SN Computer Science (2022) 3:110
https://doi.org/10.1007/s42979-021-00989-8

ORIGINAL RESEARCH

Generalised Pattern Search with Restarting Fitness Landscape Analysis

Ferrante Neri[1]

Received: 24 July 2021 / Accepted: 5 December 2021 / Published online: 23 December 2021

Abstract
Fitness landscape analysis for optimisation is a technique that involves analysing black-box optimisation problems to extract pieces of information about the problem, which can beneficially inform the design of the optimiser. Thus, the design of the algorithm aims to address the specific features detected during the analysis of the problem. Similarly, the designer aims to understand the behaviour of the algorithm, even though the problem is unknown and the optimisation is performed via a metaheuristic method. Thus, the algorithmic design made using fitness landscape analysis can be seen as an example of explainable AI in the optimisation domain. The present paper proposes a framework that performs fitness landscape analysis and designs a Pattern Search (PS) algorithm on the basis of the results of the analysis. The algorithm is implemented in a restarting fashion: at each restart, the fitness landscape analysis refines the analysis of the problem and updates the pattern matrix used by PS. A computationally efficient implementation is also presented in this study. Numerical results show that the proposed framework clearly outperforms standard PS and another PS implementation based on fitness landscape analysis. Furthermore, the two instances of the proposed framework considered in this study are competitive with popular algorithms present in the literature.

Keywords Pattern search · Local search · Fitness landscape analysis · Covariance matrix · Numerical optimisation

Introduction

Although findings in the continuous domain are not entirely conclusive [2], No Free Lunch Theorems [42] suggest that algorithms are designed to address specific optimisation problems.

Many real-world problems can be formulated as black-box optimisation problems [3]. In these cases, information about the problem is not available a priori; thus, modern implementations include mechanisms to make the algorithm suitable to the specific features of the problem. We can broadly distinguish two approaches to design algorithms. These two algorithmic philosophies, albeit ideologically different, overlap in their practical implementations.

- **adaptive algorithms**: feedback on the algorithmic behaviour regarding the specific problem is collected and used to adjust the algorithm, see [6, 7, 36]
- **fitness landscape analysis**: the optimisation problem is analysed by a method, e.g., an artificial intelligence tool, and the results are used to design the algorithm; see [17, 23, 24, 33, 34]

The feedback used by adaptive algorithms can be categorised into the following two groups.

- **Performance-based feedback**: the most successful parameter setting and/or algorithmic operator(s) are likely to be selected for the subsequent stages of the optimisation process. This is the case for many hyper-heuristic [4, 8] and self-adaptive [19] schemes.
- **Behaviour-based feedback**: some metrics associated with the functioning of the algorithm are monitored and fed back to update parameter settings and/or algorithmic operator(s). Some examples are diversity-based adaptation [26, 32] and super-fit adaptation for swarm intelligence algorithms [5–7, 16].

"This article is part of the topical collection "Applications of bioinspired computing (to real world problems)" guest edited by Aniko Ekart, Pedro Castillo, and Juanlu Jiménez-Laredo"

✉ Ferrante Neri
ferrante.neri@nottingham.ac.uk

[1] COL Laboratory, School of Computer Science, University of Nottingham, Nottingham NG8 1BB, UK

This article focuses on fitness landscape analysis. This study proposes a novel method for analysing fitness landscapes and performs the design of the optimiser based on the proposed analysis' method. The section "Related Works: Algorithmic Design based on Fitness Landscape Analysis" provides some background about algorithmic design informed by fitness landscape analysis. The section "Proposal of this Article" briefly outlines the proposed technique, explains its motivation, and describes the content of the remainder of this article.

Related Works: Algorithmic Design Based on Fitness Landscape Analysis

A fitness landscape is a tuple composed of a set/domain of candidate solutions, a notion of neighborhood, and a fitness/objective function; see [38]. Fitness landscape analysis is a popular topic that has attracted the attention of researchers in optimisation over the past 2 decades; see [22, 24]. Although the majority of studies on fitness landscape analysis focus on the combinatorial domain [35], recent studies proposed valuable contributions to the continuous domain [23, 25]. For example, an analysis of separability performed using the Pearson's coefficient was proposed in [9]. Using an interaction matrix to identify groups of strongly interacting variables has been proposed in [39]. The study in [1] proposes the construction of a graph-based abstraction of the search space representing the optimisation problem, known as a local optimisation network. In this graph, each node of the graph is a local optimum, while the edges between nodes represent the adjacency of the basins of optima.

A special mention should be given to Covariance Matrix Adaptive Evolution Strategy (CMAES) [13, 14]. This popular algorithm progressively adapts a multi-variate Gaussian distribution from which candidate solutions are sampled. This adaptation is performed to increase the likelihood of previously successful candidate solutions. While CMAES runs, its distribution adapts to the geometry of the problem/local optimum. Thus, CMAES can be considered an adaptive algorithm belonging to the performance-based feedback group and an algorithm designed on the basis of a fitness landscape analysis.

Another recent example of an algorithm designed on the basis of a fitness landscape analysis for the continuous domain is the Covariance Pattern Search (CPS) [30, 31]. This algorithm characterises the geometry of the problem by sampling points whose objective function is below a certain threshold. The covariance matrix associated with the sampled points and its eigenvectors are then calculated. These eigenvectors are then used as the search directions of the Generalised Pattern Search (GPS), see [40]. The results in [30, 31] clearly show that the pattern based on the eigenvectors of a well-estimated covariance matrix outperforms the classical pattern based on the fundamental orthonormal basis (the directions of the variables). On the other hand, the application of CPS is impractical, since it requires the setting of the above-mentioned threshold parameter for each optimisation problem. This setting is performed empirically and thus requires considerable computational effort, especially in high-dimension cases. This feature makes CPS neither versatile (over various problems) nor easily scalable.

One paper [28] overcomes this limitation using a restarting scheme that divides the run into local runs. The resulting algorithm, Adaptive Covariance Patter Search (ACPS), uses the best objective function value at each restart as the threshold for the following local run.

Proposal of this Article

The present article extends the concept of CPS by enhancing its fitness landscape analysis. Besides determining the search directions of PS, herein referred to as PS, the present study also assigns a step size to each search direction. Each step size is calculated on the basis of an estimation of the directional derivative along the associated search directions: the proposed method performs large steps when the directional derivative is low (the fitness landscape is flat) and small steps when the directional derivative is high (the fitness landscape is steep). Furthermore, the present study makes use of the restarting strategy proposed in [28] to overcome the CPS limitation of setting a threshold for each problem. Thus, the present article can be considered a generalisation of ACPS to an algorithmic framework, which is referred to as Generalised Pattern Search with Restarting Fitness Landscape Analysis (GPSRFLA).

The remainder of this article is organised as follows: The section "Basic Notation and Generalised Pattern Search" introduces the notation and describes the basics of PS and GPS. The section "Proposal of this Article" describes the proposed framework and provides a pertinent theoretical justification for the fitness landscape analysis. The section "A Computationally Efficient Instance of GPSRFLA" describes ACPS and presents it as a computationally efficient instance of GPSRFLA. The section "Numerical Results" provides the numerical results of this work. Finally, the section "Conclusion" ends with the conclusive remarks of the study.

Basic Notation and Generalised Pattern Search

Before entering the description of the algorithms, let us introduce the notation used throughout this paper. Let us indicate with $\mathbf{x}$ an n-dimensional vector of real numbers ($\mathbf{x} \in \mathbb{R}^n$). We will refer to a numerical optimisation problem

that is the minimisation of a function $f : D \to Y$ where $D \subseteq \mathbb{R}^n$ and $Y \subseteq \mathbb{R}$

$$\min_{\mathbf{x} \in D} f(\mathbf{x}).$$

In this study, we will focus on the box constrained case ($[a_1, b_1] \times [a_2, b_2] \ldots \times \ldots [a_n, b_n]$ with $\times$ indicating the Cartesian product), which includes the unconstrained case $]-\infty, +\infty[^n = \mathbb{R}^n$.

We will call the set D "decision space". Also, we will refer to the n-dimensional vector $\mathbf{x}$ as "vector", "point", or "candidate solution", while we will refer to its components as "design variables".

The PS algorithms are a family of deterministic direct search methods [40], i.e., deterministic optimisation algorithms that do not require gradient calculations. The algorithms that belong to this family have been conceptualised by means of a generalised scheme, namely GPS [40]. GPS is characterised by two elements:

- a set of search directions (a basis of vectors) spanning the decision space D;
- a trial step vector endowed with a step variation rule.

From an initial point $\mathbf{x}$, the PS algorithms perturb the solution along the search directions in an iterative manner. Let us indicate with k the iteration index. Formally, the search directions are determined by two matrices. The first is a non-singular matrix, namely the *basis matrix*, and it is indicated by $\mathbf{B} \in \mathbb{R}^{n \times n}$ where $\mathbb{R}^{n \times n}$ is the set of square matrices of real numbers of order n. The second is a rectangular matrix, namely the *generating matrix*, and it is indicated with $\mathbf{G}_k \in \mathbb{Z}^{n \times p}$ where $\mathbb{Z}^{n \times p}$ is the set of matrices of relative numbers of size n by p with $p > 2n$ and rank n.

The search directions are given by the columns of the matrix

$$\mathbf{P}_k = \mathbf{B}\mathbf{G}_k \tag{1}$$

that is referred to as the *pattern* (and has size $n \times p$). Thus, a pattern can be seen as a repository of search directions, with n of them being in the direction of a basis of $\mathbb{R}^n$ and n of them being in the same directions but with opposite orientation. There may potentially be some additional directions.

The GPS k^{th} *trial step* along the i^{th} direction is the vector $\mathbf{s}_k$, defined as

$$\mathbf{s}_k = \Delta_k \mathbf{B}\mathbf{g}_k^i, \tag{2}$$

where Δ_k is a positive real number and $\mathbf{g}_k^i$ is the i^{th} column of the matrix $\mathbf{G}_k$. The parameter Δ_k determines the step size, while $\mathbf{B}\mathbf{g}_k^i$ is the direction of the trial step.

If $\mathbf{x}_k$ is the current best solution at the iteration k, the trial point generated by means of the trial step would be

$$\mathbf{x^t}_k = \mathbf{x}_k + \mathbf{s}_k. \tag{3}$$

The set of operations that yields a current best point is called the *exploratory move*. The exploratory move succeeds when a solution with better performance is detected, and fails when no update of the current best is found. Within the GPS family, various PS implementations employ different strategies, e.g., attempting only one trial vector per step or exploring all the columns of $\Delta_k \mathbf{P}_k$.

The pseudocode of GPS is given in Algorithm 1.

Algorithm 1 Generalized Pattern Search [40]

```
INPUT x
k ← 1
x_k ← x
while condition on the budget do
    generate the trial step s_k from Δ_k P_k
    calculate x^t_k ← x_k + s_k # Exploratory Move
    if f(x^t_k) ≤ f(x_k) then
        x_{k+1} ← x^t_k
    else
        x_{k+1} ← x_k
    end if
    k ← k + 1
    update P_k and Δ_k
end while
x ← x_{k+1}
RETURN x
```

An Implementation of Pattern Search

Although GPS in [40] refers to a generic basis matrix $\mathbf{B}$, most PS implementations use the identity matrix $\mathbf{I}$ as the matrix, that moves along the directions of the problem. Furthermore, in the absence of specific information, the elements of the generating matrix $\mathbf{G}_k$ are selected to explore each direction in the same way.

