Covariance Pattern Search with Eigenvalue-determined Radii

Ferrante Neri
COL Laboratory, School of Computer Science
University of Nottingham
Nottingham, United Kingdom
ferrante.neri@nottingham.ac.uk

Yuyang Zhou
Department of Computer Science
Imperial College London
London, United Kingdom
yz10020@ic.ac.uk

***Abstract*—Effective implementations of Memetic Algorithms often integrate, within their design, problem-based pieces of information. When no information is known, an efficient MA can still be designed after a preliminary analysis of the problem. This approach is usually referred to as Fitness Landscape Analysis (FLA). This paper proposes a FLA technique to analyse the epistasis of continuous optimisation problems and estimate those directions, within a multi-dimensional space, associated with maximum and minimum directional derivatives. This estimation is achieved by making use of the covariance matrix associated with a distribution of points whose objective function value is below (in case of minimisation) a threshold. The eigenvectors and eigenvalues of the covariance matrix provide important pieces of information about the geometry of the problem and are then used to design a memetic operator that is a local search belonging to the family of generalised Pattern Search. A restarting mechanism enables a progressive characterisation of the fitness landscape. Numerical results show that the proposed approach successfully explore ill-conditioned basins of attractions and outperforms the standard pattern search as well as a pattern search recently proposed in the literature and partially based on a similar design logic. The proposed local search based on FLA also displays a performance competitive with that of other types of local search.**

***Index Terms*—Memetic Algorithms, Fitness Landscape Analysis, Pattern Search, Covariance Matrix, Local Search**

I. Introduction

In many research studies, Memetic Algorithms (MAs) are interpreted as an extension of Evolutionary Algorithms (EAs): MAs are often seen as EA frameworks endowed with Local Search (LS) to accelerate the search of the algorithm, see [1]–[3]. However, for over a decade now, MAs are no longer identified with a specific algorithm composed of an EA and LS. In recent studies, MAs are considered as instances of a broader subject namely Memetic Computing (MC). This subject investigates the design of optimisation algorithms composed of multiple and diverse search operators/memes, see [4].

The success of MC should be put in relation with the No Free Lunch (NLF) Theorems [5]: since there is no universal optimiser, the flexible nature of MC design promotes the definition of algorithmic frameworks that integrate pieces of information about the problem into algorithmic design. Typical examples of knowledge integration in MA design is in domain-specific search operators [6]–[8]. In [9] known instructions to solve the knapsack problem are the memes that evolve at run time, see [10]–[12].

An interesting modern tendency in MC is multi-tasking that is the simultaneous search within a domain of the optimum of multiple problems [13], [14]. This research sub-field is based on the idea that a solution of one problem can be related to the solution of another problem, see [15]–[17], and thus there is exportable knowledge among problems.

When a problem is unknown and thus there is no prior information available about the problem an effective MC design must analyse the problem to collect some hints that may enable the design of a high-performance MA. The process of analysing the problem and use the result of this analysis to design the search algorithm is often termed Fitness Landscape Analysis (FLA) [18] which has some early very successful implementations of MAs in the discrete domain, see [19]–[21]. In recent years, FLA in the continuous domain attracted the attention of researchers [22]–[24].

In [25] a FLA based LS was proposed. The FLA in [25] samples points in the domain to generate a data set of candidate solutions whose objective function value is below a prearranged threshold (in a minimisation scenario). This data set provides some useful pieces of information about the geometry of the problem. This information is exploited by calculating the covariance matrix associated with the data and then its eigenvectors. These eigenvectors are then used as a basis (reference system) to explore the space. More specifically, in [25], [26] the eigenvectors of the calculated covariance matrix are used to build the Pattern Matrix of an implementation of generalised Patter Search (PS) [27]. In [28], the same FLA has been successfully implemented in three LS algorithms composing Multiple Trajectory Search [29].

The present paper moves a step further with respect to the FLA and PS in [25]. We propose to use the eigenvalues of the covariance matrix to set the exploratory radius along each search direction (the directions of the eigenvectors). Furthermore, a restarting mechanism has been implemented to update the result of the FLA and progressively adapt the algorithm to the fitness landscape. This restarting mechanism also prevents from the manual setting of a problem-based

978-1-7281-8393-0/21/$31.00 ©2021 IEEE

threshold, see [25].

The remainder of this paper is organised in the following way. Section II introduces the notation, the Covariance Pattern Search (CPS) presented in [25] and outlines its theoretical principles. On the basis of these theoretical principles, Section III provides new theoretical observations which are the basis of the design of the proposed LS which is named Covariance Pattern Search with Eigenvalue-determined Radii and briefly indicated with eigenCPS. Section IV displays the numerical results of this study. Finally Section V provides the conclusions of this study.

II. Background: Covariance Pattern Search

In order to clarify the notation used in this article, we will refer to the minimisation 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 $D \subset \mathbb{R}^n$:

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

Each vector $\mathbf{x} \in D$ is expressed in the orthonormal reference system of its variables. This means that any vector $\mathbf{x}$ can be represented as the linear combination of the orthonormal basis $B_\mathbf{e} = \{\mathbf{e^1}, \mathbf{e^2}, \ldots, \mathbf{e^n}, \}$ where $\mathbf{e^k}$ is a vector of length n composed of all zeros and a 1 in the k^{th} design variable, see [30].

Generalised PS is a family of LS algorithms that makes use of a basis of vectors to explore the space by performing a step in each direction identified by the vectors composing the basis. Although in [27] it was conceptualised that any basis of vectors can be used, most of the implementations use the orthonormal basis B_e. In this paper we focus on the algorithmic structure proposed in one of the LS algorithms of of the Memetic Frameworks [29], [31], [32]. This implementation, from a candidate solution $\mathbf{x}$ and an exploratory radius ρ, for each direction i at first attempts to sample the trial solution $\mathbf{x^t}$:

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

and then if this move fails, i.e. $f(\mathbf{x^t}) \geq f(\mathbf{x})$, PS attempts to explore the opposite direction

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

If all the moves generating trial solutions failed then ρ is reduced, usually halved.

