Stochastic Environmental Research and Risk Assessment (2021) 35:1969–1984
https://doi.org/10.1007/s00477-020-01961-3

ORIGINAL PAPER

# A new principal component analysis by particle swarm optimization with an environmental application for data science

John A. Ramirez-Figueroa[1,2] · Carlos Martin-Barreiro[1,2] · Ana B. Nieto-Librero[1,3] · Victor Leiva[4] · M. Purificación Galindo-Villardón[1,3]

Accepted: 11 December 2020 / Published online: 2 January 2021


**Abstract**
In this paper, we propose a new method for disjoint principal component analysis based on an intelligent search. The method consists of a principal component analysis with constraints, allowing us to determine components that are linear combinations of disjoint subsets of the original variables. The effectiveness of the proposed method contributes to solve one of the crucial problems of multivariate analysis, that is, the interpretation of the vectorial subspaces in the reduction of the dimensionality. The method selects the variables that contribute the most to each of the principal components in a clear and direct way. Numerical results are provided to confirm the quality of the solutions attained by the proposed method. This method avoids a local optimum and obtains a high success rate when reaching the best solution, which occurs in all the cases of our simulation study. An illustration with environmental real data shows the good performance of the method and its potential applications.

**Keywords** Constrained binary particle swarm optimization · Data mining · Disjoint principal components · Evolutionary computation · R software · Singular value decomposition

## 1 Introduction

Most of multivariate techniques are based on the singular value decomposition (SVD) of the data matrix. This decomposition allows us to find a system of orthogonal principal axes (or components) that correspond to the directions of maximum variance (Beaton et al. 2014). The vectors corresponding to these new principal components are linear combinations of the original variables so that such axes are latent variables. Then, the principal component analysis (PCA) is performed by SVD, with PCA being the simplest of the eigenvectors-based multivariate techniques.

Each principal axis is related to a principal vector and the values of its coordinates give practical sense to the components in the context of the research (Hastie et al. 2009). This task can become a problem if a large number of coordinates have equal absolute values, with no loads close to zero, making it difficult to characterize the axis corresponding to the latent variable to be considered. If each of the principal axes is a linear combination of a few original variables, its interpretation is simpler within the context. Several multivariate techniques have been developed to achieve this goal. For example, the technique proposed by Vines (2000) consists of obtaining directions that are represented by vectors of integers with several of their coordinates equal to zero. The sparse PCA proposed by Zou et al. (2006) seeks factorial axes with few non-zero loads. The decomposition proposed by Mahoney and Drineas (2009) represents the data matrix as a product of three low-rank matrices. These techniques above mentioned permit us to express the principal axes based on a reduced number of columns (variables) and/or rows (individuals or objects).

✉ Victor Leiva
victorleivasanchez@gmail.com
http://www.victorleiva.cl

[1] Department of Statistics, Universidad de Salamanca, Salamanca, Spain

[2] FCNM, Universidad Politécnica ESPOL, Guayaquil, Ecuador

[3] Institute of Biomedical Research of Salamanca, Salamanca, Spain

[4] School of Industrial Engineering, Pontificia Universidad Católica de Valparaíso, Valparaíso, Chile

Another proposal is to use disjoint components that, in addition to characterizing a component as a linear combination of a few original variables, they do not appear in the other components, justifying the name disjoint. Vigneau and Qannari (2003) performed hierarchical clustering to relate a latent variable to each previously obtained cluster, proposing a disjoint PCA. A different approach to that used by Vigneau and Qannari (2003) was presented by Vichi and Saporta (2009), determining the latent variables by means of a PCA, named clustering and disjoint PCA (CDPCA). The CDPCA consists of sequentially applying cluster analysis and PCA by means of an alternating least squares algorithm. Macedo and Freitas (2015) proposed an algorithm for the CDPCA.

Another approach was proposed by Nieto-Librero et al. (2017) corresponding to the clustering and disjoint HJ biplot technique (Frutos et al. 2014). Note that a dispoint PCA facilitates the interpretation of the results, without the need of clusters to characterize them. Nieto-Librero (2015) developed a disjoint HJ biplot technique and its calculation algorithm is based on the approach proposed by Vichi and Saporta (2009) and the algorithm presented in Macedo and Freitas (2015). Ferrara et al. (2016) introduced the three following techniques to obtain disjoint components: (i) stepwise PCA; (ii) constrained PCA; and (iii) disjoint PCA, where (iii) was derived from the CDPCA. Freitas et al. (2020) compared two optimization methods for the CDPCA algorithm, one based on alternating least squares and the other one based on two-step semidefinite programming. Lou et al. (2020) proposed a method to perform PCA selecting the number of positions different from zero using particle swarm optimization (PSO).

PSO has been applied to control systems, data mining, graphical computing, neural networks, scheduling problems and robotics (Alatas and Akin 2008; Vasile and Buiu 2011). In multivariate analysis, Voss (2005) utilized PCA with PSO based on the system of principal axes being moved to the swarm, with PCA being used to determine the directions of these axes in each iteration. Chu et al. (2011) introduced a method that employs PCA to handle the bounds of the feasible region. Ma and Ji (2012) utilized PCA to reduce dimension, selecting the most important principal components and to then apply PSO. Similarly, Zhao et al. (2014) used PCA to determine efficient directions to be employed in PSO. Song et al. (2017) utilized PSO for global optimization in the presence of discontinuity. Other applications of PSO are in the cluster analysis, particularly in the detection of centroids and in the selection of the number of clusters (Van der Merwe and Engelbrecht 2003; Esmin and Matwin 2012; Gajawada and Toshniwal 2012; Wang et al. 2018). In the articles cited above, with the exception of the work on cluster analysis, the multivariate techniques utilize PSO to improve its performance in prior or posterior stages.

The present work is based on the disjoint HJ biplot technique, introducing a different approach in how the optimization is performed. Here, a new paradigm in the optimization of data science problems is exhibited to minimize the objective function using an algorithm of PSO of binary type with constraints. This technique is named here as constrained binary particle swarm optimization (CBPSO), where the term binary is due to the use of binary matrices in the PSO. Optimization by PSO is an emerging technique of evolutionary computation based on stochastic optimization and inspired by the behavior of biological species, such as starlings (Sangwook et al. 2008).

The objective of the present article is to propose a new methodology that combines PSO and PCA. Specifically, we use PSO in the solution of the optimization problem inherent to the calculation of the disjoint principal components. In addition, a new proposal is also presented here to compute the eigenvalues associated with the variability of each of the new principal axes, enabling us to construct indicators of the explicability percentages of each disjoint axis obtained.

The rest of his paper is organized as follows. Section 2 describes a dispoint PCA algorithm developed from HJ biplot, which we name DPCA in short, using an alternating least squares method. In Sect. 3, we detail the implementation of an algorithm that uses binary PSO optimization with binary constraints named here CBPSO-PCA. Section 4 summarizes the new proposed method and exposes the advantages of using PSO through simulations. In this section, we also illustrate our method with real environmental data to show its potential applications. Finally, Sect. 5 discusses the conclusions of this study and future research.

## 2 The DPCA method

In this section, we provide background about the DPCA. In addition, an algorithm that implements the calculation of disjoint principal components is provided.

### 2.1 Background

The $I \times J$ matrix $\boldsymbol{X}$ in the DPCA is expressed as in the standard PCA, that is, as the product of the transpose of the loading matrix $\boldsymbol{B}$ by the score matrix $\boldsymbol{A}$ (Jolliffe 2002, p. 31) given by

$$\boldsymbol{X} = \boldsymbol{A}\boldsymbol{B}^{\top}, \tag{1}$$

where $I$ is the number of individuals and $J$ is the number of variables; $\boldsymbol{A}$ is the $I \times Q$ score matrix, containing the coordinates of the individuals in the reduced $Q$-

dimensional space of the disjoint components; $\boldsymbol{B} = (b_{ij})$ is the $J \times Q$ component loading matrix, containing the coordinates of the variable $j$ in the reduced $Q$-dimensional space and is subject to the following constraints:

(i) $\sum_{j=1}^{J} b_{jq}^2 = 1$, for $q = 1, \ldots, Q$;
(ii) $\sum_{j=1}^{J} (b_{jq} b_{jr})^2 = 0$, for $q = 1, \ldots, Q-1$ and $r = q+1, \ldots, Q$;
(iii) $\sum_{q=1}^{Q} b_{jq}^2 > 0$, for $j = 1, \ldots, J$.

Constraint (i) indicates that each column of $\boldsymbol{B}$ has norm one. Constraint (ii) expresses that two different columns of $\boldsymbol{B}$ are orthogonal, while constraint (iii) establishes that $\boldsymbol{B}$ does not have zero vectors. Since this is a low-range approximation, we have that $\boldsymbol{X}$ defined in (1) is now given by $\boldsymbol{X} = \boldsymbol{A}\boldsymbol{B}^\top + \boldsymbol{E}$, where $\boldsymbol{E}$ is the error matrix. Minimizing the Frobenius norm squared of the error matrix $\boldsymbol{E}$ is equivalent to minimizing the objective function

$$F(\boldsymbol{A}, \boldsymbol{B}) = \|\boldsymbol{X} - \boldsymbol{A}\boldsymbol{B}^\top\|^2 = \|\boldsymbol{E}\|^2, \tag{2}$$

that corresponds to the sum of residual squares. Then, we have the optimization problem stated as

$$\begin{cases} \min F(\boldsymbol{A}, \boldsymbol{B}) = \left\|\boldsymbol{X} - \boldsymbol{A}\boldsymbol{B}^\top\right\|^2 \\ \text{subject to } \boldsymbol{A}, \boldsymbol{B} \text{ as described above.} \end{cases} \tag{3}$$

Since $\boldsymbol{X}$ is the data matrix, it is a given matrix and therefore it does not vary throughout the optimization process, so that it is constant, such as its norm $\|\boldsymbol{X}\|$ is. Note that minimizing

$$\frac{\|\boldsymbol{X} - \boldsymbol{A}\boldsymbol{B}^\top\|^2}{\|\boldsymbol{X}\|^2}$$

is equivalent to minimizing $\|\boldsymbol{X} - \boldsymbol{A}\boldsymbol{B}^\top\|^2$. The function $F$ stated in (2) is now redefined as

$$F(\boldsymbol{A}, \boldsymbol{B}) = \frac{\|\boldsymbol{X} - \boldsymbol{A}\boldsymbol{B}^\top\|^2}{\|\boldsymbol{X}\|^2}. \tag{4}$$

Thus, $F$ redefined in (4) represents the squared relative error and is used as the objective function. Note that $F$ measures the degree of fit of the low-range approximation. Hence, minimizing the objective function is equivalent to minimizing the relative error, and therefore a better fit corresponds to smaller values of the objective function.

