Ann Math Artif Intell (2010) 60:153–178
DOI 10.1007/s10472-010-9211-0

Genetic algorithms and particle swarm optimization for exploratory projection pursuit

Alain Berro · Souad Larabi Marie-Sainte · Anne Ruiz-Gazen

Published online: 4 September 2010

Abstract Exploratory Projection Pursuit (EPP) methods have been developed thirty years ago in the context of exploratory analysis of large data sets. These methods consist in looking for low-dimensional projections that reveal some interesting structure existing in the data set but not visible in high dimension. Each projection is associated with a real valued index which optima correspond to valuable projections. Several EPP indices have been proposed in the statistics literature but the main problem lies in their optimization. In the present paper, we propose to apply Genetic Algorithms (GA) and recent Particle Swarm Optimization (PSO) algorithm to the optimization of several projection pursuit indices. We explain how the EPP methods can be implemented in order to become an efficient and powerful tool for the statistician. We illustrate our proposal on several simulated and real data sets.

Keywords Clustering · Exploratory projection pursuit · Genetic algorithm · Particle swarm optimization

Mathematics Subject Classifications (2010) 62-07 · 62-09 · 62H99 · 68-04 · 68T20

A. Berro · S. Larabi Marie-Sainte
IRIT, University Toulouse 1, Toulouse, France

A. Berro
e-mail: berro@irit.fr

S. Larabi Marie-Sainte
e-mail: larabi@irit.fr

A. Ruiz-Gazen (✉)
Toulouse School of Economics (Gremaq and IMT),
University Toulouse 1, Toulouse, France
e-mail: ruiz@cict.fr

1 Introduction

Exploratory Projection Pursuit (EPP) methods are looking for interesting low dimensional projections of high dimensional multivariate data (see [5], for a recent survey). The quality or "interestingness" of a linear projection is measured by a criterion called a projection pursuit index. Thereby, the search for interesting linear projections consists in optimizing an index. The idea originated in Kruskal [26] and Friedman and Tukey [15] were the first to use the term "projection pursuit". From these pioneer papers and until the end of the nineties, the field of EPP widely expanded in the statistical literature. Let us quote Huber [21], Jones and Sibson [22], Friedman [14], Cook et al. [8], Sun [42], Posse [36], Nason [34] and Klinke [24] among the most famous references of that period. In these articles, several projection indices and optimization algorithms have been proposed and studied in detail. These studies conclude that EPP is a powerful tool for statisticians in order to detect hidden non-linear structure in high dimension data sets. The well known Principal Components Analysis (PCA) belongs to the family of EPP methods with the variance as the projection index. However PCA only takes into account second order moment and may miss interesting hidden structure that can be easily discovered by another EPP technique.

For all that, Caussinus and Ruiz-Gazen [5] noticed that the use of EPP is not widely adopted when compared with the extensive use of PCA. This point was already stressed in Nason [33]: "although there has been great interest in projection pursuit methods there does not seem to have been corresponding interest in the software" and, up to our knowledge, there is no recent proposal. In the present paper we take advantage of the high performance computers available nowadays and revisit the field of EPP. Our objective is to make the most of the existing literature and make new proposals in order to obtain a powerful tool that should be implemented in the current toolboxes of the data analyst. In order to reach our goal, we believe that three important conditions have to be fulfilled.

The first condition relies on the suitability of the implemented projection indices. In the literature, there already exist several indices that are suitable for various structures and applications (see [5], for a detailed description). In the present paper we propose a new index that complements the existing ones in the presence of clusters.

The second condition concerns the optimization algorithm. It is commonly accepted that the role played by the optimization algorithm in the field of EPP is crucial. Friedman and Turkey [15] states that "in order to be useful as a tool for exploratory data analysis on data sets with complex structures, it is important that the algorithm find several solutions that represent potentially informative projections for inspection by the researcher". So the problem is not to find only one global optimum but several local optima. Many projection indices are differentiable functions and usual gradient optimization methods have been proposed. In the present paper we focus on stochastic heuristic algorithms because they can be used for any not necessarily regular projection index and easily tuned to obtain different local optima.

The third component of a powerful EPP tool relies on the presentation of the results. In our opinion, the existing implementations are not completely satisfactory in this respect. Most of the implementations we are aware of (see for example the computational statistics toolbox in Matlab by Martinez and Martinez [32]) only rely on a few random starts and usually lead to only one optimum of the projection index. Other projections may be found by structure removal but there is no possibility of direct inspection of several local optima. On the opposite side, the software GGobi [10] is dynamic and the user can look at as many projections as he wants in an interactive way. The initial starting values are either chosen at random or by the user who looks at the point cloud rotating. For high dimension data sets with complex structure, considering only one optimum of the projection index is likely not to be sufficient while looking at rotating clouds for a long time may be tedious. We believe that a proposal in between the two existing approaches is welcome. Following Friedman and Tukey comment, we propose to run at least one hundred times the optimization algorithm and submit the potentially informative projections that follow to the inspection of the statistician.

Many evolutionary algorithms have been developed, such as genetic algorithm [20]. These algorithms are stochastic search heuristics inspired by Darwinian evolution and genetics. The main idea is to create a population of candidate solutions to an optimization problem, which is iteratively refined by alteration and selection of the good solutions. Candidate solutions are selected according to the so-called fitness function, which evaluates their quality with respect to the optimization problem. In case of GAs, the alteration consists of crossover to recombine information of different candidate solutions and mutation to randomly explore the local neighborhood of existing solutions. One promising and recently introduced approach to numerical optimization, which is rather unknown outside the search heuristics field, is particle swarm optimization (PSO) [23]. A PSO algorithm is a population-based search algorithm based on the simulation of the social behavior of birds within a flock. In PSO, individuals, referred to as particles, are flown through search space. Changes to the position of the particles within the search space are based on the social-psychological tendency of particles to emulate the success of other particles. PSO algorithm differs from the other evolutionary methods (typically, the genetic algorithms) and are based on the notion of cooperation between particles. The information exchanged between particles helps to solve difficult problems.

Up to our knowledge, only a few papers in the literature use genetic algorithms in the context of exploratory projection pursuit [1, 11, 18]. These papers do not consider several projection indices and focus on the search of one “global” optimum. Moreover, there is no paper implementing the PSO method. In the present paper, we highlight the interest of using several indices and propose the use of GA and PSO algorithms in order to find several “local” optima. We also provide some plots that help the statistician in interpreting the many different projections obtained. An important advantage of the algorithms we focus on is their ability to cope with local optima by maintaining, recombining and comparing several candidate solutions simultaneously. Moreover, our experience is that the proposed algorithms are easy to implement and have a fast convergence speed and good global and local convergence adaptability. Hence, we believe that GAs and PSO can play a role in exploratory projection pursuit and we illustrate this point in the application section.

Note that the idea of looking for several local optima is already studied in the literature on evolutionary algorithms (see [12], Chapter 9 for a survey) and on PSO [28] also. Such problems are said “multimodal” and several methods such as fitness sharing and crowding exist for tackling these optimization problems. In the present paper we do not implement such methods but we simply propose to run many times standard algorithms. As stated in Eiben and Smith ([12], pp. 158), this idea is the simplest one in order to find several possible interesting solutions. Its implementation is straightforward with no need of tuning more parameters. Another advantage of this simple method in our particular context is that it does not only provide the statistician with a list of possible interesting projections. It also gives insight into the frequency of each local optima and into the differences between the corresponding projection vectors when plotting several graphics as detailed in the application section. Such plots are very helpful for the statistician in order to check whether some projections appear accidentally or in a repeated way. However, for large number of observations and dimensions, it may be worthwhile to accelerate the procedure by using more efficient methods for multimodal problems than the one we propose.

The present paper is dedicated to the analysis of the performance of GA and PSO for the optimization of several one-dimensional projection pursuit indices. The projected data are plotted using histograms and density estimators. The structures we mainly focus on through several examples are clusters and outliers but EPP may be useful to detect other types of structures [8].

This paper is organized as follows. Section 2 introduces four univariate projection pursuit indices and describes their properties while the optimization methods we focus on are considered in Section 3 for GA and Section 4 for PSO. Section 5 reports the main results on different data sets, and finally, Section 6 concludes our study.

