Adaptive Covariance Pattern Search

Ferrante Neri[0000−0002−6100−6532]

Computational Optimisation and Learning (COL) Lab,
School of Computer Science, University of Nottingham, United Kingdom
ferrante.neri@nottingham.ac.uk

Abstract. Pattern search is a family of single solution deterministic optimisation algorithms for numerical optimisation. Pattern search algorithms generate a new candidate solution by means of an archive of potential moves, named pattern. This pattern is generated by a basis of vectors that span the domain where the function to optimise is defined. The present article proposes an adaptive implementation of pattern search that performs, at run-time, a fitness landscape analysis of the problem to determine the pattern and adapt it to the geometry of the problem. The proposed algorithm, called Adaptive Covariance Pattern Search (ACPS) uses at the beginning the fundamental orthonormal basis (directions of the variables) to build the pattern. Subsequently, ACPS saves the successful visited solutions, calculates the covariance matrix associated with these samples, and then uses the eigenvectors of this covariance matrix to build the pattern. ACPS is a restarting algorithm that at each restart recalculates the pattern that progressively adapts to the problem to optimise. Numerical results show that the proposed ACPS appears to be a promising approach on various problems and dimensions.

Keywords: Numerical Optimisation · Adaptive Algorithms · Pattern Search · Local Search · Covariance Matrix.

1 Introduction

Over the past decades, researchers in heuristic optimisation have striven to design algorithms that display a high performance on a wide set of problems. The No Free Lunch Theorems [24] indicate that high performance and versatility are conflicting features of algorithms and hence a trade-off should be found.

Researchers attempted to achieve this aim by exploiting the information available about the optimisation problem within the design of algorithm. Since generally the information about the problem is not available a priori, modern algorithms include mechanisms to make the algorithm suitable to the specific features of the problem. We identify here two algorithmic philosophies that, albeit ideologically different, may overlap in their practical implementations.

– **fitness landscape analysis**: the optimisation problem is analysed by a method, e.g. an Artificial Intelligence tool, and the results of the analysis are used to design the algorithm, see [8, 11, 12, 17, 16]

– **adaptive algorithms**: a feedback on the algorithmic behaviour on the specific problem is collected and used to adjust the algorithm, see [18, 21]

These two categories have a different focus but are not mutually exclusive: adaptive algorithms may analyse the fitness landscape. Fitness landscape analysis approaches focus on *how* the design is performed, that is on the basis of the analysis. On the other hand, adaptive approaches focus on *when* the design is performed/adjusted that is at run-time.

Furthermore, these two approaches, albeit overlapping, do not coincide. There exists a multitude of adaptive approaches that are not based on a fitness landscape analysis. For example, the self-adaptive Differential Evolution in [1] embeds the parameters into the solutions and propagates them to other candidate solutions by exploiting their evolution. Another example is in various hyper-heuristic schemes [2, 3] where the adaptation is based on the success of the parameters and the performance associated with them.

Conversely, at the intersection between fitness landscape analysis and adaptive approaches lies, for example, the Covariance Matrix Adaptive Evolution Strategy (CMAES) [6, 7]. This popular algorithm progressively adapts a multivariate Gaussian distribution from which candidate solutions are sampled. This adaptation is performed to increase the likelihood of previously successful candidate solutions. While evolving the CMAES distribution adapts to the geometry of the problem/local optimum.

By following a similar idea but fully embracing the fitness landscape analysis philosophy, paper [14] proposes a technique to analyse the geometry of the problem. This technique samples points whose objective function values are below a threshold, then calculates the covariance matrix associated with the distribution of these points and finally calculates the eigenvectors of the covariance matrix. Following this fitness landscape analysis, the directions of the eigenvectors are used to build the pattern of a pattern search, see [22] and Section 2 for definition of pattern. The resulting pattern search is called Covariance Pattern Search (CPS).

The results in [14] clearly show that the pattern based on the eigenvectors of a well-estimated covariance matrix outperforms the classical pattern based on the fundamental orthonomal basis (the directions of the variables). On the other hand, the application of CPS is impractical since it requires the setting of the above-mentioned threshold parameter for each optimisation problem. This setting is performed empirically and thus requires a considerable computational effort, especially in the high dimensional case. This feature makes CPS neither versatile (over various problems) nor easily scalable.

The present paper proposes a pattern search that makes use of the intuition of [14] about the search directions but follows an adaptive structure. The proposed algorithm, namely Adaptive Covariance Pattern Search (ACPS) at the beginning of the optimisation uses the fundamental orthonormal basis and while optimising the function stores the visited successful points. These points are then used to build the associated covariance matrix. The corresponding eigenvectors are then calculated. ACPS is then restarted with the newly calculated pattern. After

the restart, ACPS stores the visited successful points to build a more accurate pattern which is used after another restart. The procedure is repeated until exhaustion of the computational budget. Thus, ACPS can be seen as a restarting pattern search whose pattern changes at each restart and progressively adapts to the geometry of the local optimum.

The remainder of this article is organised in the following way. Section 2 provides the theoretical foundations of pattern search and its generalised abstraction which is then used in this study. Section 3 describes CPS and justifies the use of a basis of eigenvectors mentioned above to build the pattern. The limitations of CPS are also outlined in Section 3. Section 4 illustrates the proposed ACPS and provides details about its implementation and functioning. Section 5 tests the performance of ACPS and compares it against that of CPS, CMAES and a Quasi-Newtonian method. Finally, Section 6 presents the conclusions of this study.