In the next subsection, we explain how $F$ is minimized to determine the number $Q$ of principal components. This generates a $Q$-dimensional subspace where the original data are projected so that each original variable contributes to only one component. Note that these disjoint components are orthogonal.

## 2.2 Explanation of the DPCA

As mentioned, the DPCA is an adaptation of the analysis employed to construct the HJ biplot technique and the principal components (Ferrara et al. 2016). The DPCA contains the following steps:

*Step 0*. Let $\boldsymbol{X}$ be an $I \times J$ data matrix, with $Q$ being chosen as the number of disjoint components to be considered. The $J \times Q$ stochastic binary matrix $\boldsymbol{V}_0$ is randomly generated by rows, such that its elements are given by

$$v_{jq} = \begin{cases} 1, \text{if the variable } j \text{ contributes to the component } q; \\ 0, \text{otherwise;} \end{cases}$$

satisfying the constraints

$$\sum_{q=1}^{Q} v_{jq} = 1, \quad j = 1, \ldots, J, \tag{5}$$

$$\sum_{j=1}^{J} v_{jq} > 0, \quad q = 1, \ldots, Q, \tag{6}$$

where the expression defined in (5) ensures that there is a one and nothing more than one in each row, whereas the expression defined in (6) ensures that there are no columns filled with zeros. From (5) and (6), we have that

$$\sum_{j=1}^{J} v_{jq} v_{jr} = 0, \quad q \neq r,$$

which means the columns are orthogonal.

***Example 1*** Let $J = 5$ and $Q = 3$. Then,

$$\boldsymbol{V}_0 = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix}.$$

Note that $\boldsymbol{V}_0$ is a position matrix of the original data, since it indicates how the original variables are distributed in the selected disjoint components. This must ensure that any component has all null loadings. Hence, the component with the largest number of assigned variables is randomly divided into two parts and then one of the halves is passed to the component that has the column of zeros.

Once $\boldsymbol{V}_0$ is calculated, the loading matrix $\boldsymbol{B}_0$ is obtained as follows. For each component $q = 1, \ldots, Q$, a partition matrix $\boldsymbol{W}_{0q}$ is generated from the data matrix $\boldsymbol{X}$, with $\boldsymbol{W}_{0q}$ having $I$ rows, such as $\boldsymbol{X}$, and a number of columns corresponding to the variables indicated by the ones in the $q$-th column of $\boldsymbol{V}_0$, as indicated in Remark 1, where the matrix $\boldsymbol{W}$ is defined.

***Remark 1*** Note that the matrix $\boldsymbol{W}$ is generated from the original data $\boldsymbol{X}$ taking those columns corresponding to variables for which a one is present in the matrix $\boldsymbol{V}$ at the associated component. For instance, if for the component 1, the non-null elements of each column in the matrix $\boldsymbol{V}_0$ take the values 1, $j$ and $J$, then we must take the 1-th, $j$-th and $J$-th columns. Such columns are those forming the matrix $\boldsymbol{W}_{01}$. For this matrix $\boldsymbol{W}_{01}$, we apply the HJ biplot technique calculating the coordinates for the variables. From these coordinates, we must choose the first column and such values are the coordinates of the variables 1, $j$ and $J$ in the first component. The remaining variables have coordinates equal to zero for this component.

***Example 1*** (continued) Recall that $J = 5$ and $Q = 3$. Then,

$$\boldsymbol{W}_{01} = \begin{array}{c} \begin{matrix} x_1 & x_2 \end{matrix} \\ \begin{pmatrix} x_{11} & x_{12} \\ x_{21} & x_{22} \\ \vdots & \vdots \\ x_{I1} & x_{I2} \end{pmatrix} \end{array}, \boldsymbol{W}_{02} = \begin{array}{c} \begin{matrix} x_4 \end{matrix} \\ \begin{pmatrix} x_{14} \\ x_{24} \\ \vdots \\ x_{I4} \end{pmatrix} \end{array}, \boldsymbol{W}_{03} = \begin{array}{c} \begin{matrix} x_3 & x_5 \end{matrix} \\ \begin{pmatrix} x_{13} & x_{15} \\ x_{23} & x_{25} \\ \vdots & \vdots \\ x_{I3} & x_{I5} \end{pmatrix} \end{array}.$$

Note that each $\boldsymbol{W}_{0q}$ is expressed by its SVD in the form

$$\boldsymbol{W}_{0q} = \boldsymbol{R}\boldsymbol{\Lambda}\boldsymbol{T}^{\top}, \tag{7}$$

where $\boldsymbol{R}$ is a unitary matrix, $\boldsymbol{\Lambda}$ is a rectangular diagonal matrix with non-negative real numbers on the diagonal, and $\boldsymbol{T}$ is also a unitary matrix, assuming suitable dimensions for these matrices. The diagonal entries $\lambda_i$ of $\boldsymbol{\Lambda}$ are known as the singular values of $\boldsymbol{W}_{0q}$. The columns of $\boldsymbol{R}$ and $\boldsymbol{T}$ are called the left and right singular vectors of $\boldsymbol{W}$, respectively. Thus, we obtain $Q$ SVDs of the form represented in (7).

Once the matrix $\boldsymbol{W}_{0q}$ is obtained, the matrix $\boldsymbol{B}_0$ is constructed, with the $q$-th column of $\boldsymbol{B}_0$ being the right singular vector corresponding to the largest singular value of the $q$-th SVD. This column contains the loadings corresponding to the variables considered in the $q$-th column of $\boldsymbol{V}_0$, with zeros in the positions that are not taken into account for the $q$-th column. Since $Q$ SVDs are taken into account, the number of columns of $\boldsymbol{B}_0$ is $Q$. Once the matrix $\boldsymbol{B}_0$ is computed, the coordinates for the objects are obtained as $\boldsymbol{A}_0 = \boldsymbol{X}\boldsymbol{B}_0$, and then the objective function is calculated as $F_0(\boldsymbol{A}_0, \boldsymbol{B}_0)$.

*Step k*. It is assumed that $k-1$ steps of the algorithm have already been executed and then arrays $\boldsymbol{V}_{k-1}, \boldsymbol{A}_{k-1}, \boldsymbol{B}_{k-1}$ of $F_{k-1}$ are available. We start by updating the matrix $\boldsymbol{V}_k$, letting the $j$-th row, with $j = 1, \ldots, J$, to locate a one in the $q$-th position of this row, for $q = 1, \ldots, Q$, and zeros in the rest of the row. Maintaining unchanged the rest of the rows of $\boldsymbol{V}_{k-1}$, we obtain a new position matrix denoted by $\widehat{\boldsymbol{V}}_{kq}$. With this position matrix, we get the partition matrix $\widehat{\boldsymbol{W}}_{kq}$, the respective $\widehat{\boldsymbol{A}}_{kq}, \widehat{\boldsymbol{B}}_{kq}$ matrices and then the objective function $\widehat{F}_{kq}(\widehat{\boldsymbol{A}}_{kq}, \widehat{\boldsymbol{B}}_{kq})$ is evaluated. The position where $\widehat{F}_{kq}(\widehat{\boldsymbol{A}}_{kq}, \widehat{\boldsymbol{B}}_{kq})$ is minimum assigns a one in that row. This process of relocating the ones in all the rows of $\boldsymbol{V}_{k-1}$ is continued and the resulting matrix is called $\boldsymbol{V}_k$. Since in each row the position containing the one runs through all the $Q$ entries of the $j$-th row, the number of SVDs to determine $\boldsymbol{V}_k$ is $J \times Q$. If during the process some column of $\boldsymbol{V}_k$ is generated with all its elements equal to zero, it acts in the same way as explained in the construction of $\boldsymbol{V}_0$. Once the position matrix $\boldsymbol{V}_k$ is computed, the corresponding matrices $\boldsymbol{W}_{kq}$ are obtained and from them the loading matrix $\boldsymbol{B}_k$ is also reached in the way already indicated. Hence, the matrix $\boldsymbol{A}_k$ of scores is evaluated at $\boldsymbol{A}_k = \boldsymbol{X}\boldsymbol{B}_k$ and then $F(\boldsymbol{A}_k, \boldsymbol{B}_k) = F_k$ is calculated to check whether the stopping criterion is attained as indicated below.

*Stopping criteria.* Set a tolerance value $\varepsilon > 0$ and if we have $|F_k - F_{k-1}| < \varepsilon$, then the algorithm is stopped. If this is the case, the algorithm is finished and the solution is attained at this step, which is assumed as the final solution. If the stopping criterion is not reached, iterate until reaching it. Another stopping criterion is the total number of iterations to be performed. This is the stopping criterion used in the DPCA.

The matrix $\boldsymbol{V}_k$ is constructed in a non-flexible way, since when descending by rows in its construction, the positions that contain the ones of each row are fixed and do not change once they are determined. Another option is to consider all possible binary matrices for $\boldsymbol{V}_k$, but the number of possible partitions can be excessively high, even for small values of $J$ and $Q$, as shown in Table 1 assuming Example 1.

**Table 1** Number of operations to be performed for Example 1

| $J$ | $Q = 2$ | $Q = 3$ |
|---|---|---|
| 10 | $\lvert S\rvert = 1\,022$ | $\lvert S\rvert = 55\,980$ |
| 15 | $\lvert S\rvert = 32\,766$ | $\lvert S\rvert = 14\,250\,606$ |
| 20 | $\lvert S\rvert = 1\,048\,574$ | $\lvert S\rvert = 3\,483\,638\,676$ |
| 30 | $\lvert S\rvert = 1\,073\,741\,822$ | $\lvert S\rvert = 2.058879109 \times 10^{14}$ |

### 2.3 The algorithmic form of the DPCA

Algorithm 1 summarizes the DPCA.

