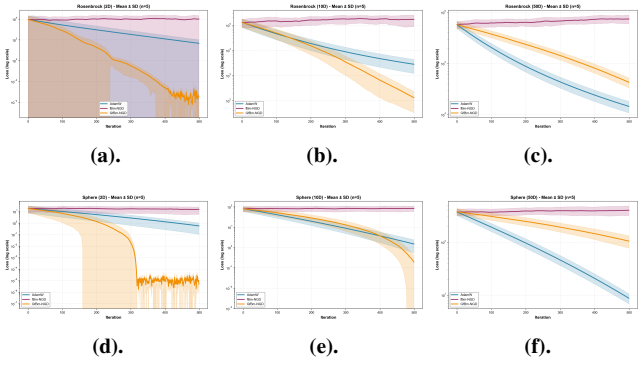
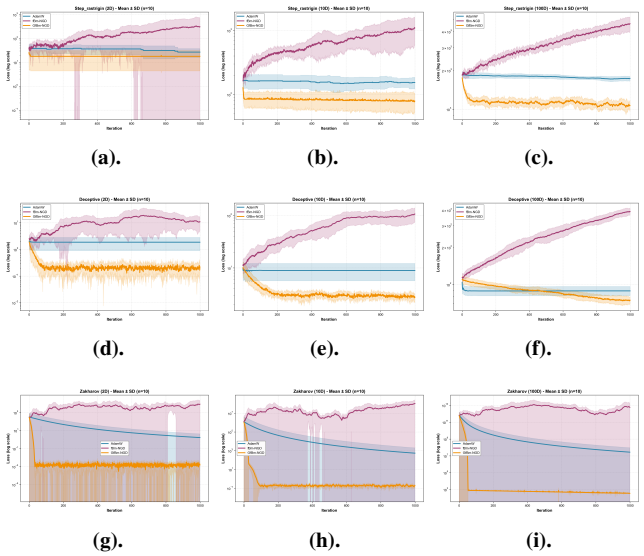


Fig. 8: Convergence Comparison II
(a). Rosenbrock 2D. (b). Rosenbrock 10D. (c). Rosenbrock 50D.
(d). Sphere 2D. (e). Sphere 10D. (f). Sphere 50D.



(a). **(b).** **(c).**

(d). **(e).** **(f).**

(g). **(h).** **(i).**

Fig. 9: Convergence Comparison III
(a). Step Rastrigin 2D. (b). Step Rastrigin 10D. (c). Step Rastrigin 100D.
(d). Deceptive 2D. (e). Deceptive 10D. (f). Deceptive 50D.
(g). Zakharov 2D. (h). Zakharov 10D. (i). Zakharov 50D.

TABLE III: Average Final Loss Summary

Surface	Dim	AdamW_final	fBm_final	GfBm_final
Ackley	2	7.971	9.907	7.971
	10	9.024	11.354	9.033
	50	8.822	10.811	8.828
Griewank	2	0.006	0.817	0.006
	10	0.016	1.062	0.016
	50	0.000	1.120	1.023
Rastrigin	2	15.720	31.292	15.721
	10	86.363	181.501	86.485
	50	399.577	958.729	401.904
Rosenbrock	2	77.345	11670.593	0.010
	10	4205.363	130533.599	190.616
	50	15271.510	703938.044	46685.426
Schwefel	2	832.038	837.409	830.237
	10	4158.382	4187.510	4165.272
	50	20794.170	20937.134	20886.483
Sphere	2	0.848	22.694	0.000
	10	2.933	99.539	0.740
	50	9.050	423.317	115.314

TABLE IV: Average Final Loss Across Benchmark Functions and Dimensions

Function	Dim	AdamW	fBm-CACO	GfBm-CACO
Ackley	2D	6.979	9.147	6.979
	10D	8.083	10.197	8.086
	50D	8.799	10.870	8.840
Griewank	2D	0.003	0.513	0.003
	10D	0.013	1.018	0.013
	50D	0.003	1.111	1.012
Rastrigin	2D	9.353	24.326	10.348
	10D	100.093	195.822	100.253
	50D	436.192	969.719	441.093
Rosenbrock	2D	47.169	10147.804	**0.000**
	10D	2812.121	171284.351	**132.927**
	50D	14522.346	743026.813	42938.308
Schwefel	2D	830.188	833.891	830.075
	10D	4157.378	4186.064	4164.626
	50D	20794.668	20943.946	20888.904
Sphere	2D	0.565	15.477	**0.000**
	10D	1.489	83.525	**0.193**
	50D	8.741	400.816	104.922

TABLE V: Optimizer Performance Comparison (Mean Final Loss $\pm$ SD)

$p < 0.05$ (statistically significant improvement over AdamW).
$p < 0.05$ (statistically significant difference from AdamW, worse performance).
Bold values indicate best performance for that configuration.

