Using Kernel Principal Component Analysis in Evolutionary Algorithms as an Efficient Multi-Parent Crossover Operator

Petr Pošík
Faculty of Electrical Engineering, Department of Cybernetics
Czech Technical University in Prague
Technická 2, 166 27 Prague 6, Czech Republic
E-mail: posik@labe.felk.cvut.cz

***Abstract*— A new crossover operator for real-valued evolutionary algorithms is proposed in this paper. It is based on the kernel principal component analysis and is very suitable for small groups of design variables with a high degree of dependency. Its efficiency is evaluated on three reference optimization problems and is compared to that of other more conventional crossover operators. The new crossover is in the majority of cases more efficient and more reliable than its competitors, including the simplex crossover.**

I. Introduction

Genetic and evolutionary algorithms (GAs, EAs) have many advantages over other techniques when solving optimization tasks. Among others we can present e.g. the ability to escape from local optima and to use the population as a facility to learn the structure of the problem. On the other hand, the basic versions of EAs suffer from the linkage (epistasis, or statistical dependency) among the design variables involved in the optimization task. In other words, the EAs usually consider each variable to be independent of the others.

In many practical optimization tasks, this assumption is certainly not fulfilled and this fact can greatly deteriorate the efficiency of the evolutionary algorithm. The reason is hidden mainly in the crossover operator. If the EA in certain point of the evolution finds a so-called building block (BB), i.e. good values of a group of linked variables, the crossover is likely not to preserve this group. Many authors, e.g. [3], [6], [11], pointed out the necessity of identifying proper BBs and using this information in the crossover which then has two roles:

- to ensure a good mixing of the identified BBs,
- and to preserve (in some sense) the good building blocks.

On the level of identified BBs, the crossover should serve as an instrument for proper building blocks mixing. This can be ensured by using the classical two-parent crossover. On the contrary, inside the individual BBs, the crossover should preserve good combinations of values. Harik [3] wrote:

"...The GA's population consists of chromosomes that have been favored by evolution and are thus in some sense good. The distribution that this population represents tells the algorithm where to find other good solutions. In that sense, the role of crossover is to generate new chromosomes that are very much like the ones found in the current population. ..."

This can be hardly achieved by a two-parent crossover. We need more information hidden in more parents or in the whole population (we need to use a multi-parent crossover). In real-valued EAs which are considered in the rest of this paper, the crossover operator often proposed for this 'inside-BB' recombination is the so-called simplex crossover (SPX) [11]. This operator randomly selects several parents which form a simplex in the search space, extends it by some factor and uniformly samples one new individual from the extended simplex.

In the rest of this paper, a new crossover operator for tightly linked building blocks is described. This operator originates in the field of estimation of distribution algorithms (EDAs). They do not create new individuals by combining the features of several parents, rather they create an explicit generative probabilistic model of the distribution of good solutions in the search space and the offsprings are then sampled from this model. The kernel principal component analysis is suggested to play the role of the distribution model and is described in section III. Section IV presents an experimental comparison of the proposed crossover with other 'more classic' crossover operators and section V summarizes and concludes the paper.

II. Linkage Learning via Coordinate Transformations

As stated above, the linkage presents a serious issue for evolutionary algorithms. However, there are possibilities how to overcome this limitation; in fact in some cases we can use the linkage to work for us if we are able to identify it and to use it in the evolution. Among other methods, I would like to point out the following two:

1) *Probabilistic models.* In the framework of EDAs we can account for the statistical dependencies by using a probabilistic model that is flexible enough to cover these interactions. The models used in EDAs ranges from very simple univariate marginal distribution models [9] which assume the independence of individual features, to much more complex ones which model the distribution of good

solutions in the search space e.g. as a finite mixture of multivariate gaussian distributions [2].

2) *Transformations of coordinates*. We can transform the individuals from the search space into another one in which the linkage is greatly reduced. The principal component analysis (PCA) can be used to create such a linear reversible transformation which will decorrelate the features of individuals. Another approach would be to use the independent component analysis (ICA) to make the features as independent as possible [13].

