A Mean-Variance Optimization Algorithm

István Erlich, *Senior Member, IEEE*, Ganesh K. Venayagamoorthy, *Senior Member, IEEE*, and Nakawiro Worawat, *Graduate Student Member, IEEE*

***Abstract*— A new stochastic optimization algorithm referred to by the authors as the 'Mean-Variance Optimization' (MVO) algorithm is presented in this paper. MVO falls into the category of the so-called "population-based stochastic optimization technique." The uniqueness of the MVO algorithm is based on the strategic transformation used for mutating the offspring based on mean-variance of the *n*-best dynamic population. The mapping function used transforms the uniformly distributed random variation into a new one characterized by the variance and mean of the *n*-best population attained so far. The searching space within the algorithm is restricted to the range - zero to one - which does not change after applying the transformation. Therefore the variables are treated always in this band but the function evaluation is carried out in the problem range. The performance of MVO algorithm has been demonstrated on standard benchmark optimization functions. It is shown that MVO algorithm finds the near optimal solution and is simple to implement. The features of MVO make it a potentially an attractive algorithm for solving many real-world optimization problems.**

I. Introduction

Optimization problems exist in many fields of science, engineering, and business. As many real-world optimization problems become increasingly complex, better optimization algorithms are always needed. Many such problems can be precisely formulated, but it is rather difficult or impossible to solve, either analytically or through conventional numerical procedures. This is the case when the problem is non-convex and, therefore, inherently nonlinear and multimodal. Unconstrained optimization problems can be formulated as a *k*-dimensional minimization problem as follows:

$$\text{Min } f(\boldsymbol{x}),\ \boldsymbol{x} = [x_1, x_2, \ldots, x_k]$$

where *k* is the number of the parameters to be optimized.

Population based stochastic search algorithms have become very popular in recent years such as particle swarm optimization, ant colony optimization, genetic algorithms, evolutionary programming, bacterial foraging algorithm, differential evolution and artificial immune systems [1]-[5]. These algorithms have been applied to solve a range of optimization problems from function optimization to complex real world problems like power system stabilizer design [5], scheduling vehicle-to-grid transactions [6] and elephant movement prediction [7].

Although a large number of techniques and approaches exist in literature to improve many of the population-based stochastic algorithms, non-convex optimization is still a challenge to the optimization community. This is mainly due to the large variability with the topological properties of the underlying objective function. Therefore, it is necessary to explore new principles allowing the resolution of a wide range of non-convex optimization problems. In addition, other desirable features for real-world applications call for algorithms with less computational overhead and hardware requirements. Along this spirit, a new optimization algorithm, named by the authors, mean-variance optimization (MVO) is presented in this paper.

Like differential evolution, genetic algorithms and particle swarm optimization, MVO falls into the category of the so-called "population-based stochastic optimization technique." However, its concepts share some similarities and differences from other known stochastic algorithms. The similarities are in borrowing ideas of selection, mutation and crossover from evolutionary computation algorithms. The main distinct features of the MVO algorithm is based on the strategic transformation of mutated genes of the offspring based on the mean-variance of the *n*-best population.

The remaining sections of the paper are organized as follows: Section II describes the MVO algorithm. Section III presents results on sample benchmark functions. Finally, conclusions are presented in Section IV.

II. The MVO Algorithm

The flowchart of the MVO algorithm is given in Fig. 1. The basic steps in the MVO algorithm are described below.

A. Initialization of MVO Algorithm and Normalization of Variables

The MVO algorithm and optimization problem parameters have to be initialized. The few MVO algorithm parameters to be initialized include:

- dynamic population size, *n*
- number of dimensions (problem variables), *m*, to be selected for mutation
- selection method used

Manuscript revised April 30, 2010. This work was supported in part by the grants - NSF CAREER ECCS #0348221 and NSF EFRI 0836017.

I. Erlich is the Chair of the Department of Electrical Power Systems, University of Duisburg-Essen, Germany (e-mail: istvan.erlich@uni-due.de).

G. K. Venayagamoorthy is the Director of the Real-Time Power and Intelligent Systems Laboratory, Missouri University of Science and Technology (http://web.mst.edu/~ganeshv & e-mail: gkumar@ieee.org).

N. Worawat is PhD candidate with the Department of Electrical Power Systems, University of Duisburg-Essen, Germany (e-mail: worawat.nakawiro@uni-due.de).

Further optional parameters used in the strategic transformation are:

- shaping scaling factor, f_s
- Asymmetry Factor AF
- initial value of shape factor s_d.

The numbers of problem variables, k, to be optimized have to be initialized within the allowed limits. It can be done by the user; otherwise the variables are initialized randomly.

The range of the search space for all optimization variables within the MVO algorithm is [0,1]. However, fitness evaluation is carried out using the actual values in the problem space (fitness evaluation space). Therefore, during the optimization, de-normalization is carried out in every single iteration.