One example is the greedy implementation proposed in [41], which states that each design variable samples a trial solution—if the first move fails, it attempts to explore the opposite direction. This greedy approach appears to be especially effective for multi-variate problems as it allows a quick enhancement of the initial solution; see [41]. Let us indicate with $\mathbf{e}^1, \mathbf{e}^2, \ldots \mathbf{e}^n$ the orthonormal basis of $\mathbb{R}^n$

$$\mathbf{e}^1 = (1, 0, \ldots 0)^{\mathbf{T}}$$
$$\mathbf{e}^2 = (0, 1, \ldots 0)^{\mathbf{T}}$$
$$\ldots$$
$$\mathbf{e}^n = (0, 0, \ldots 1)^{\mathbf{T}},$$

where the apex $^{\mathbf{T}}$ indicates the transpose and ρ is a scalar ($\rho = \Delta_1$). This greedy PS first samples (minus move)

$$\mathbf{x}^{\mathbf{t}} = \mathbf{x}_k - \rho \cdot \mathbf{e}^i, \tag{4}$$

and if this trial point is worse than the current best $\mathbf{x}_k$, it attempts to sample (plus move)

$$\mathbf{x}^{\mathbf{t}} = \mathbf{x}_k + \frac{\rho}{2} \cdot \mathbf{e}^i \tag{5}$$

before moving to the following design variable. We will say that a move succeeded if the objective function f of the trial point $\mathbf{x}^{\mathbf{t}}$ is better than that of $\mathbf{x}_k$. The number of successful and failed moves determines the cost of a full scan alongside all directions. Each scan requires between n and $2n$ objective function calls.

It can be remarked that an asymmetric exploration is carried out to avoid revisiting the same solution multiple times, see [31]. If moves in all directions fail, then radius ρ is halved. The algorithm is stopped either when the radius ρ is smaller than the tolerance value or when the computational budget is exceeded. The pseudocode of PS is reported in Algorithm 2.

Algorithm 2 Pattern Search according to the greedy implementation in [41]

INPUT $\mathbf{x}$
$k \leftarrow 1$
$\mathbf{x}_k \leftarrow \mathbf{x}$
while condition on the budget **do**
  $h \leftarrow k$
  for $i = 1 : n$ **do**
    $\mathbf{x}^{\mathbf{t}} \leftarrow \mathbf{x}_k - \rho \cdot \mathbf{e}^i$
    if $f\left(\mathbf{x}^{\mathbf{t}}\right) \leq f\left(\mathbf{x}_k\right)$ **then**
      $k \leftarrow k + 1$
      $\mathbf{x}_k \leftarrow \mathbf{x}^{\mathbf{t}}$
    else
      $\mathbf{x}^{\mathbf{t}} \leftarrow \mathbf{x}_k + \frac{\rho}{2} \cdot \mathbf{e}^i$
      if $f\left(\mathbf{x}^{\mathbf{t}}\right) \leq f\left(\mathbf{x}_k\right)$ **then**
        $k \leftarrow k + 1$
        $\mathbf{x}_k \leftarrow \mathbf{x}^{\mathbf{t}}$
      end if
    end if
  end for
  if $h = k$ #If no improvement occurred **then**
    $\rho \leftarrow \frac{\rho}{2}$
  end if
end while
$\mathbf{x} \leftarrow \mathbf{x_k}$
RETURN $\mathbf{x}$

In terms of GPS notation, this PS implementations in two dimensions ($n = 2$) are characterised by the basis matrix

$$\mathbf{B} = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix},$$

while the generating matrix $\mathbf{G}_k$ is

$$\mathbf{G}_k = \begin{pmatrix} \frac{1}{2} & 0 & -1 & 0 & \frac{1}{2} & \frac{1}{2} & -1 & -1 & 0 \\ 0 & \frac{1}{2} & 0 & -1 & \frac{1}{2} & -1 & -1 & \frac{1}{2} & 0 \end{pmatrix},$$

and $\Delta_k = \rho$. Each row of the matrix $\mathbf{G}_k$ represents a variable in the system of coordinates identified by the basis $\mathbf{B}$. Each column contains the information of a possible outcome of the for loop in Algorithm 2. For example, the first column represents a scenario in which, along the first variable, the minus move failed and the plus move succeeded and, along the second variable, both the moves failed. In a similar way, the sixth column indicates that, along the first variable, the minus move failed and the plus move succeeded, while along the second variable, the minus move succeeded. In other words, each column of $\mathbf{G}_k$ is a possible linear combination of plus and minus moves that can be potentially performed within the for loop in Algorithm 2.

landscape uses progressively more updated data to make progressively more accurate decisions about the pattern that enhances the performance of the algorithm. Thus, the proposed framework is designed to progressively learn the optimisation problem and train Patter Search accordingly.

At the beginning of the optimisation, one point $\mathbf{x}$ is sampled within the decision space D. A total budget t_b is allocated to the entire optimisation process, including the fitness landscape analysis. Then, a maximum local budget l_b is allocated to the fitness landscape analysis and PS between two consecutive restarts, which is referred to as *local run*. The inputs of the fitness landscape analysis are the domain D, the objective function f, and the current best point $\mathbf{x}$. The outputs of the fitness landscape analysis are the basis matrix $\mathbf{B}$ and the generating matrix $\mathbf{G}_k$. The latter two matrices, which compose the pattern matrix, are then used as inputs with the current best solution $\mathbf{x}$ for the Pattern Search local run. The output of the Pattern Search local run is the current best solution $\mathbf{x}$, which is then inputted into the fitness landscape analysis component again. At each restart, the radius Δ_k is reinitialised to search for the optimum with the new pattern matrix $\mathbf{P}_k$ in the following local run. Algorithm 3 describes the external framework of the proposed GPSRFLA.

Algorithm 3 External Framework of GPSRFLA

```
while Condition on the total budget t_b do
    % Local Run
    while Condition on the local budget l_b do
        INPUT objective function f(x), decision space D, and current best solution x
        Apply fitness landscape analysis as in Algorithm 4
        OUTPUT basis matrix B and generating matrix G_k
        INPUT current best solution x, basis matrix B, and generating matrix G_k (that is the
        pattern matrix P_k)
        Apply Generalised Patter Search with the pattern calculated by the fitness landscape
        analysis as in Algorithm 5
        OUTPUT the updated current best solution x
    end while
    Initialise the radius Δ_k to its initial value
end while
```

Generalised Pattern Search with Restarting Fitness Landscape Analysis

The proposed GPSRFLA framework is composed of two algorithmic components

- Fitness Landscape Analysis
- (Generalised) Pattern Search,

which are periodically restarted. At each restart, the fitness landscape is analysed and the analysis informs the setting of the pattern of PS. It is then ran. At each restart, the fitness

The following two subsections describe in detail the functioning of the fitness landscape analysis and PS, respectively.

Fitness Landscape Analysis

The fitness landscape analysis component makes use of a data structure $\mathbf{V}$, which can contain up to n_v candidate solutions. At the first local run, n_s points (with $n_s > n_v$) are sampled within the decision space D. The objective function value of these n_s points is calculated and the n_v with the best objective function values are saved in the data structure

V. In the following local runs, n_s points (with $n_s > n_v$) are sampled in the neighborhood of the best current solution **x**. If the candidate solution **x** is

$$\mathbf{x} = (x_1, x_2, \ldots, x_n),$$

the neighborhood is determined by the hyper-cube, where each side is the interval $[x_i - \delta, x_i + \delta]$ where $\delta = k_v \cdot \rho$ with k_v parameter to set and ρ is the radius of PS; see section 2.1.

The data structure **V** can be represented as

$$\mathbf{V} = \begin{pmatrix} x_{1,1} & x_{1,2} & \ldots & x_{1,n} \\ x_{2,1} & x_{2,2} & \ldots & x_{2,n} \\ \ldots & \ldots & \ldots & \ldots \\ x_{m,1} & x_{m,2} & \ldots & x_{m,n} \end{pmatrix}.$$

Using the points (vectors) in **V**, the mean vector and covariance matrix **C** are calculated. The mean vector μ is calculated as

$$\boldsymbol{\mu} = (\mu_1, \mu_2, \ldots, \mu_n) = \frac{1}{m}\left(\sum_{i=1}^{m} x_{i,1}, \sum_{i=1}^{m} x_{i,2}, \ldots, \sum_{i=1}^{m} x_{i,n}\right)^{\mathbf{T}}$$

and the generic element $c_{j,l}$ of the covariance matrix **C** is:

$$c_{j,l} = \frac{1}{m}\sum_{i=1}^{m}\left((x_{i,j} - \mu_j)(x_{i,l} - \mu_l)\right).$$

Then, the eigenvectors

$$\mathbf{P} = (\mathbf{p}^1, \mathbf{p}^2, \ldots, \mathbf{p}^n)$$

of **C** are calculated via Cholesky factorisation. Since **C** is symmetric, it is diagonalizable, and an orthogonal basis of its eigenvectors can be found; see [27]. These eigenvectors are used as the basis to explore the space in the PS logic. In other words, the matrix **P**, whose columns are the eigenvectors of **C**, is used as the basis matrix **B** of GPS; see [29].

The eigenvalues

$$\lambda_1, \lambda_2, \ldots \lambda_n$$

associated with the eigenvectors $\mathbf{p}^1, \mathbf{p}^2, \ldots, \mathbf{p}^n$, respectively, are used to update the generating matrix $\mathbf{G}_k$ of GPS. More specifically, the matrix $\mathbf{G}_k$ can be represented as a vector of row vectors; each of them associated with a design variable of the optimisation problem

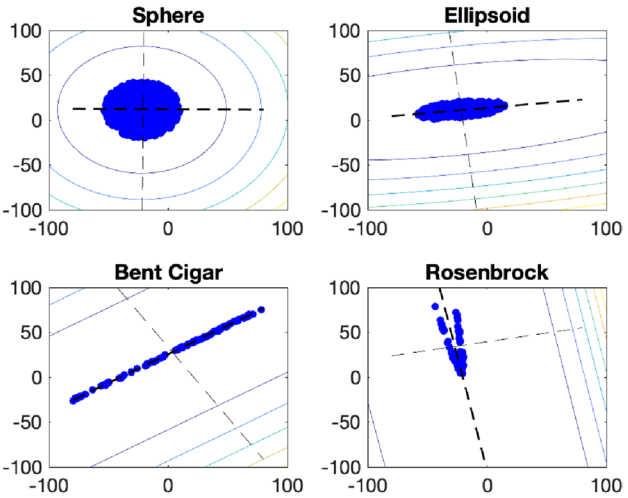


Fig. 1 Sampling of points (blue dots) within $[-100, 100]^2$ for shifted and rotated sphere, ellipsoid, bent cigar, and Rosenbrock functions below the threshold values 10^3, 3×10^4, 10^6 and 5×10^3, respectively. Each sub-figure shows the contour of the function under consideration. The pairs of dashed lines in each sub-figure indicate the directions of the eigenvectors of the Covariance matrix associated with the sampled points

$$\mathbf{G}_k = \begin{pmatrix} \mathbf{g}_{k1} \\ \mathbf{g}_{k2} \\ \ldots \\ \mathbf{g}_{kn} \end{pmatrix}.$$

The generating matrix $\mathbf{G}_k$ is then updated by multiplying each row by the square root of the corresponding eigenvalue

$$\mathbf{Q}_k = \begin{pmatrix} \sqrt{\lambda_1} \cdot \mathbf{g}_{k1} \\ \sqrt{\lambda_2} \cdot \mathbf{g}_{k2} \\ \ldots \\ \sqrt{\lambda_n} \cdot \mathbf{g}_{kn} \end{pmatrix}. \tag{6}$$

The PS is then run on current best solution **x**, with the pattern $\mathbf{P}_k$ calculated as

$$\mathbf{P}_k = \mathbf{P}\mathbf{Q}_k. \tag{7}$$

Algorithm 4 displays the pseudocode of the Fitness Landscape Analysis.