The transformations produced by both the PCA and the ICA are linear in nature and thus are not able to reduce the statistical dependency enough. For that reason some authors aimed their attention to non-linear versions of the above mentioned methods. We can e.g. employ a kind of clustering (explicit, or implicit produced by a parallel model of EA) and perform the PCA or ICA on each cluster separately [4]. As it turns out, many of the complex probabilistic models are constructed exactly in this way. In the same time they usually offer a statistically justified way of constructing the model e.g. via the maximal-likelihood estimation [5].

Many of the complex probabilistic models thus can be viewed as a combination of learning (1) a possibly non-linear coordinate transformation and (2) a much simpler probabilistic model of distribution of the transformed data points. Sampling from such a complex model then amounts to (1) sampling from the simple probabilistic model in the transformed space and (2) converting the new data points into the original space via the inverse transformation. This is also the approach used by the kernel principal component analysis that is described in the next section.

III. Kernel Principal Component Analysis

In this section, a short summary of the kernel principal component analysis (KPCA) is given. Interested reader should refer to the original paper [8], or to the book [7]. First, the required theory is presented and then the characteristics of KPCA are discussed.

A. KPCA Basics

When performing ordinary PCA it usually amounts to computing eigenvalues and eigenvectors of the data sample covariance matrix. Let us denote the set of centered data points at hand (the population) as $\mathbf{X} = (\mathbf{x}^1, \mathbf{x}^2, ...\mathbf{x}^N)$. The population matrix $\mathbf{X}$ is of size $D \times N$, where D is the dimension of the input space and N is the population size. The covariance matrix of size $D \times D$ is given by

$$\mathbf{C} = \frac{1}{N}\mathbf{X}\mathbf{X}^T. \tag{1}$$

If we find the eigendecomposition of this matrix, we find the linear transformation which decorrelates the components of individuals. However, it can be shown (see e.g. [8], [5]) that after performing the eigendecomposition of the dot product matrix of size $N \times N$ computed as

$$\mathbf{K} = \frac{1}{N}\mathbf{X}^T\mathbf{X}\ , \tag{2}$$

we arrive to the same non-zero eigenvalues and (after simple computation) to the same eigenvectors. Thus, we can express the whole PCA in terms of dot products.

The kernel methods gained their popularity mainly due to the simplicity with which they allow a broad range of originally linear methods to become non-linear. We can 'non-linearize' any linear algorithm which is expressed in terms of dot products.

Suppose we have a function $\Phi : \mathcal{R}^D \to F$ which presents a non-linear mapping from the input space to the so-called *feature* space. Then, we can define a function $k : \mathcal{R}^D \times \mathcal{R}^D \to \mathcal{R}$ as a dot product of two data points transformed to the feature space:

$$k(\mathbf{x}^i, \mathbf{x}^j) = \langle \Phi(\mathbf{x}^i), \Phi(\mathbf{x}^j) \rangle \tag{3}$$

If we compute a so-called *kernel matrix* $\mathbf{K}$ so that the individual elements of the matrix are $\mathbf{K}_{ij} = k(\mathbf{x}^i, \mathbf{x}^j)$, we have a dot product matrix of the data points transformed to the feature space. We can now perform the linear PCA in the feature space via the eigendecomposition of the kernel matrix and this whole process is called *kernel PCA*.

Suppose now we want to use the KPCA for feature extraction, i.e. for an input point $\mathbf{x}$ we want to find the projection of the image $\Phi(\mathbf{x})$ onto the principal components of the feature space F. The i-th element y_i of the image $\mathbf{y}$ (the i-th non-linear feature of $\mathbf{x}$) can be computed as an projection of the image $\Phi(\mathbf{x})$ on the i-th eigenvector of the kernel matrix $\mathbf{K}$, i.e.

$$f_i(\mathbf{x}) = y_i = \langle \mathbf{v}^i, \Phi(\mathbf{x}) \rangle = \sum_{j=1}^{N} v_j^i \, k(\mathbf{x}^j, \mathbf{x}) \, , \tag{4}$$

where $\mathbf{v}^i = (v_1^i, v_2^i, \ldots, v_N^i)$ is the i-th normalized eigenvector of the kernel matrix $\mathbf{K}$.

