SN Computer Science (2021) 2:171
https://doi.org/10.1007/s42979-021-00513-y

ORIGINAL RESEARCH

Generalised Pattern Search Based on Covariance Matrix Diagonalisation

Ferrante Neri[1] · Shahin Rostami[2]

Received: 12 June 2020 / Accepted: 8 February 2021 / Published online: 25 March 2021

Abstract
Pattern Search is a family of gradient-free direct search methods for numerical optimisation problems. The characterising feature of pattern search methods is the use of multiple directions spanning the problem domain to sample new candidate solutions. These directions compose a matrix of potential search moves, that is the pattern. Although some fundamental studies theoretically indicate that various directions can be used, the selection of the search directions remains an unaddressed problem. The present article proposes a procedure for selecting the directions that guarantee high convergence/high performance of pattern search. The proposed procedure consists of a fitness landscape analysis to characterise the geometry of the problem by sampling points and selecting those whose objective function values are below a threshold. The eigenvectors of the covariance matrix of this distribution are then used as search directions for the pattern search. Numerical results show that the proposed method systematically outperforms its standard counterpart and is competitive with modern complex direct search and metaheuristic methods.

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

Introduction

Modern numerical optimisation problems are often complex and do not allow the application of gradient-based algorithms, see [4]. Some of the main reasons why a gradient-based approach may be infeasible or impractical are the following (see [10]):

– the first derivatives or objective functions are not available, e.g. the function is the result of a simulation or an experiment;
– the numerical approximation of the gradient is computationally expensive and leads to an unacceptable overhead;
– the objective function is noisy, e.g. coming from measurements or uncertain data, and the gradient is therefore unreliable.

In these cases, a derivative-free method may be the only feasible option [3, 10]. Derivative-free methods are algorithms that search for a null-gradient point, i.e. the closest local optimum, without calculating the gradient of the objective function.

Derivative free methods can be divided into two macro-categories:

– **model-based** methods construct a surrogate model of the objective function, e.g. by sampling points and interpolating them, and then calculating the gradient of the surrogate model to search for the optimum (some modern brilliant examples are reported in [8, 9]);
– **direct search** methods use only the objective function values to explore the domain and search for the optimum.

This article is part of the topical collection “Evolution, the New AI Revolution” guest edited by Anikó Ekárt and Anna Isabel Esparcia-Alcázar”.

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

Shahin Rostami
shahin@polyra.com

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

2 Data Science Lab, Polyra Limited, Bournemouth BH8 9JN, UK

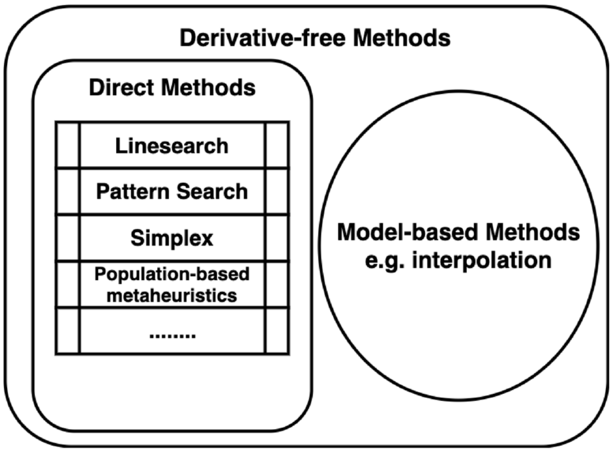


Fig. 1 Overview of gradient-free optimisation methods

Direct search methods may employ various strategies to sample and test new points [7]. Amongst the many strategies, we can consider the following to be some of the most prevalent in the literature:

- **linesearch** methods iteratively identify a unidimensional direction trans-passing the *n*-dimensional space, and then optimise the objective function along this direction, see [2];
- **pattern search** methods use multiple directions to generate a set of potential moves (a pattern) to cover the entire domain, see [23, 27];
- **simplex** methods define a geometric figure (a simplex that is the generalisation of the triangle) and makes use of its vertices to set the exploratory rules, see [37];
- **population-based metaheuristics** define a large category of methods where multiple candidate solutions are stored and recombined to sample new points. Some methods refer to an evolutionary metaphor, e.g. the classical examples of Genetic Algorithms and Evolution Strategies [13], while some others refer to the motion of animals/particles, e.g. the Particle Swarm Optimisation [28] and Differential Evolution [43] (the latter is not associated with a swarm metaphor but is often considered a swarm intelligence algorithm in terms of its operation, see [50]).

A scheme summarising this partial taxonomy is reported in Fig. 1.

Pattern search methods have been used in the past seventy years, and there have been various implementations spanning from early versions preceding electronic computers, to modern hybrid versions embedding global search mechanisms [45, 46], as well as studies on dynamic optimisation [36]. This article focuses on pattern search and proposes a novel method belonging to this family of methods. The design of the proposed method is based on a *fitness landscape analysis approach*: in order to have a problem specific solver, the optimisation problem is analysed by an Artificial Intelligence tool and the results of the analysis are used to design the algorithm, see [7, 25, 32, 33]. In the specific case, the proposed pattern search aims to detect a preferential pattern that suits the specific problems. This article extends our previous work presented in [42]. More specifically, while in [42] we introduced a specific implementation of a pattern search method that uses a fitness landscape analysis, we extensively present here the fitness landscape analysis method and its applicability to the entire pattern search family. Furthermore, we investigate for the first time the theoretical standpoint of the proposed class of methods and provide a theoretical justification of the method.

The remainder of this article is organised in the following way. "Basic Notation and Pattern Search" section provides the background of the pattern search method, describes its evolution over the years, and highlights the open problem of selecting the pattern. "The Proposed Covariance Pattern Search" section describes the proposed pattern search variant from an implementation perspective. "Theoretical Justification of Covariance Pattern Search" section analyses the theoretical foundations of the proposed pattern search and justifies the conjecture on which the method is based. "Numerical Results" section experimentally validates the proposed method by comparing it against the pattern search method (without the fitness landscape analysis approach) and against modern algorithms. "Limitations of Covariance Pattern Search and Future Developments" section highlights the limitations and points out the opportunities for future work following this preliminary study. Finally, "Conclusion" section provides the conclusions to this study.

Basic Notation and Pattern Search

To clarify the notation employed, we will refer to the minimization problem of an objective function $f(\mathbf{x})$ in the continuous domain. The candidate solution $\mathbf{x}$ is a vector of n design variables in a hyper-cubical decision space $D \subset \mathbb{R}^n$:

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

In its original implementation, Pattern Search (PS) maintains a single solution and, while moving along the axes, improves upon its performance (objective function value). This procedure was part of the experimental description carried out by Nicholas Metropolis and Enrico Fermi, see [11, 29]. They were identifying the parameters of a physical model to minimise the error between the model and the experimental data. For this purpose, they varied one parameter of the model at a time by using a certain step size, and when no improvement in this error function was detected the step size was halved until it was smaller than a predefined threshold, see [1].

More formally, a starting solution $\mathbf{x}$ is inputted. Subsequently, the current best solution $\mathbf{x}$ is varied and a trial solution $\mathbf{x^t}$ generated. The trial solution:

$$\mathbf{x^t} = \left(x_1^t, x_2^t, \dots, x_n^t\right)$$

is generated by applying the update rules/search moves of a PS for each design variable $\forall i = 1, 2, \dots, n$:

$$x_i^t = x_i \pm \rho$$

where ρ is the step size or exploratory radius. The expression $x_i \pm \rho$ means that, according to the specific PS implementation, either or both the points $x_i^t = x_i + \rho$ and $x_i^t = x_i - \rho$ are visited. Let us consider the orthonormal basis of $\mathbb{R}^n$

$$\begin{aligned}
\mathbf{e}^1 &= (1, 0, \dots 0)\\
\mathbf{e}^2 &= (0, 1, \dots 0)\\
&\dots\\
\mathbf{e}^n &= (0, 0, \dots 1)
\end{aligned}$$

and rewrite the update equations for the ith design variable in the vector form

$$\mathbf{x^t} = \mathbf{x} \pm \rho \cdot \mathbf{e}^i$$

where $\rho \cdot \mathbf{e}^i$ indicates the product of the scalar step size ρ and the versor $\mathbf{e}^i$.

If a trial solution is found to have outperformed the current best solution, it replaces it, i.e. if $f\left(\mathbf{x^t}\right) \leq f(\mathbf{x})$ then $\mathbf{x} = \mathbf{x^t}$ (or equivalently $\mathbf{x} \leftarrow \mathbf{x^t}$).

If we modify the notation to include the iteration index, k, where $\mathbf{x}_k$ is the candidate solution at the kth iteration, the trial solution can be generated by an equation of the type

$$\mathbf{x^t} = \mathbf{x}_k \pm \rho \cdot \mathbf{e}^i \tag{1}$$

and if $f\left(\mathbf{x^t}\right) \leq f\left(\mathbf{x}_k\right)$ then $\mathbf{x}_{k+1} = \mathbf{x^t}$. As a note for all the Pattern Search implementations in this section, the scalar ρ could be a vector if it is desired to individually assign a different step size to each design variable. This can happen when the algorithmic designer possesses auxiliary knowledge of the problem or if range of variability is different from one design variable to another (D is hyper-rectangular).

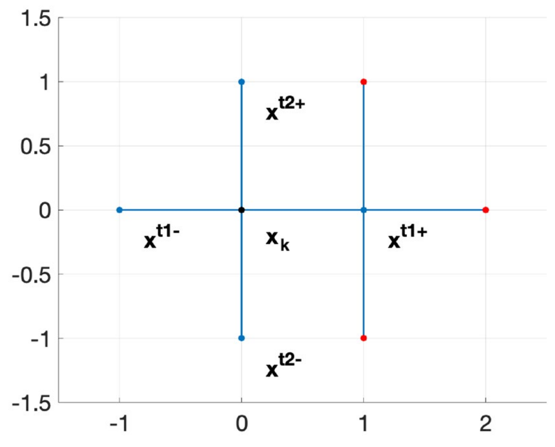


Fig. 2 The issue of revisiting solutions through Pattern Search. Let $\mathbf{x}_k$ be the current best, and $\mathbf{x}^{\mathbf{t}i+}$ $\mathbf{x}^{\mathbf{t}i-}$ $(i = 1, 2)$ (marked in blue) be the points visited during the exploration. If, for example, $\mathbf{x}^{\mathbf{t}1+}$ is the new current best then the following exploration visits the four points marked in red. Hence, the algorithm would revisit $\mathbf{x}_k$

Original Pattern Search

The early Pattern Search (PS) implementations were steepest descent methods [21]: all the variables were explored before a new current solution was selected. For this reason, $2n$ trial vectors are sampled before the best trial vector is selected as the new current best.

Let us consider two trial solutions $\mathbf{x}^{\mathbf{t}i+}$ and $\mathbf{x}^{\mathbf{t}i-}$ which have been generated at the iteration k for the design variable i, where:

$$\begin{aligned}
\mathbf{x}^{\mathbf{t}i+} &= \mathbf{x}_k + \rho \cdot \mathbf{e}^i\\
\mathbf{x}^{\mathbf{t}i-} &= \mathbf{x}_k - \rho \cdot \mathbf{e}^i
\end{aligned} \tag{2}$$

The mechanism makes the algorithm revisit the same solutions on a regular basis. For example, in $\mathbb{R}^2$, an exploration involves four trial solutions. When a new current best is selected the new exploration includes one already calculated point. Fig. 2 illustrates this issue of revisiting solutions, where it is shown that if the successful trial point is $\mathbf{x}^{\mathbf{t}i+}$, one search move would revisit the solution $\mathbf{x}_k$.

To avoid wasting a large portion of the computational budget by revisiting previously calculated candidate solutions, a memory-based mechanism is often employed. The original naive version of Pattern Search, taking into account this revisiting issue, is reported in Algorithm 1.

Algorithm 1 Pattern Search according to its original implementation [1,11]

```
INPUT $\mathbf{x}$
$k \leftarrow 1$
$\mathbf{x}_k \leftarrow \mathbf{x}$
while local budget condition do
    for $i = 1 : n$ do
        $\mathbf{x}^{\mathbf{t}i-} \leftarrow \mathbf{x}_k - \rho \cdot \mathbf{e}^i$
        Calculate $f\left(\mathbf{x}^{\mathbf{t}i-}\right)$ if not calculated previously
        $\mathbf{x}^{\mathbf{t}i+} \leftarrow \mathbf{x}_k + \rho \cdot \mathbf{e}^i$
        Calculate $f\left(\mathbf{x}^{\mathbf{t}i+}\right)$ if not calculated previously
    end for
    Select the trial solution $\mathbf{x}^{\mathbf{t}}$ with the best (minimal) objective function
    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
        $\rho \leftarrow \frac{\rho}{2}$
    end if
end while
$\mathbf{x} \leftarrow \mathbf{x}_k$
RETURN $\mathbf{x}$
```

As highlighted in [29], this approach can be accurate since it extensively explores the neighborhood before deciding on a search direction. However, this implementation is likely to require a large number of objective function evaluations before it detects a solution close to the optimum. Consequently, this approach can be impractical in high-dimensional domains.