In [25], [28], this logic has been coupled with a FLA. After having set a problem specific threshold $thre$, several points are sampled within D and their objective function calculated. Those points whose objective function value is such that $f(\mathbf{x}) < thre$ are stored in a data structure $\mathbf{V}$:

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

With the points (vectors) in $\mathbf{V}$ mean vector and covariance matrix $\mathbf{C}$ are calculated. The mean vector μ is calculated as

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

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

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

The eigenvectors $\mathbf{P} = \left(\mathbf{p^1}, \mathbf{p^2}, \ldots, \mathbf{p^n}\right)$ of $\mathbf{C}$ are calculated (by Cholesky Factorisation). Since $\mathbf{C}$ is symmetric, it is diagonalizable and an orthogonal basis of its eigenvectors can be found, see [30]. These eigenvectors are used as the basis to explore the space in a PS logic. This means that CPS differs from PS from the equations used to calculate the trial solutions, that is eq. (1) and eq. (2) are replaced with

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

and

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

The pseudocode of the resulting CPS is displayed in Algorithm 1.

Algorithm 1 Covariance Pattern Search

```
INPUT x, f(x), D, sample size, thre and budget
for s = 1 : sample size do
    Sample a point x in D and store in V
end for
Keep samples with f(x) < thre in V
Process V to calculate the covariance matrix C
Apply Cholesky Factorisation on C to calculate the eigenvectors P = (p^1, p^2, ..., p^n)

ρ = width of D
while budget and precision conditions on ρ do
    x^t = x
    for i = 1 : n do
        x^t = x − ρ · p^i
        if f(x^t) < f(x) then
            x = x^t
        else
            x^t = x + (ρ/2) · p^i
            if f(x^t) < f(x) then
                x = x^t
            end if
        end if
    end for
    if x has not been updated then
        ρ = ρ/2
    end if
end while
RETURN x
```

The rationale behind the choice of using the eigenvectors $\mathbf{p^i}$ is due to the fact that the matrix $\mathbf{P}$ whose columns are the eigenvectors $\mathbf{p^i}$

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

is the transformation matrix that diagonalizes the matrix $\mathbf{C}$ that is

$$\mathbf{\Lambda} = \mathbf{P^{-1}CP} = \mathbf{P^T CP} \tag{5}$$

where $\mathbf{\Lambda}$ is a diagonal matrix whose diagonal elements are the eigenvalues of $\mathbf{C}$ and $\mathbf{P^{-1}} = \mathbf{P^T}$ since $\mathbf{P}$ is an orthogonal matrix. 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 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 [33].

Furthermore, it was shown in [28] that, if the sampling of points in $\mathbf{V}$ describes the geometry of the basins of attraction, the directions of the eigenvectors identify the maximum and minimum directional derivative. Numerical results in [25] and [28] show that CPS consistently outperforms the PS that uses B_e.

In order to highlight the rationale of CPS, Fig. 1 displays the directional derivatives along the directions of the eigenvectors (solid line) and the orthonormal vectors of B_e (dashed lines) passing through the optimum. Fig. 1 refers to the shifted and rotated ellipsoid function in two dimensions:

$$\begin{aligned}
&\textbf{INPUT } \mathbf{x}\\
&\mathbf{z} = \mathbf{Q}\left(\mathbf{x} - \mathbf{o}\right)\\
&f\left(\mathbf{x}\right) = \textstyle\sum_{i=1}^{2} \left(10^6\right)^{\frac{i-1}{1}} z_i^2
\end{aligned}$$

where the shift vector is

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

and the rotation matrix is

$$\mathbf{Q} = \begin{pmatrix} -0.502 & -0.864 \\ -0.864 & -0.502 \end{pmatrix}.$$

We may observe that the directions of $\mathbf{p^i}$ identify the maximum and minimum directional derivatives, that is the directions alongside fitness landscape is steepest and nearly flat. The choice of these directions enables quick improvements and detection of high-quality solutions.

III. Covariance Pattern Search with Eigenvalue-determined Radii

The rationale of CPS synthetically outlined in Fig 1 induces the following observation: since the directions of eigenvectors identify the steepest and flattest directions of the fitness landscape, an effective CPS should exploit this piece of information by connecting the exploratory radius ρ with the directional gradients. More specifically, an effective LS would move large steps when the fitness landscape is flat as we know, on the basis of the FLA, that no major changes in the objective function values are expected in short steps. Conversely, alongside the steepest directions we would like that the algorithm performs small steps so that the large variations of the fitness landscape are not neglected during the search.

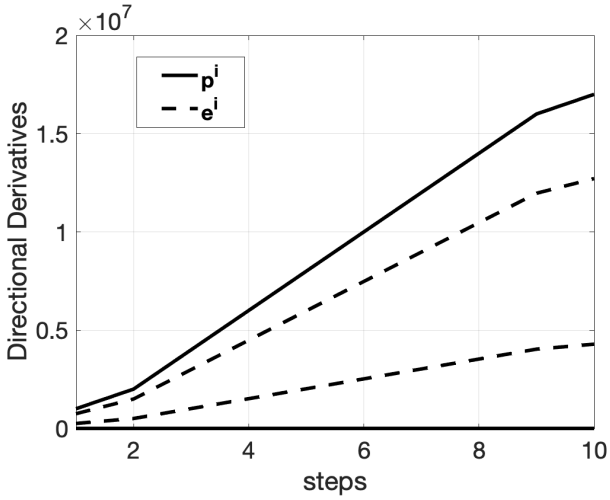

Fig. 1. Directional derivatives along the directions of the eigenvectors of the covariance matrix $\mathbf{p^i}$ and the orthonormal vectors $\mathbf{e^i}$

Thus, we propose to set an exploratory radius for each search direction, that is each eigenvector $\mathbf{p^i}$. To set the exploratory radius we use the vector

$$\mathbf{d} = \left(\ d_1, d_2, \ldots d_n\ \right)^T$$

which contains a piece of information about the directional derivative alongside the directions of the n eigenvectors.

Unfortunately, the directional derivatives are not directly known since they require the knowledge of the optimum. In order to address this issue, this paper proposes a FLA technique to estimate the directional derivative alongside the directions of the eigenvectors and integrate the achieved knowledge in the setting of the exploratory radii.

Let us consider again the covariance matrix $\mathbf{C}$ calculated as in CPS in Section II. 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

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

the corresponding eigenvalues.