However, the best thing is that we need not know the mapping function Φ. It can be shown that if the function k satisfies some conditions (Mercer's conditions), it corresponds to a dot product in some non-linearly mapped feature space F. Thus, we can compute the KPCA just by selecting any valid kernel without explicitly prescribing the mapping Φ. In the literature, the most often used kernels are the polynomial kernel, the radial basis function (RBF) kernel and the sigmoidal kernel. In the rest of this paper, the RBF kernel of the form

$$k(\mathbf{x}^i, \mathbf{x}^j) = e^{-\frac{\left\|\mathbf{x}^i - \mathbf{x}^j\right\|^2}{2\sigma^2}} \tag{5}$$

is used.

B. The Pre-Image Problem

In the last subsection, Eq. 4 provides a way how to carry out the projection of input data points onto the non-linear principal components of the feature space. To be able to use the KPCA as a crossover operator we need to address the inverse transformation as well. In general, this presents rather intricate issue. The mapping Φ induced by the selected kernel k usually maps the low-dimensional input data points to a high-dimensional (possibly even infinite dimensional)

feature space F. The feature space is usually of much higher cardinality so that a one-to-one mapping almost never exists. This means that for an input point $\mathbf{x}$ there always exists its image $\Phi(\mathbf{x})$ in the feature space, but the opposite is not true — for a randomly chosen image in feature space there only seldom exists a precise pre-image in the input space.

However, we can try to look for at least approximate pre-images. The idea is really simple: if we have an image $\mathbf{y}$ given as an expansion of training points images, $\mathbf{y} = \sum_{j=1}^{N} \alpha_j \, \Phi(\mathbf{x}^j)$, we can find its approximate pre-image $\mathbf{z}$ in such a way that the image of $\mathbf{z}$, $\Phi(\mathbf{z})$, minimizes the euclidean distance to $\mathbf{y}$ (or the distance is at least sufficiently small), i.e. we solve the following optimization problem:

$$\mathbf{z} = \arg \min_{\mathbf{x} \in \mathcal{R}^D} \|\mathbf{y} - \Phi(\mathbf{x})\| \tag{6}$$

In [8], for the gaussian kernel the following fixed point iteration algorithm was derived:

$$\mathbf{z}_{n+1} = \frac{\sum_{j=1}^{N} \alpha_j \exp(-\left\|\mathbf{x}^j - \mathbf{z}_n\right\|^2 / (2\sigma^2))\mathbf{x}^j}{\sum_{j=1}^{N} \alpha_j \exp(-\left\|\mathbf{x}^j - \mathbf{z}_n\right\|^2 / (2\sigma^2))} \tag{7}$$

C. KPCA Crossover and Its Characteristics

The principle of the KPCA crossover as used in this article can be described as follows:

1) Normalize the population (make it centered with standard deviation equal to 1)
2) Train the KPCA model on the whole population
3) Find the projection of the population onto the principal components of the feature space F via Eq. 4
4) Determine the hypercube in which the images of the individuals 'live' in the feature space
5) Randomly sample N images from the hypercube
6) Find the pre-images of the new individuals using Eq. 7 iteratively
7) Denormalize the offsprings

The KPCA crossover can be (a bit simplistically) also described as a random search in the feature space. The KPCA amounts to performing the linear PCA in the non-linearly transformed input space, thus in feature space it preserves all the characteristics of the PCA. If we wanted to visualize the non-linear components extracted from the data, we could do it e.g. via the contour plots of the components. In case of PCA we would see a set of parallel contour lines perpendicular to the respective principal component. In case of KPCA we can see rather complex set of curves which describe the data with much greater fidelity (see Fig. 1).