2 Generalised Pattern Search

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

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

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

In [22] Pattern Search has been conceptualised and interpreted as a family of direct search methods, i.e. optimisation algorithms that do not require calculations of the gradient, and indicated as Generalised Pattern Search (GPS). GPS family is characterised by two elements:

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

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

$$\mathbf{G}_k = (\mathbf{M}_k, -\mathbf{M}_k, \mathbf{L}_K)$$

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 by $(p-2n)$ matrix that contains at least the null column vector $\mathbf{o}$. The search directions are given by the columns of the matrix:

$$\mathbf{P}_k = \mathbf{B}\mathbf{G}_k = (\mathbf{B}\mathbf{M}_k, -\mathbf{B}\mathbf{M}_k, \mathbf{B}\mathbf{L}_k) \tag{1}$$

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

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

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

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

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

The set of operations that yields a current best point is called the *exploratory move*. The exploratory move succeeds when a solution with better performance is detected, and fails when no update of the current best occurs. Within GPS family, various 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 [22], 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 $(\mathbf{B}\mathbf{M}_k, -\mathbf{B}\mathbf{M}_k)$ such that $f(\mathbf{x}_k + \mathbf{y}) < f(\mathbf{x}_k)$, then the exploratory move must produce a trial step $\mathbf{s}_k$ such that $f(\mathbf{x}_k + \mathbf{s}_k) < f(\mathbf{x}_k)$.
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 1.

3 Covariance Pattern Search

Covariance Pattern Search (CPS) [14] is an algorithm belonging to the GPS family that performs a fitness landscape analysis of the problem to determine

Algorithm 1 Generalized Pattern Search [22]

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 $\mathbf{x^t}_k \leftarrow \mathbf{x}_k + \mathbf{s}_k$ # Exploratory Move
  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 Δ_k
end while
$\mathbf{x} \leftarrow \mathbf{x_{k+1}}$
RETURN x

the matrix $\mathbf{B}$ and hence the pattern in Eq. (1). The fitness landscape analysis requires a threshold *thr* empirically set for each problem. Once this threshold has been set, 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.

For example, Fig. 1 displays the scatter plot of points whose objective function value is below $thr = 10^8$ for the shifted and rotated ellipsoid in two dimensions as in [10].

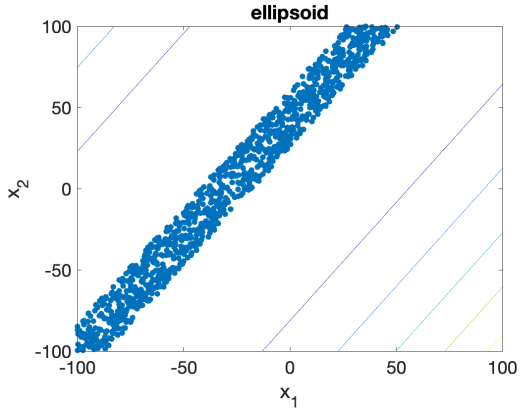


Fig. 1: Sample of points whose objective function (shifted and rotated 2D ellipsoid) is below the threshold $thr = 10^8$

For the sake of clarity the calculation of the ellipsoid is performed by means of the following procedure

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

where $\mathbf{Q}$ is a rotation matrix and $\mathbf{o_s}$ is a shift vector. Let us indicate with $\mathbf{V}$ the data structure where the m points whose objective function values are below thr (like the blue points in Fig. 1). These points can be interpreted as the samples of a multivariate distribution characterised by its mean vector μ and covariance matrix $\mathbf{C} = [c_{j,l}]$ where:

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

Subsequently, the n eigenvectors of the matrix $\mathbf{C}$ are calculated by means of Cholesky Factorisation, see [19, 13]. These eigenvectors are the columns $\mathbf{p}^i$ of a matrix $\mathbf{P} = \left(\mathbf{p}^1, \mathbf{p}^2, \ldots, \mathbf{p}^n\right)$. Algorithm 2 displays the pseudocode of the Fitness Landscape Analysis.

Algorithm 2 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 (uniform distribution) 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 μ and covariance matrix $\mathbf{C}$
Apply Cholesky Factorisation to extract the eigenvectors $\mathbf{P} = \left(\mathbf{p}^1, \mathbf{p}^2, \ldots, \mathbf{p}^n\right)$

CPS makes use of the directions of the eigenvectors as those of the basis of the decision space to build the pattern. Simply, CPS is a GPS where Eq. (1), (2), and (3) are respectively

$$\mathbf{P}_k = \mathbf{P}\mathbf{G}_k = \left(\mathbf{P}\mathbf{M}_k, -\mathbf{P}\mathbf{M}_k, \mathbf{P}\mathbf{L}_k\right) \tag{4}$$

$$\mathbf{s}_k = \Delta_k \mathbf{P}\mathbf{g}_k^i \tag{5}$$

and

$$\mathbf{x^t}_k = \mathbf{x}_k + \mathbf{s}_k. \tag{6}$$

The choice of exploring the decision space along the directions of the eigenvectors is justified by the fact that the covariance matrix $\mathbf{C}$ can be diagonalised

by means of the matrix $\mathbf{P}$. The directions of the eigenvectors can be interpreted as a new reference system characterised by a lack of correlation between pairs of variables. Consequently, the new reference system fits 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 [9]. Furthermore, it was shown in [15] 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. Thus, numerical results in [14] and [15] show that CPS consistently outperforms the standard pattern search. The latter is a pattern search that explores the space along the directions of the variables, that is a pattern search whose matrix $\mathbf{B}$ of Eq. (1) is the identity matrix.

