OPTIMIZATION VIA CLASSIFICATION

Petr Pošík

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

Abstract

The vast majority of population-based optimization algorithms use selection in such a way that the non-selected individuals do not have any effect on the evolution at all, even though they may carry a valueable information — information about the local shape of the search distribution and/or about the search space areas where the search should be suppressed. This article describes a unified way of taking advantage of the information hidden in the non-selected individuals in the framework of evolutionary algorithms: first, build a classifier discriminating between selected and non-selected individuals, then turn the description of selected individuals into a search distribution, and sample new offspring from it. The concept is verified by a simple real-valued evolutionary algorithm which outperforms the state-of-the-art evolutionary strategy with covariance matrix adaptation (CMA-ES) on selected test functions in all tested search space dimensionalities. Finally, the article proposes some guidelines for future work to make this algorithm generally applicable.

Keywords: *estimation of distribution algorithms, learnable evolution model, quadratic discriminant function, separating ellipsoid, function optimization*

1 Introduction

Many algorithms used for real-parameter black-box optimization use Gaussian distribution to sample new points. This approach started with mutative evolutionary strategies (ES) which were soon equipped with a *feedback adaptation* of the step size (Rechenberg's one fifth rule), and *self-adaptation* of the step size, coordinate-wise step sizes, and self-adaptation of the whole covariance matrix (see e.g. [10]). However, Rudolph [9] showed that self-adaptive mutations can lead to premature convergence.

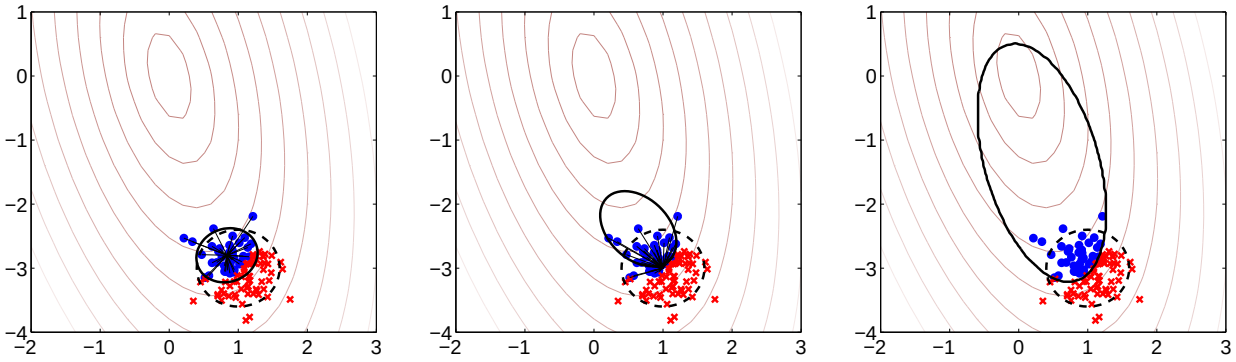
Other algorithms that use Gaussian distribution very often fall into the class of estimation of distribution algorithms (EDAs) [5]. They select better individuals and fit the Gaussian distribution to them—usually by means of the maximum likelihood estimation which is far from ideal (see Fig. 1(a) for an example of Estimation of Multivariate Normal distribution Algorithm (EMNA)). Without imposing limits on the minimal 'size' of the Gaussian, the variance of the distribution in the direction of the fitness function gradient quickly decreases, and the algorithm thus can get stuck even on the slope of the fitness function.

One way of overcoming this drawback of maximum likelihood estimation in real-valued EDAs is to estimate *the distribution of the selected mutation steps* instead of *the distribution of selected individuals* (cf. 1(a) and 1(b)). Pošík [7] applied this approach in a co-evolutionary manner. Hansen at al. [4] use similar principles in the evolutionary strategy with covariance matrix adaptation (CMA-ES) which is considered to be the state-of-the-art in real-valued black-box optimization. It adapts the step size separately from the 'directions' of the multivariate Gaussian distribution. The adaptation is based on accumulation of the previous steps that the algorithm made. Another approach for fighting premature convergence when maximum likelihood estimation is used constitutes the Adaptive Variance Scaling (AVS) of Bosman and Grahl [2]. Their method can enlarge the step size when needed.