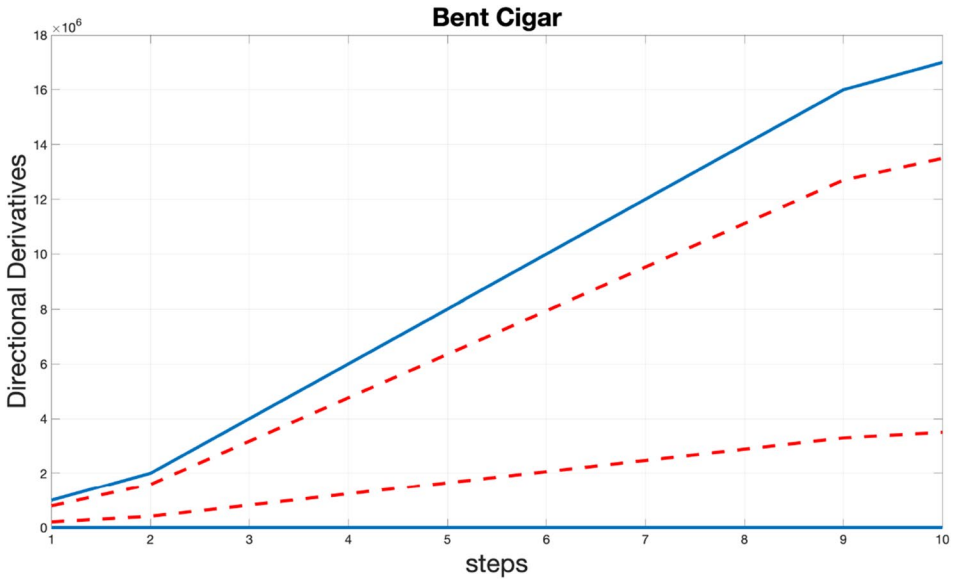


Fig. 2 Plot of the directional derivatives from the optimum of the bent cigar in two variables along the directions of the variables $\mathbf{e}^i$ (red dashed line) and along the directions of the eigenvectors $\mathbf{p}^i$ (blue solid line)

Algorithm 4 Fitness Landscape Analysis

1: **INPUT** objective function $f(\mathbf{x})$, decision space D, the generating matrix $\mathbf{G}_k$ and the parameters k_v, ρ n_v and n_s
2: $h \leftarrow 1$
3: **for** $s = 1 : n_s$ **do**
4: Sample a point $\mathbf{x}$ in $[x_i - k_v \cdot \rho, x_i + k_v \cdot \rho]^n$ (the entire decision space D for the first local run)
5: Calculate $f(\mathbf{x})$
6: **end for**
7: Select the best n_v points among the sampled n_s and save them in the data structure $\mathbf{V}$
8: Calculate the mean vector μ and the covariance matrix $\mathbf{C}$ of the points in $\mathbf{V}$
9: Apply Cholesky factorisation to extract the eigenvectors $\mathbf{P} = (\mathbf{p}^1, \mathbf{p}^2, \ldots, \mathbf{p}^n)$ and eigenvalues $\lambda_1, \lambda_2, \ldots, \lambda_n$
10: Calculate the generating matrix as $\mathbf{Q}_k \leftarrow \begin{pmatrix} \sqrt{\lambda_1} \cdot \mathbf{g}_{k1} \\ \sqrt{\lambda_2} \cdot \mathbf{g}_{k2} \\ \cdots \\ \sqrt{\lambda_n} \cdot \mathbf{g}_{kn} \end{pmatrix}$
11: **OUTPUT** pattern $\mathbf{P}_k \leftarrow \mathbf{P}\mathbf{Q}_k$

Rationale of Fitness Landscape Analysis

This section explains the rationale behind the choices made above, i.e., what the fitness landscape analysis measures and how it informs the algorithmic design of PS. First, it is important to visualise the information contained in the data structure $\mathbf{V}$. Let us consider the following four shifted and rotated objective functions in two dimensions within $[-100, 100]^2$; see [20]:

$$\begin{aligned} \text{Sphere} \quad & f(\mathbf{x}) = z_1^2 + z_2^2 \\ \text{Ellipsoid} \quad & f(\mathbf{x}) = 50z_1^2 + 200z_2^2 \\ \text{Bent Cigar} \quad & f(\mathbf{x}) = z_1^2 + 10^6 z_2^2 \\ \text{Rosenbrock} \quad & f(\mathbf{x}) = 100\left(z_1^2 - z_2\right)^2 + \left(z_1 - 1\right)^2, \end{aligned}$$

where $\mathbf{z} = \mathbf{R}(\mathbf{x} - \mathbf{o})$. The shift vector is

$$\mathbf{o} = \begin{pmatrix} -21.98 \\ 11.55 \end{pmatrix}$$

and $\mathbf{R}$ is a random rotation matrix. Figure 1 displays the plot of the bi-dimensional vectors contained in the data structure $\mathbf{V}$ and the directions of the eigenvectors of the covariance matrix associated with the sampled points.

Figure 1 shows that the data structure $\mathbf{V}$ contain pieces of information about the geometry of the problem and that the eigenvectors of the covariance matrix identify important directions for such problem. For example, for the bent cigar problem, the eigenvectors identify the longitudinal and transverse axes of the line detected by the points.

As reported in [31], the rationale behind the choice to use the eigenvectors $\mathbf{p}^i$ is due to the fact that the matrix $\mathbf{P}$, whose

columns are the eigenvectors $\mathbf{p}^i$, is the transformation matrix that diagonalises the matrix $\mathbf{C}$ that is

$$\mathbf{\Lambda} = \mathbf{P^{-1}CP} = \mathbf{P^T CP}, \tag{8}$$

where $\mathbf{\Lambda}$ is a diagonal matrix whose diagonal elements are the eigenvalues of $\mathbf{C}$ and $\mathbf{P^{-1}} = \mathbf{P^T}$ as $\mathbf{P}$ is an orthogonal matrix ($\mathbf{P^T}$ is the transpose of the matrix $\mathbf{P}$). The directions of the eigenvectors can be interpreted as a new reference system characterised by a lack of correlation between pairs of variables. Thus, the new reference system exploits the available information about the geometry of the problem. This concept is broadly used in other contexts, especially in data science, and is closely related to principal component analysis [18].

Furthermore, as reported in [31], the directions of the eigenvectors of the covariance matrix identify the maximum and minimum directional derivatives. Thus, these eigenvectors are an efficient basis for Patter Search. For example, let us consider the shifted and rotated bent cigar function in two variables $f(\mathbf{x}) = z_1^2 + 10^6 z_2^2$ with $\mathbf{z} = \mathbf{R}(\mathbf{x} - \mathbf{o})$. The shift vector is

$$\mathbf{o} = \begin{pmatrix} -21.98 \\ 11.55 \end{pmatrix}$$

and $\mathbf{R}$ is the rotation matrix

$$\mathbf{R} = \begin{pmatrix} -0.45408 & -0.89096 \\ -0.89096 & 0.45408 \end{pmatrix}.$$

Figure 2 shows the estimated directional derivative along the directions of the variables, as in the case of PS (see Algorithm 2), and along the directions of the eigenvectors $\mathbf{p}^i$ of the covariance matrix.

Figure 2 implicitly provides an interpretation of the search along the directions of the eigenvectors $\mathbf{p}^i$: the directions of these eigenvectors identify the steepest and flattest directions of the fitness landscape.

To enhance the performance of PS, it is here proposed to use large step sizes along those directions corresponding to a flat fitness landscape and small step sizes along those directions corresponding to a steep fitness landscape. Although we cannot know in advance the values of the directional derivatives, we have their estimated values by means of the eigenvalues of the covariance matrix. This is the main motivation of the proposed update of the generating matrix $\mathbf{G}_k$.

Let us consider again the covariance matrix $\mathbf{C}$ calculated as shown above. Let $\mathbf{P} = \left(\mathbf{p^1}, \mathbf{p^2}, \ldots, \mathbf{p^n}\right)$ be a matrix whose columns are the eigenvectors of $\mathbf{C}$ and let us indicate with

$$diag(\mathbf{\Lambda}) = \left(\lambda_1, \lambda_2, \ldots, \lambda_n\right)$$

the corresponding eigenvalues.

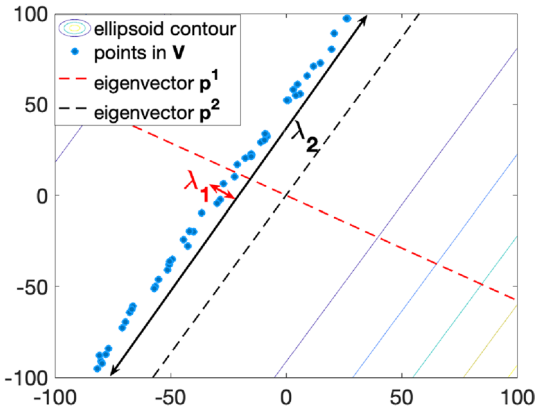


Fig. 3 Distribution of points in $\mathbf{V}$, directions of the eigenvectors and meaning of eigenvalues ($\lambda_1 = 1.5221$ and $\lambda_2 = 4324.1$) in the domain for the rotated and shifted ellipsoid in two dimensions

It must be observed that, since $\mathbf{C}$ is symmetric, the eigenvalues are all real numbers; see [27]. Furthermore, the eigenvectors can be chosen as an orthonormal basis (every pair of vectors is orthogonal and each vector has modulus equal to 1) of a vector space, which we refer to as eigenspace. These eigenvectors span the domain D.

Thus, if we consider a vector $\mathbf{x} \in D$ expressed in the basis $B_\mathbf{e} = \{\mathbf{e}^1, \mathbf{e}^2, \ldots, \mathbf{e}^n\}$, we may express it through the corresponding vector $\mathbf{y}$ in the reference system/basis of the eigenvectors

$$\mathbf{y} = \mathbf{P^T x}.$$

Since the mean vector μ calculated from the vectors in $\mathbf{V}$ is also an element of D, we can express it via eigenvectors

$$\mu_\mathbf{y} = \mathbf{P^T}\mu.$$

Let us now introduce the covariance matrix $\mathbf{C_y}$ of the data in $\mathbf{V}$ in the reference system identified by the eigenvectors. This is expressed by

$$\mathbf{C_y} = (\mathbf{P^T X_c})(\mathbf{P^T X_c})^\mathbf{T} = \mathbf{P^T X_c X_c^T P} = \mathbf{P^T CP}, \tag{9}$$

where

$$\mathbf{X_c} = (\mathbf{x_1} - \mu, \mathbf{x_2} - \mu, ..., \mathbf{x_m} - \mu).$$

From Eqs. (8) and (9), it follows that:

$$\mathbf{C_y} = \mathbf{\Lambda}. \tag{10}$$

Thus, the diagonal elements of $\mathbf{C_y}$ are the eigenvalues of $\mathbf{C}$, while the extradiagonal elements are zero. Since the diagonal elements of a covariance matrix are the variances σ_i^2 of the data along the direction $\mathbf{p^i}$, it follows that:

$$\sigma_i^2 = \lambda_i.$$

The selection of the best n_v points out of the n_s available samples can be conceptually considered the selection of points whose objective function value is below a certain threshold *thre*, where *thre* is the objective function value of the point in the data structure $\mathbf{V}$ with the worst (highest) objective function value. Thus, we may say that the fitness landscape analysis selects those points, such that $f(\mathbf{x}) \leq thre$. In a basin of attraction, these samples would be distributed around a local optimum. Let us suppose, for simplicity, the notation that the optimum is in the null vector $\mathbf{o}$ (i.e., let us apply an operation of translation). The directional derivative along some direction $\mathbf{p^i}$ is

$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{p^i}} \approx \frac{f(\mathbf{x^i}) - f(\mathbf{o})}{|\mathbf{x^i} - \mathbf{o}|} = \frac{f(\mathbf{x^i} - f(\mathbf{o}))}{|\mathbf{x^i}|}.$$

Let us observe that $f(\mathbf{o})$ is a constant, $\mathbf{x^i} = l \cdot \mathbf{p^i}$ with l modulus of $\mathbf{x^i}$ and $|\mathbf{p^i}| = 1$, since it is a versor (i.e., a vector with modulus 1). When we pose $f(\mathbf{x}) = thre$, we find that $f(\mathbf{x^i}) - f(\mathbf{o}) = thre^*$ is also a constant. Thus

$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{p^i}} \approx \frac{thre^*}{|l|} \tag{11}$$

that is the directional derivative along the eigenvector $\mathbf{p^i}$ is inversely related to the modulus l.