The fitness landscape analysis in Algorithm 2 has been applied to three local search schemes in [15] and has been successfully integrated in a greedy implementation of pattern search in [14]. This greedy implementation based on [23] updates the current best solution as soon as a better solution is found. Also, this implementation explore both the orientation of each direction only when a move along the first orientation failed. Algorithm 3 illustrates the implementation details of CPS as in [14].

Algorithm 3 Covariance Pattern Search according to the greedy implementation in [14]

INPUT $\mathbf{x}$ and matrix $\mathbf{P}$ calculated in Algorithm 2
$k \leftarrow 1$
$\mathbf{x}_k \leftarrow \mathbf{x}$
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 $\mathbf{x}$

3.1 Limitations of Covariance Pattern Search

The main limitation of CPS is that a threshold *thr* must be empirically identified for each problem. A wrong choice of *thr* would result into a poor functioning of

the algorithm. An excessively low *thr* value would cause an empty data structure $\mathbf{V}$ (at least $n+1$ linearly independent vectors are needed to build the covariance matrix $\mathbf{C}$). An excessively high *thr* value would cause a data structure $\mathbf{V}$ that covers the entire domain and does not describe the geometry of the problem.

The choice of a proper *thr* is not only fundamental to guarantee the functioning of CPS but is also computationally expensive since it requires multiple trial and error tests. The difficulty and thus computational cost of the setting of *thr* grows with the dimensionality of the problem.

Another, albeit minor, limitation of CPS is that the algorithm is inherently a local search. The fitness landscape analysis aims at describing the geometry of a unimodal problem. Hence, CPS would likely not be usable to address multimodal problems.

4 The Proposed Adaptive Covariance Pattern Search

This paper proposes an adaptive algorithm, namely Adaptive Covariance Pattern Search (ACPS), that exploits the same mathematical principles of CPS. Unlike, CPS, the proposed ACPS does not require the setting of *thr*. Furthermore, the proposed ACPS, while is still to be considered a local search, can be successfully applied to some multimodal problems.

ACPS requires an initial solution $\mathbf{x}$, an initial exploration radius ρ, and stopping criteria on the minimum radius ρ_m and two computational budgets (maximum number of function calls), one local B_l one global B_g. The algorithm initialises the matrix $\mathbf{P}$ to the identity matrix of size n (this corresponds to the fundamental orthonormal basis of $\mathbb{R}^n$). ACPS perturbs $\mathbf{x}$ along the directions of the columns of the matrix $\mathbf{P} = \left(\mathbf{p}^1, \mathbf{p}^2, \ldots, \mathbf{p}^n\right)$. By following the logic of Algorithm 3, at the step k, for each variable i the algorithm calculates

$$\mathbf{x}^\mathbf{t} = \mathbf{x}_k - \rho \cdot \mathbf{p}^i \tag{7}$$

and updates $\mathbf{x}_{k+1}$ to $\mathbf{x}^\mathbf{t}$ if the trial solution $\mathbf{x}^\mathbf{t}$ outperforms the current best solution $\mathbf{x}_k$. If the exploration in Eq. (7) is unsuccessful (the trial solution does not outperform the current best solution), ACPS attempts to explore the other orientation, that is

$$\mathbf{x}^\mathbf{t} = \mathbf{x}_k + \frac{\rho}{2} \cdot \mathbf{p}^i, \tag{8}$$

and tests the performance of the newly calculated $\mathbf{x}^\mathbf{t}$ before moving to the following design variable. Every time the trial solution $\mathbf{x}^\mathbf{t}$ outperforms the current best solution $\mathbf{x}_k$, it occurs that the solution $\mathbf{x}^\mathbf{t}$ is saved and stored in the data structure $\mathbf{V}$. If the exploration along all the n directions fail then the exploration radius ρ is halved:

$$\rho = \frac{\rho}{2}$$

The algorithm is continued until the local budget B_l is exceeded or the search radius is smaller than the minimum radius ρ_m.

When the algorithm is stopped, the covariance matrix $\mathbf{C}$ is calculated on the basis of the samples in the data structure $\mathbf{V}$ and its eigenvectors $\mathbf{p}^1, \mathbf{p}^2, \ldots, \mathbf{p}^n$ calculated by Cholesky Factorisation. Thus, the matrix $\mathbf{P} = \left(\mathbf{p}^1, \mathbf{p}^2, \ldots, \mathbf{p}^n\right)$ is updated. The initial radius ρ is initialised to its original value and the algorithm restarted on the current best solution $\mathbf{x}_k$ for up to extra B_l function calls. Also the data structure $\mathbf{V}$ is reinitialised at each restart. ACPS is stopped when the condition on the global budget B_g is exceeded. Algorithm 4 displays the functioning of the proposed ACPS.