At the opposite end of the spectrum, there are implementations of Pattern Search that are designed not to explore the entire neighborhood. In these implementations, Pattern Search can select and move as soon as a trial solution with a lower objective function value than that of the current best is detected. A greedy approach can yield good results when addressing large scale optimisation problems. For example, one search operator used in the large scale framework in [49] is fundamentally a greedy Pattern Search implementation that, for each design variable, samples a trial solution and if the first move fails it attempts to explore the opposite direction. More specifically, this greedy Pattern Search, herein referred to as gPS, first samples:

$$\mathbf{x}^{\mathbf{t}} = \mathbf{x}_k - \rho \cdot \mathbf{e}^i \tag{3}$$

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

$$\mathbf{x}^{\mathbf{t}} = \mathbf{x}_k + \frac{\rho}{2} \cdot \mathbf{e}^i \tag{4}$$

before moving to the following design variable. It must be observed that the step size is asymmetric in the two directions. This is a preventative measure to avoid revisiting previously visited solutions. This idea has also been employed within Memetic Computing frameworks, see [5, 7, 24, 52].

To illustrate the functioning of gPS a graphical representation of the moves in two-dimensional space $\mathbb{R}^2$ is shown in Fig. 3. In two dimensions there are three potential outcomes. (1) For each design variable, one move is successful: two trial solutions are accepted and then a diagonal move is performed; (2) One move is successful: the move is along one axis; (3) All the moves are unsuccessful: there is no successful move and the step size is halved. In more than two dimensions, a diagonal move is produced by at least two successes while the other two potential outcomes remain identical. Figure 3 displays the fist outcome with a diagonal move.

The details of Pattern Search according to the implementation in [49] is reported in Algorithm 2.

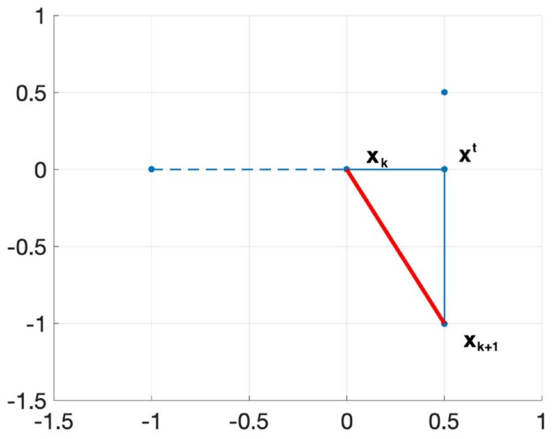


Fig. 3 Search moves of gPS: two (n) successes lead to a diagonal move. Dashed lines indicate failed moves, solid lines successful moves, and red the line the move from the previous current best to the new current best

Algorithm 2 Pattern Search according to the greedy implementation in [49]

```
INPUT $\mathbf{x}$
$k \leftarrow 1$
$\mathbf{x}_k \leftarrow \mathbf{x}$
while local budget condition do
   $h \leftarrow k$
   for $i = 1 : n$ do
      $\mathbf{x^t} \leftarrow \mathbf{x}_k - \rho \cdot \mathbf{e}^i$
      if $f(\mathbf{x^t}) \leq f(\mathbf{x}_k)$ then
         $k \leftarrow k + 1$
         $\mathbf{x}_k \leftarrow \mathbf{x^t}$
      else
         $\mathbf{x^t} \leftarrow \mathbf{x}_k + \frac{\rho}{2} \cdot \mathbf{e}^i$
         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 $\mathbf{x}$
```

Hooke–Jeeves Pattern Search

Although this paper and several recent studies [48] identify Pattern Search as a family of optimisation methods which dates back to the 1940s with Metropolis and Fermi's method, the term "Pattern Search" was only coined in the 1960s by Hooke and Jeeves [23]. In fact, Hooke and Jeeves referred to their method as "Direct Search", but it is more commonly known today as the Hooke–Jeeves Method or Hooke–Jeeves Pattern Search, see [4, 47].

The Hooke–Jeeves Pattern Search (HJPS) [23] is composed of two search moves, namely the *exploratory* and *pattern* moves, the latter of which has subsequently given the name to the method. The exploratory move scans all the decision variables and, in an approach similar to that of gPS in Algorithm 2, attempts one direction unless it fails, in which case it attempts the opposite direction. More specifically, at the generic kth iteration HJPS samples $\forall i = 1, 2 \ldots, n$ at first the trial solution:

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

and if this move fails ($f\left(\mathbf{x^t}\right) > f\left(\mathbf{x}_k\right)$), it attempts to sample:

$$\mathbf{x^t} = \mathbf{x}_k - \rho \cdot \mathbf{e}^i. \tag{6}$$

If the moves in all directions fail, then the radius (step size) is halved. HJPS addresses the revisiting issue by applying the pattern move: if the exploratory move succeeded and a new current best solution is selected, a move along the direction identified by the previous and current best (the direction connecting the two points) is attempted. Let us indicate the previous best solutions with $\mathbf{x^{old}}$ and the current best solution with $\mathbf{x}_k$, then the pattern move generates a new trial solution with:

$$\mathbf{x^p} = \mathbf{x^{old}} + \alpha\left(\mathbf{x}_k - \mathbf{x^{old}}\right) \tag{7}$$

where α is a scalar named *acceleration factor* and usually set equal to 2. A new exploration is then carried out around the point $\mathbf{x^p}$. If the exploration fails, the following exploration is centered on $\mathbf{x}_k$, while if the exploration succeeds, it continues. Algorithm 3 displays the working principles of HJPS.

Algorithm 3 Hooke-Jeeves Pattern Search [23]

```
INPUT $\mathbf{x}$
$k \leftarrow 1$
$\mathbf{x}_k \leftarrow \mathbf{x}$
while local budget condition do
    #EXPLORATORY MOVE
    $h \leftarrow k$
    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
            $\mathbf{x}^{\mathbf{old}} \leftarrow \mathbf{x}_k$
            $k \leftarrow k + 1$
            $\mathbf{x}_k \leftarrow \mathbf{x}^{\mathbf{t}}$
        else
            $\mathbf{x}^{\mathbf{t}} \leftarrow \mathbf{x}_k - \rho \cdot \mathbf{e}^i$
            if $f\left(\mathbf{x}^{\mathbf{t}}\right) \leq f\left(\mathbf{x}_k\right)$ then
                $\mathbf{x}^{\mathbf{old}} \leftarrow \mathbf{x}_k$
                $k \leftarrow k + 1$
                $\mathbf{x}_k \leftarrow \mathbf{x}^{\mathbf{t}}$
            end if
        end if
    end for
    if $k = h$ # If the exploration led to no improvement then
        $\rho \leftarrow \frac{\rho}{2}$
    else
        #PATTERN MOVE
        $\mathbf{x}^{\mathbf{P}} \leftarrow \mathbf{x}^{\mathbf{old}} + \alpha\left(\mathbf{x}_k - \mathbf{x}^{\mathbf{old}}\right)$
        Exploratory Move on $\mathbf{x}^{\mathbf{P}}$
        if $f\left(\mathbf{x}^{\mathbf{P}}\right) \leq f\left(\mathbf{x}_k\right)$ then
            $k \leftarrow k + 1$
            $\mathbf{x}_k \leftarrow \mathbf{x}^{\mathbf{P}}$
        end if
    end if
end while
$\mathbf{x} \leftarrow \mathbf{x}_k$
RETURN $\mathbf{x}$
```

An illustration of the HJPS search logic is shown in Fig. 4. It can be seen that HJPS and gPS perform very similar search moves, see Figs. 4 and 3. The main difference is the double (possibly diagonal) step in HJPS. The latter can be viewed as an attempt to exploit a promising search direction.

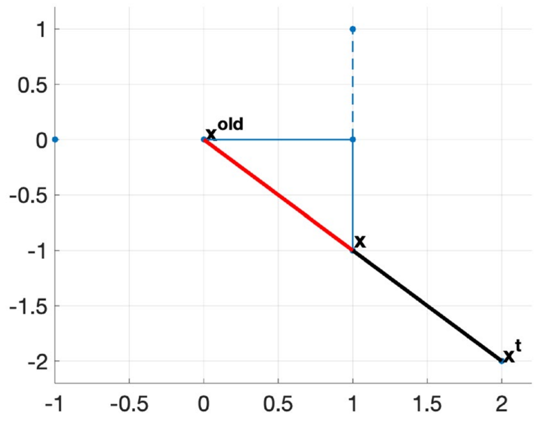


Fig. 4 Search moves of HJPS: diagonal move due to the exploratory move and double step due to the pattern move. Dashed lines indicate failed moves, solid lines successful moves, the red line the exploratory move from the previous current best to the new current best, and the black line the pattern move

Generalised Pattern Search

Virginia Torczon in [48] conceptualised Pattern Search as a family of direct search methods, i.e. optimisation algorithms that do not require calculations of the gradient. Hence, Pattern Search is a family of algorithms characterised by two elements:

– a set of search directions spanning the decision space;
– a trial step vector endowed with a step variation rule.

For example, in the original Pattern Search and gPS methods, the directions are given by the orthonormal basis $B_{\mathbf{e}} = \{\mathbf{e}^1, \mathbf{e}^2, \dots, \mathbf{e}^n\}$ in both the possible orientations, e.g. $\mathbf{e}^i$ and $-\mathbf{e}^i$. The vector of step sizes is a vector of length $2n$, whose elements are all ρ for the original Pattern Search method, or $\frac{\rho}{2}$ and ρ for the gPS method. Pattern Search algorithms differ in their implementations of the exploratory moves, e.g. a steepest descent or greedy logic.

This generalization of Pattern Search should be considered in relation to the definition of the multidirectional Pattern Search introduced in [12], where directions different from those identified by the design variables (axes of the problem) are used during the search.

The Generalized Pattern Search (GPS) introduced in [48] is a framework that includes all the algorithms within this category. Formally, with reference to the kth iteration, the search directions are determined by two matrices. The first is a non-singular matrix, namely the *basis matrix*, and it is indicated with $\mathbf{B} \in \mathbb{R}^{n \times n}$ 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 $\mathbf{C}_k \in \mathbb{Z}^{n \times p}$ where $\mathbb{Z}^{n \times p}$ is the set of matrices of relative numbers of size $n \times p$ with $p > 2n$ and rank n. The matrix $\mathbf{C}_k$ can be partitioned as:

$$\mathbf{C}_k = \left(\mathbf{M}_k, -\mathbf{M}_k, \mathbf{L}_k\right)$$

where $\mathbf{M}_k$ is a non-singular matrix of order n, $-\mathbf{M}_k$ is the opposed matrix of $\mathbf{M}_k$, and $\mathbf{L}_k$ is a $n \times (p - 2n)$ matrix that contains at least the null column vector $\mathbf{o}$. The search directions are given by the columns of the matrix:

$$\mathbf{P}_k = \mathbf{B}\mathbf{C}_k = \left(\mathbf{B}\mathbf{M}_k, -\mathbf{B}\mathbf{M}_k, \mathbf{B}\mathbf{L}_k\right) \tag{8}$$

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

The GPS kth *trial iteration* along the ith direction is the vector $\mathbf{s}_k$, defined as:

$$\mathbf{s}_k = \Delta_k \mathbf{B}\mathbf{c}_k^i \tag{9}$$

where Δ_k is a positive real number and $\mathbf{c}_k^i$ is the ith column of the matrix $\mathbf{C}_k$. The parameter Δ_k determines the step size while $\mathbf{B}\mathbf{c}_k^i$ 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:

$$\mathbf{x^t}_k = \mathbf{x}_k + \mathbf{s}_k. \tag{10}$$

The set of operations that yields a current best point is called the *exploratory move* (coherently with the implementations seen above). The exploratory move succeeds when a solution with better performance is detected, and fails when no update of the current best occurs. Different Pattern Search implementations employ different strategies, e.g. by attempting only one trial vector per step or exploring all the columns of $\Delta_k \mathbf{P}_k$. However, as explained in [48], Pattern Search implementations belong to the GPS framework only if the following hypotheses, namely the *Strong Hypotheses*, are verified.

Strong Hypotheses

Hypothesis 1 $\mathbf{s}_k$ is generated by the pattern $\mathbf{P}_k$ or, in other words, is a column vector of the matrix $\Delta_k \mathbf{P}_k$. The length is determined by the scalar Δ_k.