Along the direction of $\mathbf{p}^i$, there exists a correlation between the modulus l and the corresponding eigenvalue λ_i. Let us consider the two points $\mathbf{x^i}$ and $-\mathbf{x^i}$ belonging to the direction of $\mathbf{p^i}$. Let us assume that the objective function values of these points are *thre*. Thus, the distance between $\mathbf{x^i}$ and $-\mathbf{x^i}$ estimates the width of the distribution along the direction of $\mathbf{p}^i$ and the standard deviation estimates the average modulus of the points in $\mathbf{V}$. Considering that the standard deviation calculated along the direction of $\mathbf{p}^i$ is the square root of λ_i, it follows that:

$$\sqrt{\lambda_i} = \sigma_i = \sqrt{\frac{1}{2}((\mathbf{x^i} - \mathbf{o})^2 + (-\mathbf{x^i} - \mathbf{o})^2)} = l. \tag{12}$$

By combining Eq. (11) and (12), we may conclude that the directional derivative in the direction of an eigenvector $\mathbf{p}^i$ of the covariance matrix $\mathbf{C}$, as calculated above, is inversely proportional to the square root of the corresponding eigenvalue

$$\frac{\partial f(\mathbf{x})}{\partial \mathbf{p^i}} \approx \frac{thre^*}{\sqrt{\lambda_i}}. \tag{13}$$

Figure 3 reports an example that is useful in visualising the meaning of Eq. (12). The points belonging to the data structure $\mathbf{V}$ associated with the samples for the shifted and rotated ellipsoid in two dimensions are reported as blue dots. The dashed lines indicate the directions of the eigenvectors of the covariance matrix. The two associated eigenvalues are $\lambda_1 = 1.5221$ and $\lambda_2 = 4324.1$, respectively. These numbers reflect the distribution of the points that appear as a thin and long line. We may observe that $\lambda_2 \gg \lambda_1$ and the points in $\mathbf{V}$ span a much wider range along the direction of $\mathbf{p}^2$ (in black) than along the direction of $\mathbf{p}^1$ (in red). Therefore, we can see that the eigenvalues estimate the extent of the distribution of points along the directions of the corresponding eigenvectors.

Furthermore, as shown by the contour, along the direction of the first eigenvector, the fitness landscape is very steep. However, along the direction of the second eigenvector, the landscape is nearly flat. This statement intuitively explains the meaning of Eq. (13).

Since the square roots of the eigenvalues are inversely proportional to the directional derivative, it is proposed to use them as direct multipliers to set the step sizes along each search direction of the basis of eigenvectors. With reference to Fig. 3, along the direction of $\mathbf{p}^1$, the landscape is steep and the corresponding eigenvalue λ_1 is small. Therefore, we use λ_1 as a multiplier to ensure that small steps are performed. Conversely, along the direction of $\mathbf{p}^2$, the landscape is flat and the corresponding eigenvalue λ_2 is large. Thus, we use λ_2 as a multiplier to enable large steps along that direction. This logic explains the proposed way of modifying $\mathbf{Q}_k$ in Eq. (6).

Pattern Search Designed on the Basis of Fitness Landscape Analysis

This article proposes a restarting algorithm based on PS logic presented in Algorithm 2. At each local run, the fitness landscape analysis returns the pattern $\mathbf{P}_k = \mathbf{PQ}_k$. This means that the minus move along the ith direction is

$$\mathbf{x^t} = \mathbf{x}_k - \rho \cdot \sqrt{\lambda_i} \cdot \mathbf{p}^i \tag{14}$$

and the plus move is

$$\mathbf{x^t} = \mathbf{x}_k + \frac{\rho}{2} \cdot \sqrt{\lambda_i} \cdot \mathbf{p}^i. \tag{15}$$

We may express the same concept in terms of GPS notation using the example in two dimensions of Section 2.1. The proposed PS in two dimensions ($n = 2$) is characterised by the basis matrix

$$\mathbf{P} = \begin{pmatrix} \mathbf{p}^1 & \mathbf{p}^2 \end{pmatrix}$$

and the generating matrix $\mathbf{Q}_k$

$$\mathbf{Q}_k = \begin{pmatrix} \frac{\sqrt{\lambda_1}}{2} & 0 & -\sqrt{\lambda_1} & 0 & \frac{\sqrt{\lambda_1}}{2} & \frac{\sqrt{\lambda_1}}{2} & -\sqrt{\lambda_1} & -\sqrt{\lambda_1} & 0 \\ 0 & \frac{\sqrt{\lambda_2}}{2} & 0 & -\sqrt{\lambda_2} & \frac{\sqrt{\lambda_2}}{2} & -\sqrt{\lambda_2} & -\sqrt{\lambda_2} & \frac{\sqrt{\lambda_2}}{2} & 0 \end{pmatrix}$$

and $\Delta_k = \rho$. The trial step would be

$$\mathbf{s}_k = \Delta_k \mathbf{P}\mathbf{q}_k^i, \tag{16}$$

where $\Delta_k = \rho$ and $\mathbf{q}_k^i$ is the i^{th} column of the matrix $\mathbf{Q}_k$. We may easily verify that, by combining the moves in Eqs. (14) and (15), all the potential $\Delta_k \mathbf{P}\mathbf{q}_k^i$ can be generated.

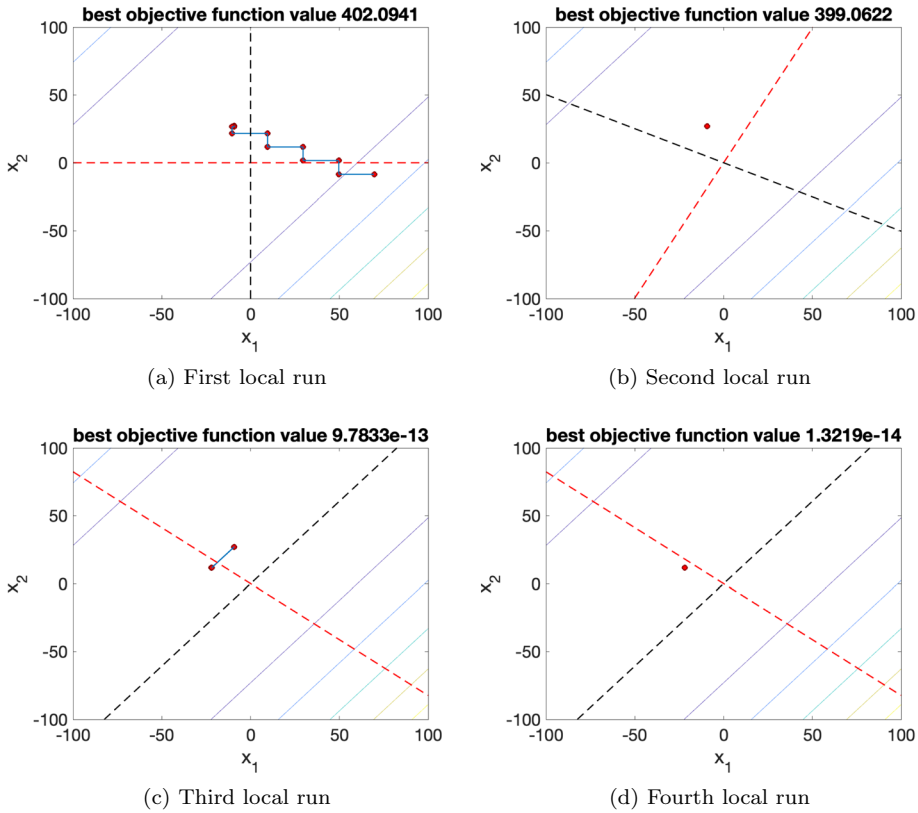
The main parameters of GPSRFLA are reported in the following.

f	objective function
$\mathbf{x}$	candidate solution
$\mathbf{x^t}$	trial solution
n_s	number of samples for the analysis
k_v	since of the space where points are sampled
$\mathbf{V}$	data set for the analysis (n_v its number of rows)
$\mathbf{C}$	covariance of the points in $\mathbf{V}$
$\mathbf{P}$	eigenvector (basis) matrix of $\mathbf{C}$
λ_i	eigenvalue of $\mathbf{C}$
$\mathbf{Q}_k$	generating matrix
$\mathbf{P}_k$	pattern matrix
ρ	radius

Algorithm 5 shows the pseudocode of the proposed GPS designed on the basis of the fitness landscape analysis.

Algorithm 5 GPS with a pattern calculated by the fitness landscape analysis.

INPUT current best solution $\mathbf{x}$, eigenvectors of the covariance matrix $\mathbf{P} = (\mathbf{p}^1, \mathbf{p}^2, \ldots \mathbf{p}^n)$, eigenvalues of the covariance matrix $\lambda_1, \lambda_2, \ldots \lambda_n$
$k \leftarrow 1$
$\mathbf{x}_k \leftarrow \mathbf{x}$
while local budget condition **do**
  $h \leftarrow k$
  for $i = 1 : n$ **do**
    $\mathbf{x^t} \leftarrow \mathbf{x}_k - \rho \cdot \sqrt{\lambda_i} \cdot \mathbf{p}^i$
    if $f(\mathbf{x^t}) \leq f(\mathbf{x}_k)$ **then**
      $k \leftarrow k + 1$
      $\mathbf{x}_k \leftarrow \mathbf{x^t}$
    else
      $\mathbf{x^t} \leftarrow \mathbf{x}_k + \frac{\rho}{2} \cdot \sqrt{\lambda_i} \cdot \mathbf{p}^i$
      if $f(\mathbf{x^t}) \leq f(\mathbf{x}_k)$ **then**
        $k \leftarrow k + 1$
        $\mathbf{x}_k \leftarrow \mathbf{x^t}$
      end if
    end if
  end for
  if $h = k$ #If no improvement occurred **then**
    $\rho \leftarrow \frac{\rho}{2}$
  end if
end while
$\mathbf{x} \leftarrow \mathbf{x}_k$
RETURN $\mathbf{x}$



(a) First local run (b) Second local run

(c) Third local run (d) Fourth local run

Fig. 4 Trajectory of ACPS in four consecutive local runs on the rotated and shifted ellipsoid. The red dots are current best solutions and the trajectory of the ACPS is shown as a blue solid line. The black and red dashed lines indicate the eigenvectors. The best objective function values are at the top of each figure

A Computationally Efficient Instance of GPSRFLA

A characterising feature of algorithms based on fitness landscape analysis is the requirement of objective function calls, which impacts the budget of the optimiser. This section presents a PS implementation that, although fits within the GPSRFLA framework in Algorithm 3, fills the data structure $\mathbf{V}$ with the points visited during the search.

More specifically, the data structure $\mathbf{V}$ is an output of the GPS and an input for the fitness landscape analysis. In the first local run, the basis matrix $\mathbf{B}$ is initialised to the identity matrix $\mathbf{I}$. GPS moves along the directions of the variables as shown in Algorithm 2. During each local run, GPS saves all the successful trial points in the data structure $\mathbf{V}$, i.e., all the points that have been current best points during the local run. The filled data structure $\mathbf{V}$ is then passed to the fitness landscape analysis component, which calculates the covariance matrix $\mathbf{C}$ and its eigenvector matrix $\mathbf{P} = \left(\mathbf{p}^1, \mathbf{p}^2, \ldots, \mathbf{p}^n\right)$. The matrix $\mathbf{P}$ is then used as the basis matrix $\mathbf{B}$ for the following local run. It must be observed that, at each restart, the data structure $\mathbf{V}$ contains the points below a threshold identified by $f\left(\mathbf{x}_k\right)$ with $\mathbf{x}_k$ starting point of that local run.

Algorithm 6 illustrates the resulting algorithm, ACPS, which was presented for the first time in [28].