**Algorithm 1** The DPCA procedure

1: Read $\boldsymbol{X}$, $Q$, `nIter` (number of iterations), and the tolerance $\varepsilon$.
2: Initialize with $\boldsymbol{V}_0$ and $k = 0$.
3: Iterate
  3.1: Setting $k = k + 1$.
  3.2: Updating $\boldsymbol{V}_k$.
  3.3: Obtaining the partition matrices $\boldsymbol{W}_{kq}$, for $q = 1, \ldots, Q$.
  3.4: Computing the SVD of $\boldsymbol{W}_{kq}$, for $q = 1, \ldots, Q$.
  3.5: Constructing $\boldsymbol{B}_k$ and $\boldsymbol{A}_k$.
  3.6: Calculating $F_k = F(\boldsymbol{A}_k, \boldsymbol{B}_k)$.
4: Repeat the procedure until $|F_k - F_{k-1}| < \varepsilon$.
5: Report the results for $\boldsymbol{A}_k$ and $\boldsymbol{B}_k$.

## 3 The CBPSO-PCA

As mentioned in Sect. 1, the PSO is an optimization technique where individuals (particles) move in the space of feasible solutions of the objective function, communicating with each other in search of the best solution (optimal approximation). The generic PSO method is presented next.

### 3.1 The PSO method

The PSO has the following individual and social behaviors:

(i) Particles or individuals are attracted to the optimum.
(ii) At any moment, individuals know their closeness to the optimum. The closeness is estimated through the so-called fit and is the value assigned by the objective function to the position where the particle is located.
(iii) Each particle or individual remembers its closest position to the food (the optimum). This is the individual's historical knowledge.
(iv) Each particle shares its information about its closest position to the optimum with the particles closest to it, which is the historical knowledge of its neighborhood.

Through two rules of interaction, particles adapt their behavior to that of the most successful particles in their environment (Imran et al. 2013). The PSO method allows the particles to explore the set of feasible solutions $S$ in search of the optimum. The initial particle population is kept constant through the search process. At the instant of time $k$, each particle has a $\texttt{position}_k$, a $\texttt{velocity}_k$, and a value of fit $F_k$. The PSO method iterates at each instant of time $k$ for each particle as follows:

(i) Local information ($\texttt{linformation}_k$): The current position where the particle reaches its best fit, that is, the smallest value of $F_k$ for the particle.
(ii) Global information ($\texttt{ginformation}_k$): The position where the neighborhood reaches the best fit, that is, the smallest value of $F_k$ for the swarm.

Each particle updates its position according to

$$\texttt{position}_{k+1} = \texttt{position}_k + \texttt{velocity}_{k+1}, \tag{8}$$

where

$$\texttt{velocity}_{k+1} = \texttt{inertia}_k + \texttt{linformation}_k + \texttt{ginformation}_k, \tag{9}$$

with $\texttt{inertia}_k$ being the component responsible for keeping the particle moving at the same direction that it is moving until $k$.

***Remark 2*** Note that, in the PSO context, inertia allows a better exploration of feasible solutions to be obtained. This inertia must not be confused with the inertia in the context of multivariate analysis, in which it is a measure of the statistical variability of the data set. It must be emphasized that PSO has absolutely no guarantee to find a global optimum.

### 3.2 Explanation of the CBPSO-PCA

Recall that particles are represented by binary position matrices of the form $\boldsymbol{V}$ as defined in Sect. 2. In each iteration, when updating the position, the particle leaves the set of feasible solutions $S$, since although the position matrix is binary, the velocity matrix is not. To solve the optimization problem given in (3), the particles are represented by binary matrices $\boldsymbol{V}$ satisfying the constraints defined in (5) and (6). We must find the binary matrix $\boldsymbol{V}^{\star}$ that minimizes the objective function $F$ and therefore this is a binary optimization problem with constraints. The search for feasible solutions leads to a problem of high computational complexity (NP-hard) due to combinatorial explosion, as mentioned at the end of Sect. 2 and exemplified in Table 1. In the present article, we use constrained binary PSO to solve the combinatorial optimization problem associated.

The first step of the PSO method is to initialize the particles, which means that they are placed at a random initial position. The number of particles $P$ is an important input parameter at this stage. The initial position of any particle is a feasible solution of the problem, that is, an element of a set of the feasible solutions $S$. In each iteration, the position and velocity of the particles must be updated according to the expressions given in (8) and (9). The velocity matrices always have their components in the interval $[-1,\ 1]$, as shown in the continuation of Example 1 provided below. By updating the position at the stage

$k+1$, we have a new position that is not a binary matrix nor satisfies the conditions of the feasible space.

***Example 1*** (continued) Recall that $J = 5$ and $Q = 3$. Suppose that any particle, in the stage $k$, has position and velocity matrices stated by

$$\texttt{position}_k = \begin{pmatrix} 1 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix},$$

$$\texttt{velocity}_{k+1} = \begin{pmatrix} -0.51 & 0.75 & 0.12 \\ -0.33 & 0.24 & 0.68 \\ -0.83 & -0.45 & 0.53 \\ 0.71 & 0.58 & 0.91 \\ 0.46 & -0.79 & 0.64 \end{pmatrix}.$$

Then, when updating the position at the stage $k+1$, according to (9), we have a new position given by

$$\begin{aligned} \texttt{position}_{k+1} &= \texttt{position}_k + \texttt{velocity}_{k+1} \\ &= \begin{pmatrix} 0.49 & 0.75 & 0.12 \\ -0.33 & 0.24 & 0.68 \\ -0.83 & 0.55 & 0.53 \\ 0.71 & 0.58 & 0.09 \\ 0.46 & -0.79 & 0.64 \end{pmatrix}. \end{aligned}$$

where $\texttt{position}_k$ is not a binary matrix nor satisfies the conditions of the feasible space. Note that the $\texttt{position}_k$ and $\texttt{velocity}_k$ matrices are chosen for illustrative purposes only, showing how the particle (in the $\texttt{position}_k$) leaves the space of feasible solutions. The calculation of both matrices is explained below.

In order to avoid the problem indicated above, we propose the following procedure:

(i) In each row, the entry with the largest absolute value is selected. The positions of these rows are occupied by ones and the rest by zeros. This procedure, as opposed to choosing random ones, allows the memory of the particle to be preserved.

(ii) In the case where a column results only with zeros, choose the column with the largest number of ones, locate the one that come from the smallest absolute value, and change it to a column full of zeros. Thus, the matrix obtained satisfies the conditions of the set of feasible solutions $S$.

(iii) If there are two columns full of zeros, apply the same procedure, that is, choose the two ones that come from the two smallest absolute values and then pass them to the columns of zeros.

(iv) If there is a tie in the largest absolute values of a row, choose the first of them from left to right.

The points indicated in (i)–(iv) above as solution proposed convert a matrix with real components into a binary matrix in the set of feasible solutions $S$, and they are executed by a matrix operator denoted by $O$. The fit function is given by the squared relative error, which represents, as in the DPCA, the change for one unit, such as defined in Sect. 2.1. In the PSO method, there are two types of variables: individual and collective (or global). For individual variables, each particle stores its own values, but for the global variables the values are shared by all the particles. As PSO is an evolutionary algorithm, it may happen that, in two or more successive iterations, there is no change of the objective function. For this reason, the stopping criterion is the number of iterations defined by the user (Martí et al. 2009).

In the sequel, $T$ is a matrix operator that transforms the binary matrix $\boldsymbol{V}$ into the loading matrix $\boldsymbol{B}$ as described in the step $k$ of Sect. 2.2. It is important to emphasize that, for calculating $\boldsymbol{B} = T(\boldsymbol{V})$, the number of SVDs performed by the operator $T$ is the same as the number $Q$ of desired disjoint components. Next, we describe how local and global informations work to reach the best fit.

#### 3.2.1 Local information

The local information of a particle quantifies its attraction to the position where it reaches its best fit. For a specific particle $p$, the following information is stored in memory:

- $\boldsymbol{V}_p$ corresponds to the binary matrix $\boldsymbol{V}$ for the particle $p$ and represents the current position of that particle (current binary position).
- $\boldsymbol{B}_p = T(\boldsymbol{V}_p)$ is the loading matrix $\boldsymbol{B}$ for the particle $p$ in the current position $\boldsymbol{V}_p$.
- $F(\boldsymbol{V}_p)$ corresponds to the value of the objective function evaluated at the current position $\boldsymbol{V}_p$ of the particle $p$, that is, $F(\boldsymbol{V}_p) = F(\boldsymbol{A}_p, \boldsymbol{B}_p)$, where $\boldsymbol{A}_p = \boldsymbol{X}\boldsymbol{B}_p$.
- $\boldsymbol{V}_p^{\star}$ is the best binary matrix $\boldsymbol{V}$ found by the particle $p$ locally.
- $\boldsymbol{B}_p^{\star} = T(\boldsymbol{V}_p^{\star})$ represents the best loading matrix found by the particle $p$ locally.
- $F(\boldsymbol{V}_p^{\star})$ is the value of the objective function evaluated at the best solution found by the particle $p$ locally.
- $\texttt{velocity}_p$ is the current velocity of the particle $p$ used to generate a disturbance of the current position of this particle to place it at a new position. Thus, $\texttt{velocity}_p$ is a $J \times Q$ matrix, and as part of the algorithm, each entry in this matrix is in the interval $[-1, 1]$.

#### 3.2.2 Global information

The global information quantifies the attraction of the particle towards its position that obtains the best global fit. In order to make it, the following information is collected at the level of the entire swarm:

- $\boldsymbol{V}^{\star}$ is the best binary matrix $\boldsymbol{V}$ found by the swarm of particles globally.
- $\boldsymbol{B}^{\star} = T(\boldsymbol{V}^{\star})$ is the best loading matrix found by the swarm of particles globally.
- $F(\boldsymbol{V}^{\star})$ represents the value of the objective function evaluated at the best solution found by the swarm of particles globally.

As mentioned, the PSO method consists of three main steps: initialization of the particles, iteration and update. The second step of the algorithm is the most important because it corresponds to the iterations. In each iteration, all particles must move to a new position. Each position is a feasible solution to the problem, that is, it is not allowed a particle to place itself in the position of an unfeasible solution. In this stage, we have the following parameters:

- `nIter`: number of iterations.
- `minIner`: minimum inertia value, which corresponds to the value of the inertia in the last iteration.
- `maxIner`: maximum inertia value, which is the value of the inertia in the first iteration.
- `wCognition`: cognitive weight that serves to control the cognitive component of the new velocity.
- `wSocial`: social weight, which is used to control the social component of the new velocity.

