ON THE CROSSOVER OPERATOR FOR GA-BASED OPTIMIZERS IN SEQUENTIAL PROJECTION PURSUIT

Soledad Espezua, Edwin Villanueva and Carlos D. Maciel
Sao Carlos School of Engineering, University of Sao Paulo, Sao Carlos, Brazil

Keywords: Projection pursuit, Sequential projection pursuit, Genetic algorithms, Crossover operators.

Abstract: Sequential Projection Pursuit (SPP) is a useful tool to uncover structures hidden in high-dimensional data by constructing sequentially the basis of a low-dimensional projection space where the structure is exposed. Genetic algorithms (GAs) are promising finders of optimal basis for SPP, but their performance is determined by the choice of the crossover operator. It is unknown until now which operator is more suitable for SPP. In this paper we compare, over four public datasets, the performance of eight crossover operators: three available in literature (arithmetic, single-point and multi-point) and five new proposed here (two hyperconic, two fitness-biased and one extension of arithmetic crossover). The proposed hyperconic operators and the multi-point operator showed the best performance, finding high-fitness projections. However, it was noted that the final selection is dependent on the dataset dimension and the timeframe allowed to get the answer. Some guidelines to select the most appropriate operator for each situation are presented.

1 INTRODUCTION

The collection of data sets with large amounts of measured features is becoming increasingly common in many industrial and science areas. This makes dimension reduction an active research topic in data mining, machine learning and statistics. Projection pursuit (PP) (Friedman and Tukey, 1974; Friedman, 1987) is a framework of methods proposed to deal with such high-dimensional data sets that has become very popular in the statistical literature. PP faces the curse of dimensionality by searching for "interesting" low-dimensional projections of the data, where the interestingness of the projections is assessed by a pre-defined function, known as *projection pursuit index* (PP index). Thus, the most interesting projection spaces are found by optimizing such function.

A wide range of data-mining problems can be tackled with PP (Daszykowski et al., 2007), depending on the PP index used. For instance, PP can perform the well known principal components analysis - PCA, with the variance being the PP index (Daszykowski et al., 2007). Several PP indices have been proposed in the literature for different applications (some reviews can be found in (Berro et al., 2010; Rodriguez-Martinez et al., 2010; Jee, 2009; Lee et al., 2005)). Indeed, there are PP indices suitable for clustering analysis (Bolton and Krzanowski, 2003; Pena and Prieto, 2001), classification (Demirci et al., 2008; Lee et al., 2005), feature selection (Guo et al., 2001) and regression analysis (Ren et al., 2007).

One of the major problem in PP lies in the optimization of the PP indices (Berro et al., 2010). Most optimization methods were developed in the context of exploratory projection pursuit - EPP, where the target space is of one, two or three dimensions (Bolton and Krzanowski, 2003; Nason, 1995; Posse, 1995a; Posse, 1995b; Posse, 1990; Friedman, 1987). Gradient-based methods were the first to be used (Jones and Sibson, 1987; Huber, 1985). However, it was noted that such methods can easily get stuck in local optima close to the starting point, capturing noise-induced structures (Friedman, 1987). This led some researches to develop more global optimization methods, such as (Pena and Prieto, 2001; Posse, 1995a) and (Friedman, 1987). Although their proposals are appropriate for EPP, they have difficulties to scale beyond three-dimensional projections, mainly due to computational constraints in computing the indices in such spaces. This difficulty was circumvented with the emergence of the sequential projection pursuit (SPP) (Guo et al., 2000), an algorithm that can perform projections onto higher-than-three dimensional spaces in a simple way. SPP opened new possibilities for PP, such as feature selection (Guo et al., 2001) and feature extraction for machine learning (Rodriguez-

Espezua S., Villanueva E. and D. Maciel C. (2012).
ON THE CROSSOVER OPERATOR FOR GA-BASED OPTIMIZERS IN SEQUENTIAL PROJECTION PURSUIT.
In *Proceedings of the 1st International Conference on Pattern Recognition Applications and Methods*, pages 93-102
DOI: 10.5220/0003784400930102

Martinez et al., 2010). In SPP, the bases of the projection space (PP factors) are sought one after the other instead of all together. Each PP factor is obtained by optimizing one-dimensional PP index over residual data (data resulting from removing the structure found in the previous PP factors). A binary genetic algorithm (GA) was presented as the index optimizer of SPP. The use of a GA for optimizing PP indices was also proposed by (Chiang et al., 2001) in the so-called projection pursuit evolutionary algorithm (PPEA). (Webb-Robertson et al., 2005) argued later that the GA optimizer in SPP is slow in obtaining the PP factors and proposed an alternative to it: the random scan sampling algorithm (RSSA). Some other global methods for optimization of PP indices were proposed in recent years, such as simulated annealing (SA) (Lee et al., 2005) and particle swarm optimization (PSO) (Berro et al., 2010).

Based in the success of GA in many problems, we believe that their potential was underexplored as optimizers for SPP. Some desirable features of GAs are attractive for SPP, such as (Srinivas and Patnaik, 1994): 1) GAs can search the solution space (that is known to be multimodal) in a parallel and multidirectional way, giving more chance to find highly informative projections; 2) at any time we can take a solution, which get better with time; 3) one can control the diversity of the population, which can be useful for EPP since many alternative solutions are required for inspection; and 4) GAs can be straightforwardly implemented in parallel and distributed platforms, expanding the applicability of PP to problems with huge data sets. Despite these desirable features, the performance of a GA is determined by the set of genetic operators and parameters used, whose determination is not an easy task and heavily dependent on the problem addressed. In particular, the choice of the crossover operator is of key importance in the success of a GA, since it is the primary search mechanism that a GA relies, responsible for the rapid exchange of useful information among solutions to locate better solutions (Srinivas and Patnaik, 1994). Despite the acceptable results reported with the crossover operators used in the original SPP and PPEA, the selection of them was arbitrary. Nevertheless, it is unknown if some other operators can perform better in the optimization of PP factors for SPP.