In this paper, it is argued that we can learn better search distributions by estimating the contour line of the fitness function that goes between the selected and discarded members of the population. A novel algorithm that learns the Gaussian distribution this way is proposed (see Fig. 1(c)). If the population does not surround a local optimum, the resulting Gaussian distribution should fit into the local neighborhood to much greater extent compared to a Gaussian learned with CMA-ES-like algorithms.

In Sec. 2, the relations with other similar algorithms are described. The main principle of the proposed method and the computational details are presented in section 3. Section 4 describes the experiments that were carried out to assess the very basic properties of the proposed method. Section 5 presents the results of the comparison of our method against the CMA-ES algorithm and against the predecesor of this algorithm that

uses the perceptron learning rule to learn the parameters of the Gaussian. Section 6 summarizes the paper, points out the main advantages of the method and discusses the directions of future work on all the things that have to be done before the algorithm is generally applicable.



(a) EMNA estimates the Gaussian from *selected points* using maximum likelihood method. It is highly prone to premature convergence even on the slope of the fitness function.

(b) CMA-ES-like algorithm estimates the Gaussian from *selected mutation steps* using maximum likelihood method. It fights the premature convergence very well, but the learned Gaussian generally does not fit to the fitness landscape contour lines.

(c) Elliptic classifier learns the Gaussian by estimating the fitness landscape contour line directly. The fit is much closer than in case of the two previous approaches.

Figure 1: An example of learning the covariance matrix using various methods. Contour lines illustrate a 2D quadratic function. Generating Gaussian (---) is used to sample new points which are then divided to selected (•) and discarded (×) points which are in turn used to learn the generating Gaussian for the next generation (—).

2 The Main Principle and Related Algorithms

Although the optimization-via-classification approach is original and distinct from other algorithms, it can be also described in the framework of EDAs as an alternative way of learning the search distribution (compare Alg. 1 and Alg. 2, the main difference is emphasized by small caps).

Algorithm 1: EDA

```
1 begin
2     Initialize and evaluate the population of size N.
3     while termination criteria are not met do
4         Select parents from the population.
5         LEARN A PROBABILISTIC MODEL DESCRIBING THEIR DISTRIBUTION.
6         Sample new individuals.
7         Replace the worst individuals.
8 end
```

Algorithm 2: Optimization via Classification

```
1 begin
2     Initialize and evaluate the population of size N.
3     while termination criteria are not met do
4         Divide the population to better and worse individuals.
5         LEARN A CLASSIFIER DISTINGUISHING BETWEEN THEM.
6         TURN THE DESCRIPTION OF BETTER INDIVIDUALS INTO PROBABILITY DISTRIBUTION.
7         Sample new individuals.
8         Replace the worst individuals.
9 end
```

As can be seen, the direct learning of the probabilistic model in EDA is in *Optimization via Classification* (OvC) replaced by a 'sidestep': first, we learn a classifier distinguishing between selected and discarded individuals, and second, we extract the probability model from the description of the selected individuals hidden in the learned classifier.

Similar principle can be found also in Learnable Evolution Model (LEM) of Wojtusiak and Michalski [13]. LEM offers several alternative ways of creating offspring (including GA-like pipeline, reinitialization pipeline, etc.), one of them consists of classifier distinguishing between good and bad individuals, and instantiation of the

concept of good individuals, which is actually the process used in OvC. LEM, however, uses AQ21 classification rules as the model; the rules divide the search space with axis-parallel splits which are not very suitable for continuous spaces, on the other hand, LEM can be applied to mixed continuous-discrete problems.

Pošík and Franc [8] proposed different model—a combination of quadratic discriminant function as the classifier, and the Gaussian distribution as the search distribution. The basic principle of the algorithm described in this paper is the same as in [8] and is actually illustrated in Fig. 1(c): after evaluation of the population, we try to model the contour line of the fitness function with an ellipsoid that would allow us to discriminate between the selected and discarded individuals. The ellipsoid—the decision boundary—is of the form $\mathbf{x}^T\mathbf{A}\mathbf{x}+\mathbf{B}^T\mathbf{x}+C=0$, where $\mathbf{x}$ is a D-dimensional vector representing a population member, $\mathbf{A}$ is a positive definite $D \times D$ matrix, $\mathbf{B}$ is a vector with D elements, and C is a scalar.