It must be observed that since $\mathbf{C}$ is symmetric then the eigenvalues are all real numbers, see [30]. 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 $B_\mathbf{e}$, we may express it by 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 then we can express it in the basis of 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

$$\mathbf{C_y} = (\mathbf{P^T X_c})(\mathbf{P^T X_c})^{\mathbf{T}} = \mathbf{P^T X_c X_c^T P} = \mathbf{P^T C P} \quad (6)$$

where

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

From Eq. (5) and Eq. (6), it follows that

$$\mathbf{C_y} = \mathbf{\Lambda}. \quad (7)$$

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 samples used to calculate the covariance matrix $\mathbf{C}$ are selected to ensure 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 of notation that the optimum is in the null vector $\mathbf{o}$. 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. When we pose $f(\mathbf{x}) = thre$, we have that $f(\mathbf{x^i}) - f(\mathbf{o}) = thre^*$ is also a constant. Thus,

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

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

The standard deviations of vectors in the basis of eigenvectors is the square root of corresponding variance. A larger deviation in a direction where samples spread wider, thus the corresponding $\mathbf{x^i}$ has a larger l and a smaller directional derivative. Consider $\mathbf{x^i}$ and $-\mathbf{x^i}$ are two points along the same direction $\mathbf{p^i}$, which are the leftmost and rightmost points whose objective function values are $thre$. The deviation of these two points is their average modulus, as

$$\sqrt{\lambda_i} = \sqrt{\frac{1}{2}((\mathbf{x^i} - \mathbf{0})^2 + (-\mathbf{x^i} - \mathbf{0})^2)} = l$$

Thus, for a given direction $\mathbf{p^i}$, the square root of eigenvalue $\sqrt{\lambda_i}$ closely relates to the average modulus of vectors along this direction. We can conclude that the modulus l of a point $\mathbf{x^i}$ along that direction is directly correlated to the eigenvalue λ_i.

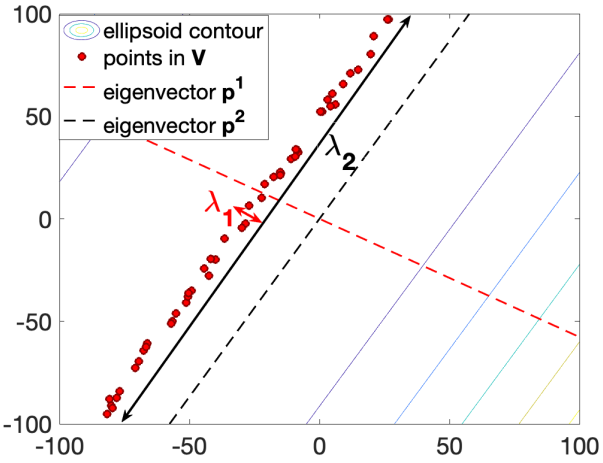

Fig. 2. Distribution of points in $\mathbf{V}$, directions of the eigenvectors and meaning of eigenvalues ($\lambda_1 = 1.5221$ and $\lambda_2 = 4324.1$) in the domain for the rotated and shifted ellipsoid in two dimensions.

The meaning of eigenvalues is highlighted in Fig. 2 where for the rotated and shifted ellipsoid in two dimensions, the points in $\mathbf{V}$, the directions of the eigenvectors, and the eigenvalues are displayed. It must be noticed that the eigenvalues are $\lambda_1 = 1.5221$ and $\lambda_2 = 4324.1$. These numbers reflect the distribution of the points which appear like a thin and long line. Also, as shown by the contour, along the direction of the first eigenvector the fitness landscape is very steep whereas along the direction of the second eigenvector the landscape is nearly flat.

For this reason, we propose to set the elements of the vector $\mathbf{d}$ associated with the direction of the eigenvector $\mathbf{p^i}$ equal to the square root of the corresponding eigenvalue λ_i:

$$d_i = \sqrt{\lambda_i}$$

Each d_i value is multiplied by the exploratory radius ρ whose functioning is the same as that of CPS described in Algorithm 1. Thus, we may consider that each search direction i has its own exploratory radius $\rho \cdot d_i$.

The proposed eigenCPS also overcomes one of the limitations of the CPS, that is the setting of the parameter $thre$ for each optimisation problem. The algorithm is run $iter$ times. In each sub-run, the FLA is performed and then CPS is run with the eigenvalue determined radii. During the first sub-run $sample\ size$ points in $\mathbf{V}$ are sampled within the entire D and the best $accept\ size$ are retained. In the following sub-runs, $sample\ size$ points in $\mathbf{V}$ are sampled within that portion of D that is within the distance $K_{\mathbf{V}} \times \rho$ from the current best $\mathbf{x}$ where the values of ρ and $\mathbf{x}$ are those returned by the previous sub-run. The best $accept\ size$ points are retained in $\mathbf{V}$.

In addition to the above issue, eigenCPS employs a restarting mechanism. The radius ρ is then re-initialised to $K_\rho \times \rho$ at each restart. The logic is to link the initialisation to the success of the previous local run. If the previous local run

was unsuccessful, a large exploratory radius searches in a a large space to possibly detect a better solution.

We may observe that the logic of CPS is retained and the threshold $thre$ is implicit: it is in each sub-run, the worst objective function value among those of the $accept\ size$ selected points. In this way, eigenCPS at each sub-run moves and, with progressive accuracy, characterises the fitness landscape.

Although eigenCPS is a LS which can be considered a valuable meme for continuous optimisation, it has some global propertied and is thus able to handle simple multimodal landscapes. Algorithm 2 displays the functioning of the proposed eigenCPS.