B. *Termination Criteria*

The MVO search process can be terminated based on a completion of a specified number of iterations (fitness evaluations), the solution, $\boldsymbol{x}$, attaining a desirable fitness or no improvement in the fitness over the last s iterations. The termination criteria is determined by the user for a given optimization problem. In this context, the MVO algorithm is not different from many of the heuristic optimization algorithms. It should be noted that the number of iterations in the MVO algorithm is equivalent to the number of offspring fitness evaluations.

C. *Dynamic Population*

The MVO utilizes a single parent-offspring pair concept but incorporates information of performance of the n best individuals stored in the archive table through mean and variance. The population size n, is taken to be a minimum of two. In order to factor population performance in the MVO operations (described below), a minimum of two initial random solutions are needed at the beginning. If n is greater than two the table of best individuals is filled up progressively over iterations in a descending order of the fitness. When the table is filled with n members an update is performed only if the fitness of the new population is better than those in the table. As fitness improves over iterations, the population members keep changing, in other words, dynamic.

Mean, $\bar{x}_i$, and variance, v_i, are computed after every update of the archive for each dimension using (1) and (2) respectively.

$$\bar{x}_i = \frac{1}{n}\sum_{j=1}^{n} x_i(j) \tag{1}$$

$$v_i = \frac{1}{n}\sum_{j=1}^{n} (x_i(j) - \bar{x}_i)^2 \tag{2}$$

where, j goes from 1 to n (population size). At the beginning $\bar{x}_i$ corresponds with the initialized value of x_i and the variance is set to $v_i = 1.0$.

Small population size will result in focusing on exploration whereas a large size leads to better exploitation.

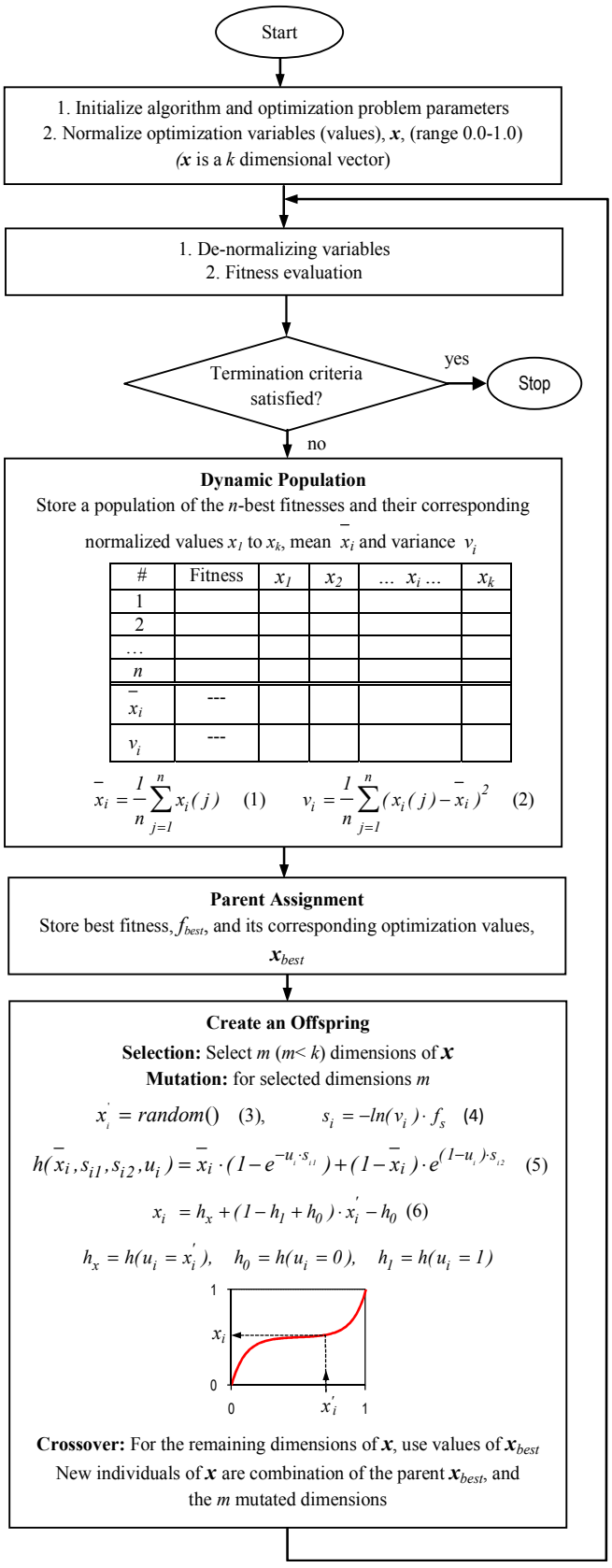

Fig. 1. MVO algorithm flowchart.

D. Parent Assignment

The individual with the best fitness f_{best} (first position in the table), and its corresponding optimization values, $\boldsymbol{x}_{best}$, are stored in memory as the '*parent*' of the population for that iteration. Generally it is possible to use any individuals saved in the table or their combinations as parent. The authors tested the mean of all individuals and some variants of the weighted means. However, in all optimization examples tested so far the best population as parent provided the best performance.