After finding the quadratic decision function, we need to turn it into the sampling Gaussian distribution. Auger et al. [1] discussed that setting the covariance matrix $\mathbf{\Sigma}$ to $\mathbf{\Sigma} = \mathbf{A}^{-1}$ is a very good (if not optimal) choice. We follow this approach since the elliptic decision boundary then corresponds to certain contour line of the Gaussian density function. The candidate members of the new population are then sampled from this distribution. Auger et al. [1] proposed this as a method of improving the CMA-ES covariance matrix adaptation using a quadratic regression model of the fitness function in the local neighborhood of the current point. Their approach, however, required in D-dimensional space at least $\frac{D(D+3)}{2}+1$ data vectors to learn the quadratic function. Moreover, it assumed that each point has its fitness value, i.e. it cannot use selection schemes based on pure comparisons of two individuals.

The works that can be described as instances of the OvC framework are surveyed in Table 1. This paper clearly builds on the previous work [8] and replaces the augmented perceptron algorithm used to learn the quadratic discrimination function with a semidefinite programming (SDP) procedure. The SDP problem formulation imposes some severe limitations on this implementation: this preliminary algorithm is able to optimize only convex quadratic functions. Despite of that, it serves as a proof-of-concept and forms a strong basis for the development of more capable learning algorithm.

Table 1: Algorithms that can be described as instances of optimization-via-classification approach

Sampling distribution	**Created from classifier**	**Classifier learning algorithm**	**Where described**
Mixture of uniform distributions	Classification rules	AQ21	Learnable Evolution Model of Wojtusiak and Michalski [13]
Gaussian distribution	Quadratic discriminant function	Perceptron algorithm	Estimation of Fitness Landscape Contour Lines in EAs of Pošík and Franc [8]
Gaussian distribution	Quadratic discriminant function	Semidefinite programming	Optimization via Classification (this article)

3 Computational Methods

This section describes in detail all the computational methods used in the algorithm: (1) learning the quadratic classifier, (2) turning it into the Gaussian distribution, and (3) modification of the Gaussian to ensure that a specified ratio of the sampled points will lie inside the ellipsoid.

3.1 Learning the Quadratic Discriminant Function

After selection, we need to learn a quadratic function which would allow us to discriminate between two classes of data points. The classifier is given as

$$\mathbf{x}^T\mathbf{A}\mathbf{x}+\mathbf{B}^T\mathbf{x}+C<0 \iff x \in \text{selected individuals} \quad (1)$$

$$\mathbf{x}^T\mathbf{A}\mathbf{x}+\mathbf{B}^T\mathbf{x}+C>0 \iff x \in \text{non-selected individuals} \quad (2)$$

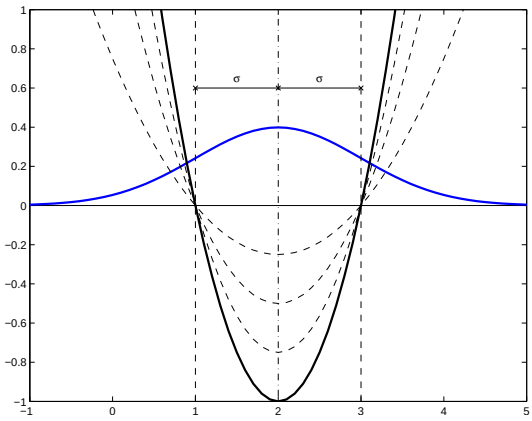


Figure 2: Non-uniqueness of the quadratic decision function. All of them define the same decision boundary.