Moving in this direction, we present in this paper a comparative experimental study of eight crossover operators: three currently used in SPP and PPEA (arithmetic crossover and single and multipoint crossover) and five new proposed here (one single extension of the arithmetic crossover, two hyperconic crossovers and two fitness-biased crossovers). The performance of the crossover operator in a prototypical GA is assessed by measuring the mean fitness of the population at different stages of evolution and the converged fitness and number of generations needed to converge. The study was carried out over four public datasets of increasing dimension ranging from 13 to 166 variables. Also, the influence of the evolutionary pressure in the performance of the different operators is analyzed. The results showed that the proposed hyperconic crossover operators tend to find projections with fitness values higher than the other operators, with one of such operators clearly excelling in higher dimensions. The multi-point crossover was also competitive in low dimensions. The final selection of the crossover operator is dependent on the dataset dimension and the tolerable time to get the answer. Some guidelines to aid in the selection of the most suitable crossover operator for SPP are presented.

The paper is organized as follows. Section 2 introduces some important concepts of PP, SPP, and GAs. Section 3 describes the crossover operators studied. Section 4 presents the experimental setup. The results and discussions are presented in Sections 5 and 6. Finally, our conclusions are presented in Section 7.

2 BACKGROUND

2.1 Projection Pursuit

The projection pursuit concept was formally introduced in the paper of (Friedman and Tukey, 1974), although the seminal ideas were originally posed by (Kruskal, 1969). To describe the idea of PP we assume that the data set is arranged in a $n \times p$ matrix $\mathbf{X}$ with n instances and p attributes or variables. PP seeks a m-dimensional projection space ($m < p$), defined by the orthonormal bases $\mathbf{A} \in \mathbb{R}^{p \times m}$, where the projected data $\mathbf{X} \cdot \mathbf{A}$ expose information of interest. The degree of interestingness of the projection is measured by the function $\mathfrak{I}$, called the *projection pursuit index* (PP index). Thus, PP can be formulated as the optimization problem:

$$\begin{aligned} \mathbf{A}^* &= \arg\max_{\mathbf{A}} \{\mathfrak{I}(\mathbf{X} \cdot \mathbf{A})\} \qquad (1) \\ s.t \quad & \mathbf{A}^T \cdot \mathbf{A} = \mathbf{I}. \end{aligned}$$

The choice of the PP index is a key consideration. A great deal of research has been centered on the construction of a globally useful and robust index, but the effectiveness of an index is often dependent on the application and characteristics of the given

dataset (Rodriguez-Martinez et al., 2010). One dominant consideration in the developing of PP indices has been the so-called *affine invariance* (Jee, 2009). A PP index $\mathfrak{I}$ is said affine invariant if $\mathfrak{I}(\mathbf{X}) = \mathfrak{I}(s\mathbf{X}+\mathbf{v})$ for a nonsingular linear transformation s and a constant vector $\mathbf{v}$. Thus, affine invariance ensures that changes in scale and location of the projected data do not affect the index value. A common approach to insure affine invariance is *sphering* the original data matrix $\mathbf{X}$ to have zero mean and identity covariance matrix. This can be done by the following transformation (Posse, 1995a):

$$\mathbf{Z} = \Lambda^{-1/2}\mathbf{Q}(\mathbf{X} - E[\mathbf{X}]) \tag{2}$$

where $\mathbf{Q}$ and Λ are respectively the eigenvector and eigenvalue matrices resulting from the eigen-decomposition of the covariance matrix $\Sigma = \mathbf{Q}\Lambda\mathbf{Q}^T$. For simplicity, in the rest of the paper we will refer to the original data matrix $\mathbf{X}$ as the sphered version of it.

In clustering applications, *entropy* is commonly used as the PP index of the projected data(Jones and Sibson, 1987). However, the entropy calculation is computationally intensive, requiring high-order integrals and a density estimator. A simpler and robust alternative to entropy is the *Holes* PP index (Cook et al., 1993), which returns comparable (and often better) results than entropy (Webb-Robertson et al., 2005). The Holes index is defined for one-dimensional projection as:

$$\mathfrak{I}_{Holes} = 1 - \frac{1}{n}\sum_{i=1}^{n} e^{-\frac{1}{2}y_i^2} \tag{3}$$

where y_i is the projection of the ith data instance $\mathbf{x}_i$ onto the direction of the basis vector $\mathbf{a}$, $y_i = \mathbf{x}_i \cdot \mathbf{a}$. We adopt Holes as the PP index to be optimized in all experiments presented in this paper.

Sequential projection pursuit (SPP) (Guo et al., 2000) tackles the m-dimensional constrained optimization problem in Equation 1 by converting it into a sequence of m one-dimensional optimization problems. The first basis (PP factor) $\mathbf{a}_1$ in $\mathbf{A}$ is obtained by searching (with a GA) a p-dimensional vector of unit length that maximizes the PP index. Once the first PP factor $\mathbf{a}_1$ is found, the data set is projected onto it, obtaining the score vector $\mathbf{y}_1 = \mathbf{X} \cdot \mathbf{a}_1$. The residual data is then computed as $\mathbf{X} = \mathbf{X} - \mathbf{y}_1 \cdot \mathbf{a}_1^T$. The process is then repeated on this residual data to obtain $\mathbf{a}_2$, $\mathbf{y}_2$ an a new residual data, subject to the constraint that $\mathbf{a}_2$ is orthogonal to $\mathbf{a}_1$. In this way, the predefined m PP factors are obtained in SPP.