2 The projection pursuit approach

The aim of projection pursuit is to reveal possible interesting structures hidden in high-dimensional data. To what extent these structures are "interesting" is measured by an index. It has been argued by Huber [21] and by Jones and Sibson [22] that the Gaussian distribution is the least interesting one, and that the most interesting directions are those that show the least Gaussian distribution. An illustration of the "interestingness" of a projection is illustrated on Fig. 1 where the data are clearly divided into two clusters. However, the first principal component which is the direction of maximum variance, would be vertical, providing no separation between the clusters. In contrast, the interesting projection pursuit direction is horizontal, providing optimal separation of the clusters.

Suppose that, for a set of N cases, P variables are recorded and denote by X the data set matrix with dimension $N \times P$. Then each case can be thought of as a point in a P-dimensional space. Let X_i denote the *ith* column vector in $\Re^P$ associated with the *ith* observation. Unfortunately, unless P is less than or equal to 3 it is not possible to visualize these points in the P-dimensional space. However, it is possible to project a P-dimensional set of points onto a one-dimensional line or a two-dimensional plane, or a three-dimensional volume. In EPP, we try to find "interesting" low-dimensional projections of the data. For this purpose, a suitable index function $I(\alpha)$, depending on a normalized projection vector α, is used. This function will be defined such that interesting views correspond to local and global maxima of the function.

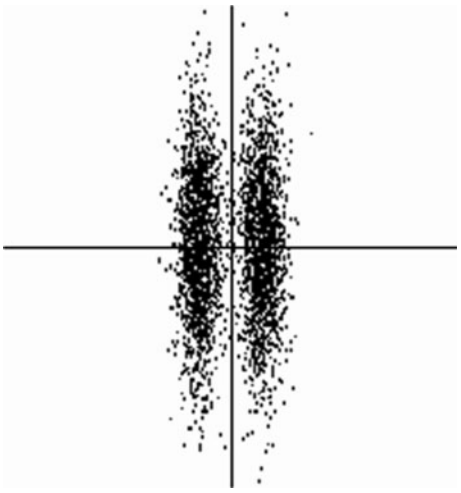

Fig. 1 An illustration of the "interestingness" of a projection

Let us assume that the P-dimensional data set X is spherical, that is the mean vector $E(X_i) = 0$ and the covariance matrix is $V(X_i) = I_P$ where I_P denotes the identity P-dimensional matrix. In practice, the expectation and the covariance of the distribution of the X_i are unknown. People usually *sphere* the data before implementing the projection pursuit procedure using the sample mean $\bar{X} = \frac{1}{N}\sum X_i$ and the sample covariance matrix $\hat{\Sigma} = \frac{1}{N-1}\sum(X_i - \bar{X})(X_i - \bar{X})'$. A simple way of obtaining spherical data is to calculate

$$Z_i = \hat{D}^{-\frac{1}{2}}\hat{U}'(X_i - \bar{X}), i = 1, \cdots, N, \tag{1}$$

where $\hat{U}\hat{D}\hat{U}' = \hat{\Sigma}$ is the spectral decomposition of $\hat{\Sigma}$. In the following, we will consider spherical data and will use the same notation X. Note that by taking spherical data, we remove the effect of location, scale and correlation structure (see [21], for more detail).

To see how EPP methods operate in the case of one-dimensional indices, it is first worth noting that projections from $\Re^P$ to $\Re$ are linear mappings. Let $X = ((x_{ij}))_{i,j}$ be a matrix of which the (i, j)th element is the jth variable for observation i. We can write the general projection from $\Re^P$ to $\Re$ as $Y = X\alpha$. Here α is a P-dimensional row vector defining the linear transformation, and Y is a N-dimensional column vector which corresponds to the coordinates of the projected observations. Choosing a projection is now a matter of choosing α.

There are many possible projection indices and the present paper focus only on four particular one-dimensional indices. The Friedman–Tukey index is the first index proposed in the context of EPP and it is interesting for the detection of outliers. The Friedman's index belongs to the family of the polynomial-based indices. When compared with other indices from the same family, this index performs particularly well in detecting separations or clusters [42]. We also consider a new proposal suited to the detection of clusters called "discriminant index" and the kurtosis index that is based on the fourth moment and has been recently studied in Peña and Prieto [35] and Achard et al. [1].

2.1 The Friedman–Tukey index

The search for interesting structures is done via a projection score like the Friedman–Tukey index

$$I_{FT}(\alpha) = s(\alpha)d(\alpha).$$

It is composed of two parts, s which depends only on the covariance structure and d which captures the "local clusters" of the data. The term s can be avoided if the P-dimensional data X are spherical.

In Jones and Sibson [22], d is expanded as a kernel density estimate:

$$\hat{f}_Y(y) = \frac{1}{Nh}\sum_{j=1}^{N} K\left(\frac{y-y_j}{h}\right) \text{ with } Y = X\alpha,\ \ y_i = \alpha' X_i \text{ and } h \text{ is a bandwidth.} \tag{2}$$

The Friedman–Tukey index can be written as:

$$\hat{I}_{FT}(\alpha) = \frac{1}{N^2 h}\sum_{i=1}^{N}\sum_{j=1}^{N} K\left(\frac{y_i - y_j}{h}\right) \tag{3}$$

It turns out that this index is an estimate of:

$$I_{FT}(\alpha) = \int_R f_Y^2(y)dy = E_Y(f_Y(y)) \tag{4}$$

It is minimized by a parabolic density which is close to a standard normal density. Thus a departure from a parabolic density is also a departure from the standard normal density. For simplicity, Klinke [24] proposes to use the following uniform kernel that we will also consider in what follows.

$$K(u) = \frac{35}{32}(1-u^2)^3 I_{\{|u|\le 1\}} \tag{5}$$

and where I_Ω denotes the dummy variable associated with the set Ω.

Silverman [40] gives the formula for the bandwidth which minimizes the asymptotic mean integrated squared error. If the kernel is a radially symmetric probability density function and the unknown density is bounded and has continuous second derivatives, h is given by:

$$h = 3.12N^{-\frac{1}{6}} \tag{6}$$

So, in our work, we focus on the univariate Friedman–Tukey index:

$$\hat{I}_{FT}(\alpha) = \frac{1}{N^2 h^2}\sum_{i=1}^{N}\sum_{j=1}^{N} K\left(\frac{\alpha'(X_i - X_j)}{h}\right) \tag{7}$$

with K defined by (5) and h defined by (6) and we look for α maximizing this index.

The main drawback of this index is that the projections associated with maxima generally lead to the detection of outliers but may miss other interesting structures such as clusters. The results will confirm this point. Another drawback of this index is that it is time-consuming.

2.2 The Friedman or Legendre index

The main idea of polynomial-based (Legendre, Hermite and Natural Hermite) indices is to approximate a criterion measuring the departure from a reference, in our case from the Gaussian, distribution using expansions based on orthogonal polynomials. Following Cook et al. [8], the index considered here can be written in the form:

$$\int_R (p_Y(y) - \phi(y))^2 g(y)dy \tag{8}$$

where p_Y is the density of the projected data, ϕ is the univariate standard normal density and g is a weight function. The orthogonal polynomials $f_n(x)$ applied in the approximation of the indices are defined by

$$\int_a^b \omega(x) f_n(x) f_m(x)dx = \begin{cases} 0 \text{ if } n \neq m \\ h_n \text{ if } n = m \end{cases} \tag{9}$$

with $n, m \in N$

In the case of the Legendre index, $a = -1$, $b = 1$, $\omega(x) = 1$ and $h_n = 2/(2n + 1)$. The Legendre index, based on Legendre polynomials L_j, was introduced by Friedman [14], with the idea of up-weighting distances in the center of the distribution rather than in the tails. In practice, it was noticed that this index is attracted by skewed distributions.