The decision boundary $\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{B}^T\mathbf{x} + C = 0$ is required to be a hyperellipsoid. As already stated, we shall approach that problem with the semidefinite programming (SDP). The problem can be described as

$$\begin{array}{lrcll} \min t & & & & \\ \text{s.t.} & t & > & 0, & \\ & \mathbf{A} & \geq & 0, & \\ & \|\mathbf{A}\| + \|\mathbf{B}\| + \|C\| & \leq & t, & \\ & \mathbf{x}_i^T\mathbf{A}\mathbf{x}_i + \mathbf{B}^T\mathbf{x}_i + C & \leq & -1 & \text{iff} \quad \mathbf{x}_i \in \text{selected individuals}, \\ & \mathbf{x}_i^T\mathbf{A}\mathbf{x}_i + \mathbf{B}^T\mathbf{x}_i + C & \geq & 1 & \text{iff} \quad \mathbf{x}_i \in \text{non-selected individuals}, \end{array} \tag{3}$$

where $\|\mathbf{A}\|$ is a Frobenius norm of matrix $\mathbf{A}$. This problem specification is taken from Boyd and Vandenberghe [3] (page 429, Quadratic discrimination). Their formulation is, however, only a feasibility problem, i.e. the solution is any quadratic discriminant function that correctly classifies the data points. The minimization of the Frobenius norm of the matrices $\mathbf{A}$, $\mathbf{B}$, and C is added to the problem so that the solution should be a quadratic function with maximum margin to the points from both classes.[1]

The algorithm was implemented in MATLAB 7.3 with the help of SeDuMi [11] and YALMIP [6] toolboxes. Compared to the previous algorithm presented in [8], this formulation of the problem automatically ensures the positive definitness of matrix $\mathbf{A}$ (which had to be enforced 'from outside' in [8]), and is able to detect the situations when the data points are not separable by ellipsoid (in which case the perceptron algorithm used in [8] did not stop at all).

3.2 From Quadratic Function to Gaussian Distribution

The quadratic function $\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{B}^T\mathbf{x} + C$ learned by the SDP procedure is not defined uniquely; all functions of the type $k(\mathbf{x}^T\mathbf{A}\mathbf{x} + \mathbf{B}^T\mathbf{x} + C)$, $k \neq 0$, have the same decision boundary (see Fig. 2 for an illustration in 1D case). One reasonable way of the standardization (assuming matrix $\mathbf{A}$ is positive definite) is to fix the function value at the minimum of the function. The minimum lies in the point $\mu = -\frac{1}{2}(\mathbf{A}^{-1}\mathbf{B})^T$. We deliberately chose the function value at the minimum to be -1, i.e. the following equation must hold:

$$k(\mu^T\mathbf{A}\mu + \mathbf{B}^T\mu + C) = -1 \tag{4}$$

$$k = -\frac{1}{\mu^T\mathbf{A}\mu + \mathbf{B}^T\mu + C} \tag{5}$$

The matrices defining the standardized quadratic function are then given as $\mathbf{A}_S = k\mathbf{A}$, $\mathbf{B}_S = k\mathbf{B}$, and $\mathbf{C}_S = k\mathbf{C}$. The multivariate Gaussian distribution $N(\mu, \mathbf{\Sigma})$ which will be used to sample new points is then given by $\mu = -\frac{1}{2}(\mathbf{A}^{-1}\mathbf{B})^T$ and $\mathbf{\Sigma} = \mathbf{A}_S^{-1}$.

3.3 Sampling from Gaussian Distribution

Sampling from the Gaussian distribution with the center μ and covariance matrix $\mathbf{\Sigma}$ is rather a standard task. The distribution, however, suffers from the curse of dimensionality in such a way that the proportion of generated vectors that lie inside the separating ellipsoid varies (drops with increasing dimensionality).

[1] It is an equivalent of the maximum margin separating hyperplane algorithm of Vapnik [12] which is learned by minimization of the weight vector of the linear decision function. The formulation in this paper thus maximizes margin to points in a quadratically mapped feature space.

Suppose we have a set of vectors generated from D-dimensional standardized Gaussian distribution. Each of the coordinates has unidimensional standardized Gaussian distribution and their sum of squares has a χ^2 distribution with D degrees of freedom, χ^2_D. Thus, if we wanted to specify the percentage p, $p \in (0,1)$, of vectors lying inside the separating ellipsoid we can employ the inverse cumulative distribution function of the χ^2 distribution, $CDF^{-1}_{\chi^2_D}$, in a way that is described in step 3 of the sampling algorithm shown as Alg. 3.