For various kernels we can set a few parameters; however, the precise form of the model is mainly data driven — the training data set is the main factor influencing the final form of the KPCA model. The KPCA is thus very efficient in modeling various types of dependencies — clusters, curves, etc. In Fig. 2 we can see that the KPCA is able to perform a kind of clustering (modeling two –or more– clusters of data points) and a kind of 'curve modeling' (describing each cluster as a ring) in the same time. This figure is an example of the ability

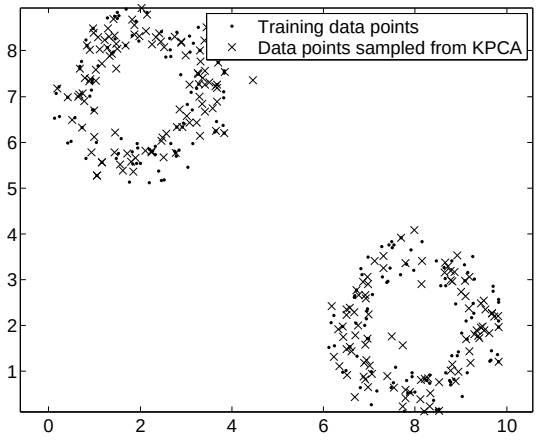

Fig. 2. Example of using the KPCA Crossover. The KPCA is able to perform clustering and a 'curve-fitting' in the same time.

of the KPCA model to generate new data points which are 'very similar' to the training data points. This makes it an ideal crossover operator for variables which are closely dependent on each other; in the same time, the KPCA crossover is not suitable for variables which are not closely related. In that case, the KPCA is very limited in introducing a new genetic material to help in the evolution (as you can see in Fig. 2, the KPCA crossover does not generate any data point in areas, where no training points are present). The search based solely on the KPCA is then strongly biased by the individuals from the initialization phase of the evolution.

IV. Empirical Comparison

A. Evolutionary Model

The carried out experiments were aimed at comparing the KPCA crossover to several other crossover operators. No mutation was used in all experiments. The following evolutionary model was used (let the population size be N):

1) Based on current population, create N new individuals via the respective crossover operator, and evaluate them.
2) Join the old and new populations to get data pool of size $2N$.
3) Use truncation selection to select the better half of data points (returning the population size back to N).
4) Repeat until the number of evaluations exceeds 50,000.

All experiments were repeated 20 times.

B. Test Suite

For the performance evaluation, a few 2-dimensional reference optimization problems with various degrees of dependency between both design variables were used. They are listed in Table I. The TwoPeaks function was used in the work of Tsutsui et al. [12]. It is a separable function. The global optimum is $F(1, 1) = 0$. The Griewangk function is an example of non-separable function. It has one global optimum ($F(0, 0) = 0$) in the origin of the coordinate system and 4 local optima surrounding (and hiding) the global one. Many evolutionary algorithms fall in trouble with this