To determine the new position of a particle $p$, it is necessary to calculate a new velocity that depends on three components: (i) the first of them is the velocity in the previous iteration that we control with inertia; (ii) the second one is the cognitive component that we handle with the cognitive weight; and (iii) the third one is the social component that we control with the social weight. The inertia decreases linearly from iteration to iteration. The iterations are initiated with `maxIner`, whereas the iterations are finished with `minIner`. Thus, for an iteration $0 \leq k \leq \texttt{nIter}$, the inertia factor value is given by

$$\texttt{finertia} = \texttt{maxIner} - \left(\frac{\texttt{maxIner} - \texttt{minIner}}{\texttt{nIter}}\right)k. \tag{10}$$

The expression defined in (10) implies that, as it is iterated, the new velocity depends less on the previous velocity, and the cognitive and social components affect it more. The value of the new velocity for a particle $p$ in the $k$-th iteration is obtained by

$$\begin{aligned}\texttt{newVel}_p =& \texttt{finertia} \times \texttt{velocity}_p \\ &+ \rho_1 \times \texttt{wCognition} \times (\boldsymbol{B}_p^{\star} - \boldsymbol{B}_p) \\ &+ \rho_2 \times \texttt{wSocial}(\boldsymbol{B}^{\star} - \boldsymbol{B}_p)\end{aligned}$$

where $\rho_1, \rho_2 \in [0, 1]$ are pseudo-random numbers so that $\texttt{newVel}_p$ has a stochastic component, $\boldsymbol{B}_p^{\star}$ is the best local loading matrix, and $\boldsymbol{B}^{\star}$ is the best global loading matrix. Note that the $J \times Q$ matrix $\texttt{newVel}_p$ generally does not comply with the constraint that the entries are in the interval $[-1, 1]$. Therefore, a transformation must be applied to satisfy this constraint. For this purpose, we define the operator $L$ as follows.

Let $z \in \mathbb{R}$ be any entry in the matrix $\texttt{newVel}_p$. The operator $L$ is defined by a convex linear combination of the sigmoid function, a particular case of the logistic function (Nezamabadi-Pour et al. 2008; Sangwook et al. 2008) defined as

$$L(z) = 2\text{sigmoid}(z) - 1,$$

where $\text{sigmoid}(z) = (1 + \mathrm{e}^{-z})^{-1} \in [0, 1]$ so that $L(z) \in [-1, 1]$. The operator $L$ is applied to all entries of $\texttt{newVel}_p$ as explained above, obtaining $\texttt{velocity}_p$. Then, a new position (in floating point format) is calculated using

$$\boldsymbol{V}_p^{(\text{temp})} = \boldsymbol{B}_p + \texttt{velocity}_p. \tag{11}$$

Again, the operator $L$ is applied to $\boldsymbol{V}_p^{(\text{temp})}$ in (11), so that its inputs are in the interval $[-1, 1]$. Now, by applying the matrix operator $O$ to $\boldsymbol{V}_p^{(\text{temp})}$, the new binary position of the particle $p$ is obtained in the set of feasible solutions as $\boldsymbol{V}_p = O(\boldsymbol{V}_p^{(\text{temp})})$ and $\boldsymbol{B}_p = T(\boldsymbol{V}_p)$.

Next, we summarize the steps of initialization, iteration and update of the PSO method in an algorithmic structure:

**Step 1 (initialization)**

1.1 For each particle $p = 1, \ldots, P$:

- 1.1.1 Generate randomly the matrix $\boldsymbol{V}_p$.
- 1.1.2 Do $\boldsymbol{B}_p = T(\boldsymbol{V}_p)$.
- 1.1.3 Compute $F(\boldsymbol{V}_p)$.
- 1.1.4 Make $\boldsymbol{V}_p^{\star} = \boldsymbol{V}_p$.
- 1.1.5 Establish $\boldsymbol{B}_p^{\star} = \boldsymbol{B}_p$.
- 1.1.6 State $F(\boldsymbol{V}_p^{\star}) = F(\boldsymbol{V}_p)$.
- 1.1.7 Express $\texttt{velocity}_p$ randomly (initial velocity).

1.2 Among all the $P$ particles, search for the particle with the best binary initial position, that corresponds to the particle with the best fit, denoted by $p^{\star}$. Then:

- 1.2.1 Do $\boldsymbol{V}^{\star} = \boldsymbol{V}_{p^{\star}}$.
- 1.2.2 Make $\boldsymbol{B}^{\star} = \boldsymbol{B}_{p^{\star}}$
- 1.2.3 State $F(\boldsymbol{V}^{\star}) = F(\boldsymbol{V}_{p^{\star}})$.

**Step 2 (iteration)**

2.1 Determine the new position of a particle $p$.
2.2 Establish the value of the new velocity for a particle $p$ in the $k$-th iteration.
2.3 Transform the entries to the interval $[-1, 1]$ using the operator $L$.
2.4 Calculate a new position using the expression given in (11) for $\boldsymbol{V}_p^{(\text{temp})}$.
2.5 Apply again the operator $L$ to $\boldsymbol{V}_p^{(\text{temp})}$ so that its inputs are in the interval $[-1, 1]$.
2.5 Obtain the new binary position of the particle $p$ in the set of feasible solutions by using the operators $O$ and $T$ in $\boldsymbol{V}_p^{(\text{temp})}$ to reach $\boldsymbol{V}_p = O(\boldsymbol{V}_p^{(\text{temp})})$ and $\boldsymbol{B}_p = T(\boldsymbol{V}_p)$.

**Step 3 (updating)**

3.1 For each iteration $k = 1, \ldots, \texttt{nIter}$ :

3.1.1 Compute `finertia`.
3.1.2 For each particle $p = 1, \ldots, P$ :

3.1.2.1 Calculate $\texttt{newVel}_p$.
3.1.2.2 Do $\texttt{velocity}_p = L(\texttt{newVel}_p)$.
3.1.2.3 Make $\boldsymbol{V}_p^{(\text{temp})} = \boldsymbol{B}_p + \texttt{velocity}_p$.
3.1.2.4 State $\boldsymbol{V}_p = O(\boldsymbol{V}_p^{(\text{temp})})$.
3.1.2.5 Establish $\boldsymbol{B}_p = T(\boldsymbol{V}_p)$.
3.1.2.6 Determine $F(\boldsymbol{V}_p)$.
3.1.2.7 If $F(\boldsymbol{V}_p) < F(\boldsymbol{V}_p^{\star})$, then

A. $\boldsymbol{V}_p^{\star} = \boldsymbol{V}_p$.
B. $\boldsymbol{B}_p^{\star} = \boldsymbol{B}_p$.
C. $F(\boldsymbol{V}_p^{\star}) = F(\boldsymbol{V}_p)$.

If $F(\boldsymbol{V}_p) < F(\boldsymbol{V}^{\star})$, then

A. $\boldsymbol{V}^{\star} = \boldsymbol{V}_p$.
B. $\boldsymbol{B}^{\star} = \boldsymbol{B}_p$.
C. $F(\boldsymbol{V}^{\star}) = F(\boldsymbol{V}_p)$.

3.2 Report the best solution found.

### 3.3 The CBPSO-PCA algorithm

The way how the CBPSO-PCA works is summarized in Algorithm 2. The function `Step` that we see in Algorithm 2 is used to represent, in a single instruction, the movement of a particle to a new position. Every time a particle moves to the next position, the best position found by that particle and the best position found by all particles are updated, if necessary. Note that the `finertia` is updated when going from an iteration to the next. All the particles of the same iteration share the same `finertia` value. In Algorithm 2, the presence of the matrices $\boldsymbol{V}_p$, $\boldsymbol{V}_p^{\star}$, and $\boldsymbol{V}^{\star}$ is omitted to simplify the operation of the CBPSO-PCA algorithm. Here, $F(\boldsymbol{B}_p) = F(\boldsymbol{A}_p, \boldsymbol{B}_p)$.

**Algorithm 2** CBPSO-PCA

1: Initiate $P$ particles at random.
2: Do $\texttt{finertia} = \texttt{maxIner}$.
3: For each iteration $k = 1, \ldots, \texttt{nIter}$:
  For each particle $p = 1, \ldots, P$:
  Do $\boldsymbol{B}_p = \texttt{Step}(\boldsymbol{B}_p, \boldsymbol{B}_p^{\star}, \boldsymbol{B}^{\star}, \texttt{wCognition}, \texttt{wSocial}, \texttt{finertia})$.
  If $F(\boldsymbol{B}_p) < F(\boldsymbol{B}_p^{\star})$, then do $\boldsymbol{B}_p^{\star} = \boldsymbol{B}_p$.
  If $F(\boldsymbol{B}_p) < F(\boldsymbol{B}^{\star})$, then do $\boldsymbol{B}^{\star} = \boldsymbol{B}_p$.
  End for each $p$.
  Update `finertia`.
  End for each $k$.
4: Report the obtained results.

### 3.4 Explained variance

Note that the CBPSO-PCA does not calculate the matrix $\boldsymbol{\Lambda}$ defined in (7). Then, the singular values of $\boldsymbol{X}$ are unknown and therefore the eigenvalues of $\boldsymbol{X}^{\top}\boldsymbol{X}$ as well. Here, we estimate the eigenvalues corresponding to each of the disjoint components by means of the calculation of the variance that the individuals exhibit in the new system of principal axes. This enables us to find the percentage of explained variance for each of the disjoint axes and the respective principal planes. Specifically, let $\boldsymbol{X}$ be a centered $I \times J$ data matrix, and let $\text{Var}(\boldsymbol{X}) = \boldsymbol{S}$ be its variance-covariance matrix. Then, the total variation of $\boldsymbol{X}$ is defined by

$$\text{trace}(\boldsymbol{S}) = \sigma_1^2 + \cdots + \sigma_J^2 = \sum_{j=1}^{J} \sigma_j^2, \tag{12}$$

where $\sigma_1^2, \ldots, \sigma_J^2$ are the variances of the $J$ variables contained in the columns of $\boldsymbol{X}$, that is, $\text{Var}(X_j) = \sigma_j^2$. Alternatively, we have that

$$\text{trace}(\boldsymbol{S}) = \alpha_1 + \cdots + \alpha_J, \tag{13}$$

where $\alpha_1, \ldots, \alpha_J$ are the eigenvalues of $\boldsymbol{S}$.