2.2 Genetic Algorithms

Genetic Algorithms (GAs) (Holland, 1975) belong to a class of algorithms inspired in Darwinian evolutionary theory. GAs start from a population of individuals (potential solutions) normally generated at random. Each individual is then assigned a fitness value by means of a fitness function (which encodes the problem objective function). The algorithm enters to an evolutionary loop, where each iteration (generation) produces a new population for the next generation. In each generation, multiple individuals are stochastically selected from the current population (based on their fitness) and recombined by a crossover operator (and possibly randomly mutated by a mutation operator) to form an offspring population which is inserted to the current population by a replacement operator. The GA used in this paper for the evaluation of the crossover operators encodes the individuals as real-value unit-length vectors. This representation is used instead of the binary representation to avoid loss of precision. Thus, an individual i is a candidate basis vector $\mathbf{a}_i = [a_{i1}, a_{i2}, \ldots, a_{ip}]^T$ to project the data $\mathbf{X} \cdot \mathbf{a}_i$. The fitness function is the Holes PP index. Several crossover operators are tested (Section 3), those that work over binary strings (single and multi-point crossover) are naturally extended to our real representation, maintaining the coefficients of the vectors as unbreakable units (genes). The selection operator is implemented as tournament selection: we randomly choose ts individuals (the tournament size) from the current population (of size w) and return the best individual. With this selection method, $w/2$ pairs of different individuals are selected to form the mating pool for crossover. For each pair of this mating pool a uniform random number is generated in the unit interval, if such number is less than a predefined number cr (crossover rate parameter) the crossover is performed and two new offspring are generated, otherwise the offspring are clones of the parents. Mutation is no used in the present study in order to no obscure the influence of the studied crossover operators. Replacement is carried out by selecting the best w individuals of the joint population of parents and offspring.

3 CROSSOVER OPERATORS

Eight crossover operators are presented here, three commonly used in the literature: arithmetic, single and multi-point crossover and five new proposed for this paper: one extension of the arithmetic crossover, two hyperconic crossovers and two fitness-biased crossovers. All operators require two parent individuals as input and return two offspring as output. Parent individuals are denoted by $\mathbf{a1} = [a1_1, \ldots, a1_p]^T$ and $\mathbf{a2} = [a2_1, \ldots, a2_p]^T$ and the offspring individuals by $\mathbf{b1} = [b1_1, \ldots, b1_p]^T$ and $\mathbf{b2} = [b2_1, \ldots, b2_p]^T$. The

function $Normalize(\mathbf{x})$ produces a unit vector in the direction of $\mathbf{x}$, thus, $Normalize(\mathbf{x}) = \mathbf{x}/\sqrt{\mathbf{x}^T \cdot \mathbf{x}}$. This operation is performed in all crossover operator in order to maintain constant the scale of the projection.

Crossover1 (Arithmetic Inner Crossover). This operator was adopted in PPEA (Chiang et al., 2001). The offspring are produced as follow:

$$\begin{aligned} \mathbf{b1} &= Normalize(r\mathbf{a1} + (1-r)\mathbf{a2}) \\ \mathbf{b2} &= Normalize((1-r)\mathbf{a1} + r\mathbf{a2}) \end{aligned} \quad (4)$$

where r is a uniform random number from $[0, 0.5]$. The offspring vectors are contained in the plane of the parents and located symmetrically in both sides of the bisectrix of the parents and limited by them.

Crossover 2 (Arithmetic Inner-outer Crossover). This operator is proposed here as an extension of Crossover1. The offspring are produced in the same way as in Equation 4. The only difference is in the random number r, which is generated uniformly in the interval $[-0.5, 0.5]$. Thus the offspring vectors are contained in the plane of the parents and located symmetrically in both sides of the bisectrix of the parents. By contrast with Crossover1, the offspring can be outside of the region delimited by the parents forming a maximum angle with respect to the bisectrix of up to 2θ, being θ the angle between any parent and the bisectrix.

Crossover3 (Single-point Crossover). This is a traditional operator in the GA that has been formulated to recombine binary strings. We implement here a real version of this operator. First, a random position r is chosen uniformly from the set $\{1, 2, \ldots, p-1\}$. Next, the offspring are created as follow:

$$\begin{aligned} \mathbf{b1} &= Normalize([a1_1, \ldots, a1_r, a2_{r+1}, \ldots, a2_p]^T) \\ \mathbf{b2} &= Normalize([a2_1, \ldots, a2_r, a1_{r+1}, \ldots, a1_p]^T) \end{aligned} \quad (5)$$

Crossover4 (Multi-point Crossover). This operator can be viewed as an extension of Crossover3. First, a random binary vectors (mask) of length p is created with similar numbers of 0s and 1s. Next, the offspring **b1** is created by placing the corresponding values of **a1** in the positions where the mask is 1 and the values of **a2** in the other positions. **b2** is created in a complementary way, i.e., **b2** has the values of **a2** in the positions where the mask is 1 and the values of **a1** in other positions.