Table 1 Objective functions used in this study

Domain	
$[-100, 100]^n$	
Objective function values for the minima of all the functions: 0	
Shift and Rotation	
INPUT x	
$\mathbf{z} \leftarrow \mathbf{R}(\mathbf{x} - \mathbf{o})$	
Function name	Function calculation
Sphere	$f_1 \leftarrow \sum_{i=1}^{n} z_i^2$
Ellipsoid 1	$f_2 \leftarrow \sum_{i=1}^{n} 50\left(i^2 z_i\right)^2$
Ellipsoid 2	$f_3 \leftarrow \sum_{i=1}^{n} \left(10^6\right)^{\frac{i-1}{n-1}} z_i^2$
Bent cigar	$f_4 \leftarrow z_1^2 + 10^6 \sum_{i=2}^{n} z_i^2$
Modified bent cigar	$f_5 \leftarrow z_1^2 + 10^6 \left(\sum_{i=2}^{n} z_i\right)^2$
Discus	$f_6 \leftarrow 10^6 z_1^2 + \sum_{i=2}^{n} z_i^2$
Modified discus	$f_7 \leftarrow 10^6 z_1^2 + \left(\sum_{i=2}^{n} z_i\right)^2$
Sum of powers	$f_8 \leftarrow \sqrt{\sum_{i=1}^{n} \lvert z_i \rvert^{\left(2+4\frac{i-1}{n-1}\right)}}$
Schwefel 2.21	$f_9 \leftarrow \max_{i=1,\ldots,n} \lvert z_i \rvert$
Rosenbrock	$f_{10} \leftarrow \sum_{i=1}^{n-1} \left(100\left(z_i^2 - z_{i+1}\right)^2 + \left(z_i - 1\right)^2\right)$
Rastrigin	$f_{11} \leftarrow 10n + \sum_{i=1}^{n} \left(z_i^2 - 10\cos\left(2\pi z_i\right)\right)$

Algorithm 6 Adaptive Covariance Pattern Search.

while Condition on the total budget t_b **do**
  $k, l \leftarrow 1$
  $\mathbf{x}_k \leftarrow \mathbf{x}$
  % *Fitness Landscape Analysis*
  INPUT current best solution $\mathbf{x}$ and data structure $\mathbf{V}$ (at the first local run $\mathbf{P} \leftarrow$ identity matrix $\mathbf{I}$ of size n)
  if the data structure $\mathbf{V}$ contains at least 3 vectors **then**
    Calculate the covariance matrix $\mathbf{C}$ from the samples in $\mathbf{V}$
    Apply Cholesky factorisation to extract the eigenvectors $\mathbf{P} = (\mathbf{p}^1, \mathbf{p}^2, \ldots, \mathbf{p}^n)$
  end if
  OUTPUT basis matrix $\mathbf{P}$
  % *Generalised Pattern Search*
  INPUT basis matrix $\mathbf{P}$ and current best solution $\mathbf{x}$
  while Condition on the local budget l_b **do**
    $h \leftarrow k$
    for $i = 1 : n$ **do**
      $\mathbf{x}^{\mathbf{t}} \leftarrow \mathbf{x}_k - \rho \cdot \mathbf{p}^i$
      if $f\left(\mathbf{x}^{\mathbf{t}}\right) \leq f\left(\mathbf{x}_k\right)$ **then**
        $k \leftarrow k + 1$
        $\mathbf{x}_k \leftarrow \mathbf{x}^{\mathbf{t}}$
        $\mathbf{V}(l) \leftarrow \mathbf{x}^{\mathbf{t}}$
        $l \leftarrow l + 1$
      else
        $\mathbf{x}^{\mathbf{t}} \leftarrow \mathbf{x}_k + \frac{\rho}{2} \cdot \mathbf{p}^i$
        if $f\left(\mathbf{x}^{\mathbf{t}}\right) \leq f\left(\mathbf{x}_k\right)$ **then**
          $k \leftarrow k + 1$
          $\mathbf{x}_k \leftarrow \mathbf{x}^{\mathbf{t}}$
          $\mathbf{V}(l) \leftarrow \mathbf{x}^{\mathbf{t}}$
          $l \leftarrow l + 1$
        end if
      end if
    end for
    if $h = k$ #If no improvement occurred **then**
      $\rho \leftarrow \frac{\rho}{2}$
    end if
  end while
  $\mathbf{x} \leftarrow \mathbf{x}_k$
  OUTPUT x
  Initialise the radius ρ to its initial value and $\mathbf{V}$ to an empty data structure
end while

The main advantage of the ACPS implementation in Algorithm 6 is that no extra objective function calls are required to perform the fitness landscape analysis. As shown above, ACPS uses the points visited during the search to perform the fitness landscape analysis. This means that, unlike the general GPSRFLA framework, ACPS may use the entire computational budget to optimise the objective function. ACPS uses some of these objective function calls to progressively analyse and learn the fitness landscape, while it refines the adaptation of the pattern matrix $\mathbf{P}_k$ to the optimisation problem.

To illustrate the functioning of the proposed ACPS, Fig. 4 shows the trajectory of the algorithm in four consecutive local runs. With the term *trajectory*, we mean the current best solutions visited by ACPS. Figure 4 refers to the shifted and rotated ellipsoid in two dimensions

INPUTx

$\mathbf{z} \leftarrow \mathbf{R}(\mathbf{x} - \mathbf{o})$

$$f \leftarrow \sum_{i=1}^{2} \left(10^6\right)^{\frac{i-1}{1}} z_i^2,$$

where the shift vector is

$$\mathbf{o} = \begin{pmatrix} -21.98 \\ 11.55 \end{pmatrix}$$

and the rotation matrix is

$$\mathbf{R} = \begin{pmatrix} -0.6358 & -0.7718 \\ -0.7718 & 0.6358 \end{pmatrix}.$$

Table 2 Thresholds *thre* for CPS in 10, 30, and 50 dimensions as reported in [31]

n	f_1	f_2	f_3	f_4	f_5	f_6	f_7	f_8	f_9	f_{10}	f_{11}
10	10^4	10^9	$5 \cdot 10^8$	$2 \cdot 10^{10}$	10^9	$2 \cdot 10^6$	10^6	10^4	10^2	$5 \cdot 10^9$	$3 \cdot 10^4$
30	$8 \cdot 10^4$	10^{12}	$2 \cdot 10^9$	10^{11}	10^9	10^6	10^6	$5 \cdot 10^5$	$1.5 \cdot 10^2$	$5 \cdot 10^{10}$	10^5
50	10^5	$5 \cdot 10^{13}$	$5 \cdot 10^9$	$2 \cdot 10^{11}$	10^{10}	$2 \cdot 10^7$	10^7	10^6	$1.5 \cdot 10^2$	10^{11}	$2 \cdot 10^5$

Table 3 Average error avg ± standard deviation σ over 51 runs for the problems listed in Table 1: PS according to the implementation in [41], CPS proposed in [31], and the implementation of GPSRFLA, as shown in Algorithms 3, 4, and 5, ACPS, as shown in Algorithm 6. GPSRFLA and ACPS, respectively, are taken as references for Wilcoxon for the comparison against PS and CPS. ACPS is taken as reference for Wilcoxon for comparison against GPSRFLA

	PS			CPS			GPSRFLA			ACPS	
	avg	σ	W	avg	σ	W	avg	σ	W	avg	σ
10 dimensions											
f_1	0.0000e+00	0.0000e+00	− =	7.2365e-29	9.6235e-29	= +	4.0871e-29	1.0783e-28	+	**0.0000e+00**	0.0000e+00
f_2	5.2230e+02	7.8797e+02	++	1.5810e-01	5.5370e-01	++	2.3275e-10	8.0699e-10	+	**4.4034e-23**	6.9361e-23
f_3	1.9218e+04	1.2130e+04	++	1.0352e+04	2.4610e+04	++	2.2942e+02	3.1479e+02	+	**7.1972e-16**	1.7443e-15
f_4	9.5270e+03	4.8450e+03	= +	2.0252e+04	1.3731e+04	++	7.8838e+03	5.4318e+03	+	**2.5260e-14**	1.1648e-13
f_5	1.7278e+04	1.3397e+04	++	3.4509e-06	5.9736e-06	++	2.1344e-20	3.5889e-20	+	**3.3716e-23**	1.2212e-22
f_6	5.5245e+04	1.2603e+04	++	1.9279e-12	2.4744e-12	+−	**1.3991e-23**	2.5838e-23	−	8.8372e-11	4.7050e-10
f_7	1.3171e+04	9.5882e+03	++	5.2635e-19	1.1885e-18	++	1.3327e-25	2.5916e-25	+	**3.0987e-27**	1.1885e-18
f_8	7.1750e-05	1.1167e-05	++	1.7431e-05	5.0694e-06	++	1.9394e-07	7.7596e-08	+	**4.1666e-08**	6.3467e-08
f_9	7.8371e+00	9.3846e+00	= +	5.7996e+00	8.9728e+00	= +	4.3593e+00	5.3730e+00	+	**1.4098e-06**	7.6446e-06
f_{10}	2.2174e+03	3.9690e+03	++	1.1752e+02	2.8670e+02	++	2.8872e+01	7.2745e+01	+	**5.3985e-27**	2.5363e-27
f_{11}	6.8320e+01	3.7388e+01	= +	6.4207e+01	3.0468e+01	==	7.0120e+01	3.6781e+01	+	**5.7508e+01**	2.5321e+01
30 dimensions											
f_1	5.0200e-31	3.9800e-31	+−	5.5283e-28	1.3526e-28	= +	5.7515e-28	3.5842e-28	+	**0.0000e+00**	0.0000e+00
f_2	2.9323e+06	1.9394e+06	++	7.1932e+05	9.4686e+05	++	3.4641e+05	1.9043e+05	+	**5.7209e-05**	1.7248e-04
f_3	7.2190e+04	3.0511e+04	++	1.0276e+05	7.1458e+04	= +	8.0714e+03	4.4338e+03	+	**8.8106e-07**	2.7855e-06
f_4	4.7509e+03	3.4985e+02	++	3.7216e+03	4.2577e+03	= +	3.3240e+03	1.8638e+03	+	**9.4800e-02**	2.8220e-01
f_5	1.6458e+04	2.4452e+04	++	1.8790e-09	5.9420e-09	−+	8.5268e-21	2.0312e-20	+	**6.2125e-24**	1.9636e-23
f_6	1.0889e+05	3.2826e+04	++	1.2576e+02	1.4733e+02	+ =	**8.4566e-24**	7.5654e-24	−	1.6064e+02	3.4634e+02
f_7	8.3226e-06	1.5585e-05	++	9.7797e-27	1.1411e-26	==	2.2606e-26	4.3771e-26	+	**5.3015e-27**	5.8250e-27
f_8	7.6628e-05	9.0514e-06	++	9.2850e-05	1.8269e-05	++	1.4546e-06	3.2435e-07	+	**9.2440e-07**	5.7487e-07
f_9	5.6482e+01	9.7784e+00	= +	4.2458e+01	1.1286e+01	= +	4.1489e+01	8.6932e+00	+	**2.2920e+00**	2.0570e+00
f_{10}	8.1998e+01	1.4070e+02	++	3.7031e+00	2.0897e+00	−+	1.8508e+02	3.7295e+02	+	**3.9870e-01**	1.2607e+00
f_{11}	3.8423e+02	9.7231e+01	= +	3.9956e+02	1.2392e+02	++	3.4507e+02	1.1963e+02	+	**2.7569e+02**	8.9984e+01
50 dimensions											
f_1	6.5738e-31	7.1289e-31	−+	1.2148e-27	6.1144e-28	= +	1.5846e-27	5.7601e-28	+	**0.0000e+00**	0.0000e+00
f_2	1.9664e+07	5.0980e+06	= +	3.5727e+07	1.9921e+07	++	1.8067e+07	9.5292e+06	+	**3.9667e+05**	3.6613e+05
f_3	1.9085e+05	4.0141e+04	++	1.9244e+05	6.5138e+04	++	4.9504e+04	2.1671e+04	+	**1.3150e+02**	3.5391e+02
f_4	1.1906e+04	2.1764e+04	= +	4.6930e+03	5.8099e+03	−+	2.9854e+04	1.9311e+04	+	**1.6111e+01**	3.273e+01
f_5	1.1034e+05	6.9674e+04	++	4.700e-03	1.4700e-02	++	8.6384e-22	1.7318e-21	+	**1.6298e-31**	4.3564e-31
f_6	1.6360e+05	4.9495e+04	++	1.2573e+02	1.2764e+02	+ =	**7.2388e-24**	1.8602e-24	−	1.0434e+02	1.3584e+02
f_7	2.6739e-11	3.3330e-11	++	1.3449e-26	2.7367e-26	==	6.2186e-27	9.6092e-27	=	**1.3501e-27**	1.8298e-27
f_8	9.4485e-05	1.2829e-05	++	1.3238e-04	1.4419e-05	++	3.2769e-06	5.0659e-07	+	**2.3891e-06**	1.2608e-06
f_9	8.1844e+01	7.1066e+00	= +	6.3083e+01	1.2169e+01	= +	6.7138e+01	1.3981e+01	+	**1.8681e+01**	9.8691e+00
f_{10}	3.0848e+02	4.7580e+02	= +	1.0012e+02	2.3364e+02	++	3.8797e+01	9.3072e+01	+	**7.9730e-01**	1.6809e+00
f_{11}	7.4604e+02	1.7833e+02	= +	8.2749e+02	2.0231e+02	= +	8.6012e+02	3.5563e+02	+	**6.2442e+01**	1.9581e+01

Best results are presented in bold

A random point $\mathbf{x}$ has been sampled within the domain. The objective function value of this starting point is $7.4385e+09$.