Let $\boldsymbol{B}$ be a $J \times J$ rotation matrix, that is, $\boldsymbol{B}$ is orthogonal in the sense that $\boldsymbol{B}^{\top}\boldsymbol{B} = \boldsymbol{I}_J$, where $\boldsymbol{I}_J$ is $J \times J$ identity matrix. The column vectors $\boldsymbol{B}$ define a new rotated system of axes. Each new axis is a linear combination of the original variables representing a new variable. Let $\boldsymbol{A}$ be a rotation transformation of $\boldsymbol{X}$ given by $\boldsymbol{A} = \boldsymbol{X}\boldsymbol{B}$. Its variance-covariance matrix is stated as

$$\boldsymbol{\Sigma} = \text{Var}(\boldsymbol{A}) = \boldsymbol{B}^{\top}\boldsymbol{S}\boldsymbol{B}.$$

Since the columns of $\boldsymbol{A}$ contain the coordinates of the individuals with respect to the new system of axes, the variance of each column represents the variance of the new variables obtained by rotation $\boldsymbol{B}$. The total variation of $\boldsymbol{A}$ is defined as

$$\begin{aligned}\text{trace}(\boldsymbol{\Sigma}) &= \text{trace}\left(\boldsymbol{B}^\top \boldsymbol{S}\boldsymbol{B}\right)\\ &= \text{trace}\left(\boldsymbol{S}\boldsymbol{B}\boldsymbol{B}^\top\right)\\ &= \text{trace}(\boldsymbol{S}\boldsymbol{I}_J)\\ &= \text{trace}(\boldsymbol{S}),\end{aligned} \tag{14}$$

that is, the total variation of $\boldsymbol{X}$ and $\boldsymbol{A}$ is the same. Note that the rotation $\boldsymbol{B}$ does not affect the total variation. If $\beta_1, \ldots, \beta_J$ are the eigenvalues of the matrix $\boldsymbol{\Sigma}$ in descending order, then the variance of the $j$-th new variable, denoted by $A_j$, is $\text{Var}(A_j) = \beta_j$, for $j = 1, \ldots, J$, and

$$\text{trace}(\boldsymbol{\Sigma}) = \beta_1 + \cdots + \beta_J. \tag{15}$$

From (12), (13), (14) and (15), we have

$$\sigma_1^2 + \cdots + \sigma_J^2 = \alpha_1 + \cdots + \alpha_J = \beta_1 + \cdots + \beta_J.$$

Therefore, the coefficients $\beta_i$ can be used to determine the percentages of explanation of the disjoint axes. Thus, the percentage of the total variation explained by the $q$-th disjoint axis is given by

$$\frac{\beta_q}{\sum_{i=1}^J \beta_i} \times 100\% = \frac{\beta_q}{\sum_{j=1}^J \sigma_j^2} \times 100\%. \tag{16}$$

The amount $\sum_{j=1}^J \sigma_j^2$ can be obtained directly from the matrix $\boldsymbol{X}$ and the coefficients $\beta_j$ may be estimated from $\text{Var}(A_j)$. In the case $Q = J$, when applying the CBPSO-PCA algorithm to obtain the disjoint components, the loading matrix $\boldsymbol{B} = \boldsymbol{I}_J$ is obtained and then

$$\boldsymbol{A} = \boldsymbol{X}\boldsymbol{B} = \boldsymbol{X}.$$

If $Q < J$, the columns of $\boldsymbol{B}$ are disjoint with norm equal to one. Thus, they form an orthonormal set of $Q$ vectors in $\mathbb{R}^J$, and additionally $\boldsymbol{B}^\top \boldsymbol{B} = \boldsymbol{I}_Q$. Hence, $\boldsymbol{B}$ can be considered as a projection matrix in the new rotated disjoint system of axes and $\boldsymbol{A} = \boldsymbol{X}\boldsymbol{B}$ is the matrix containing the coordinates of the $I$ objects found in the vector subspace generated by the first $Q$ disjoint components.

Note that our proposal is to use the variance of scores, that is, we can apply the indicators defined in (16). The variance of the columns of $\boldsymbol{A}$ is employed to calculate the variance explained by each disjoint axis obtained, and for each principal plane required. This criterion for calculating the variance of the columns of the scores matrix is utilized to determine the proportion of variability explained by each factor axis, both in the DPCA and CBPSO-PCA in the numerical examples of the following sections. Observe that is not required to calculate the matrix of singular values $\boldsymbol{\Lambda}$.

### 3.5 Application of the CBPSO-PCA algorithm

Algorithm 3 presented below is a guide to how using Algorithm 2 (CBPSO-PCA, given in Sect. 3.3) when computing the disjoint components in a practical context, that is, when a centered data matrix $\boldsymbol{X}$ is available.

**Algorithm 3** Application of the CBPSO-PCA algorithm

1: Read the $I \times J$ centered data matrix $\boldsymbol{X}$.
2: Perform a usual exploratory PCA.
3: Compute the variance explained by the principal components.
4: Find the number of disjoint components $Q$ using the investigator's criterion based on Step 3.
5: Calculate $Q$ according to the CBPSO-PCA:
  5.1 Establishing the stopping conditions.
  5.2 Stating the number of particles, maximum, minimum inertia, and social, cognitive weights.
  5.3 Applying Algorithm 2.
6: Repeat Step 5 until finding the best fit.
7: Report the obtained results.

## 4 Empirical illustrations

In this section, we show the advantages of the CBPSO-PCA algorithm, illustrating with empirical data the quality of the solutions that can be found. First, we consider two simulated examples and then a real environmental data example.

### 4.1 Computational and simulation aspects

All computational experiments are performed on a 64-bit Windows 10 computer, 8 GB of RAM, and an Intel (R) Core (TM) i7-4510U 2-2.60 GHz processor. The algorithm is implemented employing `C#.NET` and `R`. The use of the `R` statistical software (R Core Team 2018) is important primarily for performing the SVD. More specifically, the `irlba` package is used as it provides a fast and efficient way to calculate a partial SVD for large matrices, instead of using the `svd` function of the `svd` package of `R`, that provides a generic SVD of a matrix. Communication between `C#.NET` and `R` is possible using the `R.NET` middleware, which is installed in the corresponding code project as a `NuGet` package. The data matrix is located in an `Excel` sheet and is read from `C#.NET` code at runtime using `COM+`.

The stopping criterion used in the CBPSO-PCA algorithm is the number of iterations. However, each time the algorithm finds a better solution, it is stored along with its processing time. The best solution found by the algorithm usually has a processing time less than the time needed for all iterations to run.

### 4.2 Generator with disjoint component structure

A simulation algorithm is constructed to randomly generate a data matrix, with an ad-hoc structure of easy interpretation. Then, when applying a dimension reduction by

disjoint principal components, the CBPSO-PCA algorithm is able to detect it.

Let $x_1, \ldots, x_p$ be the values of original $p$ variables and $y_1, \ldots, y_q$ be the values of $q$ latent variables ($q < p$) with disjoint structure. Consider the linear combination given by $y_i = c_{1,i}x_1 + \cdots + c_{p,i}x_p$. If the $m$ original consecutive variables $x_j, x_{j+1}, \ldots, x_{j+(m-1)}$ are represented in the latent variable $y_i$, then the scalars $c_{j,i}, c_{j+1,i}, \ldots, c_{j+(m-1),i}$ are defined as independent and uniform discrete distributed random variables with support on the integers from 70 to 100. Note that the remaining scalars are defined similarly but with support on the integers from 1 to 30. This procedure is performed for each $i$ from 1 to $q$. Keep in mind that each original variable must have a strong presence in one single latent variable.

***Example 2*** Suppose that $p = 8$, $q = 3$ and $\mathrm{U_d}(a, b)$ denotes the discrete uniform distribution, with support on the integers from $a$ to $b$ ($a, b \in \mathbb{N}$, $a < b$). Note that the first linear combination is stated as $y_1 = c_{1,1}x_1 + c_{2,1}x_2 + c_{3,1}x_3 + c_{4,1}x_4 + c_{5,1}x_5 + c_{6,1}x_6 + c_{7,1}x_7 + c_{8,1}x_8$. If $x_1, x_2, x_3, x_4$ are represented in $y_1$, then $c_{1,1}, c_{2,1}, c_{3,1}, c_{4,1} \overset{\text{IID}}{\sim} \mathrm{U_d}(70, 100)$ and $c_{5,1}, c_{6,1}, c_{7,1}, c_{8,1} \overset{\text{IID}}{\sim} \mathrm{U_d}(1, 30)$, where "IID' denotes "independent and identically distributed". The second linear combination is defined as $y_2 = c_{1,2}x_1 + c_{2,2}x_2 + c_{3,2}x_3 + c_{4,2}x_4 + c_{5,2}x_5 + c_{6,2}x_6 + c_{7,2}x_7 + c_{8,2}x_8$. If $x_5, x_6, x_7$ are represented in $y_2$, then $c_{5,2}, c_{6,2}, c_{7,2} \overset{\text{IID}}{\sim} \mathrm{U_d}(70, 100)$ and $c_{1,2}, c_{2,2}, c_{3,2}, c_{4,2}, c_{8,2} \overset{\text{IID}}{\sim} \mathrm{U_d}(1, 30)$. And the third linear combination is expressed as $y_3 = c_{1,3}x_1 + c_{2,3}x_2 + c_{3,3}x_3 + c_{4,3}x_4 + c_{5,3}x_5 + c_{6,3}x_6 + c_{7,3}x_7 + c_{8,3}x_8$. We only have left the original variable $x_8$, which must be represented in $y_3$. Then, we get $c_{8,3} \sim \mathrm{U_d}(70, 100)$ and

$$c_{1,3}, c_{2,3}, c_{3,3}, c_{4,3}, c_{5,3}, c_{6,3}, c_{7,3} \overset{\text{IID}}{\sim} \mathrm{U_d}(1, 30).$$

In Example 2, the first four original variables have a strong presence in the first latent variable, the next three original variables do so in the second latent variable, and the last original variable has a strong representation in the third and last latent variable. We indicate this, in a general way, by the finite succession $\{r_k\}_{k=1}^{q}$, whose elements for the previous particular case are $r_1 = 4$, $r_2 = 3$ and $r_3 = 1$. The algorithm designed simulates a matrix with $n$ individuals and $p$ variables. The orthonormalization Gram-Schmidt process is applied to this matrix to obtain the simulated data matrix $\boldsymbol{X}$. The algorithm that builds the $n \times p$ random matrix $\boldsymbol{X}$ should, in general, implement the mapping $\phi$ defined as

$$\phi\colon \mathbb{N} \times \mathbb{N} \times \mathbb{N} \times \mathbb{N}^q \to M_{n \times p}$$
$$\left(n,\ p,\ q, \{r_k\}_{k=1}^{q}\right) \mapsto \phi\left(n,\ p,\ q, \{r_k\}_{k=1}^{q}\right)$$

subject to the constraints $n \geq p$ and $q < p$. In addition, it should be satisfied that

$$p = \sum_{k=1}^{q} r_k.$$

The mapping $\phi$ returns an $n \times p$ matrix $\boldsymbol{X}$ that has the ad-hoc structure mentioned above, which should be detected by the CBPSO-PCA algorithm.