Crossover5 (Inner Hypercone Crossover). We propose this operator to allow the offspring to be outside the plane of the parents. The offspring can be located inside the hypercone resulting from rotating the parents around their bisectrix and symmetric to it. First, the bisectrix is calculated as $\mathbf{m} = Normalize(\mathbf{a1} + \mathbf{a2})$. Next, a plane of crossover is selected containing such bisectrix. Therefore, one of the basis of that plane is the same bisectrix. The other base $\mathbf{v}$ is a vector orthogonal to it that can be constructed by randomly choosing one non-zero component of $\mathbf{m}$, m_j, and one position $l \neq j$; with this, the orthogonal vector is composed as: $v_i = 0$ $(i \neq j, l)$, $v_l = 1$, and $v_j = -m_l/m_j$. The basis of the plane of crossover are then $Q = [\mathbf{m}, \mathbf{v}]$. In this plane, the bisectrix is over the abscissa and the parents are located at angles θ and $-\theta$. Offspring $\mathbf{b1}^Q$ is chosen in polar coordinates in this plane as: e^{θ_r}, where θ_r is uniformly selected from $[0, \theta]$. Offspring $\mathbf{b2}^Q$ is symmetric to offspring $\mathbf{b1}^Q$, thus $\mathbf{b2}^Q = e^{-\theta_r}$. The offspring in the original space are recovered with the plane basis as: $\mathbf{b}i = Q \cdot \mathbf{b}i^Q$, $i = 1, 2$. In this crossover scheme there are up to $p(p-1)$ different planes where the crossover can take place, all intersecting at the parent bisectrix.

Crossover6 (Inner-outer Hypercone Crossover). This operator is proposed here as an extension of Crossover5. Like such operator, the offspring can be in one of the $p(p-1)$ possible crossover planes symmetric to the bisectrix of the parents. By contrast with Crossover5, the offspring are allowed to be outside the hypercone generated by rotating the parents around their bisectrix. The procedure to produce the offspring is similar to Crossover 5. The only difference is the interval where θ_r is sampled, which is $[0, 2\theta]$. Consequently, the offspring can born in a wider hypercone than that of Crossover5, containing this later.

Crossover7 (Fitness-biased Inner Crossover). We propose this operator to guide the creation of the offspring by the fitness of the parents. The crossover takes place in the plane of the parents, as Crossover 1 and Crossover 2. By contrast with them, the offspring are no more symmetric to the bisectrix of the parents. The basis vector in the plane are chosen so that the parent with more fitness forms a negative angle with respect to the abscissa. Let denote $\mathbf{a}_{best}$ the parent with best fitness and $\mathbf{a}_{worst}$ the other parent. The basis vectors in the plane are $Q = [\mathbf{m}, \mathbf{v}]$, where $\mathbf{m}$ is the bisectrix and $\mathbf{v}$ is computed as $Normalize(\mathbf{a}_{worst} - (\mathbf{a}_{worst}^T \cdot \mathbf{m})\mathbf{m})$. In this plane, the bisectrix is over the abscissa and the parents are located at angles $-\theta$ (the best parent) and θ (the worst parent). A uniform random angle θ_r is sampled from the interval $[-\theta, \theta]$. Then, this angle is distorted (in favor to $-\theta$) with the

following function:

$$\theta_r^{(biased)} = 2\theta((\theta_r + \theta)/2\theta)^{\beta} - \theta \qquad (6)$$

where $\beta > 1$ controls the amount of distortion. In our experiments we use $\beta = (Fitness(\mathbf{a}_{best})/Fitness(\mathbf{a}_{worst}))^4$. The function in Equation 6 produces an angle in the same interval $[-\theta, \theta]$. With $\theta_r^{(biased)}$ computed, the first offspring **b1** is constructed as explained in Crossover 5. The offspring **b2** is produced similarly choosing another θ_r.

Crossover8 (Fitness-biased Inner-outer Crossover). This operator is proposed here as an extension of Crossover7 (like Crossover2 extends Crossover1). The procedure is similar to Crossover7. The only difference is in the interval from which θ_r (and therefore $\theta_r^{(biased)}$) is sampled. This interval is $[-2\theta, 2\theta]$. Thus the offspring vectors are contained in the plane of the parents, with the tendency to fall closer to the best parent, both on its left and right sides.

4 EXPERIMENTAL STUDY

4.1 Datasets

Four public datasets were selected for our study from the UCI repository (Frank and Asuncion, 2010). The selection was performed trying to have a collection of datasets with increasing number of features (almost geometric increase) to analyze the influence of the increasing dimension over the crossover operators.

Wine Dataset. This dataset is known as the Wine recognition data. The data are the results of a chemical analysis of samples of three types of wines in Italy. The dataset consist of 178 instances and 13 continuous features (constituents found for each wine).

WDBC Dataset. This dataset is known as the Wisconsin Diagnostic Breast Cancer. Each data sample describe characteristics of the cell nuclei present in a digitized image of fine needle aspirate of breast mass. The dataset consist of 569 instances and 30 real-valued features that describe characteristics of the cell nuclei present in the image.

Sonar Dataset. This dataset content features extracted from bouncing of sonar signals collected from metal cylinders and cylindrically-shaped rocks positioned on a sandy ocean floor. The dataset consist of 208 instances and 60 real-valued attributes. Each attribute represents the energy within a particular frequency band integrated over a period of time.

Musk Dataset. In this dataset are stored features that represent different conformations of molecules that are judged by human experts. The dataset consist of 476 instances and 166 attributes labeled according to whether or not they contain a musk odor.