Let Φ be the cumulative distribution function of an $\mathcal{N}(0, 1)$ random variable (r.v.). Denote the density function of $Y = \alpha' X$ by p_α. Assume that $E(Y_i) = 0$ and $V(Y_i) = I$, the identity matrix. Under the transformation $Y \to R = 2\Phi(Y) - 1$ the density $p_\alpha(y)$ is transformed to the density function $q_\alpha(r)$ of $R = 2\Phi(\alpha' X) - 1$, and a standard normal r.v. will be transformed to a uniform r.v. on $(-1, 1)$. The m-terms *Friedman's index* I_m^F is an estimate of the sum of the first m terms of Legendre polynomial expansion of the L_2 distance between q_α and $1/2$:

$$I_m^F(\alpha) = \sum_{j=1}^{m} \frac{2j+1}{2} \left[\frac{1}{N} \sum_{i=1}^{N} L_j \left(2\Phi(\alpha' X_i) - 1 \right) \right]^2 \tag{10}$$

where the recursive definition of Legendre polynomials L_j is given as follows:

$$\begin{aligned} &L_0(r) = 1,\, L_1(r) = r,\, L_2(r) = \frac{1}{2}(3r^2 - 1), \\ &L_j(r) = \frac{1}{j}(2j - 1)rL_{j-1}(r) - (j - 1)L_{j-2}(r) \text{ for } j = 3, \cdots \end{aligned} \tag{11}$$

It is known that the Friedman's index performs better than the Friedman–Tukey's index in detecting separations or clusters and that will be confirmed in our results section.

The choice of the number m of terms included in this projection pursuit index depends on the data dimension P and sample size N. Friedman [14] suggested that we choose $2 \leq m \leq 6$ in most cases. Sun [42] showed that m should be at least 3. Here, we use $m = 3$.

2.3 The discriminant index

The discriminant index is a new proposal. It follows in some sense the idea in Friedman and Tukey [15] and, contrary to indices such as the Friedman index, it is not dedicated to the search of non-gaussian projections. It is dedicated to the research of clusters. The definition is based on the following argumentation. In the case of supervised classification, i.e., if the clusters are known on a learning data set, discriminant analysis can be easily interpreted as an EPP method [27]. For spherical data, discriminant analysis can be interpreted as the search of projections which minimizes the within-group variance of the projected data. In the present paper, we assume that if clusters are present in the data, they are unknown and so the within variance cannot be calculated and discriminant analysis is not a possible alternative. Following Art et al. [2], Caussinus and Ruiz-Gazen [3] proposed a kind of within matrix estimator in the context of Generalized Principal Components Analysis (GPCA). As principal components analysis and discriminant analysis, the proposed GPCA method does not really belong to the projection pursuit family since the involved optimization step can be solved analytically by using a spectral decomposition with no need of "pursuit". In the present paper, we propose to use the idea of minimizing the within variance of the projected data but in a true projection pursuit context. This leads to the following definition of I_D.

$$I_D(\alpha) = \frac{\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} w(\alpha'(X_i - X_j))(\alpha'(X_i - X_j))^2}{\sum_{i=1}^{N-1} \sum_{j=i+1}^{N} w(\alpha'(X_i - X_j))} \tag{12}$$

where w is a decreasing and positive weight function. Note that, contrary to the previous indices, potential interesting projections are associated with minima of I_D.

As in Caussinus and Ruiz-Gazen [3], we suggest the use of $w(x) = \exp(-x)$. The main difference between our proposal and GPCA is the argument of the weight function w which is $\alpha'(X_i - X_j)$ for I_D instead of $[(X_i - X_j)'(X_i - X_j)]^{\frac{1}{2}}$ for GPCA. Minimization of I_D in α cannot be solved analytically and requires the use of optimization algorithm. Following Lee et al. [27], the method can be easily generalized to two or three dimensional indices. The main drawback of this index is that it is time-consuming.

2.4 The kurtosis index

Peña and Prieto [35] propose a one-dimensional projection pursuit algorithm based on directions obtained by both maximizing and minimizing the kurtosis coefficient of the projected data. As detailed in Peña and Prieto [35], maximizing the kurtosis coefficient of the projected data implies detecting outliers in the projections, whereas minimizing the kurtosis coefficient implies maximizing the bimodality of the projections.

Let us assume that we are given a spherical sample of size N and dimension P, $X_i, i = 1, ..., N$. The kurtosis index is simply the fourth moment of the projected data and is defined by:

$$I_k(\alpha) = \sum_{i=1}^{N} (\alpha' X_i)^4 \tag{13}$$

As stated above local maxima and local minima may be of interest and we will consider both minimization and maximization of I_k. The main advantage of this index is its fast computation.

3 Genetic Algorithms (GA)

Genetic Algorithms (GAs) are evolutionary algorithms that use techniques inspired by Darwinian evolutionary biology such as inheritance, mutation, selection, and crossover. Evolutionary algorithms have been proposed independently by Fogel et al. [13], Rechenberg [37] and Schwefel [39]. Genetic algorithms are implemented in a computer simulation in which a population of candidate solutions (individuals or creatures) to an optimization problem evolves toward better solutions.

The original version of a GA is introduced by Holland [20]. For traditional genetic algorithms, solutions are represented in binary as strings of 0 and 1, but other encodings are also possible. The evolution usually starts from a population of randomly generated individuals covering the search space. Every individual of this population has a function called fitness function which allows to measure its quality or its weight and represents the objective function to be optimized. At each generation, the fitness of every individual in the population is evaluated, multiple individuals are stochastically selected from the current population (based on their fitness), and modified (recombined and possibly randomly mutated) to form a new population. The new population is then used in the next iteration of the algorithm. Commonly, the algorithm terminates when the termination condition has been reached. Therefore, to solve a problem using GA we need the following :

1. A system of encoding the possible solutions or chromosome structure.
2. An initial population of solutions.
3. A function to evaluate a solution's fitness.
4. A method of selection of solutions to be used to produce new solutions.
5. Recombination and mutation operators to create new solutions from the existing ones.

3.1 Solution representation: chromosome structure

In traditional genetic algorithm, solutions are represented in binary as strings of 0 and 1. A major drawback of the binary encoding [16] is that two close points in the variables space are not necessarily coded by two nearby chains of bits. This problem is generally corrected by using different encoding types. In our case, the population is encoded in real numbers. The chromosome represents the vector of projection and is denoted by projection. The projection is defined as a normalized vector of P dimension (see Section 2).

3.2 Initial population

The first population is sometimes randomly created or by using an heuristic. In our work, we randomly initialize the population of size T.

3.3 Evaluation function

This function takes a single solution as a parameter and returns a number indicating how good the solution is. The best solution will be selected by comparing the fitness value returned by all the possible solutions. In this work, we consider a projection index as the fitness function.

3.4 Selection

Selection is a genetic operator that chooses an individual from the current generation for inclusion in the next generation. During each successive generation, a proportion of the existing population is selected to produce a new generation. Certain selection methods rank the fitness of each solution and preferentially select the best solutions. Other methods only rank a random sample of the population, since this process may be very time-consuming. The selection helps keeping the diversity of the population, preventing premature convergence to poor solutions. Popular and well-studied selection methods include roulette wheel selection and tournament selection.

The roulette wheel it is the oldest selection method, where each individual occupies a section of the wheel proportionally to its fitness function, the selection probability of an individual (j) is:

$$\text{Prob}(j) = \frac{\text{Fitness}(j)}{\sum_{j=1}^{T} \text{Fitness}(j)} \tag{14}$$

If we want to select one solution, a spin of the wheel is made and proposes a candidate. Better solutions have proportionally larger section of the wheel and therefore there is a greater probability that they will be selected.

Tournament selection it consists in choosing randomly k individuals of the population, without taking into account their fitness function value, and to choose the best individual among them. The number of selected individuals has an influence on the pressure of selection. When $k = 2$, the selection is said by binary tournament.

In our version, we use tournament selection of three participants.

3.5 Crossover (recombination)

Crossover is applied to randomly paired individuals with a probability denoted p_c. Pick a pair of individuals (two parents). With probability p_c, recombine these individuals to create two new individuals, called offspring (child), that are inserted into the next population. While there are many different kinds of crossover (1-point, 2-points, n-points), the most common type is single point crossover. In single point crossover, we choose a locus at which we swap the remaining alleles from one parent to the other. The children take one section of the chromosome from each parent.

The point at which the chromosome is broken depends on the randomly selected crossover point.

In our algorithm, we apply a 2-point crossover with $p_c = 0.65$ to all the population.

3.6 Mutation

After selection and crossover, a new population of individuals is created. Some are directly copied, and others are produced by crossover. In order to ensure that the individuals are not all exactly the same, a small probability of mutation is allowed. Mutation is only executed on a single individual. It represents random and occasional modification of one allele or some part of the chromosome. We loop through all the alleles of all the individuals, and if that allele is selected for mutation, we can either change it by a small amount or replace it with a new value. Typically the mutation rate is applied with p_m less than 10% probability. Mutation is, however, vital to ensure genetic diversity within the population.