Hypothesis 2 If there exists a column vector $\mathbf{y}$ of $\left(\mathbf{B}\mathbf{M}_k, -\mathbf{B}\mathbf{M}_k\right)$ such that $f\left(\mathbf{x}_k + \mathbf{y}\right) < f\left(\mathbf{x}_k\right)$, then the exploratory move must produce a trial step $\mathbf{s}_k$ such that $f\left(\mathbf{x}_k + \mathbf{s}_k\right) < f\left(\mathbf{x}_k\right)$.

Hypothesis 3 The update of Δ_k should follow some rules. In the case of a failed exploratory move, Δ_k has to decrease, however, in the case of success Δ_k must either remain the same or increase.

The pseudocode of GPS is given in Algorithm 4.

Algorithm 4 Generalized Pattern Search [48]

```
INPUT x
$k \leftarrow 1$
$\mathbf{x}_k \leftarrow \mathbf{x}$
while local budget condition do
    generate the trial step $\mathbf{s}_k$ from $\Delta_k \mathbf{P}_k$
    calculate $\mathbf{x^t}_k \leftarrow \mathbf{x}_k + \mathbf{s}_k$
    if $f(\mathbf{x^t}_k) \leq f(\mathbf{x}_k)$ then
        $\mathbf{x_{k+1}} \leftarrow \mathbf{x^t}_k$
    else
        $\mathbf{x_{k+1}} \leftarrow \mathbf{x}_k$
    end if
    $k \leftarrow k + 1$
    update $\mathbf{P}_k$ and $\Delta_k$
end while
$\mathbf{x} \leftarrow \mathbf{x_{k+1}}$
RETURN x
```

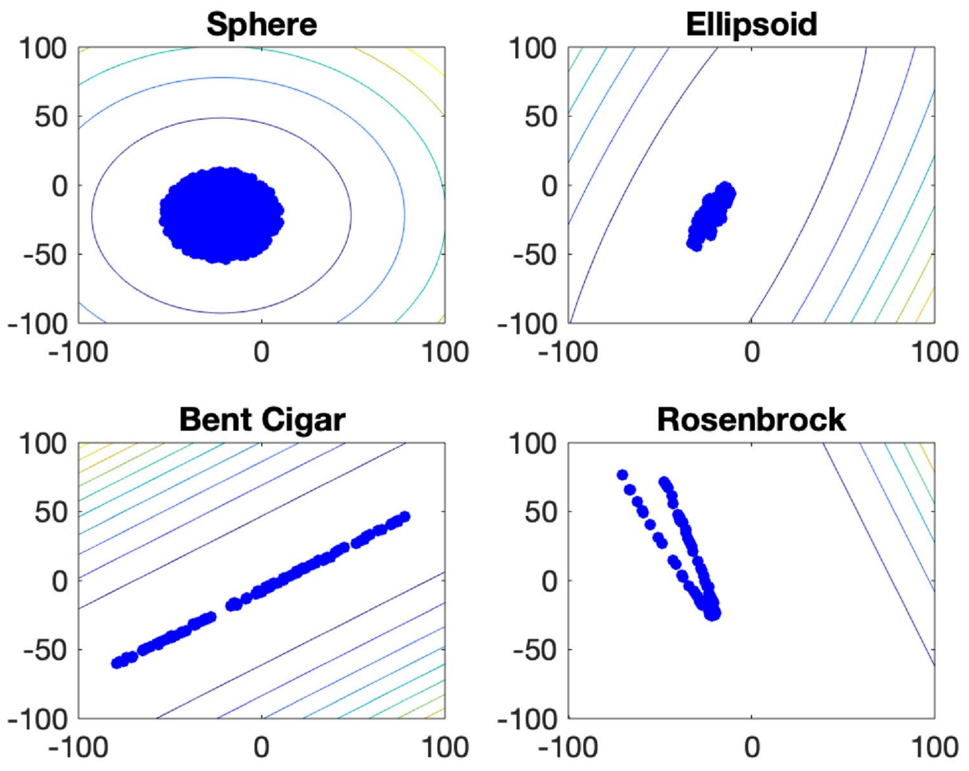


Fig. 5 Sampling of points within $[-100, 100]^2$ for shifted and rotated Sphere, Ellipsoid, Bent Cigar, and Rosenbrock functions below the threshold values $10^3, 3 \times 10^4, 10^6$, and 5×10^3, respectively

It is worth mentioning that GPS may include a Pattern Search implementation where the search directions are updated during its operation.

The Proposed Covariance Pattern Search

The proposed algorithm is a GPS that allocates the initial part of its computational budget to the analysis of the problem, in order to determine a problem-specific pattern $\mathbf{P}_k$. This section describes the implementation aspects of the proposed algorithm, namely the Covariance Pattern Search (CPS). This section, just like the implementation, is divided into two parts:

1. Fitness Landscape Analysis;
2. Algorithmic Search.

Fitness Landscape Analysis

A number of candidate solutions/points are sampled in the decision space D and their objective function values are calculated. The function values are compared with a threshold *thr* and those values that are below *thr* are saved in a data structure, while the others are discarded. The purpose of this operation is to have a sample of points whose distribution describes the geometry of the problem.

To illustrate this fact let us consider the following four popular objective functions in two dimensions within $[-100, 100]^2$, see e.g. [30].

Sphere	$f(\mathbf{x}) = x_1^2 + x_2^2$
Ellipsoid	$f(\mathbf{x}) = 50x_1^2 + 200x_2^2$
Bent Cigar	$f(\mathbf{x}) = x_1^2 + 10^6 x_2^2$
Rosenbrock	$f(\mathbf{x}) = 100\left(x_1^2 - x_2\right)^2 + \left(x_1 - 1\right)^2$

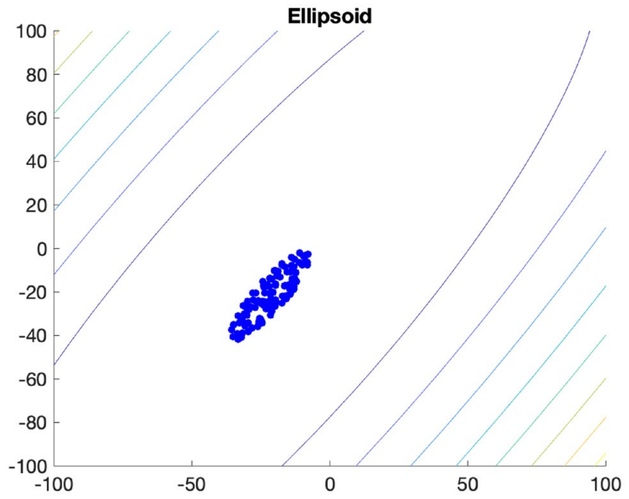


Fig. 6 Distribution of points for a rotated ellipsoid associated with a covariance matrix $\mathbf{C}$ and eigenvector matrix $\mathbf{P}$

If we apply the procedure described above for these four problems after shifting and rotation, we obtain four sets of points that are distributed as shown in Fig. 5.

Let us consider the scenario where out of the points sampled in $D \subset \mathbb{R}^n$, a set of m vectors/candidate solutions have an objective function value below the threshold *thr*:

$$\mathbf{x^1} = \left(x_1^1, x_2^1, \dots, x_n^1\right)$$
$$\mathbf{x^2} = \left(x_1^2, x_2^2, \dots, x_n^2\right)$$
$$\dots$$
$$\mathbf{x^m} = \left(x_1^m, x_2^m, \dots, x_n^m\right)$$

```
Algorithm 5 Fitness Landscape Analysis
INPUT objective function $f(\mathbf{x})$, decision space $D$, and parameters $thr$ and $samplesize$
$h \leftarrow 1$
for $s = 1 : samplesize$ do
    Sample a point $\mathbf{x}$ in the decision space $D$
    if $f(\mathbf{x}) < thr$ then
        Insert $\mathbf{x}$ into the data structure $\mathbf{V}$: $\mathbf{V}(h) \leftarrow \mathbf{x}$
        $h \leftarrow h + 1$
    end if
end for
Process the data structure $\mathbf{V}$ to calculate the mean vector $\mu$ and covariance matrix $\mathbf{C}$
Apply Cholesky Factorisation to extract the eigenvectors $\mathbf{P} = (\mathbf{p}^1, \mathbf{p}^2, \dots, \mathbf{p}^n)$
```

and are allocated in the data structure $\mathbf{V}$.

These points can be interpreted as the samples of a multivariate statistical distribution characterised by a mean vector:

$$\boldsymbol{\mu} = \left(\mu_1, \mu_2, \dots, \mu_n\right) = \frac{1}{m}\left(\sum_{i=1}^{m} x_1^i, \sum_{i=1}^{m} x_2^i, \dots, \sum_{i=1}^{m} x_n^i\right)$$

and a covariance matrix:

$$\mathbf{C} = \begin{pmatrix} c_{1,1} & c_{1,2} & \dots & c_{1,n} \\ c_{2,1} & c_{2,2} & \dots & c_{2,n} \\ \dots & \dots & \dots & \dots \\ c_{n,1} & c_{n,2} & \dots & c_{n,n} \end{pmatrix}$$

where:

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

and:

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

Subsequently, the n eigenvectors of the matrix $\mathbf{C}$ are calculated by means of Cholesky Factorisation, see [38, 44]. These eigenvectors are the columns $\mathbf{p}^i$ of a matrix $\mathbf{P}$:

$$\mathbf{P} = \left(\mathbf{p}^1, \mathbf{p}^2, \dots, \mathbf{p}^n\right).$$

The directions of the eigenvectors are then used by the algorithms to perform the search. Algorithm 5 displays the pseudocode of the Fitness Landscape Analysis.

Algorithmic Search

If $\mathbf{P}$ is the matrix of the eigenvectors, we propose a GPS whose basis matrix is $\mathbf{P}$ and its pattern matrix $\mathbf{P}_k$ is thus:

$$\mathbf{P}_k = \mathbf{P}\mathbf{C}_k = \left(\mathbf{P}\mathbf{M}_k, -\mathbf{P}\mathbf{M}_k, \mathbf{P}\mathbf{L}_k\right) \quad (11)$$

the trial step along the ith direction is:

$$\mathbf{s}_k = \Delta_k \mathbf{P}\mathbf{c}_k^i \quad (12)$$

and if $\mathbf{x}_k$ is the current best solution at the kth iteration, the corresponding trial solution $\mathbf{x^t}_k$ is determined by:

$$\mathbf{x^t}_k = \mathbf{x}_k + \mathbf{s}_k. \quad (13)$$

To exemplify the content of the proposal, let us see the proposed implementations of gPS and HJPS. Since the main novel characteristic of the proposed idea is the use of the eigenvectors of a covariance matrix, we will refer to the implemented algorithms as Covariance Pattern Search, and these two implementations as greedy Covariance Pattern Search (gCPS) and Hooke-Jeeves Covariance Pattern Search (HJCPS).

In the case of gCPS in two dimensions ($n = 2$), the basis matrix is the eigenvector matrix $\mathbf{P}$ while the generating matrix $\mathbf{C}_k$ is:

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

and $\Delta_k = \rho$ (which gets halved every time the exploration fails). Thus, the total set of possible moves is determined by $\mathbf{PC}_k$.

For example, the distribution in Fig. 6 obtained from a rotated ellipsoid function is associated with a covariance matrix:

$$\mathbf{C} = \begin{pmatrix} 55.362 & 67.026 \\ 67.026 & 109.40 \end{pmatrix}$$

whose eigenvector matrix would be:

$$\mathbf{P} = \begin{pmatrix} -0.82881 & 0.55953 \\ 0.55953 & 0.8.2881 \end{pmatrix}$$

One potential trial step could be:

$$\mathbf{s}_k = \Delta_k \mathbf{P}\mathbf{c}_k^6 = \rho \begin{pmatrix} -0.82881 & 0.55953 \\ 0.55953 & 0.82881 \end{pmatrix} \begin{pmatrix} \frac{1}{2} \\ -1 \end{pmatrix}$$
$$= \rho \begin{pmatrix} -0.97393 \\ -0.54905 \end{pmatrix}$$

which can be re-written as:

$$\mathbf{s}_k = \rho \mathbf{P}\left(\frac{1}{2}\mathbf{e}^1 - \mathbf{e}^2\right) = \rho\left(\frac{1}{2}\mathbf{e}^1\mathbf{p}^1 - \mathbf{p}^2\right).$$

In other words, in the n dimensional case every trial vector is the linear combination of the eigenvectors:

$$\mathbf{s}_k = \lambda_1 \mathbf{p}^1 + \lambda_2 \mathbf{p}^2 + \ldots + \lambda_n \mathbf{p}^n$$

where the coefficients λ_i can take only three values: $\forall i, \lambda_i \in \{0, -\rho, \frac{\rho}{2}\}$.