4.2 Evaluation

The performance of the different crossover operators are evaluated in the prototypical GA described in Subsection 2.2. Three hundred replicates are performed for each dataset to obtain consistent statistics. Figure 1 depicts a replicate, which consists in: 1) generate an initial random population by choosing $w = 10p$ random vectors from $\mathbb{R}^p$ and normalizing them to be unit length vectors; 2) starting from this initial population, evolve two instance of the GA for each crossover operator: one for $ts = 1\%$ of the population and one for $ts = 5\%$ of the population (in total 16 GA instances are evolved per replicate); 3) record the following statistics for every GA instance: the mean fitness of the evolving population at 20 generations, the mean fitness of the evolving population at 200 generations, and the mean fitness and number of generations when the population reaches convergence. We consider that a population has converged when the difference between the maximum and mean fitness of the population is less than $1e^{-7}$. In some cases convergence is achieved before 200 generations; there, the GA is kept running to reach 200 generations. Otherwise, the algorithm is stopped as soon as convergence is reached. If convergence is not achieved in a maximum of 5000 generations, the GA instance is stopped anyway. In all cases the crossover rate is set to 1 and the mutation is disabled. This implies that the couples in the mating pool always produce their offspring by crossing over. The recorded statistics are averaged over the 300 replicates in each dataset.

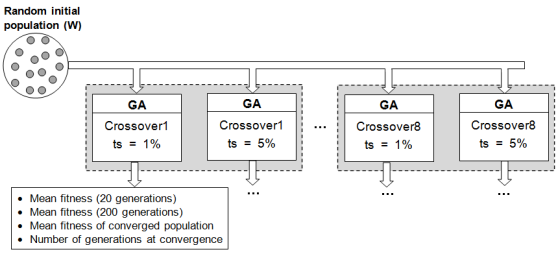
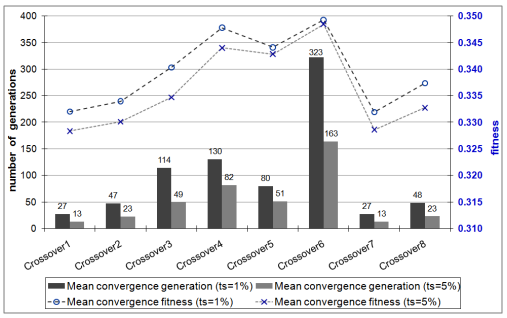
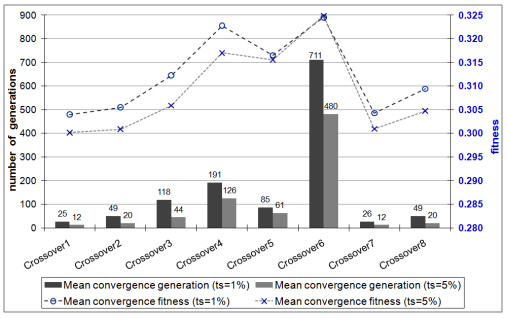


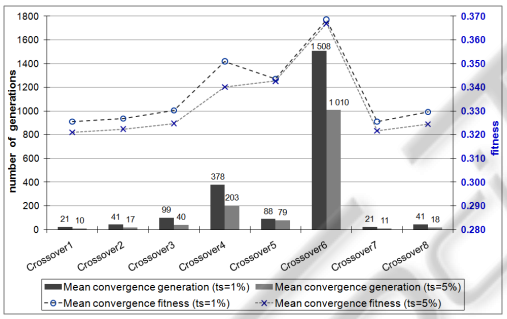
Figure 1: Example of replicate executed to get statistics of the performance of the crossover operators.



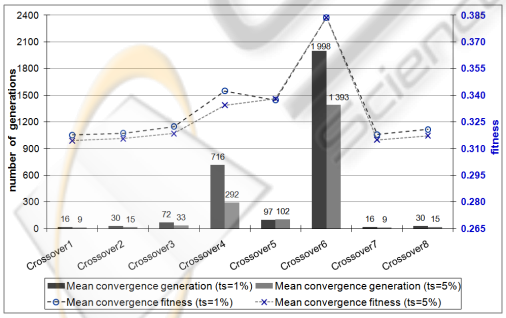
(a) Wine dataset.



(b) WDBC dataset.



(c) Sonar dataset.



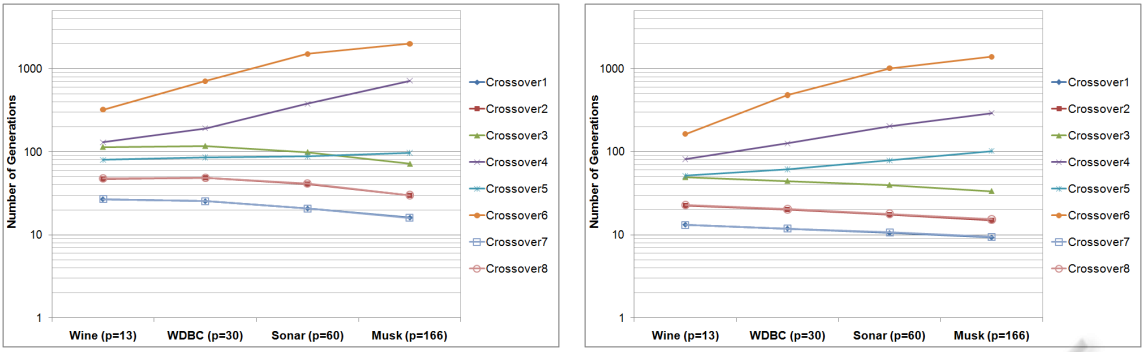
(d) Musk dataset.