The mutation operator is applied in our case to all the individuals with $p_m = 0.05$.

3.7 GA parameters

The parameters to be tuned by the user are:

1. The population size: it depends on the nature of the problem, but typically contains several hundreds or thousands of possible solutions.
2. Probabilities of crossover and mutation: the values of these probabilities can vary from an application to the other one. For example, the mutation probability is generally very low, lower than 10%, a large probability may modify the best individuals.
3. The termination criterion: either a maximum number of generations or a satisfactory fitness level for the population.

3.8 Our GA version

In this part, we summarize our GA version using the Algorithm 1. The choices of genetic operator parameters made in this study are based on some experimentation.

Algorithm 1 Genetic algorithm pseudocodes

```
input: Initial population of individual: randomly initialized and encoded in float point numbers.
       Fitness function = a projection index.
Evaluate the fitness of each individual in that population.
Determine the best individual.
while a fixed number of iterations is not reached do
    Apply a tournament selection of 3 participants.
    Breed new individuals through genetic operators :
      – Crossover with $p_c = 0.65$
      – Mutation with $p_m = 0.05$
    Evaluate the fitness of new individuals
end
Replace least-fitness population with new individuals
```

We pay attention to choose the number of individuals and the number of training iterations such that the optimum reached cannot be improved easily (by several trials and experimentation). These parameters depend on the data set and on the projection index [20].

4 Particle swarm optimization

Particle swarm optimization is a population based stochastic optimization technique developed by Kennedy and Eberhart [23], inspired by social behavior of bird flocking or fish schooling. The particle swarm paradigm has undergone many tweaks and modifications since its discovery. Various researchers have analyzed and experimented it. In the process, a certain body of lore has emerged to provide hypotheses for research as well as guidelines for application.

PSO shares many similarities with evolutionary computation techniques such as Genetic Algorithms. The system is initialized with a population of random solutions and searches for optima by updating positions. However, unlike GA, PSO has no evolution operators such as crossover and mutation. In PSO, the potential solutions are called particles. All the particles have fitness values which are evaluated by the fitness function to be optimized, and have velocities which direct the flying of the particles. The particles fly through the problem space by following the current optimum particles.

In recent years, many extensions have been suggested to improve the performance of classic PSO algorithm (e.g.: Krink et al. [25] and Lovbjerg and Krink [29]). In this study, we use the original version introduced by Kennedy and Eberhat [23].

4.1 Parameters of the particle swarm

There are several explicit parameters whose values can be adjusted to produce variations in the way the algorithm searches in the problem space. The most important of these parameters are the maximum velocity (denoted by $V_{\max}$) and the inertia. One of these two parameters has to be set at the beginning of the trial and remain constant throughout. Manipulation of these parameters can cause surprising changes in the system's behavior. In fact, $V_{\max}$ is fixed to avoid that the particles move too quickly from a region to the other one in the search space. The inertia factor controls the influence of the velocity obtained in the previous step. A large inertia factor provokes a large exploration of the search space while a small inertia factor concentrates the search on a small space.

In our implementation, we use the $V_{\max}$ parameter.

4.1.1 The fitness

The particle swarm use the concept of fitness, as do all evolutionary computation paradigms. Let $f : \Re^P \to \Re$ be the fitness function that takes a particle's solution as a parameter and returns a number indicating how good the solution is. In our work, the projection pursuit index is the fitness function.

4.1.2 Maximum velocity $V_{\max}$

The particle swarm algorithm proceeds by modifying the distance each particle covers on each dimension per iteration. Changes in the velocity are stochastic, and an undesirable result is that the particle's trajectory, uncontrolled, can expand into wider and wider cycles through the problem space, eventually approaching infinity. Something needs to be done to dampen the oscillations so that the particle searches usefully. The traditional method implements a system constant $V_{\max}$ as described in Algorithm 2.

Algorithm 2 Velocity limitation

```
if $V_{id} > V_{max}$ then $V_{id} = V_{max}$
else if $V_{id} < -V_{max}$ then $V_{id} = -V_{max}$
```

Thus the system parameter $V_{\max}$ has the beneficial effect of preventing explosion and scales the exploration of the particle's search. The choice of a value for $V_{\max}$ depends on some knowledge of the problem and will be specified in the Subsection 4.3.1.

4.1.3 Dependent and independent variables

In an experiment, an independent variable is one that is manipulated, and a dependent variable is one that is measured. PSO is initialized with a group of random particles (solutions) and then searches for optima by updating positions. At each iteration, each particle is updated by following two "best" values. The first one is the best solution (fitness) it has achieved so far (the fitness value is also stored). This value is called *pbest*. Another "best" value that is tracked by the particle swarm optimizer is the best value, obtained so far by any particle in the population. This best value is a global best and is called *gbest*. Besides, the particles move around inside a neighborhood set (defined in Subsection 4.2) in close proximity to the best particle, called the local best and denoted by *lbest*. $V_{\max}$ is an independent variable while *lbest*, *pbest* and *gbest* are dependent variables. Note that in a maximization optimization problem, "best" simply means the position with the largest objective value.

After finding the two best values (i.e., *pbest* and *lbest* if the neighborhood notion is applied or *pbest* and *gbest* otherwise) the particle updates its velocity and position according to the following equations (see [7]) :

$$V = C_1 V + A(pbest - present) + B(gbest - present) \tag{15}$$

$$present = present + V \tag{16}$$

where

V	is the particle velocity vector,
present	is the current particle (solution) vector,
pbest	is the memory of the particle's own best position vector (as defined above),
gbest	is the global best vector (as defined above).

lbest	is the local best, which is the best solution in the neighborhood of the particle.
rand()	is a random number between (0, 1),

A and B equal to $C_{\max} * rand()$ are two random matrices with each component generally equal to a uniform random number between 0 and 1 multiplied by $C_{\max}$.

The original intent was to use a different random component per dimension, rather than the same component for each dimension per particle.

1. C_1 is a learning factor that says how much the particle is directed towards good positions, usually $C_1 = 0.8$ or 0.7 [7, 23],
2. $C_{\max}$ is a learning factor which usually has a relationship with θ (defined as C_1 in the literature) through a transcendental function:

$$C_1 = \frac{1}{(\theta - 1 + \sqrt{\theta^2 - 2\theta})} \text{ and } C_{\max} = C_1 * \theta \tag{17}$$

   Optimal "confidence coefficients" are therefore approximately in the ratio scale of $C_1 = 0.7$ and $C_{\max} = 1.43$.

The particle moves in the search space according to both factors *pbest* and *gbest* (or *lbest* when using the neighborhood notion). A good choice of *pbest*, *lbest* and *gbest* guarantees the diversity within the population.

4.1.4 The number of particles

Another question faced at the implementation time is: "how many particles should we use?". There is no pat answer to this question, but the experience of some authors [7, 23] indicates some values which usually seem to work well when considering the performance of the method or the convergence time.

4.2 The neighborhood

The particles move in the search space in close proximity to the local best into the neighborhood set and do not explore the rest of the search space. So it is important to specify the neighborhood appropriately. Note that smaller neighborhoods lead to slower convergence while larger neighborhoods to faster convergence. With a global best, the representation of a neighborhood consists of the entire swarm. In our experimentation, we have tested three different versions of the neighborhood:

None neighborhood this version contains no neighborhood, i.e., the particle has no neighbor. So, the practical application of the algorithm involves using *gbest* instead of *lbest*.

Global neighborhood in this version, we consider that all the particles are neighbors of each other. Then, the number of neighbors is the number of particles minus one. Thus, in the algorithm, we use *gbest* instead of *lbest*.

Cosine neighborhood it consists in finding two particles forming an angle smaller than 30 degrees. Each particle has at most two neighbors. The measure of proximity between two vectors is the cosine of the angle between the two vectors. Note that in the case of spherical data, this cosine is also the correlation coefficient between the two linear combinations associated with the two projection vectors.

From our experience, the Cosine Neighborhood version is the most appropriate in the context of EPP because it gives several local optima (see Section 5).

4.3 The PSO algorithm

In this study, we decided to use the original version introduced by Kennedy and Eberhart [23] modified by Clerc [7].