Algorithm 4 Adaptive Covariance Pattern Search

INPUT $\mathbf{x}$ of n elements
$\mathbf{P} \leftarrow$ identity matrix $\mathbf{I}$ of size n
while global budget condition B_g **do**
  $k, l \leftarrow 1$
  $\mathbf{x}_k \leftarrow \mathbf{x}$
  # Local run
  while local budget condition B_l and $\rho > \rho_m$ **do**
    $h \leftarrow k$
    for $i = 1 : n$ **do**
      $\mathbf{x}^\mathbf{t} \leftarrow \mathbf{x}_k - \rho \cdot \mathbf{p}^i$
      if $f\left(\mathbf{x}^\mathbf{t}\right) \leq f\left(\mathbf{x}_k\right)$ **then**
        $k \leftarrow k + 1$
        $\mathbf{x}_k \leftarrow \mathbf{x}^\mathbf{t}$
        $\mathbf{V}\left(l\right) \leftarrow \mathbf{x}^\mathbf{t}$
        $l \leftarrow l + 1$
      else
        $\mathbf{x}^\mathbf{t} \leftarrow \mathbf{x}_k + \frac{\rho}{2} \cdot \mathbf{p}^i$
        if $f\left(\mathbf{x}^\mathbf{t}\right) \leq f\left(\mathbf{x}_k\right)$ **then**
          $k \leftarrow k + 1$
          $\mathbf{x}_k \leftarrow \mathbf{x}^\mathbf{t}$
          $\mathbf{V}\left(l\right) \leftarrow \mathbf{x}^\mathbf{t}$
          $l \leftarrow l + 1$
        end if
      end if
    end for
    if $h = k$ #If no improvement occurred **then**
      $\rho \leftarrow \frac{\rho}{2}$
    end if
  end while
  Locally **RETURN** $\mathbf{x}$
  if the data structure $\mathbf{V}$ contains at least $n + 1$ vectors **then**
    Calculate the covariance matrix $\mathbf{C}$ from the samples in $\mathbf{V}$
    Apply Cholesky Factorisation to extract the eigenvectors $\mathbf{P} = \left(\mathbf{p}^1, \mathbf{p}^2, \ldots, \mathbf{p}^n\right)$
  end if
  Initialise ρ to its initial value and $\mathbf{V}$ to an empty data structure
end while
$\mathbf{x} \leftarrow \mathbf{x}_k$
RETURN $\mathbf{x}$

The proposed ACPS in Algorithm 4 can be considered as a restarting version of CPS in Algorithm 3 where the exploration radius is re-initialised at each restart, the search directions of the matrix $\mathbf{P}$ are updated at each restart, and the starting solution $\mathbf{x}$ (input) is the output of the local search at the previous stage (before the restart).

Furthermore, the proposed ACPS embeds within the search a fitness landscape analysis similar to that presented in Algorithm 2. However, while in CPS the threshold *thr* is prearranged and constant, in ACPS the threshold is adaptive and integrated within the search logic. At each restart ACPS uses a different threshold that is $f(\mathbf{x}_k)$ with $\mathbf{x}_k$ starting point of that *local run* (inner while loop in Algorithm 4).

At each local run, the proposed ACPS lowers (for minimisation) the value of the threshold and progressively refines the search directions (eigenvectors) to better fit the local features of the fitness landscape.

In order to illustrate the functioning of the proposed ACPS Fig. 2 shows the trajectory of the algorithm in four consecutive local runs. With the term *trajectory* we mean the current best solutions visited by ACPS. Fig. 2 refers to the shifted and rotated ellipsoid in two dimensions:

$$\begin{array}{l}\textbf{INPUT x}\\ \mathbf{z} \leftarrow \mathbf{Q}(\mathbf{x} - \mathbf{o_s})\\ f \leftarrow \sum_{i=1}^{2} \left(10^6\right)^{\frac{i-1}{1}} z_i^2\end{array}$$