In the case of HJCPS in two dimensions ($n = 2$), the basis matrix is the eigenvector matrix $\mathbf{P}$. The generating matrix $\mathbf{C}_k$ is slightly more complex due to the presence of the pattern move. Let us partition the matrix $\mathbf{C}_k$ as:

$$\mathbf{C}_k = \left(\mathbf{C^e}_k, \mathbf{C^p}_k\right)$$

where $\mathbf{C^e}_k$ refers to the exploratory move and $\mathbf{C^p}_k$ refers to the pattern move. The first submatrix is analogous to gCPS, i.e.:

$$\mathbf{C^e}_k = \begin{pmatrix} 1 & 0 & -1 & 0 & 1 & 1 & -1 & -1 & 0 \\ 0 & 1 & 0 & -1 & 1 & -1 & -1 & 1 & 0 \end{pmatrix}$$

The pattern move attempts to search further along the direction of the successful exploratory move. With reference to Fig. 4, the successful exploratory move corresponds to the column:

$$\mathbf{c}_k^i = \begin{pmatrix} 1 \\ -1 \end{pmatrix}.$$

In this case, the pattern move adds a vector to the exploratory move as follows:

$$\alpha\left(\mathbf{x} - \mathbf{x}^{\mathbf{old}}\right) = \alpha \begin{pmatrix} 1 \\ -1 \end{pmatrix}.$$

Thus, the total possible move is:

$$\begin{pmatrix} 1 \\ -1 \end{pmatrix} + \alpha \begin{pmatrix} 1 \\ -1 \end{pmatrix} = \begin{pmatrix} 1 + \alpha \\ -1 - \alpha \end{pmatrix}$$

Thus, the second submatrix of the generating matrix is:

$$\mathbf{C^p}_k = \begin{pmatrix} 1+\alpha & 0 & -1-\alpha & 0 & 1+\alpha & 1+\alpha & -1-\alpha & -1-\alpha & 0 \\ 0 & 1+\alpha & 0 & -1-\alpha & 1+\alpha & -1-\alpha & -1+\alpha & 1+\alpha & 0 \end{pmatrix}.$$

Additionally, for HJCPS, $\Delta_k = \rho$ and follows the halving logic. The other steps are analogous to what is shown for gCPS.

In the general n dimensional case, the trial step $\mathbf{s}_k$ can be represented as a linear combination of eigenvectors:

$$\mathbf{s}_k = \alpha'\left(\lambda_1 \mathbf{p}^1 + \lambda_2 \mathbf{p}^2 + \ldots + \lambda_n \mathbf{p}^n\right)$$

where $\forall i, \lambda_i \in \{0, -\rho, \rho\}$ and $\alpha' \in \{1, \alpha\}$.

Let us consider a given problem that has been analysed as shown in Algorithm 5 with the eigenvectors made available. The variation operators composing the exploratory moves for gPS and HJPS become:

gPS	gCPS
$\mathbf{x^t} = \mathbf{x} - \rho \cdot \mathbf{e}^i$	$\mathbf{x^t} = \mathbf{x} - \rho \cdot \mathbf{p}^i$
$\mathbf{x^t} = \mathbf{x} + \frac{\rho}{2} \cdot \mathbf{e}^i$	$\mathbf{x^t} = \mathbf{x} + \frac{\rho}{2} \cdot \mathbf{p}^i$
HJPS	HJCPS
$\mathbf{x^t} = \mathbf{x} + \rho \cdot \mathbf{e}^i$	$\mathbf{x^t} = \mathbf{x} + \rho \cdot \mathbf{p}^i$
$\mathbf{x^t} = \mathbf{x} - \rho \cdot \mathbf{e}^i$	$\mathbf{x^t} = \mathbf{x} - \rho \cdot \mathbf{p}^i$
$\mathbf{x^p} = \mathbf{x^{old}} + \alpha\left(\mathbf{x} - \mathbf{x^{old}}\right)$	$\mathbf{x^p} = \mathbf{x^{old}} + \alpha\left(\mathbf{x} - \mathbf{x^{old}}\right)$

For the sake of clarity, the gPS in Algorithm 2 is then revisited and displayed in Algorithm 6.

Algorithm 6 Covariance Pattern Search according to the greedy implementation (gCPS)

```
INPUT $\mathbf{x}$
$k \leftarrow 1$
$\mathbf{x}_k \leftarrow \mathbf{x}$
while local budget condition do
    $h \leftarrow k$
    for $i = 1 : n$ do
        $\mathbf{x^t} \leftarrow \mathbf{x}_k - \rho \cdot \mathbf{p}^i$
        if $f(\mathbf{x^t}) \leq f(\mathbf{x}_k)$ then
            $k \leftarrow k + 1$
            $\mathbf{x}_k \leftarrow \mathbf{x^t}$
        else
            $\mathbf{x^t} \leftarrow \mathbf{x}_k + \frac{\rho}{2} \cdot \mathbf{p}^i$
            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 $\mathbf{x}$
```

Theoretical Justification of Covariance Pattern Search

This section provides a theoretical justification of the proposed approach. We structured this section into three parts addressing three subproblems:

1. If the covariance matrix contains enough eigenvectors to generate a pattern $\mathbf{P}_k$.
2. If the proposed method is guaranteed to converge and under what hypotheses the convergence occurs.
3. Why the choice of eigenvectors as search directions of a pattern is proposed in this study.

To address the first question, we have to recall some basic linear algebra notions. Different square matrices (or different endomorphisms) of order n can have a different number of linearly independent eigenvectors, see [38]. Thus, in the general case we might have a matrix $\mathbf{A}$ of order n and only $n-1$ linearly independent eigenvectors. In that case, the matrix $\mathbf{P}$ would be rectangular and there would not be enough directions to span the entire decision space.

In our case the covariance matrix $\mathbf{C}$ is guaranteed to be symmetric due to the commutativity of the product of numbers $\forall j, l$:

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

Since the covariance matrix $\mathbf{C}$ is symmetric, as long as it is calculated by means of at least $n+1$ linearly independent samples (vectors), the following properties are verified (see [38] for the proofs).

Proposition 1 *Properties of the covariance matrix* $\mathbf{C}$.

- *The eigenvalues of* $\mathbf{C}$ *are all real.*
- *Any two eigenvectors of* $\mathbf{C}$ *corresponding to two distinct eigenvalues are orthogonal.*
- *The covariance matrix* $\mathbf{C}$ *is always diagonalisable, that is, there exists a non singular matrix* $\mathbf{P}$ *of order n such that:*

  $$\mathbf{P}^{-1}\mathbf{C}\mathbf{P} = \mathbf{D}$$

  is diagonal.
- *The covariance matrix* $\mathbf{C}$ *can be diagonalised by means of an orthogonal matrix* $\mathbf{P}$, *that is* $\mathbf{P}^{-1} = \mathbf{P}^{\mathbf{T}}$, *whose columns are the eigenvectors of* $\mathbf{C}$.

Thus, in relation to the proposed method we can provide the following observation.

Observation 1 *For an optimisation problem and an estimated covariance matrix* $\mathbf{C}$ *from Algorithm 5, a non singular matrix of eigenvectors* $\mathbf{P}$ *and a pattern containing the direction of the eigenvectors* $\mathbf{P}_k = \left(\mathbf{PM}_k, -\mathbf{PM}_k, \mathbf{PL}_k\right)$ *can always be found.*

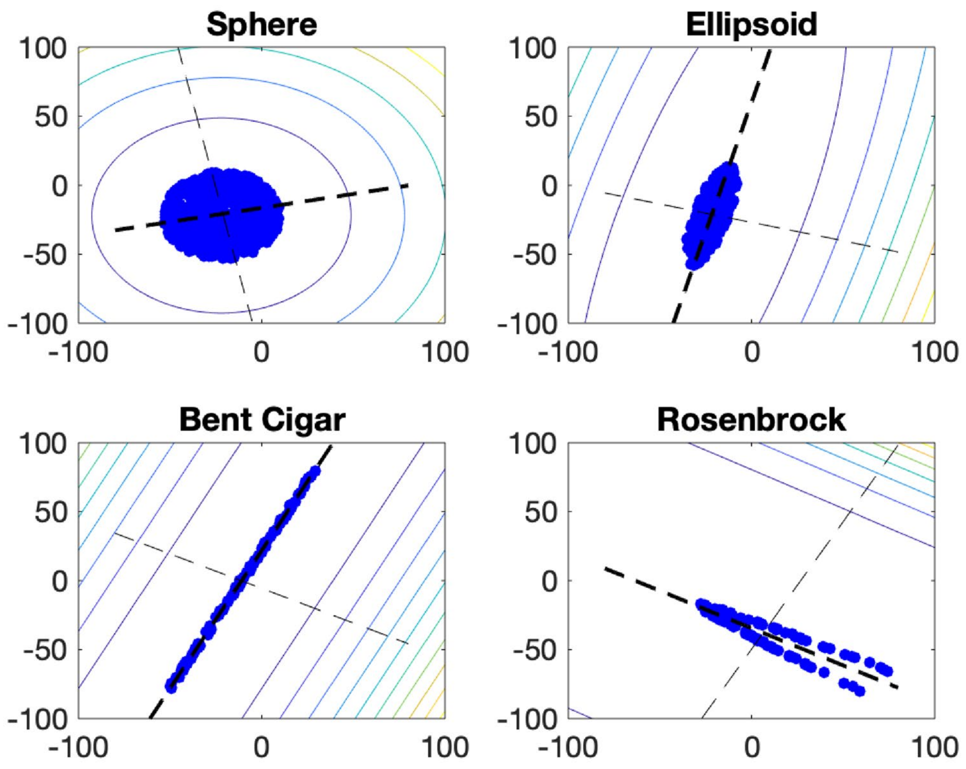


Fig. 7 Sampling of points within $[-100, 100]^2$ and corresponding eigenvector directions for shifted and rotated Sphere, Ellipsoid, Bent Cigar, and Rosenbrock functions below the threshold values 10^3, 7×10^4, 10^6, and 9×10^3, respectively

Regarding the second subproblem, the convergence of the method, we consider the study in [48] about the convergence of GPS and summarise some of the theoretical results in the following theorem.

Theorem 1 *Let f be a continuously differentiable function on a compact neighborhood* $L(\mathbf{x_0})$. *The sequence* $\{\mathbf{x}_k\}$ *produced by GPS in Algorithm 4 with respect to the Strong Hypotheses converges to a null gradient point* $\mathbf{x}$:

$$\lim_{k \to \infty} |\nabla f(\mathbf{x})| = \mathbf{o}$$

where $\mathbf{o}$ *is the null vector.*

Since from a theoretical standpoint, this article does not propose a completely new algorithm but a novel implementation of GPS (that is the use of the basis matrix $\mathbf{B} = \mathbf{P}$), the convergence of Covariance Pattern Search algorithms is subject to the same hypotheses discussed in [48] and summarised in Theorem 1.

Let us verify the strong hypotheses for gCPS.

- Hypothesis 1: $\mathbf{s}_k$ is a linear combination of the eigenvectors that is a column of $\Delta_k \mathbf{P}_k$.
- Hypothesis 2: from Algorithm 6, when a better solution is found there is an update in the value of $\mathbf{x}$. At the end of the exploratory moves the sum of all the updates contains the trial step $\mathbf{s}_k$.
- Hypothesis 3: from Algorithm 6 , ρ remains the same when there is an update and decreases when the exploratory move failed

Similar considerations can be made about HJCPS. This leads to the following observation.

Observation 2 *The proposed Covariance Pattern Search algorithms converge to a null gradient point (possibly a local optimum) under the same conditions of GPS. If the hypotheses on the problem are not valid then Covariance Pattern Search algorithms are heuristics.*

From a practical position, the guarantee of convergence as in Theorem 1 is not itself a guarantee that the algorithm is usable and successful to solve real-world problems. The first reason is that many real problems are not continuously differentiable, and often the function is not available. The second reason is that a null gradient point (not necessarily a local optimum) is guaranteed to be found after a very large number of steps. On the other hand, the demand of real-world optimisation is to detect a high-performance solution in a short time.

In this sense, the convergence rate (convergence speed is the term used for heuristics and metaheuristics) is a very important characteristic of a search algorithm. The proposal to use eigenvectors of the covariance matrix is driven by

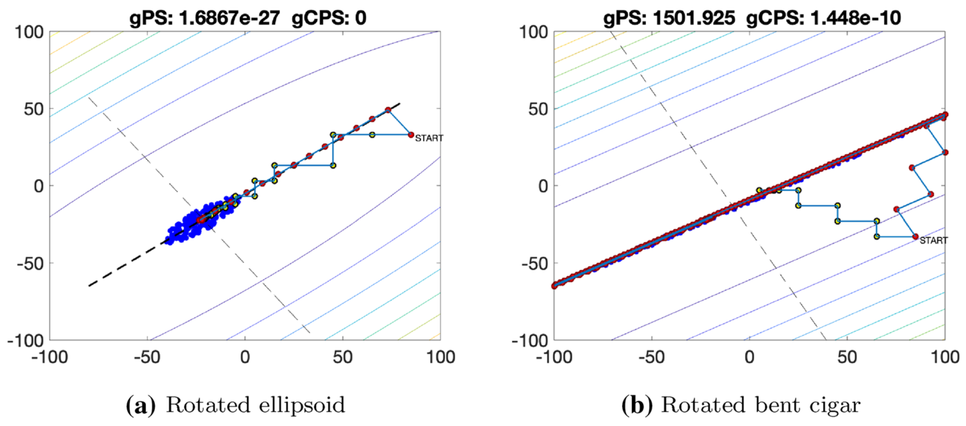


Fig. 8 Functioning of gPS (yellow markers) and gCPS (red markers). The result of the fitness landscape analysis and directions of the eigenvectors are also displayed. The error of the algorithms is reported on the top of each figure

this motivation which is then summarised in the following conjecture.