### 4.3 Simulation studies

For the first simulation, the mapping $\phi(100, 8, 3, \{4, 3, 1\})$ is executed, that is, there are 100 individuals and 8 original variables and then a $100 \times 8$ data matrix $\boldsymbol{X}$ is obtained. In addition, the latent structure is made up of three disjoint components: the first one has non-zero charges in the first four positions. The second disjoint component has non-zero charges at 5-th, 6-th and 7-th positions. The last disjoint component has a non-zero charge at the last position. The CBPSO-PCA algorithm provides the loading matrix $\boldsymbol{B}$ reported in Table 2.

The CBPSO-PCA is executed 100 times with 50 particles and 10 iterations as the stopping criterion. It should be stressed that no background process is run on the computer while the calculations are being made. In all 100 executions, the same solution is reached as shown in Table 2, needing two or three iterations at most. The CBPSO-PCA algorithm is able to detect the ad-hoc structure contained in the data. The fit obtained when applying disjoint principal components is 0.021859878 and the total variance explained by the model is 97.81%. The DPCA algorithm is also executed 100 times and the same solution given in Table 2 is reached 100% of the times. A usual PCA is also

**Table 2** Loading matrix $\boldsymbol{B}$ obtained by CBPSO-PCA and DPCA algorithms with simulated data

| | $Y_1$ | $Y_2$ | $Y_3$ |
|---|---|---|---|
| $X_1$ | 0.43413655 | 0 | 0 |
| $X_2$ | 0.53903007 | 0 | 0 |
| $X_3$ | 0.56700392 | 0 | 0 |
| $X_4$ | 0.44663027 | 0 | 0 |
| $X_5$ | 0 | 0.57919604 | 0 |
| $X_6$ | 0 | 0.56750625 | 0 |
| $X_7$ | 0 | 0.58520817 | 0 |
| $X_8$ | 0 | 0 | 1 |
| % of EV | 35.54% | 33.01% | 29.26% |

where EV denotes the explained variance

performed on the same data matrix and the loading matrix defined in Table 3 is obtained. Note that the usual PCA does not clearly detect the underlying structure in the data. For example, what happens to the second original variable $X_2$ is noteworthy: it cannot be concluded with certainty in which component it has a strong presence. The same can be said of the last original variable $X_8$.

A second simulation study is conducted to compare the performance of the CBPSO-PCA and DPCA algorithms when the matrix size is large. The simulation algorithm implements the mapping $\phi(200, 200, 3, \{50, 70, 80\})$ and a $200 \times 200$ data matrix $\boldsymbol{X}$ with three disjoint components. After executing both algorithms 100 times each, the results obtained are summarized in Table 4. Both algorithms found the best solution, with a fit of 0.02543703 and a total explained variance of 97.46%, with different success rates. The CBPSO-PCA algorithm is executed with the same input parameters as in the first simulation study, and the best time reached is 4.2 minutes in 6 iterations. In addition, for the DPCA algorithm, the best time reached is 9.8 minutes in two iterations. The last row of Table 4 reports that the DPCA is trapped in a local optimum in 7 of the 100 executions. It is important to highlight that the DPCA algorithm does not perform an exhaustive search, since starting from the binary matrix $\boldsymbol{V}$ that it initially generates randomly, it begins to move by column the number one in all possible positions.

Note that, as the number of original variables and latent variables increases, we detect a larger processing time in this algorithm. The CBPSO-PCA algorithm is better suited for large arrays, since it has the flexibility to fit its parameters allowing a better search strategy. The DPCA algorithm has only one parameter (tolerance) that is used as a stopping criterion. With respect to the percentages of success, the DPCA algorithm has a larger probability of being trapped in a local optimum, since it begins its processing with a single binary matrix $\boldsymbol{V}$, that is modified by displacements of the one row and column, not necessarily leading to an optimal matrix. As mentioned in Macedo and Freitas (2015), the algorithm can be trapped in different local optimum so that it is recommended to execute it several times. In addition, the CBPSO-PCA algorithm starts with a matrix $\boldsymbol{V}$ for each particle, and it is the joint and intelligent search of the particles, which permits the CBPSO-PCA to have a small probability of being trapped in a local optimum.

**Table 3** Loading matrix $\boldsymbol{B}$ obtained by usual PCA algorithm with simulated data

| | $Y_1$ | $Y_2$ | $Y_3$ |
|---|---|---|---|
| $X_1$ | 0.39512468 | −0.09020236 | 0.14824807 |
| $X_2$ | 0.39775809 | −0.00322755 | 0.40788776 |
| $X_3$ | 0.44254179 | −0.17190775 | 0.31755350 |
| $X_4$ | 0.35239424 | −0.11448361 | 0.25061667 |
| $X_5$ | 0.26682407 | 0.46011771 | −0.25159988 |
| $X_6$ | 0.30662152 | 0.39441186 | −0.26207760 |
| $X_7$ | 0.35705213 | 0.28282245 | −0.38998176 |
| $X_8$ | −0.27007773 | 0.70847497 | 0.60326462 |
| % of EV | 39.81% | 32.27% | 27.92% |

where EV denotes the explained variance

**Table 4** Summary of the second simulation study

| Characteristic | CBPSO-PCA | DPCA |
|---|---|---|
| Average execution time (in minutes) | 4.6 | 10.2 |
| Best execution time (in minutes) | 4.2 | 9.8 |
| Worst execution time (in minutes) | 4.8 | 10.4 |
| Success rate | 100% | 93% |

We compare the convergence of the proposed algorithm with the usual PCA. If the mathematical optimization model for obtaining disjoint components is relaxed, eliminating precisely the constraint for the calculation of these components, the minimization model for calculating the usual principal components is obtained. Therefore, the fit of this PCA is always better than the fit of a DPCA. Figure 1 shows a black curve that relates the number of iterations of the CBPSO-PCA algorithm with the fit (value of the objective function). The gray line, parallel to the horizontal axis, shows the fit of a standard PCA. For the second simulation study, the fit value of a PCA is so small that the R software represents that fit with a zero. Observe that the best fit obtained by the CBPSO-PCA algorithm in one of the executions is 0.02543703 in 6 iterations of a maximum of 7 iterations.

## 4.4 Application to real environmental data

A problem that arises when conducting studies with environmental or ecological data is that of the presence of

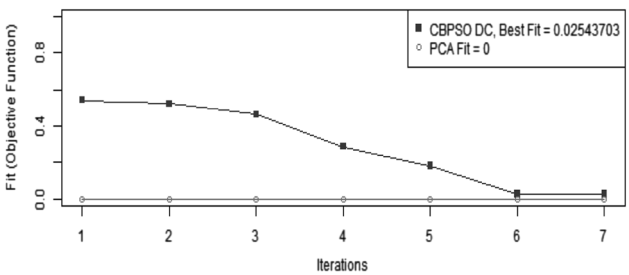


**Fig. 1** Convergence analysis for the second simulation study according to the method indicated

redundant variables, that is, variables that are an exact or approximate linear combination of one or more of the remaining variables. These variables can be eliminated without losing information or with minimal loss of it, resulting in a smaller data set, reducing the costs for the study.

Eliminating redundant variables, and retaining those that provide the most information is relevant, particularly if the data set is massive, as it is often occurs in environmental studies. The PCA allows us to reduce the dimensionality of the problem under study. Since the principal components are not correlated, they remove redundant information. The problem that arises, as already indicated, is its interpretation, because the loadings do not always approach zero in absolute value. The CBPSO-PCA permits us to alleviate this problem by presenting the variables that contribute little information (to a certain principal component) with values equal to zero. Thus, we can select the variables that provide the most information to the first disjoint principal component (that is, the one that explains a high variability) and discard the variables that are found in the remaining disjoint principal components. Obviously, this decision depends on many factors such as the type of research to be carried out, the importance of the variables to be excluded and the knowledge of the researcher who collected the data.

There is an extensive bibliography in which different methodologies are presented for reducing the number of variables in a multivariate study, among which we can highlight: the seminal work of Jolliffe (1973), who used PCA through multiple linear regression, and the work of King and Jackson (1999), which is more specific, since there different methods are proposed to discard redundant variables in environmental studies. King and Jackson (1999) showed that the Broken-Stick method is the best criterion to determine the number of principal components to retain, and then select the variables to be discarded using the different models developed by Jolliffe (1973).

When the size of a data set is decreased, a stability problem arises, which consists of the concordance between the distances of the individuals in the space generated by the original variables and the space generated by the set containing the selected variables (Sorzano et al. 2014). In order to reduce the number of the original variables using PCA in ecological data to be stable, the interested reader is referred to Grossman et al. (1991), where the ratio between original variables and selected variables is considered to be less or equal than three.

In this application, we use the Broken-Stick method and the CBPSO-PCA algorithm to select the variables to be retained in an environmental study. The data set used is taken from a database corresponding to the FOREGSEuroGeoSurveys Geochemical Baseline available at http://www.gtk.fi/publ/foregsatlas.