where the shift vector is

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

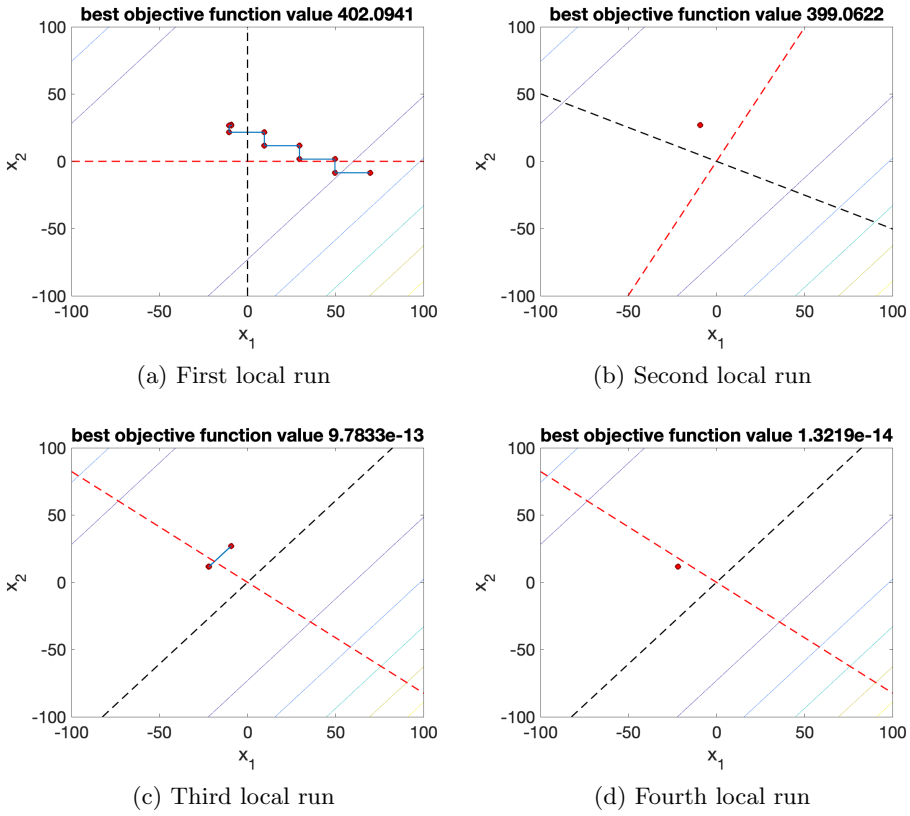
and the rotation matrix is

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

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

Fig 2 shows that in the first local run the algorithm, while moving along the directions of the variables (black and red dashed lines), approaches the optimum but remains still far from it. After the restart, ACPS uses the new search directions, i.e. the eigenvectors of the covariance matrix of the distribution of samples collected during the first local run. This system of reference appears to be ineffective. We may observe that during the second local run only a marginal improvement is achieved. However, the budget spent in the second local run is not wasted: the points sampled during the second local run enable the detection of an effective reference system (eigenvectors in the third local run). During the third local run ACPS exploits the benefits of the fitness landscape analysis and quickly detects a solution close to the optimum. The results are then refined in the fourth local run where the eigenvectors are slightly corrected.

It must be observed that the proposed ACPS resembles Rosenbrock Method [20] since they both use a basis of vector which is progressively adapted during the run (Rosenbrock Method belongs to the GPS family). However, the two algorithms are radically different in the way the basis is selected and updated.



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

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

Fig. 2: Trajectory of ACPS in four consecutive local runs on a rotated and shifted ellipsoid. The red dots are current best solutions and the trajectory of the ACPS is shown as a blue solid line. The black and red dashed lines indicate the eigenvectors. The best objective function values are on the top of each figure.

5 Numerical Results

In order to test and compare the performance of ACPS, a set of functions from the IEEE CEC2013 benchmark [10] have been selected and adapted. Since ACPS is a local search we selected all the unimodal problems, hence reproducing the testbed of CPS used in [14]. We also reproduced both the versions of ellipsoid presented in [14] (f_2 and f_3). The condition number of these two ellipsoids worsens with dimensionality at different speeds. In this paper, alongside bent cigar and discus we included their modified versions. Finally, in order to show that ACPS is capable, to some extent, to handle multimodal fitness landscapes, we included two simple multimodal functions from [10]. The list of the functions used in this study is displayed in Table 1. As shown in Table 1, each problem has been shifted and rotated: the variables $\mathbf{x}$ is transformed into $\mathbf{z}$. The shift vector

$\mathbf{o_s}$ of [10] 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 1: Objective functions used in this study

Domain	
$[-100, 100]^n$	
Shift and Rotation	
INPUT x $\mathbf{z} \leftarrow \mathbf{Q}(\mathbf{x} - \mathbf{o_s})$	
function name	function calculation
sphere	$f_1 \leftarrow \sum_{i=1}^{n} z_i^2$
ellipsoid 1	$f_2 \leftarrow \sum_{i=1}^{n} 50\left(i^2 z_i\right)^2$
ellipsoid 2	$f_3 \leftarrow \sum_{i=1}^{n} \left(10^6\right)^{\frac{i-1}{n-1}} z_i^2$
bent cigar	$f_4 \leftarrow z_1^2 + 10^6 \sum_{i=2}^{n} z_i^2$
modified bent cigar	$f_5 \leftarrow z_1^2 + 10^6 \left(\sum_{i=2}^{n} z_i\right)^2$
discus	$f_6 \leftarrow 10^6 z_1^2 + \sum_{i=2}^{n} z_i^2$
modified discus	$f_7 \leftarrow 10^6 z_1^2 + \left(\sum_{i=2}^{n} z_i\right)^2$
sum of powers	$f_8 \leftarrow \sqrt{\sum_{i=1}^{n} \lvert z_i \rvert^{\left(2+4\frac{i-1}{n-1}\right)}}$
Schwefel 2.21	$f_9 \leftarrow \max_{i=1,\ldots,n} \lvert z_i \rvert$
Rosenbrock	$f_{10} \leftarrow \sum_{i=1}^{n-1} \left(100\left(z_i^2 - z_{i+1}\right)^2 + \left(z_i - 1\right)^2\right)$
Rastrigin	$f_{11} \leftarrow 10n + \sum_{i=1}^{n} \left(z_i^2 - 10\cos\left(2\pi z_i\right)\right)$

We have compared ACPS against the following three algorithms:

- Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm [4] with an estimation of the gradient such that it may be applied to black-box problems;
- Covariance Matrix Adaptive Evolution Strategy (CMAES) [7];
- Covariance Pattern Search (CPS) [14].

The motivation behind these three competitors is the following: 1) BFGS is a Quasi-Newtonian algorithm that estimates the gradient and is here used a benchmark algorithm; 2) CMAES is a prevalent algorithm that, like CPS and ACPS are based on theoretical considerations about multivariate distributions and the covariance matrix; 3) CPS, is a pattern search whose pattern is build according to the same principles used in ACPS. Since the difference in performance between CPS and ACPS is due to the proposed adaptation, CPS can be seen as the direct competitor of ACPS in this study. All the algorithms in this study have a comparable time complexity of $\mathcal{O}\left(n^2\right)$.