Figure 2: Results for the mean fitness (curves) and number of generations (bars) at convergence (averaged over 300 replicates). The left scale is for the number of generations and the right scale is for the fitness values (Holes PP index).

With this experimental setup we try to measure the contribution of the crossover operator on the performance of the GA. For a same replicate and tournament size, all GA instances start from the same population, so the only influential factor on the GA performance is the crossover operator. One of the performance measures is the mean fitness of the population at different stages of evolution. This will tell us which operators would be more appropriate if we have a time frame to get the answer. The other performance measures are the maximum achievable fitness and the number of generations needed to reach it. With these measures, we are enabled to infer the exploratory capabilities of the operators and their efficiencies. Also important, the analysis of the crossover operators at different selective pressure levels (defined by the tournament size) is useful, since the behavior of the crossover operator can be affected by this factor.

5 RESULTS

Figure 2 shows the results of the average values of fitness and number of generations at convergence for the different crossover operators and tournament sizes analyzed. Each subfigure corresponds to a different dataset. With respect to the number of generations needed to converge, a similar pattern can be seen for all datasets and tournament sizes: Crossover6 has the highest value and clearly differentiated from the others; Crossover4 has the second highest values; Crossover1 and Crossover7 have almost identical values, which are the lowest values of all operators; Crossover2 and Crossover8 have also matching values, which are the second lowest values; Crossover3 and Crossover5 have mixed values, with Crossover3 surpassing Crossover5 only when $ts = 1\%$ in all but Musk dataset. These results can be also visualized in Figure 3, which shows the mean number of generations of convergence as a function of the dataset dimension for the two tournament sizes. It can be seen two groups of operators with different trend with respect to the dimension, independent of the tournament size. The first group, integrated by Crossover1, Crossover2, Crossover3, Crossover7 and Crossover8, shows a decrease in the number of generations for convergence as the dimensionality is increased. The second group, formed by Crossover4, Crossover5 and Crossover6, shows an increase in the number of generations for convergence as the dataset dimension is increased, with Crossover6 having the fastest increase. When the tournament size is varied from $ts = 1\%$ to $ts = 5\%$, the number of generations for convergence decreases for all crossovers operators and datasets (also observed in Figure 2), but the general



(a) ts = 1%. (b) ts = 5%.

Figure 3: Plot of the number of generations needed for convergence (averaged over 300 replicates) with respect to the dataset (ordered according to its dimension) for two selective pressure. The scale is logarithmic to visualize differences.

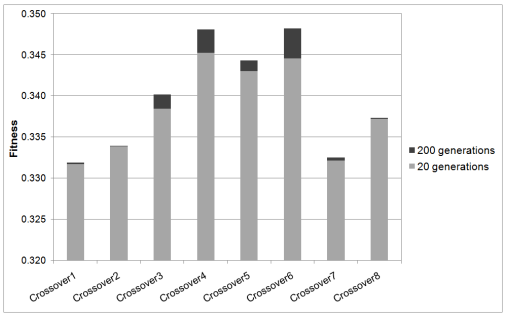
behavior of the number of generations vs. dimensionality is unchanged.

With respect to the fitness of the converged solutions, we can observe in Figure 2 that the fitness values achieved with the different crossover operators in all datasets are negatively affected by the increase of the tournament size from 1% to 5%. Crossover5 and Crossover6 are the least affected operators to this change, with Crossover6 being almost insensitive. In addition, Crossover6 converged to solutions with better fitness than the other operators, being this behavior more pronounced on datasets of higher dimensionalities (Sonar and Musk). Crossover4 presents the second highest values in its converged solutions when $ts = 1\%$. When $ts = 5\%$, the solutions produced by Crossover4 are slightly surpassed by those produced by Crossover5 in higher dimensionalities. Crossover3 has the next best fitness values, closely followed by Crossover8. Crossover1, Crossover2 and Crossover7 produce similar solutions with the lowest fitness values.

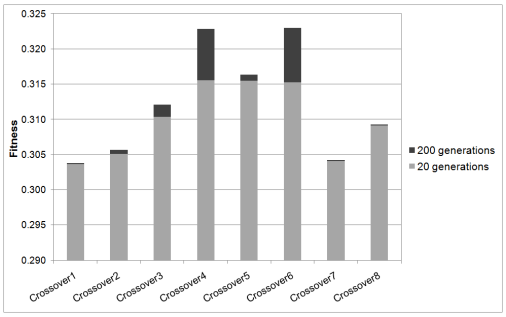
Figure 4 shows for each dataset the mean fitness of the evolving population at 20 and 200 generations for $ts = 1\%$. At both number of generations, it can be observed an interesting difference between the datasets with low dimensions (Wine and WDBC) and the datasets with higher dimensions. In the first group of datasets, Crossover4 and Crossover6 are very competitive in fitness, generating the highest values of all operators. Crossover5 is also competitive at 20 generations in such datasets, but unlike Crossover4 and Crossover6, it quickly reaches its maximum, making a poor improvement of the fitness from 20 to 200 generations. In the datasets with higher dimensionalities (Sonar and Musk), Crossover6 presents a clear superiority to the others operators at both number of generations sampled. At 200 generations in those datasets, Crossover4 continues to be the second best operator in producing high-fitness solutions, but when the number of generations is lowered to 20 it is overcome by Crossover5. Crossover3 and Crossover8 produce the next best solutions in all datasets, with Crossover3 having a better fitness enhancement with the increment of the number of generations. Crossover1, Crossover2 and Crossover7 produce the lowest-fitness solutions in all datasets, regardless the number of generations. They present an irrelevant increment of the mean fitness going from 20 to 200 generations. In addition, Crossover(1,2,3,7,8) tend to find solutions of similar fitness as the dataset dimension is increased.