Function	Dim	AdamW	fBm-CACO	GfBm-CACO
Ackley	2D	6.979	9.147	6.979
	10D	8.083	10.197	8.086
	50D	8.799	10.870	8.840
Griewank	2D	0.003	0.513	0.003
	10D	0.013	1.018	0.013
	50D	0.003	1.111	$1.012^{\dagger}$
Rastrigin	2D	9.353	24.326	10.348
	10D	100.093	195.822	100.253
	50D	436.192	969.719	441.093
Rosenbrock	2D	47.169	10147.804	**0.000**
	10D	2812.121	171284.351	$\mathbf{132.927}^{*}$
	50D	14522.346	743026.813	$42938.308^{\dagger}$
Schwefel	2D	830.188	833.891	830.075
	10D	4157.378	4186.064	$4164.626^{\dagger}$
	50D	20794.668	20943.946	$20888.904^{\dagger}$
Sphere	2D	0.565	15.477	**0.000**
	10D	1.489	83.525	$\mathbf{0.193}^{*}$
	50D	8.741	400.816	$104.922^{\dagger}$

TABLE VI: Optimizer Performance Summary (Lower is Better)

Function	D	AdamW	fBm	GfBm
Ackley	2	6.979	9.147	6.979
	10	8.083	10.197	8.086
	50	8.799	10.870	8.840
Griewank	2	0.003	0.513	0.003
	10	0.013	1.018	0.013
	50	0.003	1.111	1.012
Rastrigin	2	9.353	24.326	10.348
	10	100.093	195.822	100.253
	50	436.192	969.719	441.093
Rosenbrock	2	47.169	10147.804	**0.000**
	10	2812.121	171284.351	**132.927**
	50	14522.346	743026.813	42938.308
Schwefel	2	830.188	833.891	830.075
	10	4157.378	4186.064	4164.626
	50	20794.668	20943.946	20888.904
Sphere	2	0.565	15.477	**0.000**
	10	1.489	83.525	**0.193**
	50	8.741	400.816	104.922

VI. Discussion and Interpretation of Results

A. Experiment 1 - Discussion and Interpretation of Rastragin Surface Results

The following discussion of the above results provides additional findings and insights.

1) fBm-CACO Loss Rate Analysis: Examining the loss convergence comparison between fBm-CACO and gradient based algorithms shows time series details of convergence of loss rates across iterations, as shown in Figure 10.

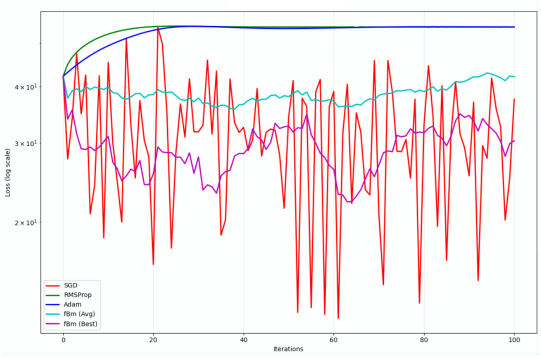


Fig. 10: Loss Convergence Comparison Between fBm and Gradient Based Algorithms

SGD: The most prominent result is that SGD has wild fluctuations between outperforming and underperforming against Adam and RMSProp, to the entent that is shows how ineffective SGD is on such a complex surface as the Rastragin surface.
Best fBm: The algorithm with the next lowest loss was the 'Best fBm', which is somewhat expected, and it exhibits some sinusoidal progression over iterations, which is due to the cyclical hills and valleys of the Rastragin surface.
fBm Average: It was surprising that the fBm average outperformed Adam and RMSProp, because whilst the Best fBM could be expected to be better than gradient based algorithms, the average of fBm paths is a more crude metric, which can lead to poorer results, due to the influence of the bad paths.
Adam and RMSProp: It was also somewhat surprising that these methods underperformed against fBm-CACO. Whilst technically, these may have performed worse than SGD, their performance may end up being better than SGD given SGD's such sporadic results on the Rastragin surface.