The problems in Table 1 have been considered in 10, 30 and 50 dimensions. For each problem in Table 1, dimensionality level, and algorithm in this study, 51 independent runs have been performed. For each run, the four algorithms under consideration have been run with the same initial solution. All the algorithms

Table 2: Thresholds thr for CPS in 10, 30, and 50 dimensions

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

in this paper have been executed with a budget of $10000 \cdot n$ function calls where n is the problem dimensionality.

As reported in [14], the budget of CPS has been split in two parts: $5000 \cdot n$ function calls have been used to build the covariance matrix $\mathbf{C}$, whilst $5000 \cdot n$ function calls have been spent to execute the algorithm along the directions of its eigenvectors. In order to set the threshold thr, we used the recommended values in [14] and tuned them in order to let the algorithm achieve the best performance. Table 2 displays the thresholds used in this work.

Both CPS and ACPS have been run with an initial radius ρ equal to 10% of the domain width. As for ACPS, as mentioned above $B_g = 10000 \cdot n$. The maximum local budget B_l has been set equal to $1000 \cdot n$ and the minimum radius to restart the algorithm has been set equal to 10^{-15}. Table 3 displays the numerical results for the four algorithms under consideration and the problems in Table 1. Table 3 shows mean value and $\pm$ standard deviation over the 51 independent runs performed. 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 [5]. 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.

Results in Table 3 show that CMAES, CPS, and ACPS are better suited than BFGS to address the black box problems (and hence without information on the gradient). Numerical results show that ACPS consistently displays a better performance than CPS, thus indicating the effectiveness of the proposed adaptation. Furthermore, ACPS has a performance comparable to that of CMAES. In the low dimensional case ($n = 10$), both the algorithms detect solutions very close to the optimum for the nine unimodal problems ($f_1 - f_9$) and detect the global optimum in several runs while they detect a local minumum for the two multimodal problems ($f_{10} - f_{11}$). In higher dimensions, we observe that the performance of both CMAES and ACPS deteriorates on some problems and remain excellent on others. For example, CMAES performs extremely well on $f_2 - f_4$ regardless of number of variables while ACPS deteriorates its performance as the number of dimensions increases. Conversely, ACPS handles the $f_5 - f_7$ problems better than CMAES. On average, ACPS and CMAES display a similar performance across the entire set of functions over the three numbers of dimensions under consideration. Thus, ACPS appears to be more competitive with CMAES than CPS. Fig. 3 displays two examples of convergence trend for two randomly selected runs. The "staircase" trend of ACPS can be noticed: some

Table 3: Average error avg ± standard deviation σ over 51 runs for the problems listed in Table 1: Adaptive Covariance Pattern Search (ACPS) with ACPS reference for Wilcoxon vs Covariance Pattern Search (CPS) [14], Covariance Matrix Adaptive Evolution Strategy (CMAES) [7], and Broyden–Fletcher–Goldfarb–Shanno (BFGS) algorithm [4].