Conjecture 1 *If a distribution of points describes the geometry of the problem, a Pattern Search performed along the directions of the eigenvectors (of the covariance matrix of this distribution) has higher convergence rate than the same Pattern Search performed along the directions of any other basis of* $\mathbb{R}^n$.

Justification. To understand the rationale of the proposal we should consider the fundamental meaning of the covariance matrix of a multivariate distribution, see [14].

The diagonal elements of the matrix directly describe the geometry of the problem since they represent the extent of the distribution along with a variable. A diagonal element much larger in value than that of other diagonal elements means that the distribution is stretched along with a design variable. In our case, the distribution approximates the basin of attraction [21] and the shape of the contour plot, and that is the geometry of the problem. With reference to Fig. 5, for the sphere the diagonal elements of the covariance matrix are very similar to each other while for a (non-rotated) ellipsoid the diagonal elements would greatly differ from each other.

The extradiagonal elements represent the correlation between pairs of variables. A large value means a high correlation while zero means no correlation. In our case, since the points are a level set, the correlation is meant with respect to the objective function: zero means that the function can be decomposed over the variables while a large value means that this decomposition is not possible. To intuitively visualise this fact, the covariance matrix associated with a sphere or a non-rotated ellipsoid would be diagonal while that associated to rotated problems, such as rotated bent cigar, would be full.

As shown above, Pattern Search algorithms generate the trial vector as a linear combination of vectors along with the directions of the basis **B**, hence they search the optimisation problem by decomposing it along with the directions of **B**. The vast majority of Pattern Search algorithms, e.g. [4, 23, 41], use **B** equal to the identity matrix **I**, i.e. a search along with the directions of the design variables[1]. This strategy is efficient in solving problems whose associated covariance matrix is diagonal, however, they are far less efficient in solving problems whose associated covariance matrix is full, e.g. rotated bent cigar.

The main idea behind this proposal is to choose those directions that diagonalise the covariance matrix. The proposed Covariance Pattern Search algorithms can be seen as methods that initially perform a change of coordinates and then search along with the variables of the new reference system. The problem can then be decomposed along

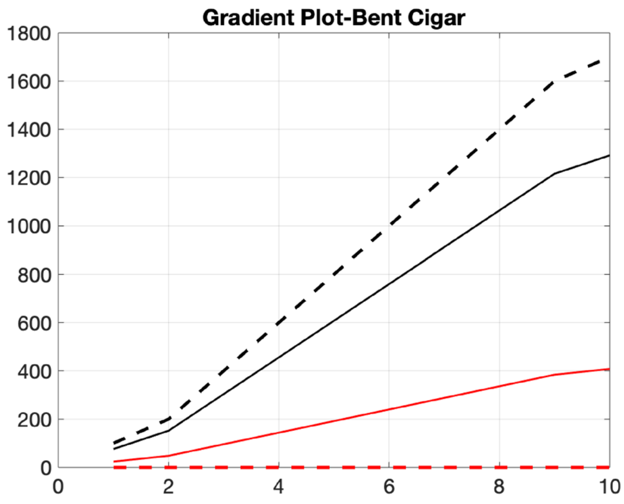


Fig. 9 Plot of the directional derivatives of gPS (continuous lines) and gCPS (dashed lines) along with their respective search directions (columns of matrix **B**) on the bent cigar problem in two dimensions

[1] An exception is multidirectional search [12] where the pattern follows a very different simplex-based logic.

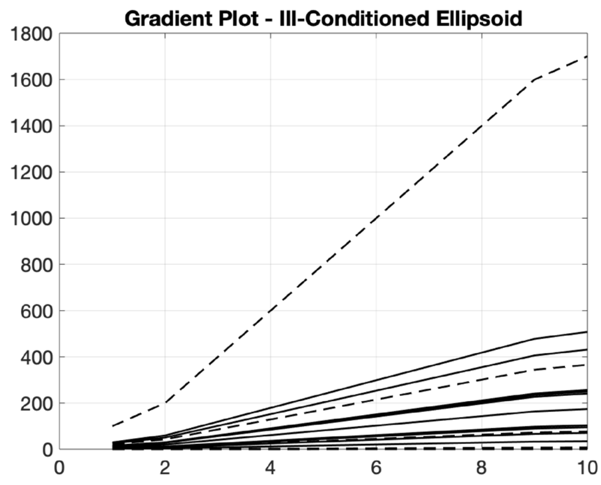


Fig. 10 Plot of the directional derivatives of gPS (continuous lines) and gCPS (dashed lines) along their respective search directions (columns of matrix **B**) on the ill-conditioned ellipsoid in ten dimensions

these new variables resulting in efficient Pattern Search Algorithms.

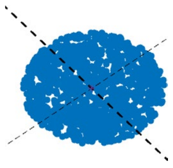
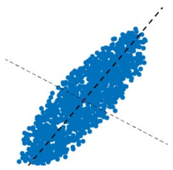
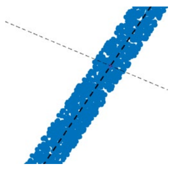
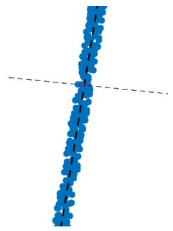
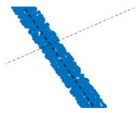
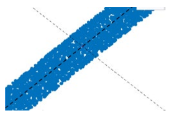
This concept is broadly used in other contexts, especially in Data Science, and is closely related to Principal Component Analysis [26]. To better understand the direction of the exploration, Fig. 7 displays distribution points and eigenvector directions for the same problems displayed in Fig. 5

To better understand the functioning of the method, Fig. 8 illustrate the points visited by gPS and gCPS respectively to solve the rotated ellipsoid and rotated bent cigar problems. Both the algorithms have been run until $\rho < 10^{-75}$ or a maximum function evaluation budget of $100000 \times n$. The points visited by gPS are marked in yellow while those visited by gCPS are marked in red. The final objective function errors are reported on the top of the figure.

The rotated ellipsoid in two dimensions is an easy problem for both the algorithms. However, we may observe that gCPS (red markers) reaches the axis of the ellipsoid in one step and then descend through it through the optimum, while gPS needs more steps to approach it. We found that gCPS reached its final value in 829 objective function calls (including the fitness landscape analysis to which a budget of 500 function calls was allocated) while gPS required 1005 objective function calls.

For the bent cigar, the advantages of the proposed Covariance Pattern Search over the standard gPS approach are more evident. The directions provided by the design variables appear unsuitable to descend the gradient along the thin stripe characterising the geometry of this problem. As a result, gPS fails to detect the local minimum. The proposed gCPS moves along the main dimension of the stripe (i.e. the direction of one eigenvector) and efficiently detects a point that is very close to the theoretical optimum.

Table 1 Basins of attraction functions

Function name	Function calculation	Basin of attraction in 2D
Sphere	**INPUTx** $\mathbf{z} \leftarrow \mathbf{Q}_p(\mathbf{x} - \mathbf{o})$ $f_1 \leftarrow \sum_{i=1}^{n} z_i^2$	
Ellipsoid	**INPUTx** $\mathbf{z} \leftarrow \mathbf{Q}_p(\mathbf{x} - \mathbf{o})$ $f_2 \leftarrow \sum_{i=1}^{n} 50\left(i^2 z_i\right)^2$	
Ill-conditioned ellipsoid	**INPUTx** $\mathbf{z} \leftarrow \mathbf{Q}_p(\mathbf{x} - \mathbf{o})$ $f_3 \leftarrow \sum_{i=1}^{n} \left(10^6\right)^{\frac{i-1}{n-1}} z_i^2$	
Bent cigar	**INPUTx** $\mathbf{z} \leftarrow \mathbf{Q}_p(\mathbf{x} - \mathbf{o})$ $f_4 \leftarrow z_1^2 + 10^6\left(\sum_{i=2}^{n} z_i\right)^2$	
Discus	**INPUTx** $\mathbf{z} \leftarrow \mathbf{Q}_p(\mathbf{x} - \mathbf{o})$ $f_5 \leftarrow 10^6 z_1^2 + \left(\sum_{i=2}^{n} z_i\right)^2$	
Sum of powers	**INPUTx** $\mathbf{z} \leftarrow \mathbf{Q}_p(\mathbf{x} - \mathbf{o})$ $f_6 \leftarrow \sqrt{\sum_{i=1}^{n} \lvert z_i \rvert^{\left(2+4\frac{i-1}{n-1}\right)}}$	

To further justify Conjecture 1, with reference to the problem in Fig. 8, we performed the following test. From the optimum we performed ten steps of size $\rho = 0.01$ along the directions of the matrix **B** for gPS and gCPS. In other words, we performed ten steps in the directions of the variables and the directions of the eigenvectors of the covariance matrix **C**. At each step, we calculated and saved the objective function value. With the calculated objective function values we calculated the numerical gradient of the objective functions along these directions. Figure 9 illustrates the directional derivative along the directions of gPS (continuous line) and gCPS (dashed lines).

The test shows that gCPS optimises the objective function along a high derivative direction and a low derivative direction. The fitness landscape analysis identifies a search direction along which small steps quickly lead to large improvements and another direction along which the problem is

Table 2 Thresholds *thr* in 10, 30, and 50 dimensions

n	f_1	f_2	f_3	f_4	f_5	f_6
10	10^4	10^9	5×10^8	10^9	10^9	10^4
30	5×10^4	5×10^{11}	2×10^9	2×10^9	10^8	10^5
50	10^5	5×10^{13}	5×10^9	2×10^9	5×10^7	3×10^5

nearly “flat”. Unlike gCPS, the standard gPS directions present intermediate gradient features.

We repeated the test for all the problems listed in Sect. 5 for 2, 10, 30, and 50 dimensions. Apart from the sphere where all the lines of the gradient plot collapse in one line, the fitness landscape analysis of gCPS systematically detected a direction whose gradient was higher than that of any variable directions of gPS. In other words, the CPS mechanism appears to be able to detect some preferential directions along which a pattern search displays a high convergence rate.

To show an example of the gradient in higher dimensions, Fig. 10 illustrates the results of the test for the rotated ill-conditioned ellipsoid in ten variables, see Section 5 for details on the function.

The results in Fig. 10 clearly show that gCPS performs its search along one direction where the derivative is higher than any variable directions of gPS.

Numerical Results

Results in this section are divided into two subsections:

– Validation of the Covariance Pattern Search.
– Comparison against other algorithms.

To simulate the local search conditions we have considered and adapted a sample of the functions from the CEC 2013 benchmark (focussing on unimodal problems), see [31]. We have used two versions of the ill-conditioned ellipsoid since the use of two versions was relevant to demonstrate the performance of the local search. The ill-conditioned ellipsoid f_3 is the one in [30] while the ellipsoid f_2 has been introduced by us in this study. The condition numbers of both f_2 and f_3 worsen with the dimensionality of the problem. However, they worsen with the dimensionality at different speeds, see Table 1 . Each problem has been scaled to 10, 30, and 50 dimensions and has been studied in $[-100, 100]^n$. Table 1 displays the functions and illustrates the shape of the corresponding basins of attraction. For each problem, a shift and a rotation has been applied: with reference to Table 1 the variable:

$$\mathbf{z} = \mathbf{Q}_p(\mathbf{x} - \mathbf{o})$$

where $\mathbf{o}$ is the shifting vector (the same used in [31]) and $\mathbf{Q}_p$ is a rotation matrix (a randomly generated orthogonal matrix) set for the pth problem.

The Pattern Search variants in this article have all been run with the initial radius $\rho = 0.1\times$ domain width $= 20$. This parameter has been set by using the indication in [49] and then tuning for our testbed. The threshold *thr* for the problems in Table 1 are reported in Table 2. The threshold values have been set empirically by testing values of the codomain that allowed that some points were stored in the data structure $\mathbf{V}$ while some others were discarded. See Section 6 for further considerations about this issue.