Figure 10 shows that SGD could not handle the complexity of the Rastragin surface. The Best fBm path outperformed SGD, Adam and RMSProp, but we also know from the previous section, that it did not outperform the Best GfBm. To discuss this further, additional loss rate results are explored in the next subsection.

2) GfBm-CACO Loss Rate Analysis: The loss rate comparison of the GfBm-CACO algorithm is shown in Figure 11.

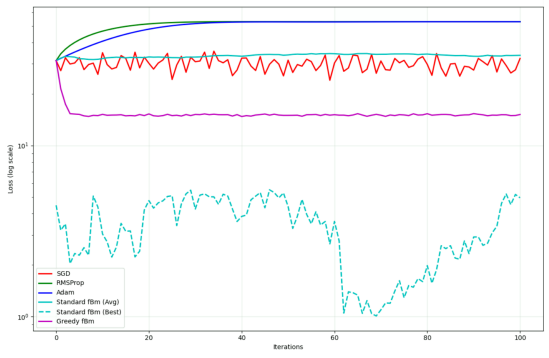


Fig. 11: Loss Convergence Comparison Between GfBm and Gradient Based Algorithms

Best fBm: Best fBm path outperformed Greedy fBm path, but this is a mixed blessing, as the volatility is greater than that of SGD.
Average fBm: As expected, the best fBm path, when averaged out with the poorer performing fBm paths, resulted in worse outcomes. Again, this metric is included to capture that not all fBm paths are always effective, but this just supports the wider search space of fBm paths.
Greedy fBm: This algorithm delivered the best consistent convergence for loss rate optimization.
Best GfBm and Average GfBm: For clarity, these concepts are not shown because they only apply for the fBm case, because in a way, GfBm takes the best step, at every step, to arrive at the best path.
SGD: As in the previous Figure 10, SGD has a hard time otimizing the loss rate, less pronounced than in the previous Figure, possibly due to the magnitude of the Best fBm, dwarfing or compressing the scale of the SGD curve. Due to this volatility, it could be the case that SGD performed worse than Adam and RMSProp.
Adam and RMSProp: These algorithms also had a tough time with the Rastragin surface.

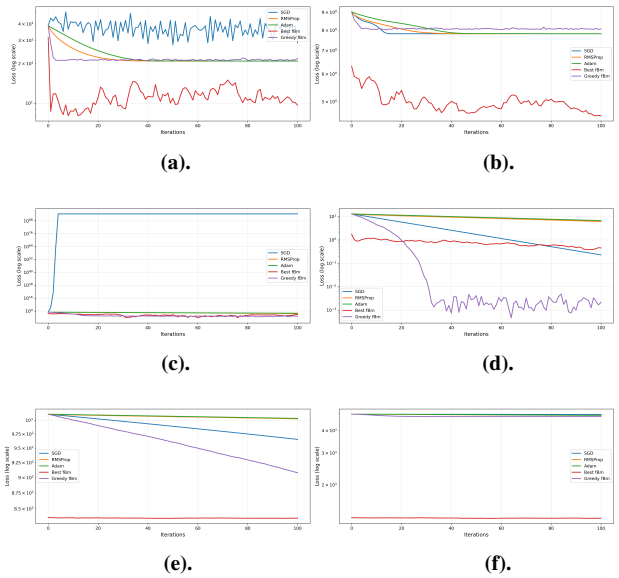
Figure 11 shows that GfBm produced the most robust results. fBm again also performed very well in relation to gradient based algorithms, however, this is dependant to some extent on the simplicity and convexity of the surface. Given that we cited that LLSs only become more complex and nonconvex with increasing dimensional contributions, fBm is still quite a legitimate prospect. In particular, Greedy fBm exhibits the following performance characteristics,

- GfBm-CACO does not always outperform the best standard fBm-CACO path.
- GfBm-CACO outperforms SGD by 2.13 times.
- GfBm-CACO outperforms RMSProp by 3.49 times.
- GfBm-CACO outperforms Adam by 3.49 times.

GfBm-CACO has the following computational efficiency findings,

- Standard fBm requires approximately 1/10 times the computation of SGD.
- GfBm requires approximately 0.5/10 times the computation of SGD.
- GfBm is approximately 2 times less computationally efficient than standard fBm.