4.3.1 Control's variables

- The dimension space of the problem is the number P of columns of the data matrix,
- The number of particles is defined according to the data set ,
- The number of neighbors: it changes according to the defined neighborhood,
- The number of iterations: max_iterations,
- The maximum velocity: $V_{\max} = (present_{\max} - present_{\min})/2$, where *present* is the current particle (solution) vector. We note $V_{\min} = -V_{\max}$,
- Initial fitness: projection index: $I = -\infty$,
- Best fitness: $best\_I = 0$;
- Individual and collective coefficients: $C_1 = 0.7$ and $C_{\max} = 1.43$,

In our case, the particle represents the vector of projection and is denoted by "projection". The projection is defined as a vector of P dimensions. This vector is normalized (see Section 2), which implies that each component of this vector lies between -1 and 1. Therefore, $present_{\max} = 1$ and $present_{\min} = -1$ which leads to $V_{\max} = 1$ (and of course $V_{\min} = -1$).

4.3.2 Principal algorithm

The initialization of the algorithm is presented in Algorithm 3 and the principal algorithm is detailed in Algorithm 4

Algorithm 3 Initialization of PSO algorithm

```
input: swarm: a vector of particles.
       getNeighbor: a function to get a particle's neighbor in the swarm.
for $j \leftarrow 1$ to $Nb\_Particles$ do
    $Projection \leftarrow swarm[j]$;
    /*Position and Velocity*/
    for $d \leftarrow 1$ to $Dimension\_Space$ do
        $Projection.NewPosition[d] \leftarrow random(0;1)$;
        $Projection.velocity[d] \leftarrow random(0;1)$;
    end
    $Projection.best\_I \leftarrow I$;
    /*Neighborhood*/
    for $k \leftarrow 1$ to $Nb\_Neighbors$ do
        $Projection.neighbor[k] \leftarrow getNeighbor(swarm, j, k)$;
    end
end
$swarm[j] \leftarrow Projection$;
```

Algorithm 4 The principal PSO algorithm

```
input: getBestNeighbor: a function to get a particle's best neighbor in the swarm.
while $i \leq max\_iteration$ do
    for $j \leftarrow 1$ to $Nb\_Particles$ do
        $Projection[j] \leftarrow swarm[j]$;
        for $d \leftarrow 1$ to $Dimension\_Space$ do
            $Projection.present[d] \leftarrow Projection.NewPosition[d]$;
        end
        $fitness \leftarrow I$;
        if $fitness > Projection.best\_I$ then
            $Projection.best\_I \leftarrow fitness$;
            for $d \leftarrow 1$ to $Dimension\_Space$ do
                $Projection.pbest[d] \leftarrow Projection.present[d]$;
            end
        end
    end
    for $j \leftarrow 1$ to $Nb\_Particles$ do
        $Projection \leftarrow swarm[j]$;
        $Projection.lbest \leftarrow getBestNeighbor(swarm; j; Nb\_Neighbors)$;
        for $d \leftarrow 1$ to $Dimension\_Space$ do
            $Projection.velocity[d] \leftarrow C_1 * V[d] + A[d][d](Projection.pbest[d] -$
            $Projection.present[d]) + B[d][d](Projection.lbest[d] - Projection.present[d])$;
            $Projection.newPosition[d] \leftarrow Projection.present[d] + Projection.velocity[d]$;
        end
    end
    $i \leftarrow i + 1$;
end
```

5 Applications

Let us consider the four projection pursuit indices defined in the second section. The present section illustrates on different data sets the use of the GA and PSO algorithms as described in Sections 3 and 4. The algorithms are implemented in the JAVA6 language.

5.1 The data sets

Let us analyze seven different data sets. Each one is represented by a matrix with N rows (observations) and P columns (variables).

Lubischew data It consists in $N = 74$ insects and $P = 6$ morphological measures [30] that are structured in three clusters. The first one (respectively the second and the third) contains observations 1 to 21 (respectively 22 to 43 and 44 to 74). This data file has already been studied in the context of EPP (see [5, 15]).

Simulated data These four data sets consist in $N = 200$ observations and $P = 8$ variables. The observations are distributed according to the following mixture:

$$\begin{cases} 200 \times (1-\varepsilon) \text{ observations follow a Normal distribution } \mathcal{N}_8(0_8, I_8) \\ 200 \times \varepsilon \text{ observations follow a normal distribution } \mathcal{N}_8((\mu, 0, ..., 0)^T, I_8) \end{cases} \quad (18)$$

Table 1 Description of the four simulated data sets

	don1	don2	don3	don4
ε	0.5	0.5	0.05	0.05
μ	10	4	10	4

where 0_8 denotes the vector of zeros in 8 dimensions, I_8 denotes the identity matrix in 8 dimensions and the values of ε and μ are defined in Table 1.

Each data set consists in two clusters. Clusters are balanced in the first and the second data sets while they are strongly unbalanced in the third and fourth data sets. Furthermore, the first and third data sets are made of two well-separated clusters while the second and fourth are made of overlapping clusters which are more difficult to distinguish.

Olive data The file consists in the percentage composition of $P = 8$ fatty acids found in the lipid fraction of $N = 572$ Italian olive oils. The 572 samples come from three different Italian regions (Southern Italy, Sardinia, Northern Italy) subdivided themselves into nine areas. This data set has been analyzed by several authors in the context of exploratory multivariate analysis (see [4, 9, 17]). The structure of the data set is quite complex with nine clusters which have different shapes in an eight-dimensional space. It has been previously processed by means of various supervised classification and clustering techniques. For example, Cook et al. [9] describe how to combine classifiers (support vector machines) with visual (tour) methods, Caussinus and Ruiz-Gazen [4] process the data by visualization/classification method without taking into account the origin of the oils. Due to the large number of classes, discovering all of them by using one-dimensional EPP is challenging but the results below are very promising. Several groups are detected by processing the data with different projection indices.

Reliability data The reliability data is a real data set from the industry under confidential agreement. It consists in 2,856 high technology chips and 166 variables. The purpose of the analysis is to detect multivariate outlying observations that may represent flawed chips.

5.2 Results

Let us first precise our choice for the neighborhood of PSO and for the size of the population and the number of iterations for GA and PSO methods. In Subsection 4.2, we proposed three neighborhood notions for PSO. After some experimentation that will not be detailed further in the present paper we conclude that the Cosine neighborhood version is the most adapted to our framework leading PSO to find several local optima. The None neighborhood and the Global neighborhood do not often lead to several local optima because they only take into account the global best particle. For small data sets like the Lubischew and the simulated data, the number of individuals for GA and particles for PSO does not need to be large. For larger data sets like the Olive and the reliability data, these values are increased. The number of iterations has been obtained by carrying out some preliminary runs on each data set and checking the convergence of the indices. Table 2 summarizes the values of both the size of the population and the number of iterations for each method GA

Table 2 Parameters of GA and PSO

Methods	Parameters	Small data sets: Lubischew & simulated	Large data set: olive	Large data set: reliability
GA	Individuals	50	100	200
	Iterations	20	50	50
PSO	Particles	20	50	50
	Iterations	50	100	200
	Fitness evaluations	1,000	5,000	10,000

and PSO. Note that the choice of the number of individuals/particles and iterations is such that GA and PSO lead to the same number of fitness evaluations which make the algorithms comparable. This number of evaluations is given in Table 2. We ran 100 times each optimization algorithm on the different data sets. We have to stress that the results would not be exactly the same for different values of the number of individuals/particles and iterations. But overall, when considering 100 runs (as we do), the method is quite robust in the sense that the structures detected in the seven data sets are the same for different values of these parameters.

In order to summarize the results in an efficient way, we draw the ranked values of the indices and the corresponding cosines of the angle between the projection vector associated with the best value of the index and the projection vector associated with the 100 other optima. In order to make the indices curves comparable we plot the opposite of the values of the minimum kurtosis and the discriminant indices since they are minimized and not maximized like the other indices. As illustrated below, such plots are particularly useful for statisticians in order to get insight into the potential complex structure of a data set. An interesting feature of these plots is that they are quite stable when we rerun the analysis which would not be the case if we reduce the number of runs. Note that the number of runs may be increased for larger or more complex data sets.

We also plot some histograms and kernel density estimators of the distributions of the projected data associated with local optima of the different indices in order to visualize possible structure(s). To save space, we do not display many projections for each of the five indices (Friedman–Tukey, Friedman, discriminant, minimum and maximum kurtosis index). We rather restrict ourselves to a selection of plots and give a summary of the obtained results for the different indices and algorithms in the text.