Algorithm 2 Covariance Pattern Search with Eigenvalue-determined Radii (eigenCPS)

INPUT $\mathbf{x}$, $f(\mathbf{x})$, D, $sample\ size$, $accept\ size$, $iter$ and $budget$
$\rho =$ width of D
for $i = 1 : iter$ **do**
  for $s = 1 : sample\ size$ **do**
    Sample a point in D around the current best $\mathbf{x}$ in radius $K_{\mathbf{V}} \times \rho$ and store in $\mathbf{V}$
  end for
  Keep $accept\ size$ samples with best values $f(\mathbf{x})$ in $\mathbf{V}$
  Process $\mathbf{V}$ to calculate the covariance matrix $\mathbf{C}$
  Apply Cholesky Factorisation on $\mathbf{C}$ to extract the eigenvectors $\mathbf{P} = \left(\mathbf{p^1}, \mathbf{p^2}, \ldots, \mathbf{p^n}\right)$ and eigenvalues $(\lambda_1, \lambda_2, \ldots, \lambda_n)$
  Calculate square root of eigenvalues λ as $\mathbf{d} = (d_1, d_2, \ldots, d_n) = \left(\sqrt{\lambda_1}, \sqrt{\lambda_2}, \ldots, \sqrt{\lambda_n},\right)$

  $\rho = K_\rho \times \rho$
  while $budget$ and precision conditions on ρ **do**
    $\mathbf{x^t} = \mathbf{x}$
    for $i = 1 : n$ **do**
      $\mathbf{x^t} = \mathbf{x} - \rho \cdot \mathbf{p^i} \cdot d_i$
      if $f\left(\mathbf{x^t}\right) < f(\mathbf{x})$ **then**
        $\mathbf{x} = \mathbf{x^t}$
      else
        $\mathbf{x^t} = \mathbf{x} + \frac{\rho}{2} \cdot \mathbf{p^i} \cdot d_i$
        if $f\left(\mathbf{x^t}\right) < f(\mathbf{x})$ **then**
          $\mathbf{x} = \mathbf{x^t}$
        end if
      end if
    end for
    if $\mathbf{x}$ has **not** been updated **then**
      $\rho = \frac{\rho}{2}$
    end if
  end while
  RETURN $\mathbf{x}$
end for

IV. Numerical Results

In order to experimentally demonstrate the effectiveness of the proposed eigenCPS, a set of functions from the IEEE CEC2013 benchmark [34] has been selected and adapted. Since eigenCPS is a LS we selected all the unimodal problems, hence reproducing the testbed of CPS used in [25]. We also reproduced both the versions of ellipsoid presented in [25] (f_2 and f_3). The condition number of these two ellipsoids worsen with dimensionality at different speeds. Finally, in order to show that eigenCPS is capable, to some extent, to handle multimodal fitness landscapes, we included two simple multimodal functions from [34]. The list of the functions used in this study is displayed in Table I. As shown in Table I, each problem has been shifted and rotated: the variables $\mathbf{x}$ is transformed into $\mathbf{z}$. The shifting vector $\mathbf{o}$ of [34] has been used. The rotation matrices $\mathbf{Q}$ have been randomly generated. One matrix $\mathbf{Q}$ has been generated for each problem and dimensionality value.

TABLE I
Objective functions used in this study

Domain	
$[-100, 100]^n$	
Shift and Rotation	
INPUT $\mathbf{x}$ $\mathbf{z} = \mathbf{Q}(\mathbf{x} - \mathbf{o})$	
function name	function calculation
sphere	$f_1 = \sum_{i=1}^{n} z_i^2$
ellipsoid 1	$f_2 = \sum_{i=1}^{n} 50\left(i^2 z_i\right)^2$
ellipsoid 2	$f_3 = \sum_{i=1}^{n} \left(10^6\right)^{\frac{i-1}{n-1}} z_i^2$
bent cigar	$f_4 = z_1^2 + 10^6 \sum_{i=2}^{n} z_i^2$
discus	$f_5 = 10^6 z_1^2 + \sum_{i=2}^{n} z_i^2$
sum of powers	$f_6 = \sqrt{\sum_{i=1}^{n} \lvert z_i \rvert^{\left(2+4\frac{i-1}{n-1}\right)}}$
Schwefel 2.21	$f_7 = \max_{i=1,\ldots,n} \lvert z_i \rvert$
Rosenbrock	$f_8 = \sum_{i=1}^{n-1} \left(100\left(z_i^2 - z_{i+1}\right)^2 + \left(z_i - 1\right)^2\right)$
Rastrigin	$f_9 = 10n + \sum_{i=1}^{n} \left(z_i^2 - 10\cos\left(2\pi z_i\right)\right)$

The numerical results are presented in two blocks of results. At first we test the effectiveness of FLA and different exploratory radii for each direction. To pursue this aim with comparing the performance of eigenCPS against that of its predecessors, that is we run

- Patter Search (PS) [29];
- Covariance Patter Search (CPS) [25];
- Covariance Pattern Search with Eigenvalue-determined Radii (eigenCPS).

These three algorithms have been run with initial $\rho = |D|$ where $|D|$ indicates the width of the domain (all the problems are hypercubical and $|D| = 200$), precision condition to stop the search $\rho \leq 10^{-15}$.

The threshold values $thre$ required by CPS are listed below

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

The eigenCPS uses $iter = 5$ outer loops, each divided into $samplesize = 200 \times n$ function calls for FLA, in which $acceptsize = 5 \times n$ best samples are used to build the

covariance matrix $\mathbf{C}$, and $budget = 800 \times n$ function calls to execute the pattern search. In the first iteration of eigenCPS, the sampling center is zero vector and radius is ρ. In addition, ρ is not increased before the local run. The parameters $K_{\mathbf{V}}$ and K_ρ are set equal to 100 and 10, respectively.

All the algorithms in this study have been run on the problems in Table I in 10, 30 and 50 dimensions. All the experiments have been run for $5000 \times n$ function calls. The budget of CPS include the budget used for the FLA: $2500 \times n$ function calls have been used to build the covariance matrix $\mathbf{C}$ whilst $2500 \times n$ function calls have been spent to execute the algorithm. The budget of eigenCPS includes $1000 \times n$ function calls to build the covariance matrix $\mathbf{C}$. The bound handling has been performed by saturating the design variable to the bound.

Each algorithm in this study for each problem had been run 30 times. We strengthen the statistical significance of the tests by the application of the Wilcoxon rank-sum test.In the Tables in this section, a "+" indicates that eigenCPS significantly outperforms the competitor algorithm indicated in the heading of the corresponding column, a "-" indicates that eigenCPS is significantly outperformed by the competitor algorithm and a "=" indicates that there is no significant difference in performance.

Numerical results in 10, 30 and 50 dimensions are displayed in Tables II, III, IV, respectively.

TABLE II
AVERAGE ERROR AVG ± STANDARD DEVIATION σ OVER 30 RUNS FOR THE PROBLEMS $f_1 - f_9$ IN 10 DIMENSIONS.