Algorithm 3: Sampling algorithm

1 **begin**
2 Eigendecompose the covarinace matrix so that $\mathbf{\Sigma} = \mathbf{R}\mathbf{\Lambda}^2\mathbf{R}^T$, where $\mathbf{R}$ is a rotational matrix of eigenvectors and $\mathbf{\Lambda}^2$ is a diagonal matrix of the eigenvalues, i.e. $\mathbf{\Lambda}$ is a diagonal matrix of standard deviations in individual principal axes.
3 Modify the standard deviations using the critical value of the χ^2 distribution,

$$\mathbf{\Lambda} = \frac{\mathbf{\Lambda}}{\sqrt{CDF^{-1}_{\chi^2_D}(p)}}. \tag{6}$$

4 Generate the desired number of vectors $\mathbf{x}_S$ from the standardized multivariate Gaussian distribution $N(\mathbf{0}, \mathbf{I})$.
5 Rescale them using the standard deviations $\mathbf{\Lambda}$ and rotate them using $\mathbf{R}$, i.e. $\mathbf{x}_C = \mathbf{x}_S\mathbf{\Lambda}\mathbf{R}$.
6 Decenter the vectors $\mathbf{x}_C$ using the center μ, i.e. $\mathbf{x} = \mathbf{x}_C + \mu$.
7 **end**

This modification of the sampling algorithm can be considered a counterpart to the step size adaptation mechanism in the CMA-ES. Although it is based on different basis, *it modifies the size of the Gaussian*. Furthermore, we should note that by fixing the percentage of points lying inside the separating hyperellipsoid, the search becomes more local with increasing dimensionality.

4 Empirical Evaluation

To assess the basic characteristics of the algorithm, two quadratic fitness functions, *spherical* and *ellipsoidal*, were chosen since the algorithm in its current state cannot cope with non-convex functions:

$$f_{\text{sphere}} = \sum_{d=1}^{D} x_d^2 \tag{7}$$

$$f_{\text{elli}} = \sum_{d=1}^{D} (10^6)^{\frac{d-1}{D-1}} x_d^2 \tag{8}$$

The behavior of our proposed method (which we shall designate as *SDP* in figures) was compared to the behavior of CMA-ES and to the behavior of the previous version of the algorithm (designated as *perceptron*).

4.1 Evolutionary Model

The evolutionary algorithm used in the experiments is the same as Alg. 2. For all three algorithms, the initial population is uniformly generated in area $\langle -10, -5\rangle^D$ in order to test not only the ability to focus the search when it resides in the area surrounding the optimal solution, but also to test the ability to efficiently shift the population toward the optimum. Our method uses elitism. The algorithms were stopped if the best fitness value in the population dropped under 10^{-8}. The results reported are taken from 20 independent runs for each algorithm and configuration.

4.2 Where to Place the Gaussian?

After finding the decision ellipsoid and turning it into a Gaussian distribution, we can decide where we want to place it. There are basically two possible decisions:

1. Place it in the center of the learnt quadratic function. This placement is reminiscent of the process done in conventional EDAs (and is actually depicted in Fig. 1(c)). Also, if the ellipsoid were fit precisely the algorithm could jump to the area of the global optimum in a few iterations.

Table 2: Population sizes used for both test problems and all algorithms

		Dimension			
Algorithm	Problem	2	4	6	8
CMA-ES	Both problems	6	8	9	10
Perceptron	Sphere	9	8	7	6
	Ellipsoidal	11	10	8	6
SDP	Sphere	7	8	9	10
	Ellipsoidal	6	7	8	11

2. Place it around the best individual of the population. Such an approach is similar to mutative ES and the search is more local.