6 DISCUSSION

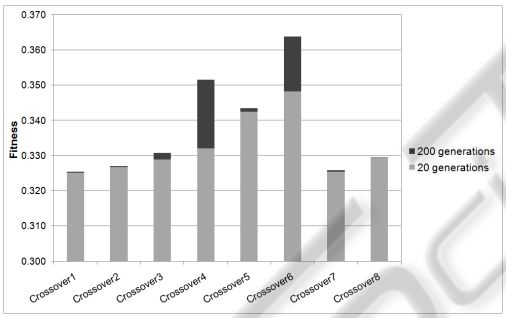
From the above results, it is clear that the performance of the analyzed crossover operators is affected by the dimension of the original data and the selective pressure used to evolve the GA. Based on the fitness of the converged solution, it can be inferred that producing the offspring somewhere in the hypercone around the bisectrix of the parents (Crossover(5,6)) improves the idea of producing the offspring only in the plane of the parents (as Crossover(1,2,7,8)), giving an increased exploratory power to the GA. This strengthening of the search power is even more remarkable if the offspring is allowed to born outside the hypercone of the parents (generated by rotating them around their bisectrix), as Crossover6. The negligible sensitivity of Crossover(5,6) to the change of the selective pressure can also be explained by their good exploratory properties, since they produce offspring in a wider space, being less prone to converge to the first local optimum they found as the selective pressure is increased. The drawback of the hyper-conical operators, especially Crossover6, is the high number of generations



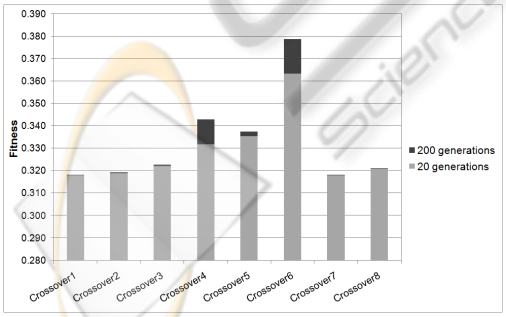
(a) Wine dataset.



(b) WDBC dataset.



(c) Sonar dataset.



(d) Musk dataset.

Figure 4: Results for the mean fitness of the evolving populations (averaged over 300 replicates) at 20 and 200 generations.

needed to converge, which rapidly grows with the dimension of the data. However, if the convergence criterion is a fixed number of generations, Crossover6 gives solutions with competitive fitness, outperforming considerably the other operators in high dimensional datasets.

The idea of recombining the parents by transferring some coordinates of one parent and the rest of coordinates of the other parent, as in Crossover(3,4), proved also to be effective for the purpose of PP factor optimization. It was found that transferring the coordinates as independent units (Crossover4) gives more search performance than transferring the coordinates in blocks, as in Crossover3. Unlike Crossover(5,6), the search performance of these operators is noticeably degraded by the increase of the selective pressure. This suggests that such operators heavily rely on the diversity of the population (given by low selective pressure) to produce better results. The average number of generations to achieve convergence is also considerable in Crossover4 (is the second highest after Crossover6) and, as Crossover6, it grows quickly with the dimension of the data. For a fixed number of generations and in low-dimensional datasets Crossover4 is as competitive as Crossover6 in fitness of the solutions. But in high dimensions it performs worse.

The proposal of biasing the production of the offspring by the fitness of the parents, as in Crossover(7,8), proved to be not as effective as the production of the offspring inside (or outside) the hypercone of the parents. However, some gain is obtained in fitness with almost the same number of generations of convergence of their non-biased versions (Crossover(1,2)). The number of generations to achieve convergence with these non-hyperconic operators (Crossover(1,2,7,8)) is very low in all datasets and decrease slightly with the dimension of the data. This behavior could be explained by the fact that the dimension of the space where the offspring are generated (the plane of the parents) does not grow with the dimension of the data (as in the other operators), so such space is more and more restrictive as the dimension increases, turning them more prone to find local optima.

Based on the above findings we can infer some guidelines to select the crossover operator for a GA as optimizer of PP factors. First, if we have a fixed time frame to get the answer and the dataset has a low dimension (as Wine or WDBC), it would be more advisable to choose Crossover4 with low selective pressure (1% of the population size) or Crossover6 with low to moderate selective pressure (1-5% of the population size). Second, if we have a fixed time frame for the answer but the dataset has moderate to high dimension (as Sonar or Musk), the most appropriate option would be Crossover6. Third, if the time is not a constraint and the dataset has low dimension, the best

options to consider would be Crossover4 with low selective pressure or Crossover(5,6) with moderate selective pressure. Finally, if the time is not a constraint and the dataset is high-dimensional, the best option to consider would be Crossover6 with moderate selective pressure.

7 CONCLUSIONS

Sequential Projection Pursuit - SPP is a useful method to find interesting linear projections of multidimensional data. The main problem in SPP is the optimization of the PP index function, which measures how interesting the projection is. Genetic algorithms are promising optimizer of SPP, but their success is dependent on the selection of their genetic operators. One of the most important operators is the crossover, which is responsible for the rapid exchange of useful information among solutions. This article addressed the problem of which crossover to choose in the design of a GA-based optimizer for SPP. An experimental study was presented, comparing the performance of eight crossover operators: three available in literature (arithmetic crossover and single and multi-point crossover) and five new proposed here (one single extension of the arithmetic crossover, two hyperconic crossovers and two fitness-biased crossovers). The study was carried out over four public datasets of increasing dimension. The results showed that one of the proposed hyperconic operators tends to find projections with higher fitness than the other operators, clearly excelling in higher dimensions. This interesting performance was also observed at different stages of evolution. The multipoint crossover operator presented the second best performance in fitness, being competitive with the hyperconic operator in low dimensional datasets. The final selection of the crossover operator is dependent on the precision required, the dimension of the dataset and the tolerable time to get the answer. Some guidelines to select the most appropriate operator for each situation were presented.