The data correspond to samples of contents of minerals taken in the subsoil of different countries of the Union European. The list of variables is shown in Table 5. These data can be used to measure subsurface contamination and obtained from the file `C_XRF_data_2v6_8Fe-b06.xls`, available at http://weppi.gtk.fi/publ/foregsatlas/ForegsData.php.

For the analysis, the columns corresponding to the coordinates of their geographical location and the country to which they correspond have been eliminated. The data matrix consists of 19 columns or variables (the elements and chemical compounds) and 788 rows (locations where the samples are collected). The variables represent the content of mineral found in the soil at each sampled location. Since the units of measurement differ, depending on the variable, we standardize the variables for the analysis.

We proceed as follows. First, we apply a usual PCA to the data set. Second, we select $Q$, the number of principal components to consider using the Broken-Stick method. Third, we calculate the $Q$ disjoint principal components with the CBPSO-PCA algorithm. And fourth, we take into account the variables that make up the first disjoint principal component. The results of the PCA of the data matrix are reported in Table 6.

**Table 5** List of variables under study

| Variable | Chemical compound/element | Measurement units |
|---|---|---|
| SIO2_SI | Silicon oxide | kg |
| TIO2_SI | Titanium oxide | kg |
| AL2O3_SI | Aluminum oxide | kg |
| FE2O3_SI | Iron oxide | kg |
| MNO_SI | Manganese oxide | kg |
| MGO_SI | Magnesium oxide | kg |
| CAO_SI | Calcium oxide | kg |
| NA2O_SI | Sodium oxide | kg |
| K2O_SI | Potassium oxide | kg |
| P2O5_SI | Phosphorus oxide | kg |
| BA_SI | Barium | mg |
| CR_SI | Chrome | mg |
| RB_SI | Rubidium | mg |
| SN_SI | Tin | mg |
| SR_SI | Strontium | mg |
| W_SI | Tungsten | mg |
| Y_SI | Yttrium | mg |
| ZN_SI | Zinc | mg |
| ZR_SI | Zirconium | mg |

Figure 2 shows a graph corresponding to the Broken-tick method (black line) superimposed on a gray curve that indicates the explained variance percentage by considering the principal components indicated on the x-axis. The point cutting is close to the third principal component. For this reason, we take into account three principal components to retain (that is, $Q = 3$). These first three principal components explain 51.66 % of the total variability. Therefore, we use $Q = 3$ to apply the CBPSO-PCA algorithm, whose results are reported in Tables 7 and 8. The three disjoint components explain 46.78 % of the total variability. If we consider the first three principal components suggested by the Broken-Stick method, which explain 51.96 % of the variability, we have that the disjoint components explain 90.01 % of the variance by using the usual first three principal components. This low percentage explained by the disjoint components is due to the fact that they must satisfy a set of additional restrictions to the usual principal components; see Sect. 2. In Fig. 3, the results of Table 7 are represented, where we clearly appreciate how the original variables are distributed in the disjoint principal components, their weight and their sign.

The convergence analysis shows that the CBPSO-PCA algorithm takes six iterations to converge to the value 0.5322, whereas he CBPSO algorithm converges in just 5

**Table 6** Results of the PCA of the environmental data matrix

| #PC | Eigenvalue | % of EV | % of AEV |
|---|---|---|---|
| 1 | 5.095 | 26.815 | 26.815 |
| 2 | 2.941 | 15.477 | 42.292 |
| 3 | 1.838 | 9.674 | 51.966 |
| 4 | 1.488 | 7.833 | 59.799 |
| 5 | 1.301 | 6.845 | 66.644 |
| 6 | 1.025 | 5.397 | 72.041 |
| 7 | 0.904 | 4.759 | 76.800 |
| 8 | 0.797 | 4.193 | 80.993 |
| 9 | 0.722 | 3.801 | 84.794 |
| 10 | 0.650 | 3.423 | 88.217 |
| 11 | 0.489 | 2.572 | 90.789 |
| 12 | 0.412 | 2.166 | 92.955 |
| 13 | 0.387 | 2.039 | 94.994 |
| 14 | 0.325 | 1.708 | 96.703 |
| 15 | 0.267 | 1.408 | 98.110 |
| 16 | 0.125 | 0.657 | 98.768 |
| 17 | 0.105 | 0.554 | 99.322 |
| 18 | 0.079 | 0.417 | 99.739 |
| 19 | 0.050 | 0.262 | 100.000 |

where *PC* principal component, *EV* explained variance, and *AEV* accumulated explained variance

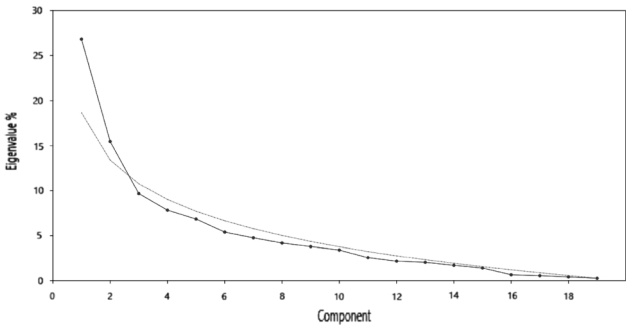


**Fig. 2** Broken-Stick plot with environmental data (in black) and % of explained variance by the indicated principal component (in gray)

**Table 7** Results of the CBPSO-PCA for the indicated variable and component with environmental data

| | # disjoint principal component | | |
|---|---|---|---|
| Variable | 1 | 2 | 3 |
| SIO2_SI | 0 | 0 | 0.57973524 |
| TIO2_SI | 0.44020387 | 0 | 0 |
| AL2O3_SI | 0 | −0.46399195 | 0 |
| FE2O3_SI | 0.49562162 | 0 | 0 |
| MNO_SI | 0.44137161 | 0 | 0 |
| MGO_SI | 0 | 0 | −0.34305531 |
| CAO_SI | 0 | 0 | −0.55125267 |
| NA2O_SI | 0 | −0.28220297 | 0 |
| K2O_SI | 0 | −0.49633089 | 0 |
| P2O5_SI | 0.27387266 | 0 | 0 |
| BA_SI | 0 | −0.43570013 | 0 |
| CR_SI | 0.18954184 | 0 | 0 |
| RB_SI | 0 | −0.47588306 | 0 |
| SN_SI | 0 | −0.15767984 | 0 |
| SR_SI | 0 | 0 | −0.34117431 |
| W_SI | 0 | −0.13253821 | 0 |
| Y_SI | 0.42150555 | 0 | 0 |
| ZN_SI | 0.27779775 | 0 | 0 |
| ZR_SI | 0 | 0 | 0.35488125 |
| % of EV | 17.34% | 16.86% | 12.58% |

where EV denotes the explained variance

**Table 8** Results of the CBPSO-PCA of the environmental data matrix

| #PC | % of EV | % of AEV |
|---|---|---|
| 1 | 17.34 | 17.34 |
| 2 | 16.86 | 34.20 |
| 3 | 12.58 | 46.78 |

where *PC* principal component, *EV* explained variance, and *AEV* accumulated explained variance

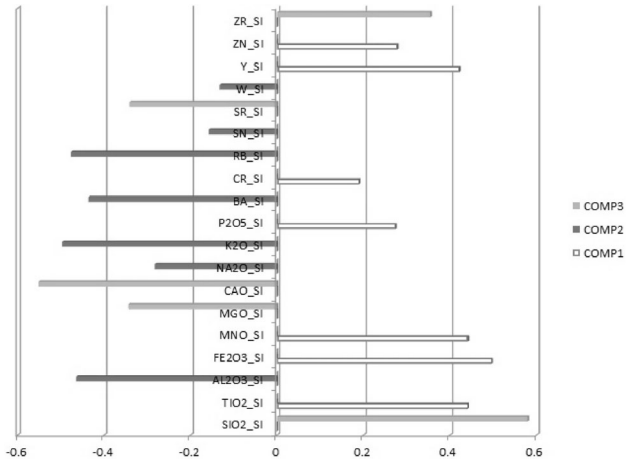


**Fig. 3** Disjoint components obtained by the CBPSO-PCA algorithm for environmental data

iterations, reaching its best fit that differs by 0.051 from the fit achieved when using the usual PCA algorithm, indicating a lost in fit but a gain in interpretation; see Fig. 4. The variables to select, according to our proposal, are the variables that appear in the first disjoint principal component, which is the one with the largest percentage of explained variance. There are seven variables with non-zero loadings, which are:

- TIO2_SI,
- FE2O3_SI,
- MNO_SI,
- P2O5_SI,
- CR_SI,
- Y_SI,
- ZN_SI.

The ratio between the number of original variables and the number of selected variables is $19/7 = 2.71$, which satisfies the recommendation given in Grossman et al. (1991).

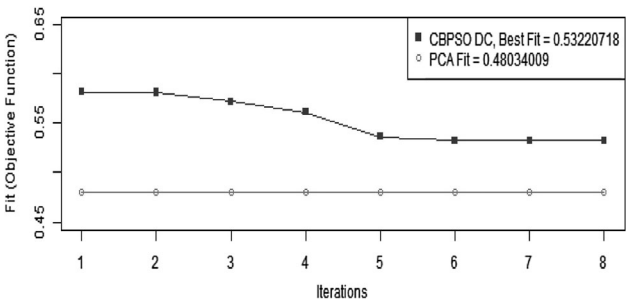


**Fig. 4** Convergence of the CBPSO-PCA algorithm with environmental data according to the method indicated

## 5 Conclusions, discussion and future research

This paper reported the following findings:

(i) A new methodology for calculating principal components based on an intelligent search is proposed, which uses binary optimization with constrains by means of particle swarm.
(ii) A stochastic optimization method was used to find solutions in cases of high computational complexity.
(iii) A numerical evaluation of the proposed methodology was considered by means of Monte Carlo simulations.
(iv) By using a case study with real-world environmental data, we have illustrated the new methodology for DPCA.