Although Pattern Search is a deterministic algorithm, its performance can depend on the initial point and, for Covariance Pattern Search, its performance depends also on the

Table 3 Average error avg ± standard deviation σ over 51 runs for the problems listed in Table 1: greedy Pattern Search (gPS) vs greedy Covariance Pattern Search (gCPS) with gCPS reference for Wilcoxon Test (W)

	gPS		gCPS		
	avg	σ	avg	σ	W
10 dimensions					
f_1	**0.0000e+00**	0.0000e+00	2.7768e-29	1.4329e-29	−
f_2	1.8198e+03	1.9132e+03	**1.1062e-03**	3.4915e-03	+
f_3	7.0241e+04	9.5716e+04	**4.1064e+03**	6.6534e+03	+
f_4	5.3135e+03	3.7177e+03	**3.3017e-06**	9.9933e-06	+
f_5	9.3944e+03	1.2996e+04	**3.0984e-25**	5.3080e-25	+
f_6	5.5475e+01	9.4900e+01	**2.6276e-05**	1.3087e-05	+
30 dimensions					
$f1$	**0.0000e+00**	0.0000e+00	1.3562e-28	5.1167e-29	−
$f2$	1.1636e+08	2.3036e+08	**6.9435e+05**	6.4717e+05	+
$f3$	2.9155e+05	2.4771e+05	**1.8116e+04**	1.1143e+04	+
$f4$	5.2064e+03	5.3776e+03	**4.6182e-13**	3.2833e-12	+
$f5$	8.4596e+03	3.4177e+04	**5.3340e-28**	3.2325e-27	+
$f6$	1.2427e+02	1.7049e+02	**6.4462e-05**	1.7549e-05	+
50 dimensions					
$f1$	**0.0000e+00**	0.0000e+00	4.1998e-28	9.9033e-29	−
$f2$	4.3878e+08	8.6339e+08	**1.6151e+07**	1.3937e+07	+
f_3	3.7522e+05	2.7408e+05	**5.1237e+04**	3.4165e+04	+
f_4	5.6398e+03	7.4905e+03	**5.5627e-22**	1.4649e-21	+
f_5	2.8261e+01	1.3792e+02	**7.8822e-27**	1.2792e-26	+
f_6	1.8153e+02	2.0042e+02	**1.0584e-04**	2.3244e-05	+

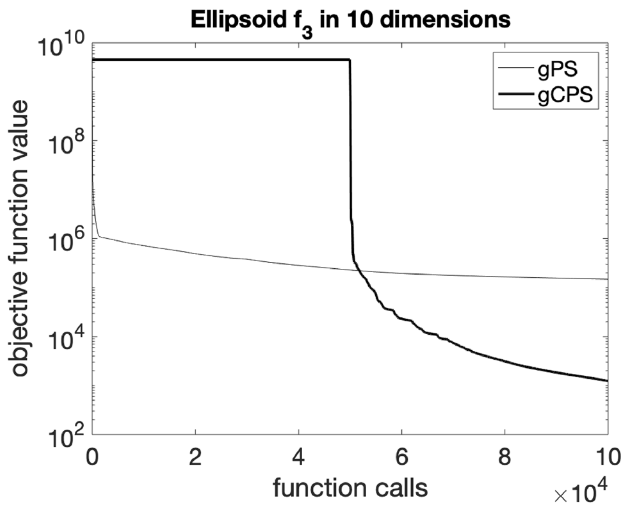


Fig. 11 Performance trend (logarithmic scale) of gPS vs gCPS for the ellipsoid f_3 in 10D

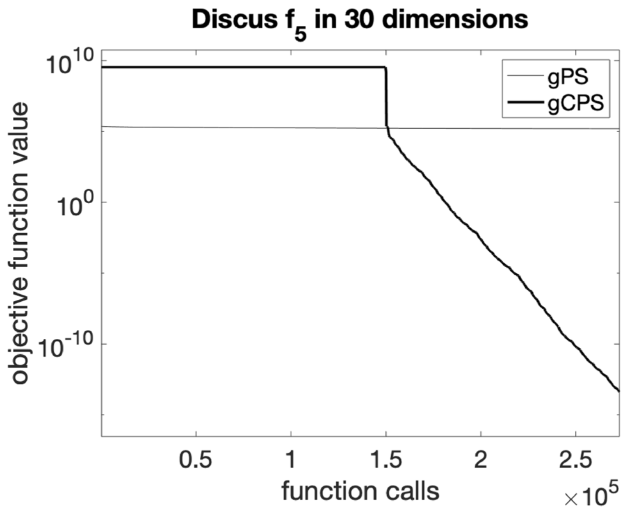


Fig. 12 Performance trend (logarithmic scale) of gPS vs gCPS for the discus f_5 in 30D

sampled points used to estimate the covariance matrix. Therefore, for each experiment, 51 independent runs have been performed and the associated mean value and ± standard deviation have been displayed.

Furthermore, to statistically investigate the question of whether the application of the proposed method results in performance gains, the Wilcoxon rank-sum test has been applied, see [17, 51]. In the Tables in this section, a "+" indicates that gCPS significantly outperforms competitor, a "-" indicates that the competitor significantly outperforms gCPS, and a "=" indicates that there is no significant difference in performance.

Table 4 Average error avg ± standard deviation σ over 51 runs for the problems listed in Table 1 subject to a random rotation at each run: greedy Pattern Search (gPS) vs greedy Covariance Pattern Search (gCPS) with gCPS reference for Wilcoxon Test

	gPS		gCPS		
	avg	σ	avg	σ	W
10 dimensions					
f_1	**0.0000e+00**	0.0000e+00	3.6603e-29	2.0133e-29	−
f_2	1.2340e+03	1.2462e+03	**3.7683e-04**	1.1634e-03	+
f_3	1.3739e+05	2.3991e+05	**2.9764e+03**	2.7313e+03	+
f_4	3.5615e+03	6.8974e+03	**3.8205e+01**	1.0569e+02	+
f_5	7.8684e+03	1.1461e+04	**1.0087e-23**	3.1002e-23	+
f_6	1.3676e+02	4.3249e+02	**2.6698e-05**	1.0627e-05	+
30 dimensions					
$f1$	**0.0000e+00**	0.0000e+00	1.3810e-28	5.9062e-29	−
$f2$	9.8109e+07	2.6281e+08	**7.2563e+05**	9.0521e+05	+
$f3$	1.7506e+05	1.8632e+05	**1.7248e+04**	1.0875e+04	+
$f4$	5.8954e+03	1.0909e+04	**2.2632e-01**	1.6163e+00	+
$f5$	1.1985e+03	5.2373e+03	**7.9322e-26**	4.2633e-25	+
$f6$	1.3485e+02	1.8275e+02	**6.4239e-05**	1.8127e-05	+
50 dimensions					
f_1	**0.0000e+00**	0.0000e+00	4.2394e-28	9.7414e-29	−
$f2$	8.2205e+08	1.7628e+09	**1.5302e+07**	1.2051e+07	+
f_3	3.1913e+05	2.8232e+05	**5.8034e+04**	3.0679e+04	+
f_4	9.3975e+03	1.0959e+04	**3.8535e-21**	1.4037e-20	+
f_5	1.2639e+02	4.5309e+02	**2.8811e-26**	1.2290e-25	+
f_6	1.8761e+02	1.8060e+02	**9.1000e-05**	1.8870e-05	+

All the algorithms in this section have been executed with a budget of $10000 \times n$ function calls where n is the problem dimensionality. To guarantee a fair comparison, the budget of the proposed Covariance Pattern Search has been split into two parts: $5000 \times n$ function calls have been used to build the covariance matrix $\mathbf{C}$, whilst $5000 \times n$ function calls have been spent to execute the algorithm. Due to the nature of Pattern Search, i.e. deterministic local search, the bound handling has been performed by saturating the design variable to the bound. We preferred the saturation to the bound over the toroidal insertion or reflection [13] since the latter two mechanisms would be equivalent to the sampling of a point. This sampling would disrupt the gradient estimation logic of Pattern Search.

Validation of Covariance Pattern Search

To validate the functioning of Covariance Pattern Search we compared gPS and gCPS detailed in Algorithms 2 and 6, respectively. It must be remarked that the algorithms have the same structure except for the matrix $\mathbf{B}$.

Table 3 shows the numerical results of gPS vs gCPS.

Table 5 Average error avg ± standard deviation σ over 51 runs for the problems listed in Table 1: greedy Covariance Pattern Search (gCPS) with gCPS reference for Wilcoxon vs Probabilistic Global Search Lausanne (PGSL) [45], Covariance Matrix Adaptive Evolution Strategy (CMAES) [20], and Whale Optimisation Algorithm (WOA) [34]

	PGSL			CMAES			WOA			gCPS	
	avg	σ	W	avg	σ	W	avg	σ	W	avg	σ
10 dimensions											
f_1	9.3911e-11	5.1542e-11	+	2.1589e-15	2.9083e-15	+	1.0214e-03	6.1787e-04	+	**2.7768e-29**	1.4329e-29
f_2	8.0344e+04	8.4760e+04	+	**1.4746e-15**	1.1443e-15	−	6.0491e+05	9.2856e+05	+	1.1062e-03	3.4915e-03
f_3	9.6851e+04	6.7100e+04	+	**1.0489e-15**	8.2847e-16	−	4.7535e+05	5.6977e+05	+	4.1064e+03	6.6534e+03
f_4	4.4803e+02	1.0111e+03	+	**1.8441e-14**	8.2847e-16	=	1.4087e+03	2.1832e+03	+	3.3017e-06	9.9933e-06
f_5	5.2501e+00	2.9491e+01	+	1.5539e-14	2.8869e-14	+	2.1305e+03	6.7707e+03	+	**3.0984e-25**	5.3080e-25
f_6	1.5045e-03	6.0847e-04	+	**9.8517e-13**	9.4715e-13	−	2.3042e-02	8.1963e-03	+	2.6276e-05	1.3087e-05
30 dimensions											
f_1	1.0067e-07	6.2153e-08	+	1.2862e-15	2.3923e-16	+	8.7062e-03	3.6387e-03	+	**1.3562e-28**	5.1167e-29
f_2	7.0010e+07	3.2542e+07	+	**1.2574e-15**	3.6076e-16	−	3.4145e+07	2.2654e+07	+	6.9435e+05	6.4717e+05
f_3	7.1768e+05	3.0764e+05	+	**1.1737e-15**	2.4995e-16	−	6.7736e+05	5.3299e+05	+	1.8116e+04	1.1143e+04
f_4	1.3380e-01	3.0262e-01	+	**1.2556e-14**	2.0370e-14	=	2.5780e+02	3.3855e+02	+	4.6182e-13	3.2833e-12
f_5	1.6386e-01	3.3514e-01	+	8.7935e-15	1.1728e-14	+	6.8149e-02	3.6128e-01	+	**5.3340e-28**	3.2325e-27
f_6	2.2218e-02	7.4020e-03	+	**1.1134e-11**	9.1361e-12	−	6.2324e-02	1.1138e-02	+	6.4462e-05	1.7549e-05
50 dimensions											
f_1	8.0474e-07	4.4488e-07	+	1.1017e-15	4.7700e-16	+	2.2366e-02	6.9947e-03	+	**4.1998e-28**	9.9033e-29
f_2	1.9910e+09	7.2381e+08	+	**9.1890e-15**	2.0913e-15	−	3.8112e+08	1.7733e+08	+	1.6151e+07	1.3937e+07
f_3	3.8261e+06	1.1263e+06	+	**1.2302e-15**	7.1779e-16	−	1.1605e+06	6.5088e+05	+	5.1237e+04	3.4165e+04
f_4	1.2430e-01	2.2500e-01	+	1.2606e+04	3.3045e+04	+	1.1822e+03	3.5112e+03	+	**5.5627e-22**	1.4649e-21
f_5	4.4505e-01	1.3905e+00	+	9.0248e-03	4.4211e-02	+	2.5731e+01	1.4155e+02	+	**7.8822e-27**	1.2792e-26
f_6	1.7938e+00	3.3942e+00	+	**4.8125e-11**	2.8066e-11	−	9.1245e-02	1.9504e-02	+	1.0584e-04	2.3244e-05

Numerical results in Table 3 show that, apart from f_1, the proposed gCPS consistently outperforms the standard gPS across the three dimensions under consideration. The problem f_1 is characterised by a central symmetry, meaning that the rotation is ineffective and any reference system would broadly perform in the same way. In this sense, gCPS "wastes" half of its budget to find a search direction whilst gPS uses its entire budget to search for the optimum. With regards to the other five problems, the use of the eigenvectors of $\mathbf{C}$ as search directions not only systematically improves upon gPS but appears to solve some problems for which gPS would fail, see f_4 and f_5.

Figures 11 and 12 illustrate the performance trends of gPS and gCPS for the ill-conditioned ellipsoid f_3 and the discus f_5 functions, respectively (see Table 1).