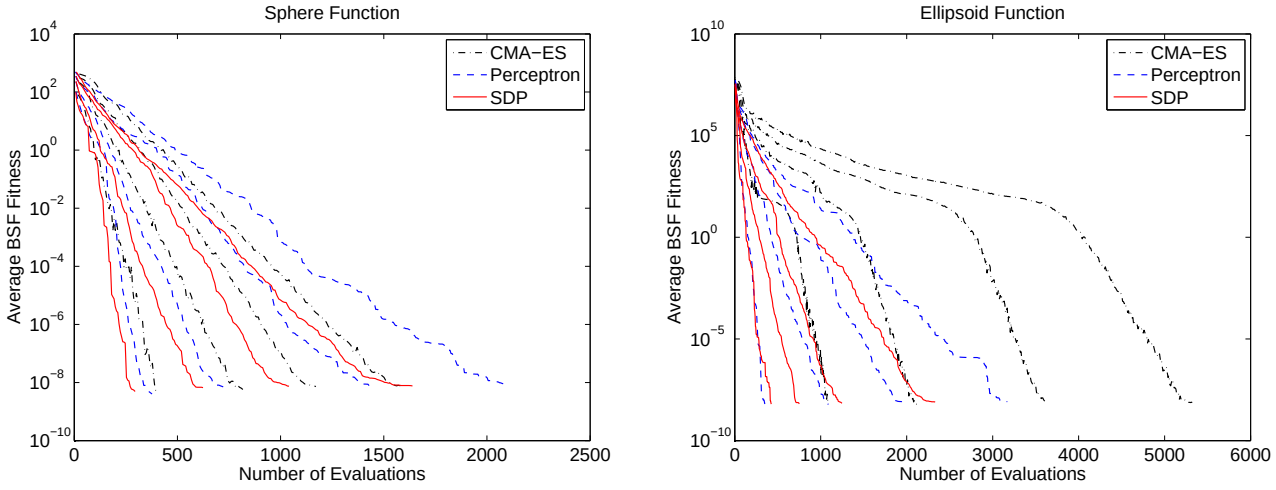
Since we plan to compare our algorithm to CMA-ES which uses the second option, we use it as well and center the learned Gaussian distribution around the best individual in the population.

4.3 Population Sizes

The CMA-ES uses a population sizing equation of the form $N = 4 + \lfloor 3log(D) \rfloor$. For the proposed algorithm, we do not have any population sizing model yet. However, we want to evaluate the potential hidden in our method, and thus we decided to tune the population size for individual test problems and individual dimensionalities.[2] The best settings found for the previous version of the algorithm (that used the perceptron learning algorithm) and for the algorithm proposed in this paper (that uses SDP to learn the ellipsoid) along with the population size used by CMA-ES are presented in Table 2.

5 Results and Discussion

The comparison of the OvC algorithm with perceptron and SDP learning, respectively, and the CMA-ES is depicted in Fig. 3. As can be seen, for the sphere function OvC with SDP is the best for all dimensionalities, although the difference with the CMA-ES gets smaller with increasing dimensionality. For the ellipsoid function, the OvC with SDP algorithm clearly outperforms both the CMA-ES and the OvC with perceptron learning for all tested dimensions (2, 4, 6, 8).



(a) Sphere function.

(b) Ellipsoid function.

Figure 3: Comparison of average evolution traces for the OvC algorithm with ellipsoid learned by SDP (—), the OvC algorithm with ellipsoid learned by perceptron (---) and the CMA-ES (-·-·-). The individual lines belong to 2, 4, 6, and 8 dimensional versions, respectively, from left to right.

Coming back to Table 2, the population sizing for OvC with perceptron was counterintuitive—optimal population size drops with increasing dimensionality. A possible reason for this behavior was given in [8]. For

[2]This is not a good practice for production systems but in this early stage of the research such a tuning is acceptable for discovering the potential of the proposed method.

OvC with SDP no such phenomenon is observed and the optimal poulation sizes have a reasonable relationship with the problem dimensionality.

Another interesting observation is that the rate of convergence is constant during the whole evolution (which can be seen in Fig. 3(b)). The CMA-ES has to adapt the Gaussian to all dimensions (slow progress phase) before it can aim for the global optimum (fast progress phase); it resembles the steepest descent algorithm. OvC, on the contrary, exhibits a constant progress which suggests better fit of the Gaussian distribution to the local neighborhood; it resembles second-order quazi-newton methods.

6 Summary and Future Work

This paper builds on the previous work [8], which introduced the concept of learning a sampling distribution by creating a classifier distinguishing between good and bad individuals. The original work used the perceptron learning rule as an initial attempt to demonstrate the features of such an algorithm, but the perceptron had many drawbacks. In this article, the task of learning the classifier was reformulated as an instance of semidefinite programming and was substituted instead the perceptron. It was shown experimentaly, that this algorithm has better features than its predecesor and than th CMA-ES, but the test function set is rather limited (due to restrictions of the algorithm).