Additional comparison metrics between GfBm-CACO and gradient based algorithms are listed in Table VII.



(a). (b). (c). (d). (e). (f).

Fig. 12: Algorithm Convergence Comparison
fBm & GfBm outperforms SGD, RMSProp and Adam over the following surfaces,
(a). Rastragin surface.
(b). Ackley surface.
(c). Rosenbrock surface.
(d). Sphere/Convex surface.
(e). Schwefel surface.
(f). Griewank surface.

TABLE VII: Additional Metrics Comparing GfBm-CACO and Gradient-Based Algorithms

Method	Final Loss	Computational Cost Relative to SGD
SGD	32.2712	1.00
RMSProp	52.8909	1.20
Adam	52.9120	1.50
Standard fBm	33.6591	1.00
Best fBm Path	4.9420	1.00
Greedy fBm	15.1683	1.00

Table VII further shows that GfBm outperforms fBm, and that these non-gradient based algorithms are no more computationally expensive than SGD (even though RMSProp and Adam are more expensive than SGD).

3) Comprehensive Comparisons between fBm, GfBm and Gradient Based Algorithms on Rastragin Surface: We now finalise our comprehensive analysis across our six reference surfaces by presenting Figure 12.

Figure 12 highlights clear differences in optimization behavior across algorithms. Adam shows fast, reliable early convergence on most functions, while SGD and RMSProp follow more direct but sometimes unstable trajectories. fBm-based methods, though noisier, frequently reach better final solutions on highly multimodal landscapes such as Rastrigin and Schwefel.

Path visualizations reinforce this pattern: SGD oscillates, Adam and RMSProp head steadily toward minima, and fBm methods explore a broader region of the landscape. The greedy fBm variant in particular balances exploration and exploitation, reducing the chance of becoming trapped.

Final loss comparisons show Adam excelling on simple functions like Sphere, whereas fBm approaches dominate on complex surfaces with many local minima. RMSProp often improves on SGD, indicating greater stability. Global minimum discovery rates further reveal the limitations of gradient methods on multimodal problems and the superior reliability of fBm methods in such settings.

Performance across random initializations also differs: gradient-based methods show high sensitivity to starting points, while fBm methods remain consistent. Greedy fBm is the most stable and competitive overall. Its ability to escape local minima stems from structured exploration, whereas standard gradient methods frequently become trapped.

Although Adam is more efficient on simple tasks, fBm methods —despite higher computational cost (not due to any one fBm path cost, but due to the ensemble of paths' cost) —offer better results on complex landscapes, making the extra expense worthwhile. As function complexity increases, gradient-based performance degrades sharply, while fBm methods maintain robustness.

Overall, gradient-based algorithms (especially Adam) perform best on smooth, simple functions, whereas fBm approaches excel on challenging multimodal problems. Greedy fBm provides an effective middle ground, and the most suitable method ultimately depends on problem characteristics and computational budget.

B. Experiment 2 - Discussion and Interpretation of Complex Surface Results

Across all benchmark optimization tasks, clear and consistent patterns emerged when comparing the deterministic gradient-based optimizer AdamW with the two noise-driven methods, fBm-CACO and its guided variant GfBm-CACO. The overall results highlight the expected trade-off between deterministic convergence and stochastic exploration: fractional noise can enhance optimization on complex multimodal landscapes, but only when it is directionally constrained.

AdamW delivered the most reliable convergence and the lowest mean losses across most test functions, reaffirming its strength on smooth surfaces with well-behaved gradients. However, GfBm-CACO often matched or exceeded AdamW's performance—particularly on non-convex functions such as Rosenbrock and Sphere—indicating that guided stochasticity can improve navigation through curved or rugged regions of the loss landscape. In contrast, the unguided variant fBm-CACO consistently performed the worst, producing higher losses, larger variability, and statistically significant degradation across nearly all experiments (typically with $p < 10^{-3}$). These results confirm that while noise can aid exploration, unstructured fractional perturbations primarily introduce instability rather than useful search behaviour.