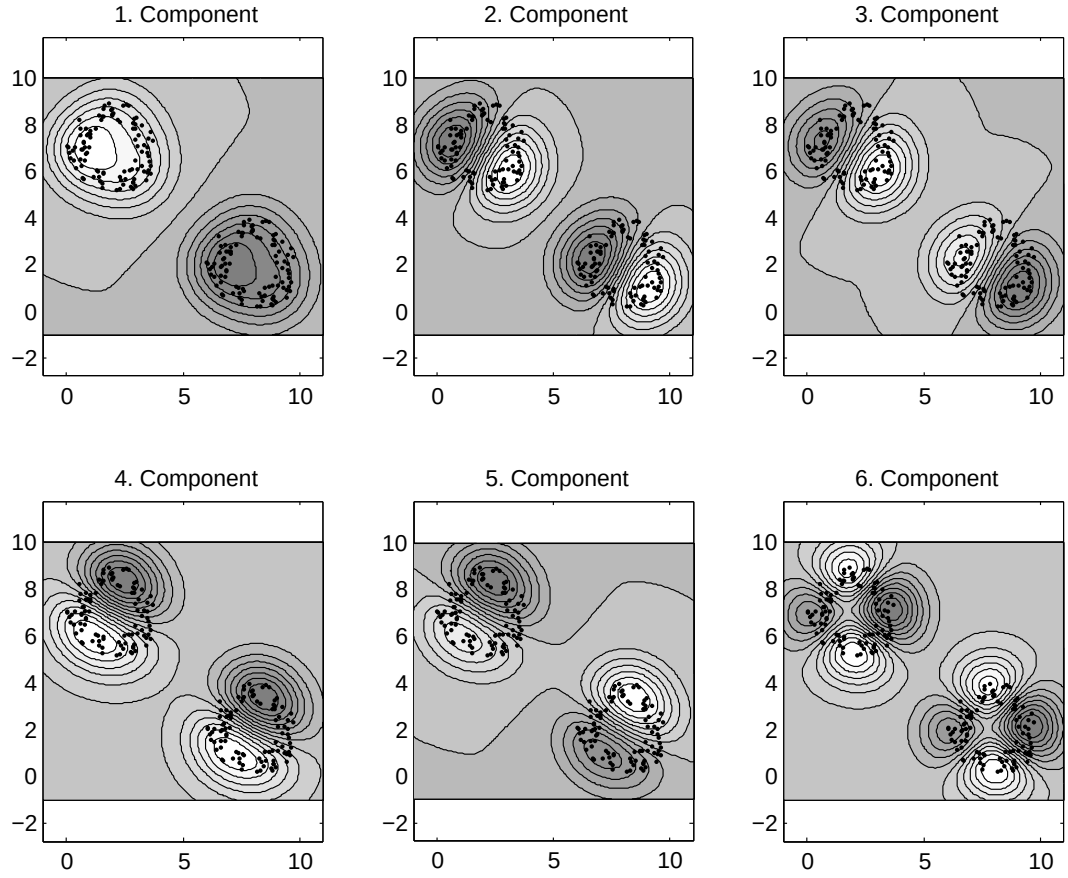

Fig. 1. First six nonlinear components of the toy data set

TABLE I

TEST FUNCTIONS AND RELATED PARAMETERS. THE f FUNCTION CAN BE DESCRIBED AS POLYLINE GOING THROUGH POINTS $(0,0)$, $(1,5)$, $(2,0)$, $(7,4)$, $(12,0)$.

Function	Expression	Domain
Two Peaks	$F = 10 - \sum_{i=1}^{2} f(x_i)$	$\langle 0, 12\rangle^2$
Griewangk	$F = 1 + \sum_{i=1}^{2} \frac{x_i^2}{4000} + \prod_{i=1}^{2} \cos \frac{x_i}{\sqrt{i}}$	$\langle -5, 5\rangle^2$
Rosenbrock	$F = 100 \times (x_1^2 - x_2)^2 + (1 - x_1)^2$	$\langle -2.05, 2.05\rangle^2$

function. The Rosenbrock (a.k.a. Banana) function has high degree of dependency between variables. It is very hard to optimize using ordinary evolutionary algorithms. The optimum is $F(1,1) = 0$.

C. Involved Crossover Operators

Experiments on the following EAs with various crossover operators were carried out:

- GA. A genetic algorithm with the uniform crossover. Binary representation with a resolution allowing the GA to find such a solution the distance of which from the global optimum is not larger than 2.22^{-16}.
- UMDA. The univariate marginal distribution algorithm is described e.g. in [9] in more detail. The distribution of each variable is estimated via the univariate marginal distribution, in this case by means of the empirical equi-height histogram. The number of histogram bins was set to such a number that if all the bins had equal width, none of them was larger then 0.1.
- DiT. EDA with the distribution tree model (in more detail described in [10]) The parameters of the model used for comparison were $minToSplit = 5$, $minInNode = 3$, and $maxPValue = 0.001$.
- SPX. EA with the simplex crossover operator (in more detail described in [11]).
- KPCA. EA with the KPCA crossover presented in this article. The kernel parameter σ was set to 1. When performing the inverse transformation, the number of principal components used was set as the minimal number of components able to describe at least 99.99% of the variance in the feature space, but not less then 10.

D. Monitored Statistics

In all experiments several efficiency measures were tracked:

- *BSF* (Best-so-far fitness). Average fitness of the best individual after 50,000 evaluations in all 20 runs.
- *StdevBSF*. The standard deviation of the best fitness after 50,000 evaluations in all 20 runs.
- *Found0.1* (*Found0.01*, *Found0.001*). As the evolution progresses, it is checked how many times (out of the 20

runs) the best solution is in the '0.1 neighborhood' (0.01, 0.001 respectively) of the global optimum. E.g. for 0.1 neighborhood, the condition $|x_i^{BSF} - x_i^{OPT}| < 0.1$ for all i must hold.