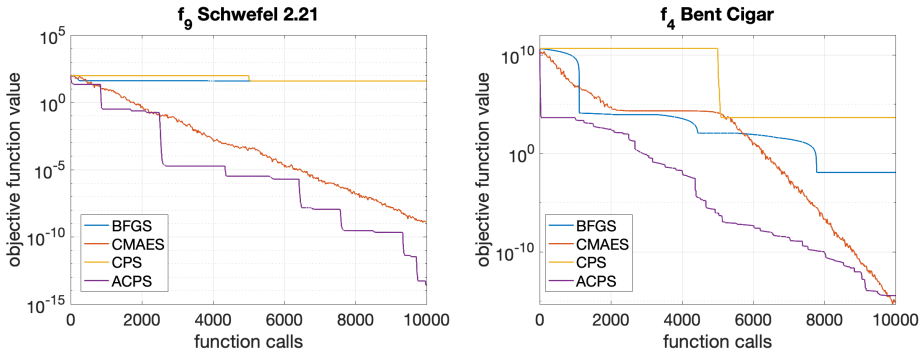
	BFGS			CMAES			CPS			ACPS	
	avg	σ	W	avg	σ	W	avg	σ	W	avg	σ
10 dimensions											
f_1	1.8757e-20	2.4576e-21	+	1.6841e-15	1.2496e-15	+	7.2365e-29	9.6235e-29	+	**0.0000e+00**	0.0000e+00
f_2	1.8956e-13	4.1016e-14	+	1.5739e-15	1.1095e-15	+	1.5810e-01	5.5370e-01	+	**4.4034e-23**	6.9361e-23
f_3	6.6365e-11	2.6183e-12	+	1.7152e-15	1.0734e-15	+	1.0352e+04	2.4610e+04	+	**7.1972e-16**	1.7443e-15
f_4	4.8479e-01	1.6494e+00	+	**1.4131e-15**	1.1850e-15	-	2.0252e+04	1.3731e+04	+	2.5260e-14	1.1648e-13
f_5	1.2818e-09	2.0603e-09	+	8.8699e-15	1.0347e-14	+	3.4509e-06	5.9736e-06	+	**3.3716e-23**	1.2212e-22
f_6	3.2306e-10	5.0825e-11	+	**1.3737e-15**	1.3659e-15	=	1.9279e-12	2.4744e-12	-	8.8372e-11	4.7050e-10
f_7	3.9439e-12	7.2576e-12	+	6.6387e-15	1.0289e-14	+	5.2635e-19	1.1885e-18	+	**3.0987e-27**	1.1885e-18
f_8	1.1967e-07	4.3589e-07	+	**9.8002e-13**	8.9374e-13	-	1.7431e-05	5.0694e-06	+	4.1666e-08	6.3467e-08
f_9	7.1074e+01	2.9732e+01	+	**7.7677e-10**	2.1798e-10	-	5.7996e+00	8.9728e+00	+	1.4098e-06	7.6446e-06
f_{10}	1.1960e+00	1.8581e+00	+	7.973e-01	1.62193+00	+	1.1752e+02	2.8670e+02	+	**5.3985e-27**	2.5363e-27
f_{11}	8.7917e+02	4.2825e+02	+	**1.6069e+01**	1.6267e+01	-	6.4207e+01	3.0468e+01	=	5.7508e+01	2.5321e+01
30 dimensions											
f_1	4.3857e-20	1.6043e-20	+	1.4429e-15	1.0251e-15	+	5.5283e-28	1.3526e-28	+	**0.0000e+00**	0.0000e+00
f_2	1.3352e-09	3.3772e-10	-	**2.9032e-15**	5.7236e-16	-	7.1932e+05	9.4686e+05	+	5.7209e-05	1.7248e-04
f_3	3.2738e-11	1.5479e-12	-	**2.0379e-15**	2.0379e-15	-	1.0276e+05	7.1458e+04	+	8.8106e-07	2.7855e-06
f_4	4.5544e+00	2.4117e+01	+	**1.3962e-15**	4.6239e-16	-	3.7216e+03	4.2577e+03	+	9.4800e-02	2.8220e-01
f_5	1.8926e-08	2.5487e-08	+	5.9103e+03	1.5913e+04	+	1.8790e-09	5.9420e-09	+	**6.2125e-24**	1.9636e-23
f_6	1.5200e-10	1.9637e-11	-	**4.1546e-15**	2.3165e-15	-	1.2576e+02	1.4733e+02	=	1.6064e+02	3.4634e+02
f_7	5.9737e-13	1.2614e-12	+	1.0770e-14	1.2033e-14	+	9.7797e-27	1.1411e-26	=	**5.3015e-27**	5.8250e-27
f_8	1.2390e-05	2.84323e-06	+	**1.5671e-11**	2.0251e-11	-	9.2850e-05	1.8269e-05	+	9.2440e-07	5.7487e-07
f_9	1.1706e+02	3.2399e+01	+	**1.3042e+00**	2.8263e+00	=	4.2458e+01	1.1286e+01	+	2.2920e+00	2.0570e+00
f_{10}	1.7275e+00	2.0093e+00	+	7.9730e-01	1.6809e+00	+	3.7031e+00	2.0897e+00	+	**3.9870e-01**	1.2607e+00
f_{11}	2.0621e+03	6.1136e+02	+	**5.0942e+01**	1.0997e+01	-	3.9956e+02	1.2392e+02	+	2.7569e+02	8.9984e+01
50 dimensions											
f_1	8.5482e-20	1.9014e-20	+	1.1890e-15	3.3954e-16	+	1.2148e-27	6.1144e-28	+	**0.0000e+00**	0.0000e+00
f_2	1.5972e-07	3.7724e-08	-	**2.1614e-14**	4.5515e-15	-	3.5727e+07	1.9921e+07	+	3.9667e+05	3.6613e+05
f_3	9.2288e-011	4.6933e-12	-	**1.3092e-15**	5.9487e-16	-	1.9244e+05	6.5138e+04	+	1.3150e+02	3.5391e+02
f_4	9.8654e-01	3.4845e+e00	-	**1.2358e-15**	5.8376e-16	-	4.6930e+03	5.8099e+03	+	1.6111e+01	3.273e+01
f_5	1.9741e-08	2.8660e-08	+	5.2536e+04	1.5800e+05	+	4.700e-03	1.4700e-02	+	**1.6298e-31**	4.3564e-31
f_6	**1.1937e-10**	7.3146e-12	-	3.0884e+04	9.7663e+04	+	1.2573e+02	1.2764e+02	=	1.0434e+02	1.3584e+02
f_7	1.5341e-13	3.2062e-13	+	1.8735e+00	4.4116e+00	+	1.3449e-26	2.7367e-26	=	**1.3501e-27**	1.8298e-27
f_8	1.1253e-05	1.8968e-06	+	**6.5703e-11**	3.2925e-11	-	1.3238e-04	1.4419e-05	+	2.3891e-06	1.2608e-06
f_9	1.2855e+02	2.7862e+01	+	**9.6761e-06**	2.8014e-05	-	6.3083e+01	1.2169e+01	+	1.8681e+01	9.8691e+00
f_{10}	1.9933e+00	2.0274e+00	+	1.1960e+00	1.9257e+00	+	1.0012e+02	2.3364e+02	+	**7.9730e-01**	1.6809e+00
f_{11}	3.3820e+03	1.0348e+03	+	1.1004e+02	1.3796e+01	+	8.2749e+02	2.0231e+02	+	**6.2442e+01**	1.9581e+01

sharp improvements coincide with a restart and thus the identification of a new pattern.

6 Conclusion

The present paper proposes an adaptive pattern search that uses the search directions given by the eigenvectors of the covariance matrix built by means of successful visited points. The proposed ACPS is a restarting local search algorithm that at each restart adapts the search directions on the basis of the information gathered in the previous local run. Hence, ACPS progressively adapts the search directions to the problem under study.