5.2.1 The Lubischew example

Figure 2 plots the ranked indices (left curves) and the associated cosines of the projection vector with the “best” projection vector (right curves) for the Friedman (top curves), the minimum kurtosis (middle curves) and the discriminant index (bottom curves) with GA. For the first two indices, the displays are quite similar. They reveal that for more than 70 runs over 100, the obtained projection vectors are the same (cosines equal to 1) but are different for the other 30 runs (cosines different from 1) which mean that there are at least two potentially interesting views of the data. For the discriminant index, the curves are quite different. The values of the index decrease much faster and when looking at the cosines curve, there is much more variability leading us to look at three or four projections. Note that the large dot on the curves corresponds to the maximum of the indices over the 100 runs while

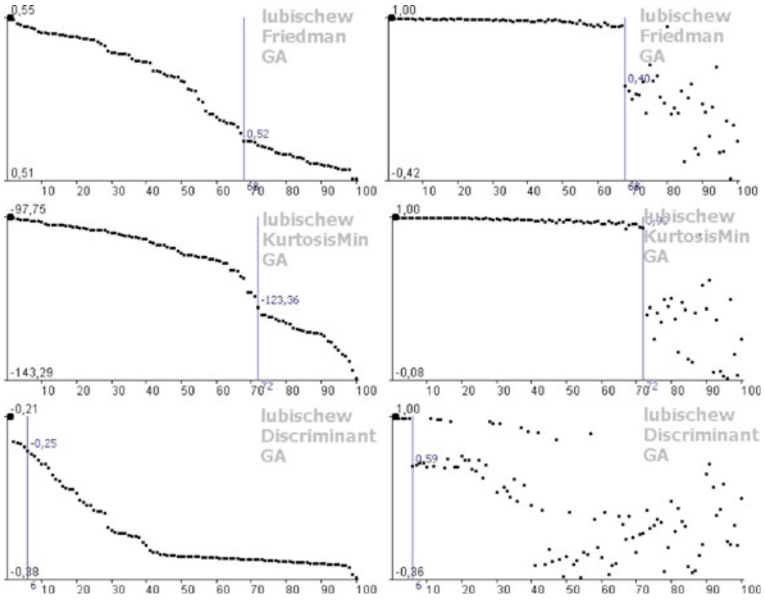

Fig. 2 Lubischew example: plots of the ranked indices (*left*) and the associated cosines of each projection vector with the “best” projection vector (*right*) for the Friedman (*top curves*), the minimum kurtosis (*middle curves*) and the discriminant index (*bottom curves*) with GA

the vertical line corresponds to the objective value of a second best projection that has been selected (run 68, resp. 72 and 6 for the Friedman, resp. minimum kurtosis and discriminant index).

The histograms of the distribution of the projected data corresponding to the maximum (resp. the second selected maximum) of the Friedman, minimum kurtosis and discriminant indices are displayed on the left (resp. right) of Fig. 3. The vertical bars below the histograms are helpful to understand the discovered structures since they give the groups the observations belong to. We recall that this information is not taken into account for the analysis but is used a posteriori to validate our results.

When looking at Fig. 3, we notice that the three indices do not lead to the identification of the same cluster(s). On the “best” projection (left histograms of Fig. 3), the Friedman index and the minimum kurtosis (resp. the discriminant) index detects the third (resp. the second) cluster as different from the other two clusters. On the second “best” projection (right histograms of Fig. 3), the minimum kurtosis and the discriminant index detect the third cluster while the Friedman index detects the second one.

No cluster structure is detected using the Friedman–Tukey index and the maximum kurtosis index. This result confirms the fact that these indices are more adequate to detect outliers than clusters.

From this figure, we conclude that clusters are identified with the Friedman, the minimum kurtosis and the discriminant indices. Moreover, this example suggests the use of the different indices in a complementary manner for larger or more complex data sets.

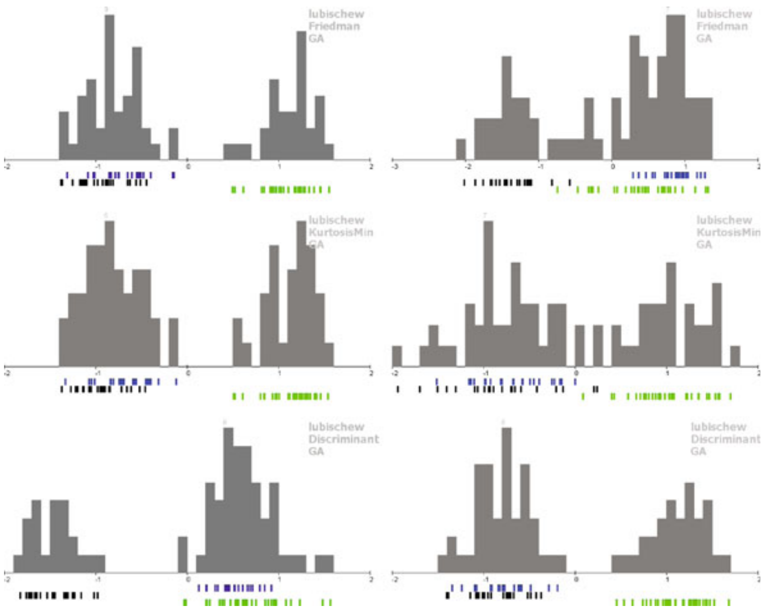

Fig. 3 Lubischew example: histograms of the distribution of the projected data on the "best" projection (*left*), and on a second "best" projection (*right*) for the Friedman (*top plots*), minimum kurtosis (*middle plots*) and discriminant indices (*bottom plots*) for GA

Results obtained with the PSO algorithm on this example are very similar to the ones presented for the GA and are omitted.

5.2.2 Simulated data

The Friedman index gives the best results for both GA and PSO : it detects the four types of clusters on the projection associated with the maximum value of the index for the 100 runs as displayed on the histograms of Fig. 4.

If we enter into details, we notice that the structure in two groups for the *don1* data set is detected by approximately 30 runs over 100 for both GA and PSO (see Fig. 4). The other projections are associated with smaller values of the indices and the histograms do not display a clear structure. For the *don3* data set, the structure is perfectly identified by the 100 runs for GA and PSO. Concerning the *don2* and the *don4* data sets, the structure is not so easy to detect and around ten runs only lead to the discovery of the structure for GA and PSO.

Concerning the other indices, the minimum kurtosis index detects the groups for the data sets *don1* and *don2* which correspond to balanced clusters but not for the *don3* and *don4* examples. On the contrary, the Friedman–Tukey and the maximum kurtosis indices give nice results when using the *don3* and *don4* examples which correspond to unbalanced clusters with 190 observations in one cluster and ten observations (that may be considered as outliers) in the other cluster. Finally, our

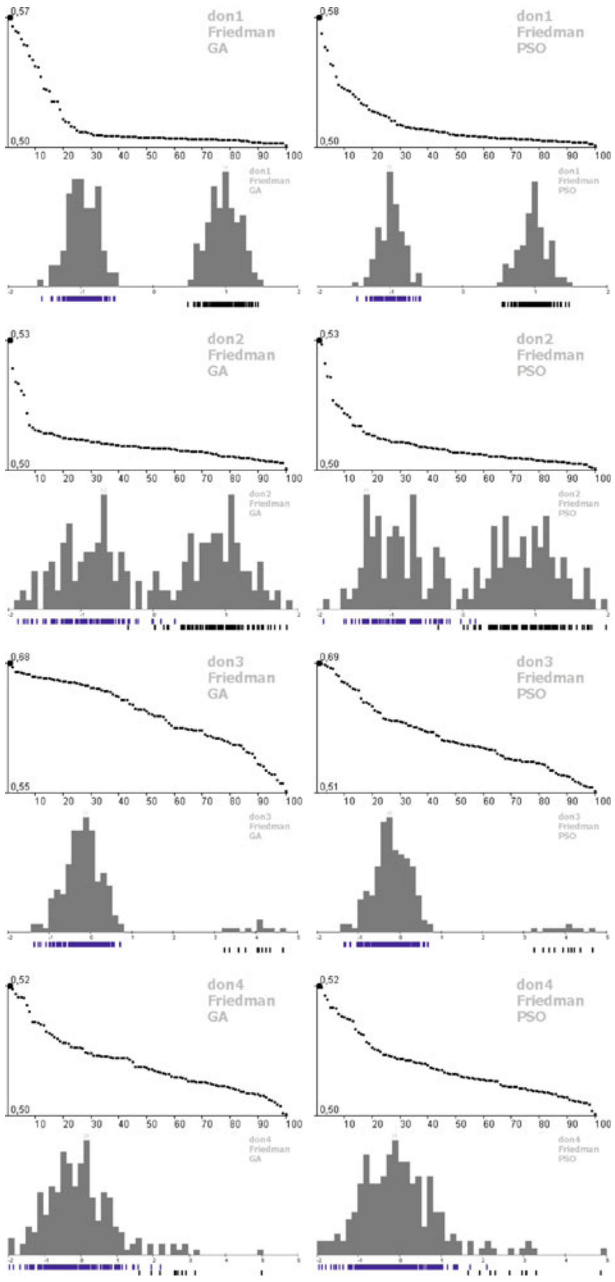

Fig. 4 Simulated data sets: plots of the Friedman index and histogram for the "best" projection for GA (*left*) and PSO (*right*) for each of the four data sets

results indicate that the discriminant index is only able to detect clusters for the third data set. This result is quite disappointing since this index has been defined in order to detect clusters and the reasons for such bad results deserve further research.