Figure 4 shows that, in the first local run of the algorithm, while moving along the directions of the variables (black and red dashed lines), it approaches the optimum but still remains far from it. After the restart, ACPS uses the new search directions, i.e., the eigenvectors of the covariance matrix of the distribution of samples collected during the first local run. This system of reference appears to be ineffective. We may observe that, during the second local run only, a marginal improvement is achieved. However, the budget spent in the second local run is not wasted; the points sampled during the second local run enable the detection of an effective reference system (eigenvectors in the third local run). During the third local run, ACPS exploits the benefits of the fitness landscape analysis and quickly detects a solution close to the optimum. The results are then refined in the fourth local run where the eigenvectors are slightly corrected.

It must be observed that the proposed ACPS resembles the Rosenbrock method [37] as both use a basis of vector that is progressively adapted during the run (the Rosenbrock Method belongs to the GPS family). However, the two algorithms are radically different in terms of the way the basis is selected and updated. More specifically, while ACPS makes use of the eigenvectors of the covariance matrix of a set of samples, the Rosenbrock Algorithm stores the successful moves and determines a new orthonormal basis guided by previous successful moves.

It must be remarked that, although ACPS can be considered an instance of the GPSRFLA framework, it does not make use of the eigenvalues to update the generating matrix $\mathbf{G}_k$. This decision has been made considering the preliminary results that we obtained. The data structure $\mathbf{V}$ is likely to obtain a small number of points. On one hand, these points are enough to correctly estimate, through the eigenvectors of the covariance matrix associated with them, the directions with maximum and minimum directional derivatives (multiple steps as illustrated in Fig. 4). On the other hand, these points are usually not enough to correctly estimate the values of the directional derivatives through the calculation of the eigenvalues. Since these wrong estimations tend to jeopardise the performance of the algorithm, it was decided not to exclude the update of the generating matrix $\mathbf{G}_k$ from ACPS.

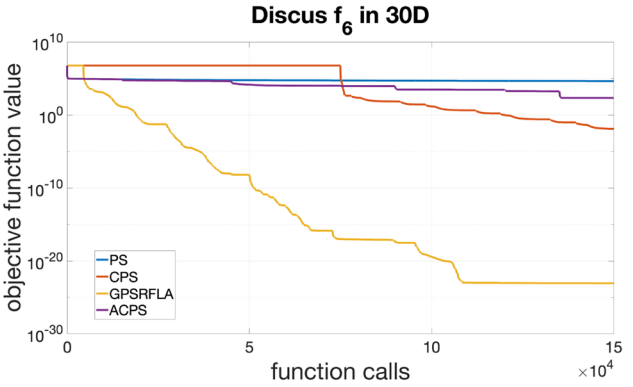


Fig. 6 Performance trend (logarithmic scale) of the PS variants in this study for the discus f_6 in 30 dimensions

Numerical Results

To test and compare the performance of the GPSRFLA framework and ACPS, a set of functions from the IEEE CEC2013 benchmark [20] was selected and adapted. Since PS is a local search, we selected all the unimodal problems, hence reproducing the testbed of CPS used in [30]. We also reproduced both the versions of ellipsoid presented in [30] (f_2 and f_3). The condition number of these two ellipsoids

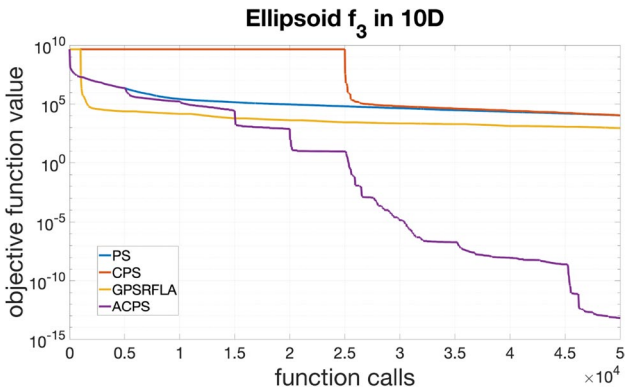


Fig. 5 Performance trend (logarithmic scale) of the PS variants in this study for the ellipsoid f_3 in 10 dimensions

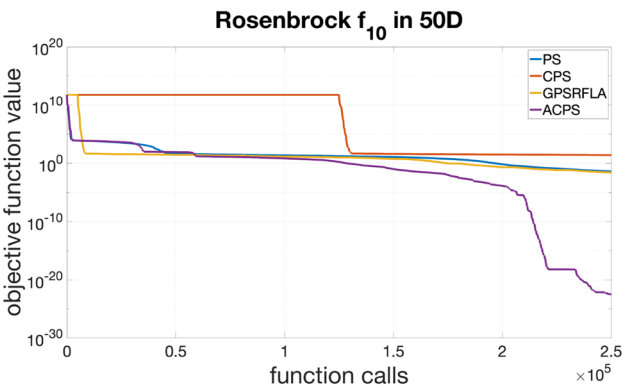


Fig. 7 Performance trend (logarithmic scale) of the PS variants in this study for Rosenbrock f_{10} in 50 dimensions

Table 4 Average error avg ± standard deviation σ over 51 runs for the problems listed in Table 1: BFGS algorithm [11], CMAES [14], the implementation of GPSRFLA as shown in Algorithms 3, 4, and 5, ACPS, as shown in Algorithm 6. GPSRFLA and ACPS, respectively, are taken as references for Wilcoxon for the comparison against PS and CPS. ACPS is taken as reference for Wilcoxon for the comparison against GPSRFLA

	BFGS			CMAES			GPSRFLA			ACPS	
	avg	σ	W	avg	σ	W	avg	σ	W	avg	σ
10 dimensions											
f_1	1.8757e-20	2.4576e-21	++	1.6841e-15	1.2496e-15	++	4.0871e-29	1.0783e-28	+	**0.0000e+00**	0.0000e+00
f_2	1.8956e-13	4.1016e-14	++	1.5739e-15	1.1095e-15	++	2.3275e-10	8.0699e-10	+	**4.4034e-23**	6.9361e-23
f_3	6.6365e-11	2.6183e-12	−+	1.7152e-15	1.0734e-15	−+	2.2942e+02	3.1479e+02	+	**7.1972e-16**	1.7443e-15
f_4	4.8479e-01	1.6494e+00	−+	**1.4131e-15**	1.1850e-15	−−	7.8838e+03	5.4318e+03	+	2.5260e-14	1.1648e-13
f_5	1.2818e-09	2.0603e-09	++	8.8699e-15	1.0347e-14	++	2.1344e-20	3.5889e-20	+	**3.3716e-23**	1.2212e-22
f_6	3.2306e-10	5.0825e-11	++	1.3737e-15	1.3659e-15	+ =	**1.3991e-23**	2.5838e-23	−	8.8372e-11	4.7050e-10
f_7	3.9439e-12	7.2576e-12	++	6.6387e-15	1.0289e-14	++	1.3327e-25	2.5916e-25	+	**3.0987e-27**	1.1885e-18
f_8	1.1967e-07	4.3589e-07	= +	**9.8002e-13**	8.9374e-13	−−	1.9394e-07	7.7596e-08	+	4.1666e-08	6.3467e-08
f_9	7.1074e+01	2.9732e+01	++	**7.7677e-10**	2.1798e-10	−−	4.3593e+00	5.3730e+00	+	1.4098e-06	7.6446e-06
f_{10}	1.1960e+00	1.8581e+00	−+	7.973e-01	1.62193+00	−+	2.8872e+01	7.2745e+01	+	**5.3985e-27**	2.5363e-27
f_{11}	8.7917e+02	4.2825e+02	++	**1.6069e+01**	1.6267e+01	−−	7.0120e+01	3.6781e+01	+	5.7508e+01	2.5321e+01
30 dimensions											
f_1	4.3857e-20	1.6043e-20	++	1.4429e-15	1.0251e-15	++	5.7515e-28	3.5842e-28	+	**0.0000e+00**	0.0000e+00
f_2	1.3352e-09	3.3772e-10	−−	**2.9032e-15**	5.7236e-16	−−	3.4641e+05	1.9043e+05	+	5.7209e-05	1.7248e-04
f_3	3.2738e-11	1.5479e-12	−−	**2.0379e-15**	2.0379e-15	−−	8.0714e+03	4.4338e+03	+	8.8106e-07	2.7855e-06
f_4	4.5544e+00	2.4117e+01	−+	**1.3962e-15**	4.6239e-16	−−	3.3240e+03	1.8638e+03	+	9.4800e-02	2.8220e-01
f_5	1.8926e-08	2.5487e-08	++	5.9103e+03	1.5913e+04	++	8.5268e-21	2.0312e-20	+	**6.2125e-24**	1.9636e-23
f_6	1.5200e-10	1.9637e-11	+−	4.1546e-15	2.3165e-15	+−	**8.4566e-24**	7.5654e-24	−	1.6064e+02	3.4634e+02
f_7	5.9737e-13	1.2614e-12	++	1.0770e-14	1.2033e-14	++	2.2606e-26	4.3771e-26	=	**5.3015e-27**	5.8250e-27
f_8	1.2390e-05	2.84323e-06	++	**1.5671e-11**	2.0251e-11	−−	1.4546e-06	3.2435e-07	+	9.2440e-07	5.7487e-07
f_9	1.1706e+02	3.2399e+01	++	**1.3042e+00**	2.8263e+00	− =	4.1489e+01	8.6932e+00	+	2.2920e+00	2.0570e+00
f_{10}	1.7275e+00	2.0093e+00	−+	7.9730e-01	1.6809e+00	−+	1.8508e+02	3.7295e+02	+	**3.9870e-01**	1.2607e+00
f_{11}	2.0621e+03	6.1136e+02	++	**5.0942e+01**	1.0997e+01	−−	3.4507e+02	1.1963e+02	+	2.7569e+02	8.9984e+01
50 dimensions											
f_1	8.5482e-20	1.9014e-20	++	1.1890e-15	3.3954e-16	++	1.5846e-27	5.7601e-28	+	**0.0000e+00**	0.0000e+00
f_2	1.5972e-07	3.7724e-08	−−	**2.1614e-14**	4.5515e-15	−−	1.8067e+07	9.5292e+06	+	3.9667e+05	3.6613e+05
f_3	9.2288e-011	4.6933e-12	−−	**1.3092e-15**	5.9487e-16	−−	4.9504e+04	2.1671e+04	+	1.3150e+02	3.5391e+02
f_4	9.8654e-01	3.4845e+e00	−−	**1.2358e-15**	5.8376e-16	−−	2.9854e+04	1.9311e+04	+	1.6111e+01	3.273e+01
f_5	1.9741e-08	2.8660e-08	++	5.2536e+04	1.5800e+05	++	8.6384e-22	1.7318e-21	+	**1.6298e-31**	4.3564e-31
f_6	1.1937e-10	7.3146e-12	+−	3.0884e+04	9.7663e+04	++	**7.2388e-24**	1.8602e-24	−	1.0434e+02	1.3584e+02
f_7	1.5341e-13	3.2062e-13	++	1.8735e+00	4.4116e+00	++	6.2186e-27	9.6092e-27	=	**1.3501e-27**	1.8298e-27
f_8	1.1253e-05	1.8968e-06	++	**6.5703e-11**	3.2925e-11	−−	3.2769e-06	5.0659e-07	+	2.3891e-06	1.2608e-06
f_9	1.2855e+02	2.7862e+01	++	**9.6761e-06**	2.8014e-05	−−	6.7138e+01	1.3981e+01	+	1.8681e+01	9.8691e+00
f_{10}	1.9933e+00	2.0274e+00	−+	1.1960e+00	1.9257e+00	−+	3.8797e+01	9.3072e+01	+	**7.9730e-01**	1.6809e+00
f_{11}	3.3820e+03	1.0348e+03	++	1.1004e+02	1.3796e+01	−+	8.6012e+02	3.5563e+02	+	**6.2442e+01**	1.9581e+01

Best results are presented in bold

worsens with dimensionality at different speeds. In this paper, alongside bent cigar and discus, we included their modified versions.