Dimensionality played a central role in magnifying these differences. In low-dimensional settings (2D), all methods showed broadly similar behaviour, but higher dimensions

(10D and 50D) amplified both the strengths and weaknesses of each approach. On highly multimodal surfaces such as Ackley and Rastrigin, AdamW achieved the lowest losses, yet GfBm-CACO frequently remained statistically comparable, suggesting that its guided noise helps escape shallow minima without compromising long-term convergence. Conversely, fBm-CACO tended to plateau or diverge, reflecting an inability to regulate noise magnitude in large search spaces. Similar patterns appeared on Griewank, where AdamW and GfBm-CACO maintained stable convergence while fBm-CACO exhibited pronounced instability. Notably, Rosenbrock and Sphere revealed cases where GfBm-CACO outperformed AdamW outright, achieving the lowest mean losses (e.g., 132.93 on Rosenbrock-10D and 0.1933 on Sphere-10D). These results suggest that guided noise may facilitate adaptive movement along narrow, curved valleys that often hinder deterministic optimizers.

The Schwefel function reinforced these trends. Although AdamW generally produced the best final losses, GfBm-CACO displayed smoother and more consistent convergence trajectories, with reduced oscillation amplitude across repeated runs. This behaviour indicates that while deterministic methods can converge more efficiently, guided stochastic approaches offer improved robustness—an appealing property in non-stationary or irregular optimization environments.

A key insight is that the benefits of guided noise become increasingly pronounced as dimensionality grows. AdamW retained strong overall performance in higher dimensions but occasionally adapted slowly to complex topologies, whereas GfBm-CACO leveraged structured noise to preserve exploratory diversity without destabilizing convergence. In contrast, fBm-CACO deteriorated sharply with dimension, demonstrating that fractional-noise perturbations require directional guidance to remain effective.

These trends were consistently supported by statistical testing, confirming the reliability of the observed differences across multiple seeds and repetitions. The strong performance of GfBm-CACO on several benchmarks, combined with its lower variance and smoother trajectories, indicates that directional guidance transforms fractional noise from a destabilizing factor into a controllable exploration mechanism. Conversely, the weak performance of fBm-CACO underscores that stochasticity alone is insufficient and can easily hinder optimization if not properly structured.

Overall, the results show that guided stochastic optimization represents a promising extension to standard gradient methods. GfBm-CACO effectively balances exploration and exploitation, performs competitively or superiorly on non-convex, curved, or multimodal surfaces, and maintains greater stability across runs. These findings support the broader hypothesis that controlled stochastic perturbations can improve traversal of complex landscapes and point toward future hybrid optimizers that dynamically regulate noise structure in response to local gradient geometry.

VII. Conclusion

This study evaluated the non-gradient optimizers fBm-CACO and its guided variant GfBm-CACO across several classical benchmark landscapes, ranging from highly non-convex functions (Rastrigin, Ackley, Rosenbrock, Schwefel, Griewank) to the convex Sphere function. While gradient-based methods such as AdamW perform well on smooth, low-dimensional surfaces, their advantages diminish in the high-dimensional, multimodal landscapes characteristic of modern neural networks and large-scale learning systems. In such settings, deterministic methods frequently become trapped or slowed by flat, irregular, or ill-conditioned regions.

Our findings show that stochastic exploration can overcome these limitations, but only when noise is directionally controlled. Across all non-convex benchmarks —and increasingly so in higher dimensions —GfBm-CACO outperformed both unguided fBm-CACO and gradient-based baselines. Its strength stems from converting fractional noise into structured exploration, allowing the algorithm to evaluate multiple candidate directions, escape poor regions, and exploit favourable subspaces. This advantage grows with dimensionality, where guided sampling becomes essential and unguided random walks deteriorate rapidly, which is why we adopt GfBm-CACO and call it GB-CACO.

Although GfBm-CACO may relinquish its lead on simple convex surfaces where gradients are fully informative, it consistently excelled on the complex, rugged, and anisotropic landscapes that better reflect practical optimization scenarios. These results demonstrate that guided stochasticity provides a powerful complement to gradient methods.

Overall, GfBm-CACO offers a promising foundation for future optimization research. By balancing exploration and stability, it highlights how structured randomness can enhance performance in challenging high-dimensional problems and suggests a natural path toward hybrid optimizers that integrate gradient information with guided stochastic search.

Funding Information

The first author was supported by an Australian Government Research Training Program (RTP) scholarship and an Australian Defence Innovation Network (DIN) top-up scholarship.