- *WhenFound0.1* (*WhenFound0.01*, *WhenFound0.001*). The average number of evaluations needed to get to the 0.1 neighborhood (0.01, 0.001 respectively) is computed only from the runs in which the algorithm succeeded to get so close to the global optimum.
- *PopSizeUsed*. All the experiments were carried out with a wide range of population sizes (pop. sizes of 20, 50, 100, 200, 400, 600, and 800 were used). The statistics for that population size which resulted in the best score after 50,000 evaluations were selected to be included in Table II.

Each cell of the Table II contains all the above mentioned statistics with the following layout:

BSF (StdevBSF)		
Found0.1	Found0.01	Found0.001
WhenFound0.1	WhenFound0.01	WhenFound0.001
PopSizeUsed		

Fig. 3. Layout of monitored statistics in the results table.

E. Results and Discussion

The overall results are shown in Table II.

TABLE II

COMPARISON OF VARIOUS CROSSOVER OPERATORS ON SELECTED TEST PROBLEMS

Alg.	Function								
	Two Peaks			Griewangk			Rosenbrock		
GA	2,5e-11 (3,9e-11)			1,3e-7 (5,1e-7)			9e-7 (3,1e-6)		
	20	20	20	20	20	18	20	20	19
	1801	5371	9601	1921	28261	35268	2881	15441	24843
	600			600			800		
UMDA	0 (0)			0,0024 (0,0026)			0,0044 (0,0042)		
	20	20	20	14	0	0	16	3	0
	541	1621	2621	3887			2351	2134	
	200			800			800		
DiT	4,8e-13 (2,3e-13)			0 (0)			5e-5 (9,6e-5)		
	20	20	20	20	20	20	20	16	5
	1771	6241	10111	961	3421	6441	2131	16014	29041
	600			400			600		
SPX	0,5063 (0,5114)			0 (0)			0 (0)		
	10	9	6	20	20	20	20	20	20
	5541	10112	8751	431	996	1481	1481	2981	3961
	100			100			200		
KPCA	0 (0)			0 (0)			0 (0)		
	20	20	20	20	20	20	20	20	20
	756	3261	4221	941	21211	24891	289	694	1036
	100			100			50		

The Two Peaks function is separable (having no interactions between the variables) and thus the UMDA was expected to solve such a problem very efficiently. Surprisingly, the EA using KPCA crossover was able to solve this problem with comparable efficiency and reliability as the UMDA (see Fig. 4). This is, however, caused by the low dimensionality of the problem and by the ability of the KPCA to implicitly perform clustering. With increasing dimensionality of this problem the UMDA scales well [9], but I do not expect the KPCA to preserve its superior results in that case (greater population sizes would be needed). The other algorithms solved this problem much worse than the UMDA and KPCA; the SPX crossover failed to solve this task finding a deceptive attractor of the Two Peaks function in a half of all the runs.