As shown in both Figs. 11 and 12, the performance trend of gCPS does not lead to any improvement for the first half of the budget. This is due to the computational cost that we have allocated to the fitness landscape analysis. Following the fitness landscape analysis, gCPS quickly improves upon the initial value and reaches a solution whose objective function value is orders of magnitude lower than that detected by the standard gPS.

To ensure that the results are not biased by specific rotation matrices, the experiments have been repeated on the problems in Table 1 with a modified experimental condition. For each run a new rotation matrix has been generated and both gPS and gCPS have been run on the rotated problem. An additional 51 independent runs have been executed under this new condition. Table 4 displays the results for this set of experiments with random matrices.

Numerical results in Table 4 show that gCPS maintains the same performance irrespective of the rotation matrix. These results allow us to conjecture that the proposed mechanism of optimising along the direction of the eigenvectors is effective and that the generation of the covariance matrix is robust.

Comparison Against Other Algorithms

We have compared gCPS against the following three algorithms:

– Probabilistic Global Search Lausanne (PGSL) [45];
– Covariance Matrix Adaptive Evolution Strategy (CMAES) [20];
– Whale Optimisation Algorithm (WOA) [34].

Table 6 Average error avg ± standard deviation σ over 51 runs for the problems listed in Table 1 subject to random rotation at each run: greedy Covariance Pattern Search (gCPS) with gCPS reference for Wilcoxon vs Probabilistic Global Search Lausanne (PGSL) [45], Covariance Matrix Adaptive Evolution Strategy (CMAES) [20], and Whale Optimisation Algorithm (WOA) [34]

	PGSL			CMAES			WOA			gCPS	
	avg	σ	W	avg	σ	W	avg	σ	W	avg	σ
10 dimensions											
f_1	9.3911e-11	5.1542e-11	+	1.6014e-15	1.3692e-15	+	1.0884e-03	7.2892e-04	+	**3.6603e-29**	2.0133e-29
f_2	8.0344e+04	8.4760e+04	+	**1.2555e-15**	7.2449e-16	−	1.3625e+06	1.5057e+06	+	3.7683e-04	1.1634e-03
f_3	9.6851e+04	6.7100e+04	+	**1.5290e-15**	1.1994e-15	−	1.0848e+06	1.2060e+06	+	2.9764e+03	2.7313e+03
f_4	4.4803e+02	1.0111e+03	+	**6.5782e-15**	1.1870e-14	−	1.3327e+03	2.4924e+03	+	3.8205e+01	1.0569e+02
f_5	5.2501e+00	2.9491e+01	+	9.9735e-15	1.1232e-14	+	3.2346e+03	1.1942e+04	+	**1.0087e-23**	3.1002e-23
f_6	1.5045e-03	6.0847e-04	+	**1.0527e-12**	8.4632e-13	−	2.9221e-02	3.4593e-02	+	2.6698e-05	1.0627e-05
30 dimensions											
f_1	7.9845e-08	5.4124e-08	+	1.2502e-15	5.7056e-16	+	8.6939e-03	3.4908e-03	+	**1.3810e-28**	5.9062e-29
f_2	7.3259e+07	3.5457e+07	+	**3.8067e-15**	1.3425e-15	−	4.7345e+07	3.7792e+07	+	7.2563e+05	9.0521e+05
f_3	1.1240e+06	5.8522e+05	+	**1.3093e-15**	5.4862e-16	−	1.5773e+06	1.8218e+06	+	1.7248e+04	1.0875e+04
f_4	2.2207e-01	4.4356e-01	+	**1.2729e-01**	1.4109e-01	=	1.1381e+03	2.6930e+03	+	2.2632e-01	1.6163e+00
f_5	5.4108e-01	1.0309e+00	+	9.1100e+02	4.9137e+03	+	6.2115e+02	3.0196e+03	+	**7.9322e-26**	4.2633e-25
f_6	2.2442e-02	6.8551e-03	+	**1.9809e-11**	2.4626e-11	−	5.7210e-02	1.7384e-02	+	6.4239e-05	1.8127e-05
50 dimensions											
f_1	7.4289e-07	2.8571e-07	+	1.0433e-15	3.9174e-16	+	2.2087e-02	7.2395e-03	+	**4.2394e-28**	9.7414e-29
f_2	1.9884e+09	8.9922e+08	+	**1.3719e-14**	6.3332e-15	−	3.5048e+08	2.9147e+08	+	1.5302e+07	1.2051e+07
f_3	4.3028e+06	1.4041e+06	+	**1.4062e-15**	5.6871e-16	−	1.1245e+06	+ 8.5818e+05	+	5.8034e+04	3.0679e+04
f_4	2.3907e+00	1.6129e+01	+	7.1375e+03	1.9090e+04	+	9.8717e+01	2.4355e+02	+	**3.8535e-21**	1.4037e-20
f_5	2.5581e-01	3.3437e-01	+	1.9817e+04	1.0947e+05	+	1.5174e+03	8.4013e+03	+	**2.8811e-26**	1.2290e-25
f_6	1.1738e+00	2.4084e+00	+	**4.7564e-11**	2.6402e-11	−	8.4904e-02	2.1078e-02	+	9.1000e-05	1.8870e-05

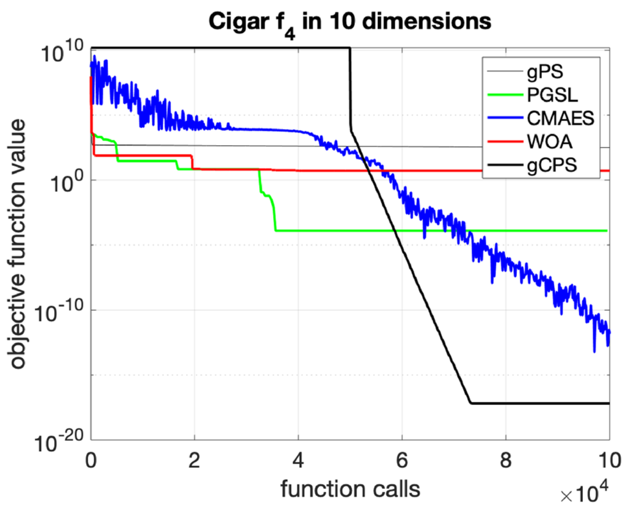


Fig. 13 Performance trend (logarithmic scale) for the Bent Cigar f_4 in 10 dimensions

The motivation behind these three competitors is the following: (1) PGSL, like gCPS, is a modern Pattern Search implementation; (2) CMAES is a prevalent algorithm that, like gCPS, is based on theoretical considerations about multivariate distributions and the covariance matrix; (3) WOA is a population-based metaheuristic recently proposed in the literature.

Furthermore, it must be remarked that as shown in [19], CMAES frameworks display excellent performance on unimodal problems and can be used as high-performance local search, see [35]. In order to provide further experimental evidence of this fact and that the comparison against CMAES is fair, we reported in Appendix the performance of two local search algorithms against the proposed gCPS. Also, the WOA, as shown in [34], is an emerging metaheuristic that can solve unimodal problems more efficiently than a large number of metaheuristics used for comparison.

We used the parameters suggested by the authors of the original papers/software: PGSL has been run with 20 intervals, 6 sub-intervals, and scale factors 0.96 and $n^{-\frac{1}{n}}$, respectively (see [45]); CMAES has been run according to the implementation in [18] and the default parameters therein (initial σ set to one third of the domain); WOA has been run with 30 agents and for $334 \times n$ iterations (to make the budget equal to that of gCPS). Numerical results with fixed and random rotation are displayed in Tables 5 and 6, respectively.

Numerical results show that the proposed gCPS outperforms PGSL and WOA consistently for all the problems

Table 7 Holm-Bonferroni Procedure with gCPS as reference (Rank 4.1944e+00)

	R_j	z_j	p_j	$\frac{\delta_j}{j}$	Test
CMAES	4.1667e+00	-7.4536e-02	9.4058e-01	5.00e-02	Failed to Reject
PGSL	2.4444e+00	-4.6957e+00	2.6564e-06	2.50e-02	Rejected
gPS	2.2222e+00	-5.2920e+00	1.2097e-07	1.67e-02	Rejected
WOA	1.9722e+00	-5.9628e+00	2.4788e-09	1.25e-02	Rejected

under consideration. It must be observed that while gCPS is designed as a "precision tool" to handle unimodal problems with rotation, WOA is a global optimiser to detect a solution close the optimum in mult*i*-modal scenarios. In this sense we expected gCPS to outperform WOA. The comparison between gCPS and PGSL is more interesting. PGSL contains a pattern search logic with the same search directions of PS (directions of the variables) and various other elements integrated to improve its performance. We observe that for the problems under examination, the simple exploratory logic of gCPS appears more effective than that of the PGSL which is more sophisticated. This fact confirms that the selection of proper search directions is fundamental in GPS.

Finally, the comparison of gCPS vs CMAES has been included with the deliberate intent of emphasising the limitations and potentials of the proposed approach. For about half of the problems gCPS is competitive or outperforms CMAES, whilst for some other problems CMAES achieves results that are orders of magnitude better in performance than gCPS. This fact happens especially in cases of steep gradients, such as the ellipsoid functions f_2 and f_3. The superiority of CMAES for these problems highlights an area for improvement of gCPS: unlike CMAES that has an ingenious adaptive mechanism, CPS employs a naive search operation based on the same step size in all the directions. The latter is likely not suited for highly ill-conditioned problems. Nonetheless, the proposed logic can be exported to other schemes and more sophisticated operators, e.g. an adaptive search radius, can enhance upon the current performance.

An example of performance trends for all the algorithms involved in this study is illustrated in Fig. 13.

To further strengthen the statistical analysis of the presented results, we performed the Holm-Bonferroni [22] procedure for the five algorithms and thirty-six problems (6 objective functions ×3 levels of dimensionality ×2 variants of rotation) under consideration. The results of the Holm-Bonferroni procedure are presented in Table 7. The Holm-Bonferroni procedure consists of the following. A score R_j for $j = 1, \dots, N_A$ (where N_A is the number of algorithms under analysis, $N_A = 5$ in this paper) has been assigned. The score has been assigned in the following way: for each problem, a score of 5 is assigned to the algorithm displaying the best performance, 4 is assigned to the second best, 3 to the third, and so on. For each algorithm, the scores obtained on each problem are summed up and averaged over the 36 test problems. With the calculated R_j values, gCPS has been taken as the reference algorithm. R_0 indicates the rank of gCPS, and with R_j for $j = 1, \dots, N_A - 1$ 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 the 36 test problems. By means of the z_j values, the corresponding cumulative normal distribution values p_j have been calculated, see [15]:

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

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 7 displays the ranks, z_j values, p_j values, and corresponding δ/j obtained. Moreover, it is indicated whether the null-hypothesis (that the two algorithms have indistinguishable performance) is "Rejected", i.e. the algorithms have statistically different performance, or "Failed to Reject" if 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 7 show that gCPS and CMAES have better performance than the other algorithms and that gCPS significantly outperforms PGSL, gPS, and WOA for the 36 problems under consideration. Conversely, the ranking values of gCPS and CMAES are very similar with slightly better performance for gCPS. However, the Holm-Bonferroni procedure shows that, statistically, gCPS has the same performance as CMAES.

Limitations of Covariance Pattern Search and Future Developments

Although gCPS and the concept of Covariance Pattern Search appear to be extremely promising, we acknowledge that there are opportunities for improvement. Hence, this section highlights the limitations of the proposed approach and some ideas on how the research can be moved forward.