Author's Contributions

Aldo Taranto: Conceptualization, methodology, software, investigation, writing - original draft, writing - review and editing, formal analysis and visualization.
Ron Addie: Validation, writing - review and editing.

Ethics

The corresponding author confirms that all of the other authors have read and approved the manuscript and that there are no ethical issues involved.

References

[1] A. Akhavan, E. Chzhen, M. Pontil, and A. Tsybakov, "Gradient-free optimization of highly smooth functions: Improved analysis and a new algorithm," *arXiv*, vol. arXiv:2306.02159, pp. 1–50, 2023, https://arxiv.org/pdf/2306.02159.

[2] C. Andrieu, N. Chopin, E. Fincato, and M. Gerber, "Gradient-free optimization via integration," *arXiv*, vol. arXiv:2408.00888, pp. 1–36, 2024, https://arxiv.org/pdf/2408.00888.

[3] D. Golberg, *Genetic algorithms in search, optimization, and machine learning*. Addion Wesley, 1989.

[4] R. Storn and K. Price, "Differential Evolution: A Simple and Efficient Heuristic for Global Optimization," *Journal of global optimization*, vol. 11, pp. 341–359, 1997, https://link.springer.com/content/pdf/10.1023/a:1008202821328.pdf.

[5] N. Hansen, "The cma evolution strategy: A comparing review," *Towards a new evolutionary computation: Advances in the estimation of distribution algorithms*, pp. 75–102, 2006, https://link.springer.com/chapter/10.1007/3-540-32494-1_4.

[6] J. Kennedy and R. Eberhart, "Particle swarm optimization," *International Conference on Neural Networks*, vol. 4, p. 1942–1948, 1995.

[7] A. Taranto, B. Nunes, and R. Addie, "Survey of continuous ant colony optimization: Theory, applications and algorithms," *TechRxiv*, pp. 1–42, 2025, https://www.techrxiv.org/doi/pdf/10.36227/techrxiv.175693561.14761595/v1.

[8] R. Lewis, V. Torczon, and M. Trosset, "Direct search methods: then and now," *Journal of computational and Applied Mathematics*, vol. 124, no. 1-2, pp. 191–207, 2000, https://www.sciencedirect.com/science/article/pii/S0377042700004234.

[9] S. Singer and J. Nelder, "Nelder-mead algorithm," *Scholarpedia*, vol. 4, no. 7, p. 2928, 2009.

[10] M. Powell, "An iterative method for finding the minima of a function of several variables without calculating derivatives," 1964.

[11] Y. Liu, Y. Hu, H. Qian, C. Qian, and Y. Yu, "Zoopt: Toolbox for derivative-free optimization," *arXiv*, vol. arXiv:1801.00329, pp. 1–13, 2017, https://arxiv.org/pdf/1801.00329.

[12] E. Aarts, "Simulated annealing: Theory and applications," *Reidel*, 1987.

[13] X. Chen, S. Liu, K. Xu, X. Li, X. Lin, M. Hong, and D. Cox, "Zo-adamm: Zeroth-order adaptive momentum method for black-box optimization," *Advances in neural information processing systems*, vol. 32, pp. 1–12, 2019, https://proceedings.neurips.cc/paper_files/paper/2019/file/576d026223582a390cd323bef4bad026-Paper.pdf.

[14] S. Na and H. Yang, "Curse of dimensionality in neural network optimization," *arXiv*, vol. arXiv:2502.05360, pp. 1–30, 2025, https://arxiv.org/pdf/2502.05360.

[15] M. D. et al., "Recent theoretical advances in non-convex optimization," *High-Dimensional Optimization and Probability*, vol. 191, 2022, https://link.springer.com/chapter/10.1007/978-3-031-00832-0_3.

[16] D. Wolpert and W. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.

[17] H. Petzka and C. Sminchisescu, "Non-attracting regions of local minima in deep and wide neural networks," *Journal of Machine Learning Research*, vol. 22, pp. 1–34, 2021, https://jmlr.csail.mit.edu/papers/volume22/19-586/19-586.pdf.

[18] J. Ornstein, R. Hammond, and M. P. et al., "Rugged landscapes: complexity and implementation science," *Implementation Science*, vol. 15, no. 85, pp. 1–9, 2020, https://doi.org/10.1186/s13012-020-01028-5.