	PS		CPS		eigenCPS
	$\mu \pm \sigma$	W	$\mu \pm \sigma$	W	$\mu \pm \sigma$
f_1	**0.00e+00**±0.00e+00	=	7.74e-29 ±1.18e-28	+	6.73e-30 ±3.63e-29
f_2	1.64e+04 ± 1.02e+04	+	1.10e+03 ± 2.75e+03	+	**7.96e-03** ± 2.42e-02
f_3	9.48e+04 ± 9.67e+04	+	9.92e+03 ± 9.23e+03	+	**1.31e+03** ± 1.83e+03
f_4	1.36e+04 ±8.31e+03	+	1.08e+04 ±8.14e+03	=	**8.84e+03** ±6.33e+03
f_5	6.12e+04 ± 1.65e+04	+	4.21e+02 ± 8.22e+02	+	**1.97e-13** ± 1.06e-12
f_6	1.07e-04 ± 2.95e-05	+	6.42e-05 ± 2.67e-05	+	**6.04e-07** ± 3.61e-07
f_7	1.56e+01 ± 1.99e+01	=	**3.70e+00** ± 5.02e+00	-	9.36e+00 ± 1.29e+01
f_8	6.06e+01 ± 1.67e+02	+	3.30e+01 ± 1.10e+02	+	**8.59e+00** ± 3.58e+01
f_9	6.79e+01 ± 3.47e+01	=	**5.98e+01** ±2.78e+01	=	6.40e+01 ±2.89e+01

TABLE III
AVERAGE ERROR AVG ± STANDARD DEVIATION σ OVER 30 RUNS FOR THE PROBLEMS $f_1 - f_9$ IN 30 DIMENSIONS.

	PS		CPS		eigenCPS
	$\mu \pm \sigma$	W	$\mu \pm \sigma$	W	$\mu \pm \sigma$
f_1	**2.37e-31** ±3.62e-31	-	4.93e-28 ±2.95e-28	=	3.71e-28 ±2.23e-28
f_2	2.63e+06 ±1.80e+06	+	2.15e+06 ±1.67e+06	+	**1.11e+06** ±9.86e+05
f_3	2.97e+05 ±1.06e+05	+	1.10e+05 ±7.82e+04	+	**2.67e+04** ±1.56e+04
f_4	1.19e+04 ±7.16e+03	+	**7.65e+03** ±7.03e+03	=	7.89e+03 ±7.19e+03
f_5	1.59e+05 ±3.45e+04	+	3.64e+01 ±6.14e+01	+	**1.15e-17** ±2.65e-17
f_6	1.51e-04 ±2.84e-05	+	1.47e-04 ±3.40e-05	+	**4.76e-06** ±1.12e-06
f_7	6.07e+01 ±1.80e+01	+	**2.09e+01** ±8.45e+00	-	4.54e+01 ±1.39e+01
f_8	2.80e+03 ±4.62e+03	+	1.22e+03 ±2.68e+03	+	**1.21e+02** ±2.87e+02
f_9	3.72e+02 ± 1.05e+02	=	3.69e+02 ±1.34e+02	=	**3.23e+02** ±1.18e+02

Numerical results show that eigenCPS consistently outperforms PS. The only exception is f_1 in 30 and 50 dimensions. This result is expected since PS already uses the optimal directions and step size for the sphere function f_1. The proposed eigenCPS employs part of budget to analyse a fitness landscape to identify the working conditions of PS. The comparison between eigenCPS and CPS show that eigenCPS on average outperforms CPS. The performance of eigenCPS appears to be superior to that of CPS for problems with one sensitive direction such as the discus function f_5, see [34]. The only case where CPS outperforms eigenCPS is f_7 Schwefel 2.21. This problem is characterised by a central symmetry. Thus, after the rotation angle (that is the basis of eigenvectors) has been identified, the best exploratory is the same alongside all the directions identified by the basis. However, Since eigenCPS estimates the eigenvalues on the basis of the samples in $\mathbf{V}$, the elements of the vector $\mathbf{d}$ (which determine the exploratory radii) are similar but not identical to each other.

TABLE IV
AVERAGE ERROR AVG ± STANDARD DEVIATION σ OVER 30 RUNS FOR THE PROBLEMS $f_1 - f_9$ IN 50 DIMENSIONS.

	PS		CPS		eigenCPS
	$\mu \pm \sigma$	W	$\mu \pm \sigma$	W	$\mu \pm \sigma$
f_1	**6.11e-31** ± 5.35e-31	-	1.01e-27 ± 5.06e-28	=	9.92e-28 ± 3.22e-28
f_2	8.78e+07 ± 6.12e+07	+	3.65e+07 ± 3.46e+07	=	**2.76e+07** ± 2.10e+07
f_3	5.31e+05 ± 1.34e+05	+	1.74e+05 ± 7.60e+04	+	**7.92e+04** ± 2.94e+04
f_4	2.04e+04 ± 1.37e+04	+	1.90e+05 ± 6.86e+05	+	**9.99e+03** ± 9.64e+03
f_5	1.97e+05 ± 2.77e+04	+	2.30e+02 ± 4.29e+02	+	**2.78e-19** ± 1.38e-18
f_6	1.97e-04 ± 3.71e-05	+	2.42e-04 ± 5.95e-05	+	**1.02e-05** ± 2.68e-06
f_7	7.53e+01 ± 1.15e+01	+	**4.03e+01** ± 7.64e+00	-	5.30e+01 ± 1.09e+01
f_8	**1.81e+01** ± 2.60e+01	=	6.43e+02 ± 1.83e+03	+	6.89e+01 ± 1.78e+02
f_9	**7.26e+02** ± 1.71e+02	=	8.16e+02 ± 2.93e+02	=	7.67e+02 ± 1.67e+02

Fig. 4 depicts one example of performance trend of the PS, CPS, and eigenCPS. We may observe that eigenCPS at each restarts better characterises the fitness landscape and detects new promising search strategies.