We expect the concept of estimating the fitness contour line to have a number of advantages over the conventional evolutionary algorithms. The most important are:

1. the Gaussian distribution should fit the local neighborhood better than in case of EMNA or CMA-ES,
2. it does not need the fitness values for population members (as is the case in [1])—to know which individuals are selected is sufficient,
3. it estimates a 'reasonable' Gaussian even from a small number of individuals (much less than $D(D+3)/2+1$ that are needed by [1]).

Although the employment of SDP is an improvemnt over previous version, the algorithm is still an initial attempt to proof the concept of modeling the fitness landscape contour line. As such it has strong assumptions which must be relaxed before the algorithm is generally applicable. At present state it is able to optimize only convex quadratic functions. A number of promissing topics for future work remain to be addressed:

- Using SDP, it is possible to employ the soft margin which would allow the algorithm to be applied on non-convex functions, i.e. in cases when the good and bad points are not separable with an ellipsoid.
- The possibility to use the learned center of the quadratic function as the center of the Gaussian was not tested yet.
- With a learning algorithm that finds an elliptic decision boundary even for non-separable cases, there are possibilities to create a learning algorithm for a whole mixture of Gaussians which would allow us to successfully apply the algorithm on multimodal functions. Moreover, such an algorithm can be able to automaticly select the number of Gaussian components. It would be more time demanding, however, it is a generalization worth trying.

To conclude, the experiments carried out suggest there is a big potential in this method if our objective is to find a solution of certain quality using the least possible number of fitness function evaluations. For high-dimensional fitness functions the process of learning the separating ellipsoid may be higly time demanding so that this method should be mainly applicable in cases when the fitness function evaluation is expensive or takes a long time to compute. Nevertheless, we believe that this method can play a signifficant role in the future development of the real parameter optimization field.

7 Acknowledgments

The project was supported by the Ministry of Education, Youth and Sport of the Czech Republic with the grant No. MSM6840770012 entitled "Transdisciplinary Research in Biomedical Engineering II".

References

[1] A. Auger, M. Schoenauer, and N. Vanhaecke. LS-CMA-ES: A second-order algorithm for covariance matrix adaptation. In X. Y. et al., editor, *Parallel Problem Solving from Nature VIII*, number 3242 in LNCS, pages 182–191. Springer Verlag, 2004.

[2] P. A. N. Bosman, J. Grahl, and F. Rothlauf. SDR: a better trigger for adaptive variance scaling in normal EDAs. In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 492–499, New York, NY, USA, 2007. ACM Press.

[3] S. Boyd and L. Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[4] N. Hansen and A. Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[5] P. Larrañaga and J. A. Lozano, editors. *Estimation of Distribution Algorithms*. GENA. Kluwer Academic Publishers, 2002.

[6] J. Löfberg. YALMIP : A toolbox for modeling and optimization in MATLAB. In *Proceedings of the CACSD Conference*, Taipei, Taiwan, 2004.

[7] P. Pošík. Real-parameter optimization using the mutation step co-evolution. In *IEEE Congress on Evolutionary Computation*, pages 872–879. IEEE, 2005. ISBN 0-7803-9364-3.

[8] P. Pošík and V. Franc. Estimation of fitness landscape contours in EAs. In *GECCO '07: Proceedings of the 9th annual conference on Genetic and evolutionary computation*, pages 562–569, New York, NY, USA, 2007. ACM Press.

[9] G. Rudolph. Self-adaptive mutations may lead to premature convergence. *IEEE Trans. on Evolutionary Computation*, 5(4):410–413, August 2001.

[10] H.-P. Schwefel. *Evolution and Optimum Seeking*. Wiley, New York, 1995.

[11] J. F. Sturm. Using SeDuMi, a matlab toolbox for optimization over symmetric cones. *Optimization Methods and Software*, 11–12:625–653, 1999.

[12] V. Vapnik and A. Lerner. Pattern recognition using generalized portrait method. *Automation and Remote Control*, 24:774–780, 1963.

[13] J. Wojtusiak and R. S. Michalski. The LEM3 system for non-darwinian evolutionary computation and its application to complex function optimization. Reports of the Machine Learning and Inference Laboratory MLI 04-1, George Mason University, Fairfax, VA, February 2006.