At first we wish to comment on a feature of gCPS as opposed to a strict limitation: gCPS works as a local search

Table 8 Average error avg ± standard deviation σ over 51 runs for the problems listed in Table 1: greedy Covariance Pattern Search (gCPS) with gCPS reference for Wilcoxon vs Nelder-Mead Algorithm and Broyden–Fletcher–Goldfarb–Shanno algorithm

	NMA			BFGS			gCPS	
	avg	σ	W	avg	σ	W	avg	σ
10 dimensions								
f_1	2.8054e+02	7.2855e+02	+	1.8757e-20	2.4576e-21	+	**2.7768e-29**	1.4329e-29
f_2	1.2455e+07	2.8567e+07	+	**1.8956e-13**	4.1016e-14	−	1.1062e-03	3.4915e-03
f_3	1.5440e+06	3.1652e+06	+	**6.6365e-11**	2.6183e-12	−	4.1064e+03	6.6534e+03
f_4	2.9546e+00	1.7939e+01	+	**1.2818e-09**	2.0603e-09	−	3.3017e-06	9.9933e-06
f_5	1.0910e-08	9.9139e-09	+	3.9439e-12	7.2576e-12	+	**3.0984e-25**	5.3080e-25
f_6	5.1821e+02	4.8882e+02	+	**1.1967e-07**	4.3589e-07	=	2.6276e-05	1.3087e-05
30 dimensions								
f_1	1.6594e+04	8.2122e+03	+	4.3857e-20	1.6043e-20	+	**1.3562e-28**	5.1167e-29
f_2	2.3459e+10	1.4851e+10	+	**1.3352e-09**	3.3772e-10	−	6.9435e+05	6.4717e+05
f_3	2.4294e+07	2.0055e+07	+	**3.2738e-11**	1.5479e-12	−	1.8116e+04	1.1143e+04
f_4	1.2650e-01	6.5822e-01	+	1.8926e-08	2.5487e-08	+	**4.6182e-13**	3.2833e-12
f_5	1.6725e-08	2.3167e-08	+	5.9737e-13	1.2614e-12	+	**5.3340e-28**	3.2325e-27
f_6	5.1073e+03	4.2297e+03	+	**1.2390e-05**	2.84323e-064	=	6.4462e-05	1.7549e-05
50 dimensions								
f_1	2.9945e+04	1.4066e+04	+	8.5482e-20	1.9014e-20	+	**4.1998e-28**	9.9033e-29
f_2	3.6255e+11	1.9270e+11	+	**1.5972e-07**	3.7724e-08	−	1.6151e+07	1.3937e+07
f_3	7.0929e+07	4.2330e+07	+	**9.2288e-11**	4.6933e-12	−	5.1237e+04	3.4165e+04
f_4	4.4606e+00	3.1855e+01	+	1.9741e-08	2.8660e-08	+	**5.5627e-22**	1.4649e-21
f_5	1.3769e-08	3.1929e-08	+	1.5341e-13	3.2062e-13	+	**7.8822e-27**	1.2792e-26
f_6	5.9676e+03	2.9671e+03	+	**1.1253e-05**	1.8968e-006	=	1.0584e-04	2.3244e-05

algorithm and thus may be inefficient on its own to address multimodal problems. This is of course a limitation of local search algorithms and can be overcome by integrating gCPS within a global search framework, see [39, 40], or by simply using a resampling mechanism to periodically restart the search in an unexplored area of the decision space, see [6].

A second issue is that the performance of Covariance Pattern Search depends on the threshold parameter *thr*, see Algorithm 5 . This parameter should allow the generation of a data set that describes the geometry of the problem. With reference to minimisation, a value that is too low would result in a small or empty data set, while a value that is too high would generate a dataset that does not contain the geometrical features of the basin of attraction. Furthermore, this parameter is problem dependent as it depends on the specific objective function values. This means that to ensure high performance of gCPS, a tuning of this parameter is currently performed. Besides being inelegant, this procedure can also be computationally expensive in the high dimensional domain.

The third limitation that we identified (especially when we compared the performance of gCPS against CMAES) is in the exploratory radius (step size) and the way it is handled. The current naive implementation that uses the same radius along all the exploratory directions is inefficient in the case of high ill-conditioning. Although the preliminary analysis allows the detection of search directions with very different gradients, as shown in Figs. 9 and 10, this information is not fully exploited during the optimisation. Qualitatively, we may say that we would like small steps when the gradient is high and large steps when it is low. Further works will investigate the use of gradient-based and adaptive step sizes.

Finally, further work will be made to integrate within the proposed gCPS techniques for constraint handling, such as the use of external penalties.

Conclusion

This paper proposes an implementation of Generalised Pattern Search (GPS) in which search directions are given by the eigenvectors of the covariance matrix associated with the distribution of points whose objective function value falls below a prearranged threshold.

The theoretical analysis carried out in this study shows that the proposed algorithm, namely Covariance Pattern Search, is always applicable and converges to a null-gradient point under the same conditions of GPS. The conjecture that the eigenvectors enable fast convergence is confirmed by the experimental results that show a clear superiority of Covariance Pattern Search with respect to its standard version. It should be observed that the proposed directions (eigenvectors) allow the identification of a high gradient direction which would enable the quick improvement upon an initial point.

Numerical results show that the specific implementation of Covariance Pattern Search proposed in this article outperforms, on a set of rotated problems, modern algorithms based on pattern search and swarm intelligence.

Appendix: Comparison Against Local Search

In this section, we report some additional results displaying the performance of gCPS against two local search algorithms, that is

– Nelder-Mead Algorithm (NMA) [37]
– Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm [16] with an estimation of the gradient such that it may be applied to black-box problems

Both NMA and BFGS have been run on the problems listed in Table 1 under the same experimental conditions of gCPS as in Tables 3 and 5, including computational budget, starting points, and rotation matrices. Numerical results in Table 8 show that gCPS outperforms NMA for all of problems considered in this section. The comparison between gCPS and BFGS shows that gCPS outperforms BFGS in some problems and is outperfomed on others. The comparison of gCPS and BFGS is analogous to the comparison of gCPS and CMAES. The results in this section confirm the study reported in [19] which shows that for unimodal problems, algorithms based on CMAES tend to outperform NMA and BFGS. Furthermore, our results confirm that as reported in [19] Nelder-Mead deteriorates its performance when the number of dimensions increase while BFGS is able to display good performance on some problems including those in higher dimensions.

Compliance with Ethical Standards

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

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

References

1. Beale E. On an iterative method for finding a local minimum of a function of more than one variable. Tech. rep.: Princeton University; 1958.
2. Box MJ, Davies D, Swann WH. Non-linear optimisation techniques. London: Oliver & Boyd; 1969.
3. Brent RP. Algorithms for minimization without derivatives. Englewood Cliffs: Prentice-Hall; 1973.
4. Caponio A, Cascella GL, Neri F, Salvatore N, Sumner M. A fast adaptive memetic algorithm for on-line and off-line control design of PMSM drives. IEEE Trans Syst Man Cybern Part B. 2007;37(1):28–41.
5. Caraffini F, Neri F, Iacca G, Mol A. Parallel memetic structures. Inf Sci. 2013;227:60–82.
6. Caraffini F, Neri F, Passow BN, Iacca G. Re-sampled inheritance search: high performance despite the simplicity. Soft Comput. 2013;17(12):2235–56.
7. Caraffini F, Neri F, Picinali L. An analysis on separability for memetic computing automatic design. Inf Sci. 2014;265:1–22.
8. Cartis C, Fiala J, Marteau B, Roberts L. Improving the flexibility and robustness of model-based derivative-free optimization solvers. ACM Trans Math Softw. 2019;45(3):32:1–32:41.
9. Cartis C, Roberts L. A derivative-free gauss-newton method. Math Program Comput. 2019;11(4):631–74.
10. Conn AR, Scheinberg K, Vicente LN. Introduction to derivative-free optimization. MPS-SIAM book series on optimization. London: SIAM; 2009.
11. Davidon W. Variable metric method for minimization. SIAM Journal of Optimization **1**, 1–17 (1991). The article was originally published as Argonne National Laboratory Research and Development Report 5990, May 1959 (revised November 1959)
12. Dennis J Jr, Torczon V. Direct search methods on parallel machines. SIAM J Optim. 1991;1(4):448–74.
13. Eiben AE, Smith JE. Introduction to evolutionary computation. 2nd ed. Berlin: Springer; 2015.
14. Feller W. An introduction to probability theory and its applications, vol. 1. Amsterdam: Wiley; 1968.
15. Fisher RA. The design of experiments, 9th edn. Macmillan (1935); 1971.
16. Fletcher R. Practical methods of optimization. 2nd ed. New York: Wiley; 1987.
17. 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.
18. Hansen, N. The CMA evolution strategy. 2012.http://www.lri.fr/~hansen/cmaesintro.html
19. Hansen N, Auger A, Ros R, Finck S, Pošík, P. Comparing results of 31 algorithms from the black-box optimization benchmarking bbob-2009. In: GECCO'10: Proceedings of the 12th annual conference companion on Genetic and evolutionary computation, pp. 1689–1696. ACM 2010.
20. Hansen N, Ostermeier A. Completely derandomized self-adaptation in evolution strategies. Evol Comput. 2001;9(2):159–95.
21. Hart WE, Krasnogor N, Smith JE. Memetic evolutionary algorithms. In: Hart WE, Krasnogor N, Smith JE, editors. Recent advances in memetic algorithms. Berlin: Springer; 2004. p. 3–27.

22. Holm S. A simple sequentially rejective multiple test procedure. Scand J Stat. 1979;6(2):65–70.
23. Hooke R, Jeeves TA. Direct search solution of numerical and statistical problems. J ACM. 1961;8:212–29.
24. Iacca G, Neri F, Mininno E, Ong YS, Lim MH. Ockham's Razor in memetic computing: three stage optimal memetic exploration. Inf Sci. 2012;188:17–43.
25. Jana ND, Sil J, Das S. Continuous fitness landscape analysis using a chaos-based random walk algorithm. Soft Comput. 2018;22:921–48.
26. Jolliffe IT. Principal Component Analysis, 2nd edn. Springer Series in Statistics, 2002. Springer, New York.
27. Kaupe F Jr. Algorithm 178: direct search. Commun ACM. 1963;6(6):313–4.
28. Kennedy J, Eberhart RC. Particle swarm optimization. In: Proceedings of IEEE International Conference on Neural Networks,1995; pp. 1942–1948.
29. Lewis RM, Torczon V, Trosset MW. Direct search methods: then and now. J Comput Appl Math 2000;124(1), 191–207. Numerical Analysis 2000. Vol. IV: Optimization and Nonlinear Equations
30. 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.
31. Liang JJ, Qu BY, Suganthan PN, Hernández-Díaz AG. Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session on Real-Parameter Optimization. Tech. Rep. 201212, Zhengzhou University and Nanyang Technological University, Zhengzhou China and Singapore, 2013.
32. Malan KM, Engelbrecht AP. Quantifying ruggedness of continuous landscapes using entropy. In: 2009 IEEE Congress on Evolutionary Computation, 2009; pp. 1440–1447.
33. Malan KM, Engelbrecht AP. A survey of techniques for characterising fitness landscapes and some possible ways forward. Inf Sci. 2013;241:148–63.
34. Mirjalili S, Lewis A. The whale optimization algorithm. Adv Eng Softw. 2016;95:51–67.
35. Molina D, Lozano M, Garcia-Martinez C, Herrera F. Memetic algorithms for continuous optimization based on local search chains. Evol Comput. 2010;18(1):27–63.
36. Moser I. Hooke-jeeves revisited. In: Proceedings of the Eleventh Conference on Congress on Evolutionary Computation, CEC'09, 2009; pp. 2670–2676. IEEE Press.
37. Nelder A, Mead R. A simplex method for function optimization. Comput J. 1965;7:308–13.
38. Neri F. Linear algebra for computational sciences and engineering, 2nd edn. Springer, New York, 2019.
39. Neri F, Cotta C. Memetic algorithms and memetic computing optimization: a literature review. Swarm Evolut Comput. 2012;2:1–14.
40. Neri F, Cotta C, Moscato P. Handbook of Memetic Algorithms, Studies in Computational Intelligence, vol. 379. Springer, New York, 2011.
41. Neri F, Garcia XdT. Cascella GL, Salvatore N. Surrogate assisted local search on PMSM drive design. COMPEL: Int J Comput Math Electr Electron Eng. 2008;**27**(3):573–592.
42. Neri F, Rostami S. A local search for numerical optimisation based on covariance matrix diagonalisation. In: Castillo PA, Laredo JLJ, de Vega FF editors. 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, Springer, New York, 2020;12104, 3–19.
43. Neri F, Tirronen V. Recent advances in differential evolution: a review and experimental analysis. Artif Intell Rev. 2010;33(1–2):61–106.
44. Press WH, Teukolsky SA, Vetterling WT, Flannery BP. Numerical recipes in C: the art of scientific computing, 2nd edn. Cambridge University Press, New York, NY (1992)
45. Raphael B, Smith I. A direct stochastic algorithm for global search. Appl Math Comput. 2003;146(2):729–58.
46. Rios-Coelho A, Sacco W, Henderson N. A metropolis algorithm combined with hooke-jeeves local search method applied to global optimization. Appl Math Comput. 2010;217(2):843–53.
47. Stanimirovic P, Tasic M, Ristic M. Symbolic implementation of the hooke-jeeves method. Yugoslav J Oper Res. 1999;9:285–301.
48. Torczon V. On the convergence of pattern search algorithms. SIAM J Optim. 1997;7(1):1–25.
49. Tseng LY, Chen C. Multiple trajectory search for Large Scale Global Optimization. In: Proceedings of the IEEE congress on evolutionary computation, 2008; pp. 3052–3059.
50. Weber M, Neri F, Tirronen V. Distributed differential evolution with explorative–exploitative population families. Genet Program Evolvable Mach. 2009;10(4):343–71.
51. Wilcoxon F. Individual comparisons by ranking methods. Biometr Bull. 1945;1(6):80–3.
52. Zhang G, Rong H, Neri F, Pérez-Jiménez MJ. An optimization spiking neural P system for approximately solving combinatorial optimization problems. Int J Neural Sys. 2014; 24(5).