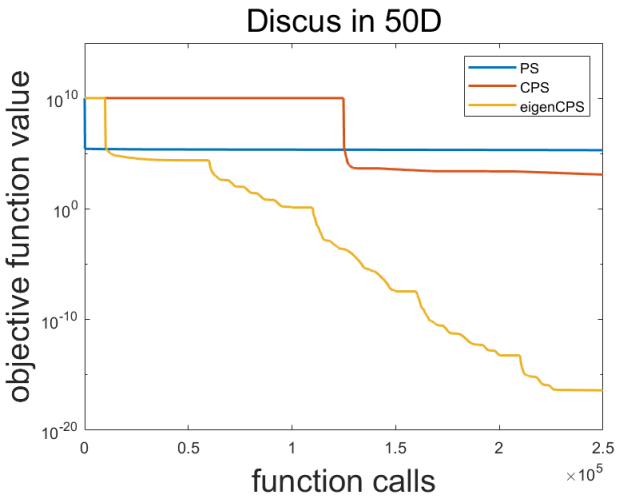

Fig. 3. Performance trend (logarithmic scale) of PS vs CPS vs eigenCPS for the Discus Function f_5 in 50D

To further study the performance of eigenCPS, we compared it against the following LS algorithms

- Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm [35] with an estimation of the gradient such that it may be applied to black-box problems;
- Rosenbrock Method (Rosenbrock) [36].

We chose two algorithms that search the closest local optimum by employing alternative logics with respect to eigenCPS. More specifically, BFGS has been chosen since it is a Quasi-Newtonian algorithm that estimates the gradient while Rosenbrock modified the search directions on the basis of their success and performs an orthogonalisation to build a new basis. Tables V, VI, and VII display the results in 10, 30 and 50 dimensions respectively.

TABLE V
AVERAGE ERROR AVG $\pm$ STANDARD DEVIATION σ OVER 30 RUNS FOR THE PROBLEMS $f_1 - f_9$ IN 10 DIMENSIONS.

	BFGS		Rosenbrock		eigenCPS
	$\mu \pm \sigma$	W	$\mu \pm \sigma$	W	$\mu \pm \sigma$
f_1	3.87e-21 $\pm$ 5.33e-21	+	8.84e-30 $\pm$ 3.76e-29	=	**6.73e-30** $\pm$3.63e-29
f_2	**8.07e-13** $\pm$ 3.10e-13	-	4.14e+04 $\pm$ 3.92e+04	+	7.96e-03 $\pm$ 2.42e-02
f_3	**5.52e-11** $\pm$ 1.04e-11	-	1.34e+05 $\pm$ 1.16e+05	+	1.31e+03 $\pm$ 1.83e+03
f_4	**2.47e-01** $\pm$ 5.92e-01	-	1.28e+04 $\pm$ 9.47e+03	=	8.84e+03 $\pm$6.33e+03
f_5	1.99e-10 $\pm$ 2.52e-11	+	3.81e+04 $\pm$ 1.86e+04	+	**1.97e-13** $\pm$ 1.06e-12
f_6	**6.92e-08** $\pm$ 2.77e-08	-	1.01e-04 $\pm$ 3.50e-05	+	6.04e-07 $\pm$ 3.61e-07
f_7	6.72e+01 $\pm$ 3.84e+01	+	**1.04e+00** $\pm$ 4.75e+00	-	9.36e+00 $\pm$ 1.29e+01
f_8	**9.30e-01** $\pm$ 1.69e+00	-	1.65e+02 $\pm$ 2.81e+02	+	8.59e+00 $\pm$ 3.58e+01
f_9	6.27e+02 $\pm$ 2.54e+02	+	6.79e+01 $\pm$ 3.23e+01	=	**6.40e+01** $\pm$2.89e+01

TABLE VI
AVERAGE ERROR AVG $\pm$ STANDARD DEVIATION σ OVER 30 RUNS FOR THE PROBLEMS $f_1 - f_9$ IN 30 DIMENSIONS.

	BFGS		Rosenbrock		eigenCPS
	$\mu \pm \sigma$	W	$\mu \pm \sigma$	W	$\mu \pm \sigma$
f_1	1.19e-20 $\pm$ 1.49e-20	+	**5.26e-31** $\pm$ 2.32e-30	-	3.71e-28 $\pm$2.23e-28
f_2	**5.28e-10** $\pm$ 1.22e-10	-	3.40e+07 $\pm$ 3.54e+07	+	1.11e+06 $\pm$9.86e+05
f_3	**3.26e-11** $\pm$ 1.84e-12	-	5.66e+05 $\pm$ 3.38e+05	+	2.67e+04 $\pm$1.56e+04
f_4	**5.69e-01** $\pm$ 2.94e+00	-	7.22e+03 $\pm$ 7.16e+03	=	7.89e+03 $\pm$7.19e+03
f_5	1.90e-10 $\pm$ 1.65e-11	+	1.29e+05 $\pm$ 3.39e+04	+	**1.15e-17** $\pm$2.65e-17
f_6	**9.99e-08** $\pm$ 1.22e-08	-	1.23e-04 $\pm$ 2.39e-05	+	4.76e-06 $\pm$1.12e-06
f_7	1.19e+02 $\pm$ 3.97e+01	+	8.11e+01 $\pm$ 1.50e+01	+	**4.54e+01** $\pm$1.39e+01
f_8	**1.33e+00** $\pm$ 1.88e+00	-	1.48e+03 $\pm$ 4.31e+03	=	1.21e+02 $\pm$2.87e+02
f_9	1.98e+03 $\pm$ 4.80e+02	+	3.76e+02 $\pm$ 1.05e+02	+	**3.23e+02** $\pm$1.18e+02

TABLE VII
AVERAGE ERROR AVG $\pm$ STANDARD DEVIATION σ OVER 30 RUNS FOR THE PROBLEMS $f_1 - f_9$ IN 50 DIMENSIONS.