The new methodology consists of a PCA with constraints, allowing us to determine components that are linear combinations of disjoint subsets of the original variables. Because the components are of norm one and disjoint, the matrix of factorial loads is orthogonal, enabling us to interpret it as a rotation of the space of individuals. This permits us to preserve the original variability in the rotated space, and to use, in a natural way, the variance of the new variables as an indicator of the variance explained by the disjoint components. By replacing the local search with the intelligent search, the set of feasible solutions is better explored, enabling us high-quality solutions to be reached regardless of the size of the matrix. In addition, the particles have shared memory enabling them to ignore routes that lead to solutions that represent local optima, something which is seen in the percentage of success in obtaining the best solution for the empirical illustration with simulated data. The numerical evaluations of the proposed methodology with simulated and real data sets allowed us to show its good performance and its potential applications. We obtained a new technique which can be a useful knowledge addition to the tool-kit of diverse practitioners, environmental engineers, applied statisticians, and data scientists.

Some open problems that arose from this study are the following:

(i) DPCA seeks the global optimum through successive local optimal choices, that is, it is an algorithm of the type known as greedy algorithms. For small instances, a greedy algorithm usually provides the optimal solution, but in larger instances, it is usually trapped in a local optimum. In general, greedy algorithms provide a good solution, but the quality of this solution usually degrades as the size of the data set increases. This

is shown in the two empirical illustrations with simulated data in Sect. 4. When the number of individuals is small, as in the first simulation (100 individuals and 8 variables), both the DPCA algorithm (that uses alternating least squares) and the proposed algorithm (CBPSO-PCA) reach the best solution at 100% of the simulated cases. Note that by increasing the size of the $\boldsymbol{X}$ matrix (200 individuals and 200 variables), the DPCA algorithm achieves the best solution in 93% of the cases, while the CBPSO-PCA algorithm does so in 100% of the cases. It should be mentioned that the fit obtained with the usual PCA is slightly better than the fit obtained with the DPCA. However, what is lost in fit is compensated by obtaining a factor structure that is easier to interpret. This is an aspect to be improved for the new methodology.

(ii) Regression modeling, errors-in-variables, functional data analysis and PLS regression, based on the proposed methodology, are also of interest (Huerta et al. 2019; Martinez et al. 2019; Carrasco et al. 2020).

(iii) In the present work, the global topology has been used (all particles communicate with all). An open area of research on this topic is to determine other topologies to determine if there are any that accelerate the convergence of the algorithm.

(iv) Inertia decreases in a linear way. Then, it is possible to test our methodology with another type of variation of the coefficients of inertia, such as slowed exponential decrease. In addition, we can try using other functions instead of the sigmoidal function, for example, we can use the probit function.

(v) Other applications of the algorithm developed in the context of multivariate analysis are: discriminant analysis, correspondence analysis, and cluster analysis, as well as the already mentioned functional data analysis and PLS.

(vi) There is also a promising field of applications in the so-called statistical learning; for example, for image compression.

(vii) Another frequently studied problem in environmental sciences is to identify the spatio-temporal variability in environmental fields by using the PCA. A good example is given by Barnston and Livezey (1987) for atmospheric circulation patterns. Nevertheless, Hsieh (2004) stated that the drawback of linear methods is that only linear structures can be correctly extracted from the data, but some environmental data can be highly nonlinear. An impact of nonlinearity in time series of environmental data on the performance of the method proposed in the present study is of interest and deserves further research.

Therefore, the new methodology proposed in this study promotes new challenges and opens issues to be explored from the theoretical and numerical perspectives. Research on these and other issues are in progress and their findings will be reported in future articles.

**Acknowledgements** The authors thank the editors and reviewers for their constructive comments on an earlier version of this manuscript. The research was partially supported by ESPOL Polytechnic University, Escuela Superior Politécnica del Litoral, ESPOL, Guayaquil, Ecuador (J.A. Ramirez-Figueroa and C. Martin-Barreiro), and by FONDECYT (grant 1200525) from the National Agency for Research and Development (ANID) of the Chilean government (V. Leiva).

## References

Alatas B, Akin E (2008) Rough particle swarm optimization and its applications in data mining. Soft Comput 12:1205–1218

Barnston AG, Livezey RE (1987) Classification, seasonality and persistence of low-frequency atmospheric circulation patterns. Mon Weather Rev 115:1083–1126

Beaton D, Chin Fatt C, Abdi H (2014) An exposition of multivariate analysis with the singular value decomposition in R. Computatl Stat Data Anal 72:176–189

Carrasco JMF, Figueroa-Zuniga JI, Leiva V, Riquelme M, Aykroyd RG (2020) An errors-in-variables model based on the Birnbaum-Saunders and its diagnostics with an application to earthquake data. Stoch Environ Res Risk Assess 34:369–380

Chu W, Gao X, Sorooshian S (2011) Fortify particle swarm optimizer with principal components analysis: a case study in improving bound-handling for optimizing high-dimensional and complex problems. IEEE Congr Evolut Comput 2011:1644–1648

Esmin A, Matwin S(2012) Data clustering using hybrid particle swarm optimization. In: *Proceedings of the 13th international conference on intelligent data engineering and automated learning*, pp. 159–1662

Ferrara C, Martella F, Vichi M (2016) Dimensions of well-being and their statistical measurements. Studies in theoretical and applied statistics. Springer, NY, pp 85–99

Freitas A, Macedo E, Vichi M (2020) An empirical comparison of two approaches for CDPCA in high-dimensional data. *Statistical Methods and Applications*, pages in press available at https://doi.org/10.1007/s10260-020-00546-2

Frutos E, Galindo MP, Leiva V (2014) An interactive biplot implementation in R for modeling genotype-by-environment interaction. Stoch Environ Res Risk Assess 28:1629–1641

Gajawada S, Toshniwal D (2012) Projected clustering using particle swarm optimization. Proc Technol 4:360–364

Grossman GD, Nickerson DM, Freeman MC (1991) Principal component analyses of assemblage structure data: utility of tests based on eigenvalues. Ecology 72:341–347

Hastie T, Tibshirani R, Friedman J (2009) The elements of statistical learning. Springer, NY

Hsieh WW (2004) Nonlinear multivariate and time series analysis by neural network methods. Rev Geophy 42(RG1003):1–25

Huerta M, Leiva V, Liu S, Rodriguez M, Villegas D (2019) On a partial least squares regression model for asymmetric data with a chemical application in mining. Chemom Intell Lab Syst 190:55–68

Imran M, Hashim R, Khalid NEA (2013) An overview of particle swarm optimization variants. Proc Eng 53:491–496

Jolliffe IT (1973) Discarding variables in a principal component analysis. II: real data. J R Stat Soc C 22:21–31

Jolliffe IT (2002) Principal component analysis. Springer, New York

King JR, Jackson DA (1999) Variable selection in large environmental data sets using principal components analysis. Environmetrics 10:67–77

Lou S, Wu P, Guo L, Duan Y, Zhang X, Gao J (2020) Sparse principal component analysis using particle swarm optimization. J Chem Eng Jpn 53:327–336

Ma B, Ji H (2012) Particle swarm optimization algorithm establish the model of tobacco ingredients in near infrared spectroscopy quantitative analysis. Int Conf Comput Comput Technol Agric 393:92–98

Macedo E, Freitas A (2015) The alternating least-squares algorithm for CDPCA. In: Plakhov A, Tchemisova T, Freitas A (eds) Optimization in the natural sciences. Springer, NY, pp 173–191

Mahoney MW, Drineas P (2009) CUR matrix decompositions for improved data analysis. Proc Natl Acad Sci U. S. Am 106:697–702

Martí L, García J, Berlanga A, Molina JM (2009) An approach to stopping criteria for multi-objective optimization evolutionary algorithms: The mgbm criterion. In: *2009 IEEE congress on evolutionary computation*, pp. 1263–1270

Martinez S, Giraldo R, Leiva V (2019) Birnbaum-Saunders functional regression models for spatial data. Stoch Environ Res Risk Assess 30:1765–1780

Nezamabadi-Pour H, Rostami-Sharbabaki M, Farsangi M (2008) Binary particle swarm optimization: challenges and new solutions. J Comput Soc Iran 6:21–32

Nieto-Librero AB (2015) *Inferential version of biplot methods based on bootstrap resampling and its application to three-way tables.*. PhD thesis, Universidad de Salamanca, Salamanca, Spain (In Spanish)

Nieto-Librero AB, Sierra-Fernández C, Vicente-Galindo MP, Ruíz-Barzola O, Galindo-Villardón MP (2017) Clustering disjoint hj-biplot: a new tool for identifying pollution patterns in geochemical studies. Chemosphere 176:389–396

R Core Team (2018) R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria

Sangwook L, Sangmoon S, Sanghoun O, Witold P, Moongu J (2008) Modified binary particle swarm optimization. Prog Natl Sci 18:1161–1166

Song S, Wang Q, Chen J, Li Y, Zhang W, Ruan Y (2017) Fuzzy c-means clustering analysis based on quantum particle swarm optimization algorithm for the grouping of rock discontinuity sets. J Civ Eng 21:1115–1122

Sorzano COS, Vargas J, Montano AP (2014) A survey of dimensionality reduction techniques. Retrieved September 5, 2020, from http://arxiv.org/abs/1403.2877

Van der Merwe DW, Engelbrecht A (2003) Data clustering using particle swarm optimization. In: *The 2003 congress on evolutionary computation, 2003*, vol. 1, pp. 215–220

Vasile CI, Buiu C (2011) A software system for collaborative robotics applications and its application in particle swarm optimization implementations. Appl Soft Comput J 11:5498–5507

Vichi M, Saporta G (2009) Clustering and disjoint principal component analysis. Comput Stat Data Anal 53:3194–3208

Vigneau E, Qannari EM (2003) Clustering of variables around latent components. Commun Stat Simul Comput 32:1131–1150

Vines SK (2000) Simple principal components. J R Stat Soc C 49:441–451

Voss MS (2005) Principal component particle swarm optimization: a step towards topological swarm intelligence. In: *2005 IEEE congress on evolutionary computation*, vol. 1, pp. 298–305

Wang L, Liu X, Sun M, Qu J, Wei Y (2018) A new chaotic starling particle swarm optimization algorithm for clustering problems. Math Probl Eng 2018:1–14

Zhao X, Lin W, Zhang Q (2014) Enhanced particle swarm optimization based on principal component analysis and line search. Appl Math Comput 229:440–456

Zou H, Hastie T, Tibshirani R (2006) Sparse principal component analysis. J Comput Gr Stat 15:265–286