This adaptive logic is here opposed to the fitness landscape analysis of CPS that occurs separately and prior to the optimisation. Two important advantages of ACPS with respect to CPS are: 1) ACPS does not require the setting of a threshold; 2) ACPS uses its entire computational budget to perform the optimisation, that is a budget dedicated to the fitness landscape analysis does not need to be allocated.



(a) f_9 in ten dimensions (b) f_4 in ten dimensions

Fig. 3: Two examples of convergence trend (logarithmic scale), i.e. objective function values vs function calls.

Numerical results show that, thanks to its adaptive logic, ACPS systematically outperforms CPS for the problems considered in this study. Thanks to its restarting logic, ACPS, albeit a local search, can successfully tackle simple multimodal optimisation problems. In ten dimensions ACPS displays a performance comparable to that of CMAES for all the problems under study. In higher dimensions (thirty and fifty), the performance of ACPS and CMAES diverge on the single problems: for some problems ACPS appears to be better suited than CMAES and for some other problems the opposite happens. Over the entire set of test problems, ACPS and CMAES display a comparable performance.

References

1. Brest, J., Greiner, S., Bošković, B., Mernik, M., Žumer, V.: Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems. IEEE Trans. Evolutionary Computation **10**(6), 646–657 (2006)
2. Burke, E.K., Hyde, M., Kendall, G., Ochoa, G., Ozcan, E., Woodward, J.: Classification of hyper-heuristic approaches. In: Handbook of Meta-Heuristics, pp. 449–468. Springer (2010)
3. Caraffini, F., Neri, F., Epitropakis, M.G.: Hyperspam: A study on hyper-heuristic coordination strategies in the continuous domain. Inf. Sci. **477**, 186–202 (2019)
4. Fletcher, R.: Practical Methods of Optimization. John Wiley & Sons, New York, NY, USA, second edn. (1987)
5. 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 Computing **13**(10), 959–977 (2008)
6. Hansen, N., Ostermeier, A.: Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In: Proceedings of the IEEE International Conference on Evolutionary Computation. pp. 312–317 (1996)

7. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation **9**(2), 159–195 (2001)
8. Jana, N.D., Sil, J., Das, S.: Continuous fitness landscape analysis using a chaos-based random walk algorithm. Soft Computing **22**, 921–948 (2018)
9. Jolliffe, I.T.: Principal Component Analysis. Springer Series in Statistics, Springer, 2nd edn. (2002)
10. Liang, J., Qu, B., Suganthan, P., Hernández-Díaz, A.: Problem definitions and evaluation criteria for the cec 2013 special session on real-parameter optimization (01 2013)
11. Malan, K.M., Engelbrecht, A.P.: Quantifying ruggedness of continuous landscapes using entropy. In: 2009 IEEE Congress on Evolutionary Computation. pp. 1440–1447 (2009)
12. Malan, K.M., Engelbrecht, A.P.: A survey of techniques for characterising fitness landscapes and some possible ways forward. Information Sciences **241**, 148 – 163 (2013)
13. Neri, F.: Linear Algebra for Computational Sciences and Engineering. Springer, second edn. (2019)
14. Neri, F., Rostami, S.: A local search for numerical optimisation based on covariance matrix diagonalisation. In: Castillo, P.A., Laredo, J.L.J., de Vega, F.F. (eds.) Applications of Evolutionary Computation - 23rd European Conference, EvoApplications 2020, EvoStar 2020, Seville, Spain, April 15-17, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12104, pp. 3–19. Springer (2020)
15. Neri, F., Zhou, Y.: 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)
16. Ochoa, G., Malan, K.: Recent advances in fitness landscape analysis. In: López-Ibáñez, M., Auger, A., Stützle, T. (eds.) Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO 2019, Prague, Czech Republic, July 13-17, 2019. pp. 1077–1094. ACM (2019)
17. Ochoa, G., Malan, K.M., Blum, C.: Search trajectory networks of population-based algorithms in continuous spaces. In: Applications of Evolutionary Computation - 23rd European Conference, EvoApplications 2020, Held as Part of EvoStar 2020, Seville, Spain, April 15-17, 2020, Proceedings. pp. 70–85 (2020)
18. Ong, Y.S., Lim, M.H., Zhu, N., Wong, K.W.: Classification of Adaptive Memetic Algorithms: A Comparative Study. IEEE Transactions On Systems, Man and Cybernetics - Part B **36**(1), 141–152 (2006)
19. Press, W.H., Teukolsky, S.A., Vetterling, W.T., Flannery, B.P.: Numerical Recipes in C (2Nd Ed.): The Art of Scientific Computing. Cambridge University Press, New York, NY, USA (1992)
20. Rosenbrock, H.H.: An automatic Method for finding the greatest or least Value of a Function. The Computer Journal **3**(3), 175–184 (1960)
21. Tirronen, V., Neri, F., Kärkkäinen, T., Majava, K., Rossi, T.: An enhanced memetic differential evolution in filter design for defect detection in paper production. Evolutionary Computation **16**, 529–555 (2008)
22. Torczon, V.: On the convergence of pattern search algorithms. SIAM Journal on Optimization **7**(1), 1–25 (1997)
23. Tseng, L.Y., Chen, C.: Multiple trajectory search for Large Scale Global Optimization. In: Proceedings of the IEEE Congress on Evolutionary Computation. pp. 3052–3059 (2008)
24. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation **1**(1), 67–82 (1997)