	BFGS		Rosenbrock		eigenCPS
	$\mu \pm \sigma$	W	$\mu \pm \sigma$	W	$\mu \pm \sigma$
f_1	3.03e-20 $\pm$ 2.59e-20	+	**1.74e-30** $\pm$ 9.06e-30	-	9.92e-28 $\pm$ 3.22e-28
f_2	**7.52e-08** $\pm$ 1.18e-08	-	3.70e+08 $\pm$ 4.45e+08	+	2.76e+07 $\pm$ 2.10e+07
f_3	**1.07e-10** $\pm$ 7.21e-12	-	7.69e+05 $\pm$ 5.03e+05	+	7.92e+04 $\pm$ 2.94e+04
f_4	**3.60e-01** $\pm$ 1.52e+00	-	1.07e+04 $\pm$ 9.99e+03	=	9.99e+03 $\pm$ 9.64e+03
f_5	1.18e-10 $\pm$ 8.29e-12	+	1.45e+05 $\pm$ 2.75e+04	+	**2.78e-19** $\pm$ 1.38e-18
f_6	**1.25e-07** $\pm$ 1.58e-08	-	1.40e-04 $\pm$ 2.64e-05	+	1.02e-05 $\pm$ 2.68e-06
f_7	1.25e+02 $\pm$ 3.39e+01	+	9.76e+01 $\pm$ 1.23e+01	+	**5.30e+01** $\pm$ 1.09e+01
f_8	**9.30e-01** $\pm$ 1.69e+00	-	4.23e+02 $\pm$ 1.79e+03	=	6.89e+01 $\pm$ 1.78e+02
f_9	2.91e+03 $\pm$ 7.72e+02	+	9.88e+02 $\pm$ 1.69e+02	+	**7.67e+02** $\pm$ 1.67e+02

Numerical results show that eigenCPS has a similar or a better performance than Rosenbrock in most cases. It can be noticed that Rosenbrock performs very well in f_1, which is the multi-dimensional sphere. Like in the case of PS, Rosenbrock initialises its search directions along the variables and with the same step size in all directions. This choice is the one the best suits the geometry of the sphere and Schwefel 2.21.

The comparison between eigenCPS and BFGS highlights the different working logic of the two algorithms. The results clearly show that in about half cases eigenCPS outperforms BFGS while in the other half of cases BFGS outperforms eigenCPS. Most importantly, the comparison between eigenCPS and BFGS highlights the limitations and thus room for improvement for the proposed eigenCPS. Problems $f_2 - f_4$, where BFGS yields results orders of magnitude better than those detected by eigenCPS, are characterised by some directions with high (directional) derivatives and other directions with low derivatives. Furthermore f_2 and f_3 are ill-conditioned. Although the eigenvalue-determined radii of eigenCPS appear to improve upon the CPS logic, the direct estimation of the gradient of BFGS seems to be more efficient than the estimation of derivative through data set proposed in this study. Further studies are needed to enhance the estimation of directional derivatives through the proposed FLA approach.

Fig. 4 displays the performance trend of BFGS, Rosenbrock, and eigenCPS for the discus function f_5. In this case, unlike $f_2 - f_4$ the proposed FLA effectively detects the sensitive direction of the discus function, see [34], and leads to a performance superior to that of BFGS. According to our interpretation, the proposed FLA is very effective to detect one direction whose derivative is larger than others. Conversely, when the derivative are more similar to each other (as in the case of ellipsoid) a correct estimation of derivatives according to the proposed FLA would require a large number of data in $\mathbf{V}$.

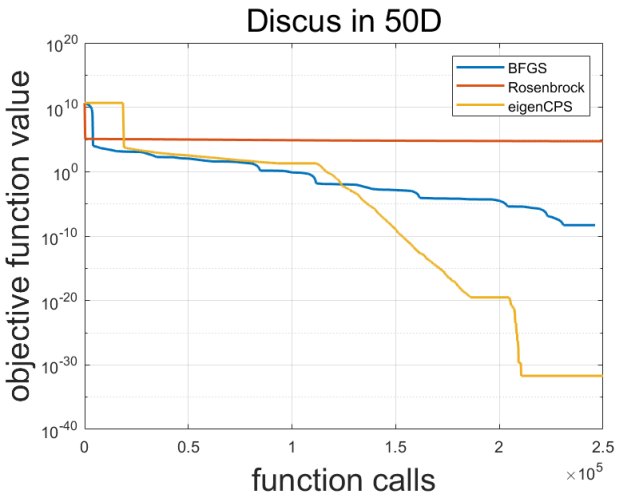

Fig. 4. Performance trend (logarithmic scale) of BFGS vs Rosenbrock vs eigenCPS for the Discus Function f_5 in 50D

V. CONCLUSION

This paper proposes a novel FLA to determine the directions of the landscape characterised by minimum and maximum derivatives and to estimate the value of derivatives alongside

these directions. The proposed FLA is based on sampling points below an implicit threshold and interpreting their distribution. The directions of interest are those of the eigenvectors of the covariance matrix associated with the sampled points while the derivatives alongside these directions are identified with the corresponding eigenvalues. This FLA is integrated in a resampling LS belonging to the family of generalised Pattern Search whose pattern is built through the basis of eigenvectors while a direction-specific exploratory radius related to the corresponding eigenvalue is associated with each search direction. A large exploratory radius is used to explore those directions corresponding to a flat landscape while a small radius is employed where the landscape is estimated to be steep.

Numerical results prove that the proposed logic outperforms its predecessor that uses eigenvectors to explore the space but the same radius alongside all the directions considered during the exploration. The comparison with other LS algorithms shows that the proposed eigenCPS displays a performance comparable to that of a gradient-based (Quasi-Newtonian) algorithm. The proposed eigenCPS can be considered as a LS with a learning element in it to be integrated into Memetic Frameworks. Future research directions will include more sophisticated mechanisms to collect data to support the accuracy of FLA.

REFERENCES

[1] P. Moscato and M. Norman, "A Competitive and Cooperative Approach to Complex Combinatorial Search," Tech. Rep. 790, 1989.

[2] F. Neri and C. Cotta, "Memetic algorithms and memetic computing optimization: A literature review," *Swarm and Evolutionary Computation*, vol. 2, pp. 1–14, 2012.

[3] C. Blum, R. Chiong, M. Clerc, K. A. D. Jong, Z. Michalewicz, F. Neri, and T. Weise, "Evolutionary optimization," in *Variants of Evolutionary Algorithms for Real-World Applications* (R. Chiong, T. Weise, and Z. Michalewicz, eds.), pp. 1–29, Springer, 2012.

[4] Y. Ong, M. H. Lim, and X. Chen, "Memetic computation—past, present future [research frontier]," *IEEE Computational Intelligence Magazine*, vol. 5, pp. 24–31, May 2010.

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

[6] Y. Yang, Y. Sun, and Z. Zhu, "Multi-objective memetic algorithm based on request prediction for dynamic pickup-and-delivery problems," in *2017 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1728–1733, 2017.

[7] L. Ma, J. Li, Q. Lin, M. Gong, C. A. Coello Coello, and Z. Ming, "Cost-aware robust control of signed networks by using a memetic algorithm," *IEEE Transactions on Cybernetics*, vol. 50, no. 10, pp. 4430–4443, 2020.