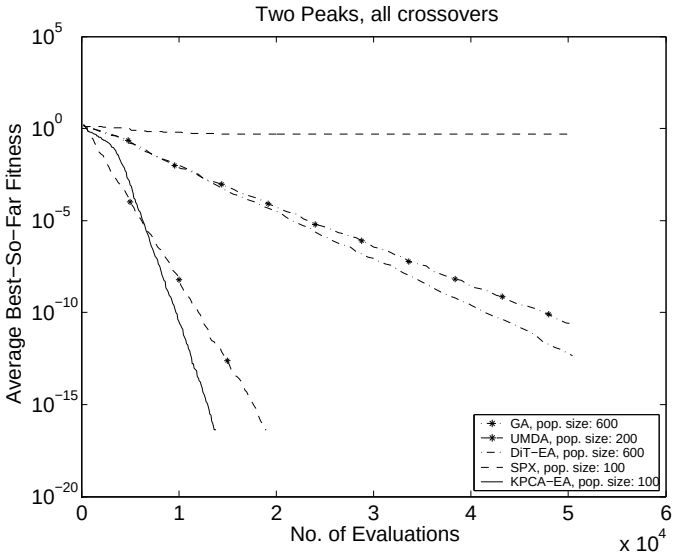

Fig. 4. Comparison of the crossovers on the Two Peaks function

The Griewangk function is not separable and the UMDA is not able to solve this function (see Fig. 5). The SPX crossover solves this problem best, however this is due to the fact that the global optimum lies in the middle of the other local optima. SPX creates many offsprings in between the parents and thus in the neighborhood of the global optima. As was shown by the Two Peaks function in previous paragraph, when solving a general multimodal function the SPX is not very efficient. The EDA using the DiT model is very efficient solving this task as well, although it needs larger population than SPX and KPCA. The KPCA solves this task reliably with relatively small population, however it needs longer time preserving a 'subpopulation' on each of the local optima. After the 'subpopulations' vanish due to the selection the global optimum is reached very quickly.

The Rosenbrock function has the strongest dependency between the variables among all three test problems. The SPX and KPCA were the only algorithms able to solve this task precisely. The EA using KPCA crossover needs much smaller population and is able to solve this 'EA-hard' problem very quickly. It needs only hundreds of fitness function evaluations which is a number that is still worse, but already comparable with more classic, gradient-based optimization techniques.

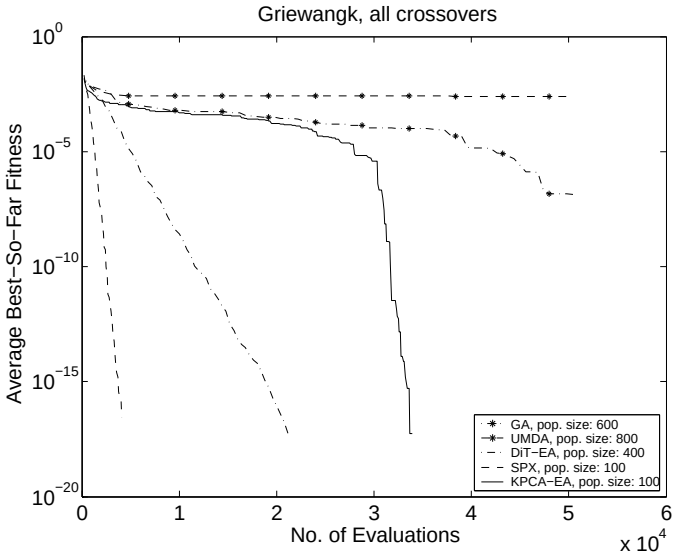

Fig. 5. Comparison of the crossovers on the Griewangk function

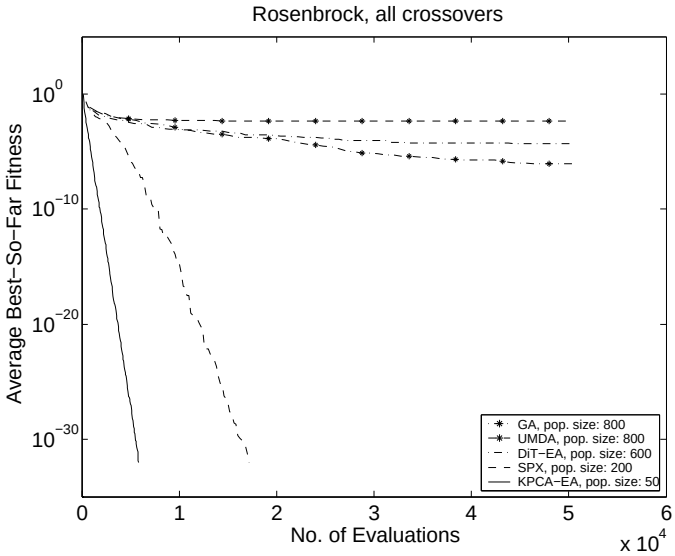

Fig. 6. Comparison of the crossovers on the Rosenbrock function

V. Summary and Conclusions

In this paper, the KPCA crossover was proposed for tightly linked groups of variables. The basics of kernel methods theory were described, along with the kernel principal component analysis and its use as an efficient multi-parent crossover operator. Finally, the performance of KPCA crossover is evaluated on a small set of reference test problems.