Finally, to show that GPSRFLA is capable, to some extent, at handling multimodal fitness landscapes, we included two simple multimodal functions from [20]. The list of the functions used in this study is displayed in Table 1. As shown in Table 1, each problem has been shifted and rotated; the vector $\mathbf{x}$ is transformed into $\mathbf{z}$. The shift vector $\mathbf{o}$ of [20] has been used. The rotation matrices $\mathbf{R}$ have been

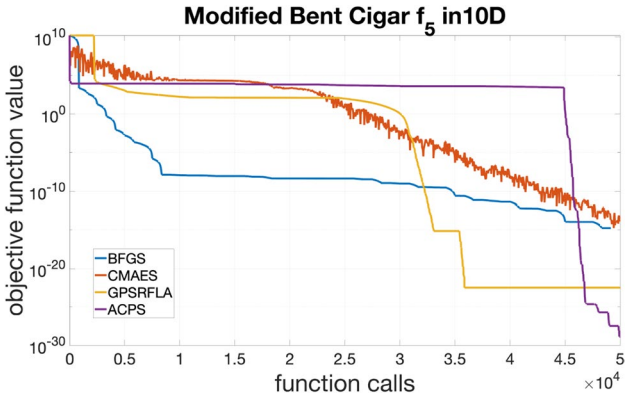


Fig. 8 Performance trends (logarithmic scale) of BFGS, CMAES, GPSRFLA, and ACPS for the modified bent cigar f_5 in 10 dimensions

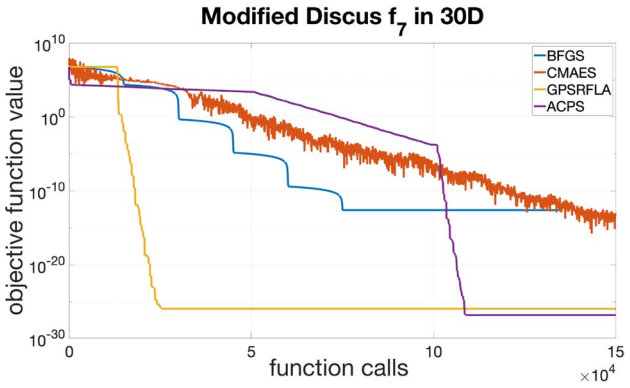


Fig. 9 Performance trends (logarithmic scale) of BFGS, CMAES, GPSRFLA, and ACPS for the modified discus f_7 in 30 dimensions

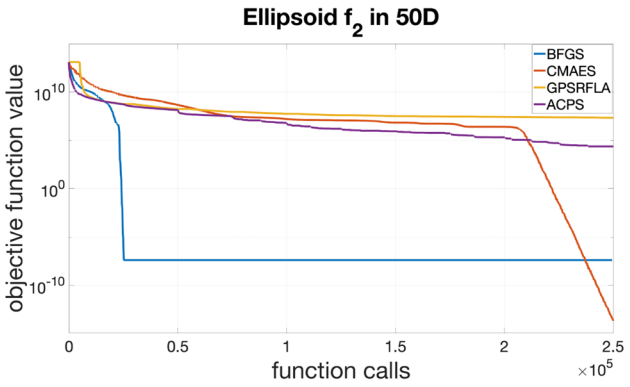


Fig. 10 Performance trends (logarithmic scale) of BFGS, CMAES, GPSRFLA, and ACPS for ellipsoid f_2 in 50 dimensions

randomly generated. One matrix $\mathbf{Q}$ has been generated for each problem and dimensionality value.

The results are divided into the following categories:

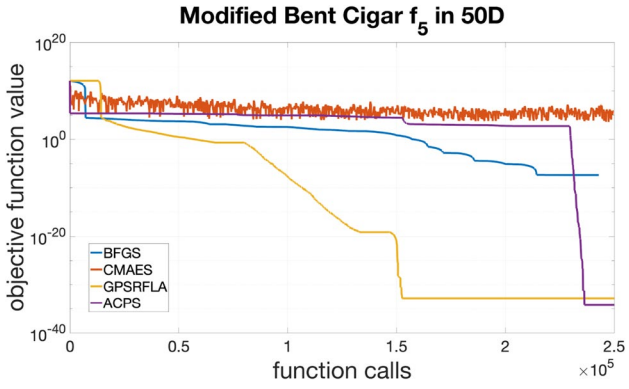


Fig. 11 Performance trends (logarithmic scale) of BFGS, CMAES, GPSRFLA, and ACPS for modified bent cigar f_5 in 50 dimensions

– Comparison among PS algorithms.
– Comparison against other algorithms.

The problems in Table 1 have been considered in 10, 30, and 50 dimensions. For each problem in Table 1, each dimensionality level, and each algorithm in this study, 51 independent runs were performed. For each run, the algorithms under consideration, if single solution, have been run with the same initial solution. All the algorithms in this paper have been executed with a budget of $10000 \cdot n$ function calls, where n is the problem dimensionality. The results for each algorithm and problem are expressed in terms of mean value and $\pm$ standard deviation over the 51 independent runs performed. Furthermore, to statistically investigate whether the application of the proposed method results in performance gain, the Wilcoxon rank-sum test was applied; see [12]. In the tables in this section, a "+" indicates that the proposed algorithm (GPSRFLA/ACPS) significantly outperformed its competitor, a "−" indicates that the competitor significantly outperformed the proposed algorithm, and a "=" indicates that there is no significant difference in performance.

Comparison among Pattern Search Algorithms

This section highlights the benefits of fitness landscape analysis on PS algorithms. To achieve this aim, the following algorithms have been tested on the problems in Table 1:

– the original PS according to the implementation in [41] and the implementation Algorithm 2
– CPS presented in [30, 31]
– GPSRFLA according to the framework in Algorithm 3, the analysis component in Algorithm 4 and the GPS implementation in Algorithm 5
– ACPS [28] as reported in Algorithm 6

Table 5 Holm–Bonferroni Procedure with ACPS as a reference (Rank 5.1515e+00)

	R_j	z_j	p_j	$\frac{\delta_j}{j}$	Test
CMAES	4.6061e+00	−1.1843e+00	2.3629e-01	5.0000e-02	Failed to Reject
GPSRFLA	3.6667e+00	−3.2240e+00	1.2643e-03	2.5000e-02	Rejected
BFGS	3.1818e+00	−4.2767e+00	1.8970e-05	1.6667e-02	Rejected
CPS	2.7273e+00	−5.2636e+00	1.4125e-07	1.2500e-02	Rejected
PS	2.0000e+00	−6.8427e+00	7.7716e-12	1.0000e-02	Rejected

All PS variants in this article have been run with the initial radius $\rho = 0.1 \cdot$ domain width $= 20$. This parameter has been set using the indication in [41] and then tuning for our testbed.

As reported in [30], the budget of CPS has been split in two parts: $5000 \cdot n$ function calls have been used to build the covariance matrix $\mathbf{C}$, while $5000 \cdot n$ function calls have been spent to execute the algorithm along the directions of its eigenvectors.

The threshold *thre* for the problems in Table 1 are reported in Table 2. The threshold values were set empirically by testing values of the codomain that allowed for some points to be stored in the data structure $\mathbf{V}$, while some others were discarded; see [31].

GPSRFLA uses a local budget of $l_b = 1000 \cdot n$ objective function calls, $n_s = 200 \cdot n$ slots (and thus GPS is run with a budget of $800 \cdot n$ objective function calls), $n_v = 5 \cdot n$ slots, and $k_v = 100$. GPS is also stopped if $\rho \leq 10^{-15}$.

ACPS is run with a maximum local budget $l_b = 1000 \cdot n$ and is stopped if $\rho \leq 10^{-15}$.

Table 3 shows the numerical results of the four PS variants. The best results for each problem are highlighted in bold.

The results in Table 3 show that both GPSRFLA and ACPS significantly outperform PS in the vast majority of cases. Only for the sphere function, f_1 PS have an excellent performance. This is due to the fact that the identity matrix is already the ideal choice of basis matrix. For all other problems, PS either performs slightly worse or much worse than GPSRFLA and ACPS. The comparisons of GPSRFLA and ACPS against CPS also show that the proposed algorithms outperform their predecessors for almost all problems considered. This suggests that the restarting fitness landscape analysis logic is beneficial to the performance of the algorithm. Finally, the comparison of ACPS to GPSRFLA shows that, in most cases, the computational efficient logic embedded in ACPS yields significantly better results than GPSRFLA, which uses part of the budget to solely analyse the problem. However, the exploitation of the information regarding the directional gradients is potentially very powerful, as shown in the case of the rotated discus function f_6.

Three examples of performance trends for the variants of PS included in this study are illustrated in Figs. 5, 6, and 7.

Comparison Against Other Algorithms

We have compared GPSRFLA and ACPS against the following two algorithms:

- Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm [11] with an estimation of the gradient that may make it applicable to black-box problems;
- Covariance Matrix Adaptive Evolution Strategy (CMAES) [14].

The motivations behind these two competitors are as follows: (1) BFGS is a Quasi–Newtonian (i.e., based on a solid theoretical foundation) algorithm that estimates the gradient and is here used as a benchmark algorithm; and (2) CMAES is a prevalent algorithm that, like GPSRFLA and ACPS, is based on theoretical considerations of multi-variate distributions and the covariance matrix. Table 4 lists the results of this comparison (the best results are highlighted in bold).

Numerical results in Table 4 show that, on average, CMAES, GPSRFLA, and ACPS are better suited than BFGS to address the black-box problems (without information on the gradient). However, BFGS is an excellent algorithm for several problems, especially the multi-variate ellipsoid function. The results show that CMAES and GPSFRLA have almost comparable performances, with CMAES performing better that GPSFRLA on average for seven problems and worse for four problems. ACPS is more competitive with CMAES as ACPS outperforms CMAES in 16 cases out of 33, while it was outperformed in 15. The algorithms have the same performance for the remaining two problems. Overall, we may conclude that ACPS and CMAES have a comparable performance for the problems under investigation.

Some further considerations can be made regarding the scalability of the algorithms. In the low-dimensional case ($n = 10$), both the algorithms detect solutions very close to the optimum for the nine unimodal problems ($f_1 - f_9$) and detect the global optimum in several runs. They also detect a local minimum for the two multimodal problems ($f_{10} - f_{11}$). In higher dimensions, we observe that the performances of both CMAES and ACPS deteriorate for some problems and remain excellent for others. For example, CMAES performs

extremely well on $f_2 - f_4$ regardless of the number of variables, while ACPS deteriorates as the number of dimensions increases. Conversely, ACPS handles the $f_5 - f_7$ problems better than CMAES.