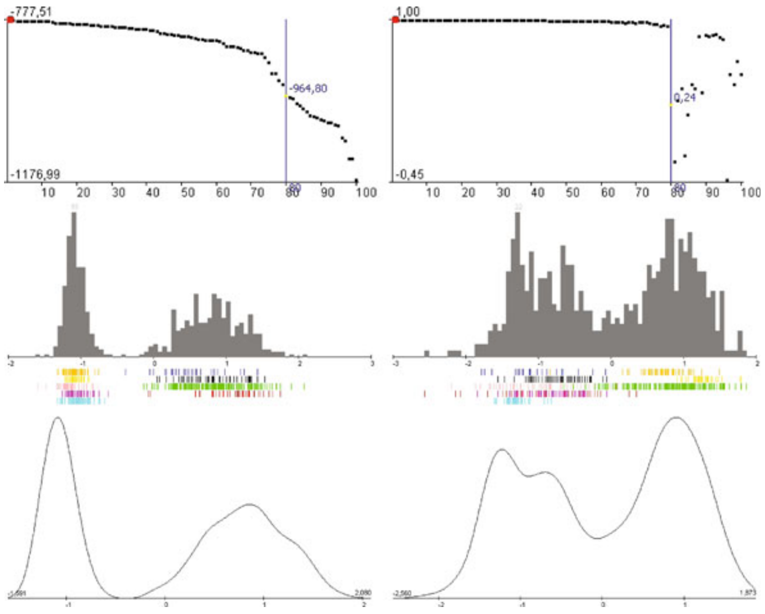

Fig. 5 Olive example: plots of the indices (*top left*) and the cosines (*top right*) for the minimum kurtosis index and PSO. Histograms (resp. kernel density estimator) for the "best" projection (*middle left*, resp. *bottom left*) and for a second "best" projection (80th run, *middle right*, resp. *bottom right*)

These data sets illustrate clearly that there are two families of projection indices: one family is dedicated to clusters detection (Friedman and the minimum kurtosis) while the second family is dedicated to outliers detection (Friedman–Tukey, the maximum kurtosis and Friedman to a less extent). Note that the results obtained by GA and PSO are very similar for the four data sets.

5.2.3 Olive data

The previous examples were small data sets structured in two or three groups. The olive oils data set is larger and more complex. The main goal of our analysis is to stress the importance of using several runs for different optimization methods and several projection indices when exploring large and complex data sets. We tested both stochastic methods (GA and PSO) using the five indices and we obtained interesting results.

On Fig. 5, the index and cosine plots lead us to look at at least two projections. We give (middle and bottom left plots) a projection obtained by PSO associated with the minimum of the minimum kurtosis (or the maximum of minus the minimum kurtosis) indices obtained from 100 runs. The plots lead clearly to the identification of two groups. To be more precise, when looking at the histogram on the middle left of the Fig. 5, the group on the left corresponds to olive oils from the southern region of Italy which contains five areas while the group on the right corresponds to olive oils from Sardinia and Northern Italy and contains four areas. When looking at the middle and bottom plots on the right which correspond to a second best projection,

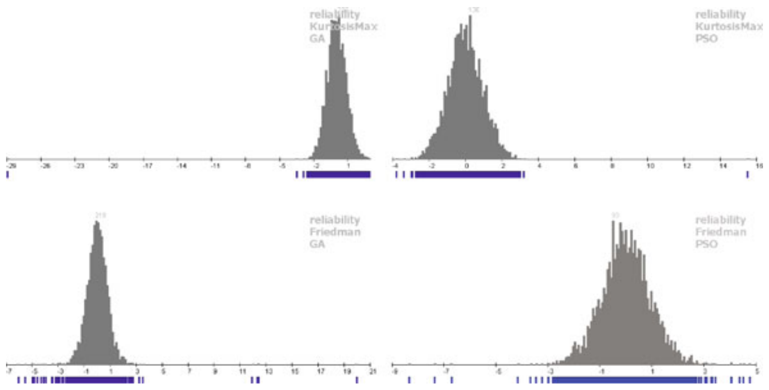

Fig. 6 Reliability example: projections obtained by the GA (*left*) and the PSO (*right*) method for the maximum kurtosis (*top*) and the Friedman (*bottom*) index

they lead us to detect another group structure. The same result is obtained using the GA method.

Note that it is also possible to identify other clusters by considering local optima of the Friedman index and other indices such as the Friedman–Tukey or the discriminant indices. There is no cluster identified using the maximum kurtosis index but some atypical oils are detected.

Another possibility from Fig. 5 is to split the data set in two groups and analyze each group separately in order to detect more groups.

This idea of partitioning the data set in two groups once they have been identified on a one-dimensional projection and iterate the procedure until no more group is detected, has been developed in Peña and Prieto [35] and can be applied automatically. It is an interesting procedure that could be used if the objective of the data analyst is clustering. In the present paper, we do not go further in that direction but it is clear that EPP is an important first step in order to detect whether clustering is adapted or not to a given data set. It could even be used in order to provide guidelines in the choice of the number of groups.

5.2.4 Reliability data

As explained previously, the data set comes from the industry and has high dimension ($P = 166$). The problem is the detection of outliers in a high dimension data set of chips. This problem is difficult and there is a significant literature on the subject in robust statistics (see for example [38]). A projection pursuit approach for anomalies detection has already been considered in the domain of hyperspectral imagery [1, 6, 31]. Because of the very high number of dimensions of the reliability data set, we also consider a PP approach. We focus on the maximum kurtosis index which is fast to compute and dedicated to the detection of outliers (see [35]) and on the Friedman index which is also fast to compute. This study is still a work in progress in partnership with the industry and we only give some projection plots. In particular, we do not provide details on the automatic procedure that will lead to the identification of a certain percentage of outliers, taking into account different indices, optimization methods and runs.

Figure 6 displays projections obtained by GA and PSO using the maximum kurtosis and the Friedman indices. A few chips are very far from the main bulk of the data when one looks at the histograms of the selected projections. These high technology chips may be considered as outlying and may be taken out of production.

6 Conclusion

In this paper, we use GA and PSO to optimize five projection pursuit indices. We validate the performance of the methods on data sets that are structured in clusters or contain outliers but other structures could also be considered. The main result of our study is that it is crucial to use several projection pursuit indices, several optimization methods and several runs in order to obtain a powerful EPP tool. This result is all the more important if the data set is large and contains a complex structure. By using several runs, we can have an insight into the number of local optima of the projection index that may be interesting to consider.

From our experience, it seems that PSO (with a cosine neighborhood) and GA are more or less equivalent in terms of finding several local optima. However, it happens sometimes that GA gives better results than PSO. Some improvements of the PSO method could be considered such that restart strategies for PSO particles or a new version of PSO like Tribes [7]. This point will deserve further attention. Note that both methods need several parameters to be tuned and in the present paper we propose some particular choices of parameters that give good results on all the data set considered so far.

Other evolutionary algorithms such as CMA-ES [19] or DE [41] could also be considered. CMA-ES is a global optimization method known to be very robust to local optima and DE is also a global optimization method. Such methods should be adapted in order to detect local optima and this point will deserve further investigation.