[8] Z. Zhou, X. Ma, and Z. Zhu, "Multi-objective memetic algorithm based on correlation priority for pickup-and-delivery problems," in *2019 IEEE Congress on Evolutionary Computation (CEC)*, pp. 990–997, 2019.

[9] L. Feng, A. Gupta, and Y.-S. Ong, "Compressed representation for higher-level meme space evolution: a case study on big knapsack problems," *Memetic Comp.*, vol. 11, pp. 3—17, 2019.

[10] G. Zhang, H. Rong, F. Neri, and M. J. Pérez-Jiménez, "An optimization spiking neural P system for approximately solving combinatorial optimization problems," *International Journal of Neural Systems*, vol. 24, no. 5, pp. 1440006:01–16, 2014.

[11] J. Dong, H. Rong, F. Neri, Q. Yang, M. Zhu, and G. Zhang, "An adaptive memetic p system to solve the 0/1 knapsack problem," in *2020 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8, 2020.

[12] M. Zhu, Q. Yang, J. Dong, G. Zhang, X. Gou, H. Rong, P. Paul, and F. Neri, "An adaptive optimization spiking neural p system for binary problems," *International Journal of Neural Systems*, vol. 31, 2021. 2050054.

[13] R. Chandra, Y.-S. Ong, and C.-K. Goh, "Co-evolutionary multi-task learning for dynamic time series prediction," *Applied Soft Computing*, vol. 70, pp. 576 – 589, 2018.

[14] L. Bai, Y.-S. Ong, T. He, and A. Gupta, "Multi-task gradient descent for multi-task learning," *Memetic Comp.*, vol. 12, pp. 355—369, 2020.

[15] Z. Zhou, X. Ma, Z. Liang, and Z. Zhu, "Multi-objective multi-factorial memetic algorithm based on bone route and large neighborhood local search for vrptw," in *2020 IEEE Congress on Evolutionary Computation (CEC)*, pp. 1–8, 2020.

[16] Z. Liang, J. Zhang, L. Feng, and Z. Zhu, "A hybrid of genetic transform and hyper-rectangle search strategies for evolutionary multi-tasking," *Expert Systems with Applications*, vol. 138, p. 112798, 2019.

[17] L. Feng, L. Zhou, A. Gupta, J. Zhong, Z. Zhu, K. C. Tan, and K. Qin, "Solving generalized vehicle routing problem with occasional drivers via evolutionary multitasking," *IEEE Transactions on Cybernetics*, pp. 1–14, 2019. to appear.

[18] K. M. Malan and A. P. Engelbrecht, "A survey of techniques for characterising fitness landscapes and some possible ways forward," *Information Sciences*, vol. 241, pp. 148 – 163, 2013.

[19] P. Merz, "Advanced fitness landscape analysis and the performance of memetic algorithms," *Evolutionary Computation*, vol. 12, no. 3, pp. 303–325, 2004.

[20] C. Reeves and J. E. Rowe, *Genetic Algorithms: Principles and Perspectives*. Springer, 2002.

[21] P. Merz and B. Freisleben, "Fitness landscape analysis and memetic algorithms for the quadratic assignment problem," *IEEE Transactions on Evolutionary Computation*, vol. 4, no. 4, pp. 337–352, 2000.

[22] K. M. Malan and A. P. Engelbrecht, "Quantifying ruggedness of continuous landscapes using entropy," in *2009 IEEE Congress on Evolutionary Computation*, pp. 1440–1447, 2009.

[23] K. M. Malan and A. P. Engelbrecht, "A progressive random walk algorithm for sampling continuous fitness landscapes," in *2014 IEEE Congress on Evolutionary Computation (CEC)*, pp. 2507–2514, 2014.

[24] N. D. Jana, J. Sil, and S. Das, "Continuous fitness landscape analysis using a chaos-based random walk algorithm," *Soft Computing*, vol. 22, pp. 921–948, 2018.

[25] F. Neri and S. Rostami, "A local search for numerical optimisation based on covariance matrix diagonalisation," in *Applications of Evolutionary Computation - 23rd European Conference, EvoApplications 2020, Held as Part of EvoStar 2020, Seville, Spain, April 15-17, 2020, Proceedings* (P. A. Castillo, J. L. J. Laredo, and F. F. de Vega, eds.), vol. 12104 of *Lecture Notes in Computer Science*, pp. 3–19, Springer, 2020.

[26] F. Neri and S. Rostami, "Generalised pattern search based on covariance matrix diagonalisation," *SN COMPUT. SCI*, vol. 2, 2021. 171.

[27] V. Torczon, "On the convergence of pattern search algorithms," *SIAM Journal on Optimization*, vol. 7, no. 1, pp. 1–25, 1997.

[28] F. Neri and Y. Zhou, "Covariance local search for memetic frameworks: A fitness landscape analysis approach," in *IEEE Congress on Evolutionary Computation, CEC 2020, Glasgow, United Kingdom, July 19-24, 2020*, pp. 1–8, IEEE, 2020.

[29] L.-Y. Tseng and C. Chen, "Multiple trajectory search for Large Scale Global Optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation*, pp. 3052–3059, 2008.

[30] F. Neri, *Linear Algebra for Computational Sciences and Engineering*. Springer, second ed., 2019.

[31] G. Iacca, F. Neri, E. Mininno, Y. S. Ong, and M. H. Lim, "Ockham's Razor in Memetic Computing: Three Stage Optimal Memetic Exploration," *Information Sciences*, vol. 188, pp. 17–43, 2012.

[32] F. Caraffini, F. Neri, and L. Picinali, "An analysis on separability for memetic computing automatic design," *Information Sciences*, vol. 265, pp. 1–22, 2014.

[33] I. T. Jolliffe, *Principal Component Analysis*. Springer Series in Statistics, Springer, 2nd ed., 2002.

[34] J. J. Liang, B. Y. Qu, P. N. Suganthan, and A. G. Hernández-Díaz, "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.

[35] R. Fletcher, *Practical Methods of Optimization*. New York, NY, USA: John Wiley & Sons, second ed., 1987.

[36] H. H. Rosenbrock, "An automatic Method for finding the greatest or least Value of a Function," *The Computer Journal*, vol. 3, no. 3, pp. 175–184, 1960.