With reference to the results in Table 4, Figs. 8, 9, 10, and 11 show some examples of performance trends of GPSRFLA and ACPS against BFGS and CMAES. These plots confirm the reports on Table 4; at higher dimensions, CMAES and ACPS both appear to be inadequate at solving some problems but are very well suited for others. Figure 10 shows an example for which GPSRFLA and ACPS perform poorly compared to CMAES. Conversely, Fig. 11 shows an example for which both GPSRFLA and ACPS display an excellent performance, while CMAES appears to be inadequate.

Statistical Ranking via the Holm–Bonferroni Procedure

To further strengthen the statistical analysis of the presented results, we performed the Holm–Bonferroni [15] procedure for the six algorithms and 33 problems (11 objective functions ×3 levels of dimensionality) under investigation. The results of the Holm–Bonferroni procedure are presented in Table 5. A score R_j for $j = 1, \ldots, N_A$ (where N_A is the number of algorithms under analysis, $N_A = 6$, in this paper) has been assigned. The score has been assigned in the following way: for each problem, a score of 6 is assigned to the algorithm displaying the best performance, 5 is assigned to the second best, 4 to the third, and so on. For each algorithm, the scores obtained for each problem are summed up and averaged over the 33 test problems. With the calculated R_j values, ACPS has been taken as the reference algorithm. R_0 indicates the rank of ACPS and, with R_j for $j = 1, \ldots, N_A - 1$, the rank of the remaining four algorithms. Let j be the index of the algorithm. The values z_j have been calculated as

$$z_j = \frac{R_j - R_0}{\sqrt{\frac{N_A(N_A+1)}{6N_{TP}}}},$$

where N_{TP} is the number of test problems (33 in this study). By means of the z_j values, the corresponding cumulative normal distribution values p_j have been calculated; see [10]

$$p_j = \frac{2}{\sqrt{\pi}} \int_{\frac{-z_j}{\sqrt{2}}}^{\infty} e^{-t^2} dt.$$

These p_j values have then been compared with the corresponding δ/j, where δ is the level of confidence, set to 0.05 in this case. Table 5 displays the ranks, z_j values, p_j values, and corresponding δ/j obtained. Moreover, it is indicated whether the null hypothesis (which states that the two algorithms have indistinguishable performance) is 'Rejected', i.e., the algorithms have statistically different performance, or 'Failed to Reject', meaning that the test failed to assess that there is different performance (one does not outperform the other).

The results of the Holm–Bonferroni procedure in Table 5 show that ACPS achieved the highest rank. ACPS and CMAES have comparable performances, while ACPS has a better performance than the other four algorithms in this study.

Conclusion

GPS is a family of single solution deterministic algorithms that search for the optimum by moving along a set of moves stored in the pattern matrix. The choice of pattern matrix is still an open issue. This article proposes a restarting scheme where, at each restart, the pattern matrix is updated following a fitness landscape analysis.

Two algorithmic implementations encompassing the two novel contributions of this study with respect to the literature are proposed. The first is the development of a criterion to estimate the directional derivatives (steep and flat directions) through the eigenvalues of the covariance matrix associated with a sample of points, as well as a mechanism to exploit this information within the search. The second is an algorithmic mechanism that uses the objective function calls performed during the search for the purpose of the fitness landscape analysis. The latter contribution enhances the computational efficiency and consequently the performance of the algorithm.

The two proposed implementations outperform their predecessors and are competitive with a Quasi-Newtonian method and a popular high-performing metaheuristic. The second implementation proved to have remarkably good performance. While the two novel contributions of this paper appear to be separately effective, more work is required to effectively combine them with the purpose of superposing their respective benefits.

Funding This study was funded by the institutions indicated in the list of affiliations.

Declarations

Conflict of Interest The authors declare that they have no conflict of interest.

References

1. Adair J, Ochoa G, Malan KM. Local optima networks for continuous fitness landscapes. In: López-Ibáñez et al. [21], pp. 1407–1414.
2. Auger A, Teytaud O. Continuous lunches are free! In: Proceedings of the 9th annual conference on Genetic and evolutionary computation, pp. 916–922. ACM 2007.
3. Blum C, Chiong R, Clerc M, Jong KAD, Michalewicz Z, Neri F, Weise T. Evolutionary optimization. In: R. Chiong, T. Weise, Z. Michalewicz (eds.) Variants of Evolutionary Algorithms for Real-World Applications, pp. 1–29. Springer 2012.
4. Burke EK, Hyde M, Kendall G, Ochoa G, Ozcan E, Woodward J. Classification of hyper-heuristic approaches. In: Handbook of Meta-Heuristics, pp. 449–468. Springer 2010.
5. Caponio A, Neri F, Tirronen V. Super-fit control adaptation in memetic differential evolution frameworks. Soft Comput Fusion Found Methodol Appl. 2009;13:811–31.
6. Caraffini F, Iacca G, Neri F, Picinali L, Mininno E. A CMA-ES super-fit scheme for the re-sampled inheritance search. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2013, Cancun, Mexico, June 20-23, 2013, pp. 1123–1130 2013.
7. Caraffini F, Neri F, Cheng J, Zhang G, Picinali L, Iacca G, Mininno E. Super-fit multicriteria adaptive differential evolution. In: Proceedings of the IEEE Congress on Evolutionary Computation, CEC 2013, Cancun, Mexico, June 20-23, 2013, pp. 1678–1685 2013.
8. Caraffini F, Neri F, Epitropakis MG. Hyperspam: A study on hyper-heuristic coordination strategies in the continuous domain. Inf Sci. 2019;477:186–202.
9. Caraffini F, Neri F, Picinali L. An analysis on separability for memetic computing automatic design. Inf Sci. 2014;265:1–22.
10. Fisher RA. The Design of Experiments, ninth edn. Macmillan 1971 1935.
11. Fletcher R. Practical Methods of Optimization. 2nd ed. New York: John Wiley & Sons; 1987.
12. Garcia S, Fernandez A, Luengo J, Herrera F. A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability. Soft Comput. 2008;13(10):959–77.
13. Hansen N, Ostermeier A. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In: Proceedings of the IEEE International Conference on Evolutionary Computation, pp. 312–317 1996.
14. Hansen N, Ostermeier A. Completely derandomized self-adaptation in evolution strategies. Evol Comput. 2001;9(2):159–95.
15. Holm S. A simple sequentially rejective multiple test procedure. Scand J Stat. 1979;6(2):65–70.
16. Iacca G, Mallipeddi R, Mininno E, Neri F, Suganthan PN. Super-fit and Population Size Reduction Mechanisms in Compact Differential Evolution. In: Proceedings of IEEE Symposium on Memetic Computing, pp. 21–28 2011.
17. Jana ND, Sil J, Das S. Continuous fitness landscape analysis using a chaos-based random walk algorithm. Soft Comput. 2018;22:921–48.
18. Jolliffe IT. Principal Component Analysis, Springer Series in Statistics. 2nd ed. New York: Springer; 2002.
19. Krasnogor N, Smith J. A tutorial for competent memetic algorithms: model, taxonomy, and design issues. IEEE Trans Evol Comput. 2005;9(5):474–88.
20. Liang J, Qu B, Suganthan P, Hernández-Díaz A. Problem definitions and evaluation criteria for the cec 2013 special session on real-parameter optimization 2013.
21. López-Ibáñez M, Auger A, Stützle T. (eds.): Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO 2019, Prague, Czech Republic, July 13-17, 2019. ACM 2019.
22. Malan KM. A survey of advances in landscape analysis for optimisation. Algorithms. 2021;14(2):40.
23. Malan KM, Engelbrecht AP. Quantifying ruggedness of continuous landscapes using entropy. In: 2009 IEEE Congress on Evolutionary Computation, pp. 1440–1447 2009.
24. Malan KM, Engelbrecht AP. A survey of techniques for characterising fitness landscapes and some possible ways forward. Inf Sci. 2013;241:148–63.
25. Malan KM, Engelbrecht AP. A progressive random walk algorithm for sampling continuous fitness landscapes. In: 2014 IEEE Congress on Evolutionary Computation (CEC), pp. 2507–2514 2014.
26. McGinley B, Morgan F, O'Riordan C. Maintaining diversity through adaptive selection, crossover and mutation. In: C. Ryan, M. Keijzer (eds.) Genetic and Evolutionary Computation Conference, GECCO 2008, Proceedings, Atlanta, GA, USA, July 12-16, 2008, pp. 1127–1128. ACM 2008.
27. Neri F. Linear Algebra for Computational Sciences and Engineering. 2nd ed. New York: Springer; 2019.
28. Neri F. Adaptive covariance pattern search. In: P.A. Castillo, J.L.J. Laredo (eds.) Applications of Evolutionary Computation - 24th International Conference, EvoApplications 2021, Held as Part of EvoStar 2021, Virtual Event, April 7-9, 2021, Proceedings, *Lecture Notes in Computer Science*, vol. 12694, pp. 178–193. Springer 2021.
29. Neri F. Teaching mathematics to computer scientists: Reflections and a case study. SN Comput Sci. 2021;2(2):75.
30. Neri F, Rostami S. A local search for numerical optimisation based on covariance matrix diagonalisation. In: P.A. Castillo, J.L.J. Laredo, F.F. de Vega (eds.) Applications of Evolutionary Computation - 23rd European Conference, EvoApplications 2020, Held as Part of EvoStar 2020, Seville, Spain, April 15-17, 2020, Proceedings, *Lecture Notes in Computer Science*, vol. 12104, pp. 3–19. Springer 2020.
31. Neri F, Rostami S. Generalised pattern search based on covariance matrix diagonalisation. SN Comput Sci. 2021;2(3):171.
32. Neri F, Tirronen V, Karkkainen T, Rossi T. Fitness diversity based adaptation in multimeme algorithms:a comparative study. In: 2007 IEEE Congress on Evolutionary Computation, pp. 2374–2381 2007.
33. Ochoa G, Malan K. Recent advances in fitness landscape analysis. In: López-Ibáñez et al. [21], pp. 1077–1094.
34. Ochoa G, Malan KM, Blum C. Search trajectory networks of population-based algorithms in continuous spaces. In: Applications of Evolutionary Computation - 23rd European Conference, EvoApplications 2020, Held as Part of EvoStar 2020, Seville, Spain, April 15-17, 2020, Proceedings, pp. 70–85 2020.
35. Ochoa G, Vérel S, Daolio F, Tomassini M. Clustering of local optima in combinatorial fitness landscapes. In: C.A.C. Coello (ed.) Learning and Intelligent Optimization - 5th International Conference, LION 5, Rome, Italy, January 17-21, 2011. Selected Papers, *Lecture Notes in Computer Science*, vol. 6683, pp. 454–457. Springer 2011.
36. Ong YS, Lim MH, Zhu N, Wong KW. Classification of Adaptive Memetic Algorithms: A Comparative Study. IEEE Trans Syst Man Cybern Part B. 2006;36(1):141–52.
37. Rosenbrock HH. An automatic Method for finding the greatest or least Value of a Function. Comput J. 1960;3(3):175–84.

38. Stadler PF. Fitness landscapes. In: Lässig M, Valleriani A, editors. Biological Evolution and Statistical Physics, vol. 585. Lecture Notes in Physics. Berlin/Heidelberg, Germany: Springer; 2002. p. 183–204.
39. Sun Y, Kirley M, Halgamuge SK. Quantifying variable interactions in continuous optimization problems. IEEE Trans Evol Comput. 2017;21(2):249–64.
40. Torczon V. On the convergence of pattern search algorithms. SIAM J Optim. 1997;7(1):1–25.
41. Tseng LY, Chen C. Multiple trajectory search for Large Scale Global Optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation, pp. 3052–3059 2008.
42. Wolpert DH, Macready WG. No free lunch theorems for optimization. IEEE Trans Evol Comput. 1997;1(1):67–82.