Although, the results presented in this paper are very promising, the KPCA crossover still deserves further evaluation and investigation. It should be studied how large tightly linked building blocks the KPCA is able to model efficiently with respect to the population size. Furthermore, when solving practical problems of higher dimensionality, the KPCA should be accompanied with a 'device' for detecting the dependencies between variables (i.e. learning the problem structure). I hypothesize that the KPCA will not scale well in general case and without problem structure learning and decomposition its superior efficiency and reliability would vanish.

Nevertheless, it should be pointed out that no special parameter tuning for the KPCA model was performed and the KPCA crossover solved all the problems with the same parameter settings. There is an open space for research in the field of creating good heuristics to set the parameters, or adapting the parameters during the EA run. The first experiments presented in this paper suggest that the KPCA crossover has a great potential and when used appropriately it belongs to the best operators for competent evolutionary algorithms.

Acknowledgments

This research was supervised by Doc. Ing. Jiří Lažanský, CSc. and was supported by the Grant agency of the Czech Republic with the grant No. GACR 102/02/0132 entitled "Use of Genetic Principles in Evolutionary Algorithms". All experiments with the KPCA were carried out using the Statistical Pattern Recognition Toolbox for MATLAB by Vojtěch Franc [1].

References

[1] Vojtěch Franc. Statistical pattern recognition toolbox. Available on-line, http://cmp.felk.cvut.cz/˜xfrancv/stprtool/index.html.

[2] Marcus R. Gallagher, Marcus Frean, and Tom Downs. Real-valued evolutionary optimization using a flexible probability density estimator. In *Genetic and Evolutionary Computation Conference (GECCO-1999)*, pages 840–864. Morgan Kaufmann, 1999.

[3] Georges Harik. Linkage learning via probabilistic modeling in the ECGA. Technical Report IlliGAL Report No. 99010, University of Illinois, Urbana-Champaign, 1999.

[4] Tomoyuki Hiroyasu, Mitsunori Miki, Hisashi Shimosaka, Masaki Sano, and Shigeyoshi Tsutsui. Distributed probabilistic model-building genetic algorithm. *IPSJ Trans. Mathematical Modeling and Its Applications*, 45(SIG02-008), 2002.

[5] Perry Moerland. *Mixture Models for Unsupervised and Supervised Learning*. PhD thesis, Computer Science Department, Swiss Federal Institute of Technology, Lausanne, 2000.

[6] Martin Pelikan, David E. Goldberg, and Eric Cantú-Paz. Linkage problem, distribution estimation, and bayesian networks. Technical Report IlliGAL Report No. 98013, University of Illinois, Urbana-Champaign, 1998.

[7] Bernhard Schölkopf and Alexander J. Smola. *Learning with Kernels*. MIT Press, Cambridge, Massachusetts, 2002.

[8] Bernhard Schölkopf, Alexander J. Smola, and Klaus-Robert Müller. Nonlinear component analysis as a kernel eigenvalue problem. Technical report, Max-Planck-Institute für biologische Kybernetik, 1996.

[9] Petr Pošík. Comparing various marginal probability models in evolutionary algorithms. In Pavel Ošmera, editor, *MENDEL 2003*, volume 1, pages 59–64, Brno, 2003. Brno University. ISBN 80-214-2411-7.

[10] Petr Pošík. Distribution tree–building real-valued evolutionary algorithm. 2004. Submitted to Parallel Problem Solving From Nature VIII.

[11] Shigeyoshi Tsutsui, David E. Goldberg, and Kumara Sastry. Linkage learning in real-valued GAs with simplex crossover. In *Proceedings of EA'01*, Le Creusot, France, 2001.

[12] Shigeyoshi Tsutsui, Martin Pelikan, and David E. Goldberg. Evolutionary algorithm using marginal histogram models in continuous domain. Technical Report IlliGAL Report No. 2001019, University of Illinois, Urbana-Champaign, March 2001.

[13] Quingfu Yhang, Nigel M. Allison, and Hijun Jin. Population optimization algorithm based on ICA.