This study is an important step towards the design of efficient GA-optimizers for SPP. We are currently investigating other PP indices and the influence of mutation on the performance of the presented crossover operators. Also, different evolutionary strategies are being studied to take advantage of the features of the proposed operators.

ACKNOWLEDGEMENTS

The authors acknowledge the CAPES/ PEC-PG - Brazil for the scholarship granted to the first author of this article.

REFERENCES

Berro, A., Marie-Sainte, S. L., and Ruiz-Gazen, A. (2010). Genetic algorithms and particle swarm optimization for exploratory projection pursuit. *Annals of Mathematics and Artificial Intelligence*, 60(1-2, SI):153–178.

Bolton, R. and Krzanowski, W. (2003). Projection pursuit clustering for exploratory data analysis. *Journal of Computational and Graphical Statistics*, 12(1):121–142.

Chiang, S.-S., Chang, C.-I., and Ginsberg, I. (2001). Unsupervised target detection in hyperspectral images using projection pursuit. *Geoscience and Remote Sensing, IEEE Transactions on*, 39(7):1380 –1391.

Cook, D., Buja, A., and Cabrera, J. (1993). Projection pursuit indexes based on orthonormal function expansions. *Journal of Computational and Graphical Statistics*, 2(3):225–250.

Daszykowski, M., Kaczmarek, K., Heyden, Y. V., and Walczak, B. (2007). Robust statistics in data analysis - a review: Basic concepts. *Chemometrics and Intelligent Laboratory Systems*, 85(2):203– 219.

Demirci, O., Clark, V. P., and Calhoun, V. D. (2008). A projection pursuit algorithm to classify individuals using fMRI data: Application to schizophrenia. *Neuroimage*, 39(4):1774–1782.

Frank, A. and Asuncion, A. (2010). UCI machine learning repository.

Friedman, J. H. (1987). Exploratory projection pursuit. *American Statistical Association*, 82(397):249–266.

Friedman, J. H. and Tukey, J. W. (1974). A projection pursuit algorithm for exploratory data analysis. *IEEE Trans. Comput.*, 23(9):881–890.

Guo, Q., Wu, W., Massart, D., Boucon, C., and de Jong, S. (2001). Feature selection in sequential projection pursuit. *Analytica Chimica Acta*, 446(1-2):85–96.

Guo, Q., Wu, W., Questier, F., Massart, D., Boucon, C., and de Jong, S. (2000). Sequential projection pursuit using genetic algorithms for data mining of analytical data. *Analytical Chemestry*, 72(13):2846–2855.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems*. University of Michigan Press, Ann Arbor, MI, USA.

Huber, P. J. (1985). Projection pursuit. *Annals of Statistics*, 13(2):435–475.

Jee, J. R. (2009). Projection pursuit. *Wiley Interdisciplinary Reviews: Computational Statistics*, 1(2):208–215.

Jones, M. C. and Sibson, R. (1987). What is projection pursuit? *Journal of the Royal Statistical Society. Series A (General)*, 150(1):pp. 1–37.

Kruskal, J. (1969). 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 Computation*, pages 427–440.

Lee, E., Cook, D., Klinke, S., and Lumley, T. (2005). Projection pursuit for exploratory supervised classification. *Journal of Computational and Graphical Statistics*, 14(4):831–846.

Nason, G. (1995). Three-dimensional projection pursuit. *J. Royal Statistical Society, Series C*, 44:411–430.

Pena, D. and Prieto, F. (2001). Cluster identification using projections. *Journal of the American Statistical Association*, 96(456):1433–1445. 160th Annual Meeting of the American-Statistical-Association, BOSTON, MASSACHUSETTS, FEB, 2000.

Posse, C. (1990). An Effective 2-dimensional Projection Pursuit algorithm. *Communications in Statistics-Simulation and Computation*, 19(4):1143–1164.

Posse, C. (1995a). Projection pursuit exploratory data analysis. *Computational Statistics and Data Analysis*, 20:669–687.

Posse, C. (1995b). Tools for two-dimensional exploratory projection pursuit. *Computational and Graphical Statistics*, 4(2):83–100.

Ren, Y., Liu, H., Yao, X., and Liu, M. (2007). Prediction of ozone tropospheric degradation rate constants by projection pursuit regression. *Analytica Chimica Acta*, 589(1):150–158.

Rodriguez-Martinez, E., Goulermas, J. Y., Mu, T., and Ralph, J. F. (2010). Automatic induction of projection pursuit indices. *IEEE Transactions on Neural Networks*, 21(8):1281–1295.

Srinivas, M. and Patnaik, L. (1994). Genetic Algorithms - A Survey. *Computer*, 27(6):17–26.

Webb-Robertson, B. J. M., Jarman, K. H., Harvey, S. D., Posse, C., and Wright, B. W. (2005). An improved optimization algorithm and bayes factor termination criterion for sequential projection pursuit. *Chemometrics and Intelligent Laboratory Systems*, 77(1-2):149–160.