[19] E. Levin, J. Kileel, and N. Boumal, "The effect of smooth parametrizations on nonconvex optimization landscapes," *Mathematical Programming*, vol. 209, p. 63–111, 2025, https://doi.org/10.1007/s10107-024-02058-3.

[20] R. Katende and H. Kasumba, "Efficient saddle point evasion and local minima escape in high-dimensional non-convex optimization," *arXiv*, vol. arXiv:2409.12604v2, pp. 1–20, 2024, https://arxiv.org/pdf/2409.12604.

[21] A. Hollender and M. Zampetakis, "The computational complexity of finding stationary points in non-convex optimization," *Mathematical Programming*, pp. 1–61, 2024, https://doi.org/10.1007/s10107-024-02139-3.

[22] M. Binda, "Why Do N-Sphere's Hypervolume Tend to 0 As the Dimensions Increase?" *SSRN*, pp. 1–12, 2023, https://ssrn.com/abstract=4473625.

[23] A. Taranto, R. Addie, and B. Nunes, "Brownian Bridge - Continuous Ant Colony Optimization (BB-CACO)," *TechRxiv*, pp. 1–31, 2025, https://www.techrxiv.org/doi/full/10.36227/techrxiv.175752063.30563682/v1.

[24] X.-S. Yang, "Ten new benchmarks for optimization," *arXiv*, vol. arXiv:2309.00644v1, pp. 1–11, 2023, https://arxiv.org/pdf/2309.00644.

[25] L. Rastrigin, *Systems of extremal control*. Mir Publishers, Moscow, 1974.

[26] D. Ackley, *A connectionist machine for genetic hillclimbing*. Kluwer Academic Publishers, Boston, 1987.

[27] H. Rosenbrock, "An automatic method for finding the greatest or least value of a function," *The Computer Journal*, vol. 3, no. 3, pp. 175–184, 1960.

[28] K. D. Jong, "An analysis of the behavior of a class of genetic adaptive systems," *PhD Thesis, University of Michigan*, 1975.

[29] H.-P. Schwefel, *Numerical Optimization of Computer Models*. John Wiley & Sons, 1981.

[30] A. Griewank, "Generalized descent for global optimization," *Journal of Optimization Theory and Applications*, vol. 34, no. 1, pp. 11–39, 1981.

[31] M. Jamil and X.-S. Yang, "A literature survey of benchmark functions for global optimization problems," *Int. Journal of Mathematical Modelling and Numerical Optimisation*, vol. 4, no. 2, pp. 150–194, 2013.

[32] D. E. Goldberg, "Simple genetic algorithms and the minimal deceptive problem," in *Genetic Algorithms and Simulated Annealing*, L. Davis, Ed. Morgan Kaufmann, 1987, pp. 74–88.

[33] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs*, 3rd ed. Springer, 1996.

[34] M. Jamil and X.-S. Yang, "A literature survey of benchmark functions for global optimization problems," *International Journal of Mathematical Modelling and Numerical Optimisation*, vol. 4, no. 2, pp. 150–194, 2013.

[35] M. Jamil and X. Yang, "A literature survey of benchmark functions for global optimization problems," *International Journal of Mathematical Modelling and Numerical Optimisation*, vol. 4, no. 2, pp. 150–194, 2013.

Dr. Aldo Taranto Aldo is undertaking postdoctoral research at the Australian National University (ANU) in advanced optimization techniques for high-dimensional machine learning, under an Australian Defence innovation scholarship. He is currently Director of AI Research & Development at MetaModelR Corporation. Aldo holds a BSc(Math) from Monash University (1996), GradDipEd from University of Melbourne (1997), MBSys from Monash University (1998), MB(Acc) from RMIT University (2006) and was awarded a PhD(Math) from the University of Southern Queensland (2022), for his research in stochastic differential equations and their application in mathematical finace and algorithmic trading.

A/Prof. Ron Addie Ron is an Adjunct Associate Professor at the University of Southern Queensland (UniSQ). He began his research career at Telstra Research Laboratories, where he completed a PhD in Markov Additive Processes and co-developed virtual paths —now foundational to ATM broadband networks. He also advanced performance models for Gaussian traffic in core networks. Joining UniSQ in 1993, he served as Head of Mathematics and Computing (2004–2006), taught across multiple IT and science programs, and supervised over 10 PhD students. His Netml software supported 1000+ students and 19+ publications over 15 years. Though retired in 2022, he remains active in research and postgraduate supervision.