Concerning the indices results, the Friedman and the minimum kurtosis indices give the best results with both GA and PSO. The discriminant index works fine for the Lubischew and the third simulated data sets but not for the other simulated data sets and this point deserves more attention. As far as the running times are concerned, the minimum kurtosis is always the fastest index to compute while the discriminant and the Friedman–Tukey indices are the slowest ones. Moreover, when considering the same number of evaluations of the objective function, in our implementation, GA is generally faster than PSO. For small data sets such as the Lubischew or the simulated data, all the times are negligible. But for larger data sets like the olive data set, the Friedman–Tukey and the discriminant indices take quite a long time especially for PSO. The reliability data set is voluminous, so the running times are larger than for the previous sets. The Friedman–Tukey and the discriminant indices were not considered further since their computation times are prohibitive. Notice however that in a context of large data sets, EPP is suitable for parallelization by simply considering a run as a task and distributing each task to a processor of a parallel computer.

From this study, our perspective is now to propose an automatic software that allows the data analyst to be efficient in exploring multivariate data sets. Several indices together with several optimization methods and the possibility of choosing the number of runs will be offered. The software should include an automatic choice of parameters (number of individuals, particles, iterations) depending on the size of the data set. We plan also to propose guidelines to choose the appropriate indices according to the objective of the user but it will be highly recommended to use several of them.

Acknowledgements We thank Professor Christine Thomas for precious advices and Professor David Tyler for constructive discussions. We are also grateful to two anonymous referees who helped us to improve the presentation and the contents of the paper.

References

1. Achard, V., Landrevie, A., Fort, J.-C.: Anomalies detection in hyperspectral imagery using projection pursuit algorithm, image and signal processing for remote sensing X. In: Bruzzone, L. (ed.) Proceedings of the SPIE, vol. 5573, pp. 193–202 (2004)
2. Art, D., Gnanadesikan, R., Kettenring, J.R.: Data-based metrics for cluster analysis. Util. Math. **21A**, 75–99 (1982)
3. Caussinus, H., Ruiz-Gazen, A.: Metrics for finding typical structures by means of principal component analysis. In: Escoufier, Y. et al. (eds.) Data Science and its Applications, pp. 177–192. Academic (1995)
4. Caussinus, H., Ruiz-Gazen, A.: Classification and generalized principal component analysis. In: Brito et al. (eds.) Selected Contributions in Data Analysis and Classification, pp. 539–548. Springer (2007)
5. Caussinus, H., Ruiz-Gazen, A.: Exploratory projection pursuit. In: Govaert, G. (ed.) Data Analysis (Digital Signal and Image Processing series). ISTE (2009)
6. Chiang, S., Chang, C.: Unsupervised target detection in hyperspectral images using projection pursuit. IEEE Trans. Geosci. Remote Sens. **39**(7), 1380–1391 (2001)
7. Clerc, M.: Particle swarm optimization. International Scientific and Technical Encyclopaedia. Wiley, Hoboken (2006)
8. Cook, D., Buja, A., Cabrera, J.: Projection pursuit indices based on orthogonal function expansions. J. Comput. Graph. Stat. **2**(3), 225–250 (1993)
9. Cook, D., Caragea, D., Honavar, H.: Visualization in classification problems. In: Antoch, J. (ed.) Proceeding in Computational Statistics (COMPSTAT 2004), pp. 799–806. Springer, Berlin (2004)
10. Cook, D., Swayne, D.F.: Interactive and Dynamic Graphics for Data Analysis. Springer, New York (2007)
11. Crawford, S.L.: Genetic optimization for exploratory projection pursuit. In: Computer Science and Statistics: Proc. 23rd Symp. Interface (1992)
12. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer (2003)
13. Fogel, L.J., Owens, A.J., Walsh, M.J.: Artificial Intelligence Through Simulated Evolution. Wiley, New York (1966)
14. Friedman, J.H.: Exploratory projection pursuit. J. Am. Stat. Assoc. **82**(1), 249–266 (1987)
15. Friedman, J.H., Tukey, J.W.: A projection pursuit algorithm for exploratory data analysis. IEEE Trans. Comput. **C-23**, 881–889 (1974)
16. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning, 412 p. Addison Wesley Longman (1989)
17. Glover, D.M., Hopke, P.K.: Exploration of multivariate chemical data by projection pursuit. Chemometr. Intell. Lab. Syst. **16**, 45–59 (1992)
18. Guo, Q., Wu, W., Massart, D., Boucon, S., de Jong, S.: Sequential projection pursuit using genetic algorithms for data mining of analytical data. Anal. Chem. **72**(13), 2846–2855 (2000)
19. Hansen, N.: The CMA Evolution Strategy. Website http://www.lri.fr/~hansen/cmaesintro.html (1996)
20. Holland, J.H.: Adaptation in Natural and Artificial Systems. University of Michigan Press, Ann Harbor (1975)
21. Huber, P.J.: Projection pursuit. Ann. Stat. **13**(2), 435–475 (1985)
22. Jones, M.C., Sibson, R.: What is projection pursuit? (with discussion). J. R. Stat. Soc. A **150**, 1–36 (1987)
23. Kennedy, J., Eberhart, R.C.: Swarm Intelligence. Yuhui Shi (1995)

24. Klinke, S.: Data Structures in Computational Statistics. Physica-Verlag (1997)
25. Krink, T., Vesterstrom, J., Riget, J.: Particle swarm optimization with spatial particle extension. In: Proceedings of the Fourth Congress on Evolutionary Computation (CEC-2002), vol. 2, pp. 1474–1479. IEE, Piscataway (2002)
26. Kruskal, J.B.: Toward a practical method which helps uncover the structure of a set of multivariate observations by finding the linear transformation which optimizes a new index of condensation. Statistical Computing, pp. 427–440. Academic, New York (1969)
27. Lee, E., Klinke, S., Cook, D., Thomas, L.: Projection pursuit for exploratory supervised classification. J. Comput. Graph. Stat. **14**(4), 831–846 (2005)
28. Li, X.: A multimodal particle swarm optimizer based on fitness euclidean-distance ratio. In: Thierens, D. (ed.) Proceeding of Genetic and Evolutionary Computation Conference 2007 (GECCO'07), pp. 78–85. ACM (2007)
29. Lovbjerg, M., Krink, T.: Extending particle swarms with self-organized criticality. In: Proceedings of the Fourth Congress on Evolutionary Computation (CEC-2002), vol. 2, pp. 1588–1593. IEE, Piscataway (2002)
30. Lubischew, A.A.: On the use of discriminant functions in Taxonomy. Biometrics **18**, 455–477 (1962)
31. Malpica, J.A., Rejas, J.G., Alonso, M.C.: A projection pursuit algorithms for anomaly detection in hyperspectral imagery. Pattern Recogn. **41**, 3313–3327 (2008)
32. Martinez, W., Martinez, A.: Computational Statistics Handbook with Matlab. CRC (Taylor and Francis Group) (2001)
33. Nason, G.P.: Design and choice of projection indices. Ph.D. dissertation, University of Bath (1992)
34. Nason, G.P.: Three-dimensional projection pursuit. J. R. Stat. Soc. C **44**, 411–430 (1995)
35. Peña, D., Prieto, F.: Cluster identification using projections. J. Am. Stat. Assoc. **96**(456), 1433–1445 (2001)
36. Posse, C.: Tools for two-dimensional exploratory projection pursuit. J. Comput. Graph. Stat. **4**(2), 83–100 (1995)
37. Rechenberg, R.I.: Evolutionsstrategie: Optimierung Technischer Systeme Nach Prinzipien Der Biologischen Evolution. Frommann-Holzboog, Stuttgart (1973)
38. Riani, M., Atkinson, A.C., Cerioli, A.: Finding an unknown number of multivariate outliers. J. R. Stat. Soc. B **71**(2), 447–466 (2009)
39. Schwefel, H.P.: Numerical Optimization of Computer Models. Wiley, New York (1981)
40. Silverman, B.W.: Density Estimation for Statistics and Data Analysis. Chapman and Hall, New York (1986)
41. Storn, R., Price, K.: Differential evolution a simple and efficient heuristic for global optimization over continuous space. J. Glob. Optim. **11**(4), 341–359 (1997)
42. Sun, J.: Some practical aspects of exploratory projection pursuit. SIAM J. Sci. Comput. **14**(1), 68–80 (1993)