E. Offspring Creation

Creation of an offspring, of k dimensions involves three common evolutionary computation algorithms' operations, namely: selection, mutation and crossover.

Selection - m of k dimensions of the optimization problem are selected for mutation operation. The following four selection strategies were studied:

1) Random selection
2) Neighbor group selection
   a) moving the group forward in multiple steps
   b) moving the group forward in single steps
3) Sequential selection of the first variable and the rest, randomly.

The two versions of strategy 2 are demonstrated in Fig 2.

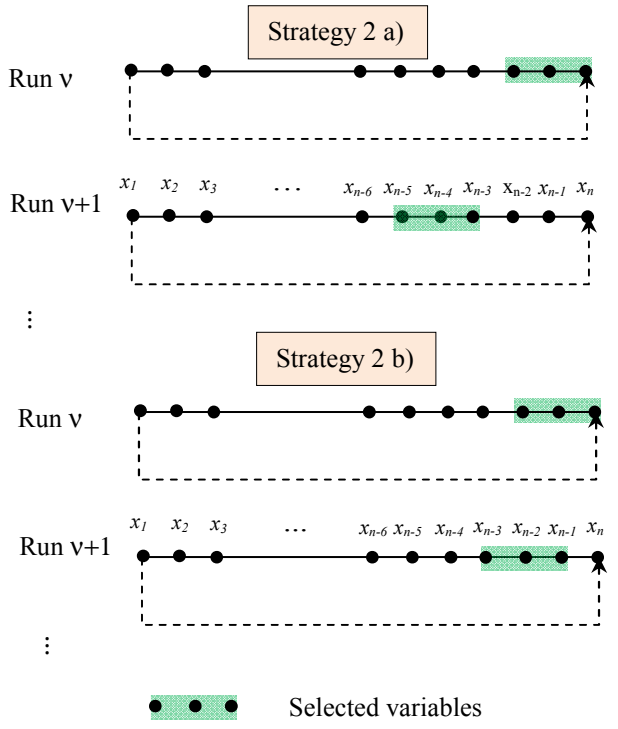

Fig. 2. Selection strategies 2a and 2b.

Different strategies can be applied for selection and the right strategy depends on the optimization problem. According to the authors experience the strategy 2 and 3 always performed better than strategy 1. However the conclusion can not be generalized.

Mutation -First the values of the selected dimensions m are randomized from a uniform distribution in the range of [0, 1].

$$x_i' = random() \tag{3}$$

The generated random values are transformed then based on the mean and variance of the m best population stored in the archive table. The transformation and the corresponding function are key features of the MVO algorithm.

A transformation function, h, is introduced based on the mean and shape factor, s_i, given in (4).

$$s_i = -ln(v_i) \cdot f_s \tag{4}$$

The scaling factor f_S allows for controlling the search process. A small value of f_s (between 0.9 and 1.0) allows the slope of the mapping curve to increase (see below) and thus better exploration. Values of f_s above 1.0 (and up to 10) will result in a flat curve and thus improved exploitation. The parameter f_S allows also to define different slopes s_{i1}, s_{i2} to focus on the space to be searched below or even above the mean value.

Dimensions m of the randomly generated variables are transformed using (5) and (6).

$$h(\bar{x}_i, s_{i1}, s_{i2}, u_i) = \bar{x}_i \cdot (1 - e^{-u_i \cdot s_{i1}}) + (1 - \bar{x}_i) \cdot e^{(1-u_i)\cdot s_{i2}} \tag{5}$$

$$x_i = h_x + (1 - h_1 + h_0) \cdot x_i' - h_0 \tag{6}$$

where

$$h_x = h(u_i = x_i'), \quad h_0 = h(u_i = 0), \quad h_1 = h(u_i = 1)$$

The transformation function is illustrated in Fig. 3.

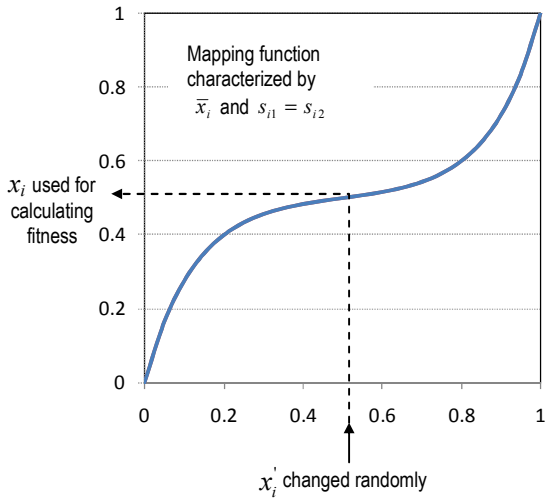

Fig. 3. Illustration of mapping function and transformation.

By extending the function $h(*)$ by h_1 and h_0 according (6) the output range of the function become exactly [0,1] as the case for the input. The effects of the mean and shape factor on the transformation function are shown in Figs. 4 and 5.

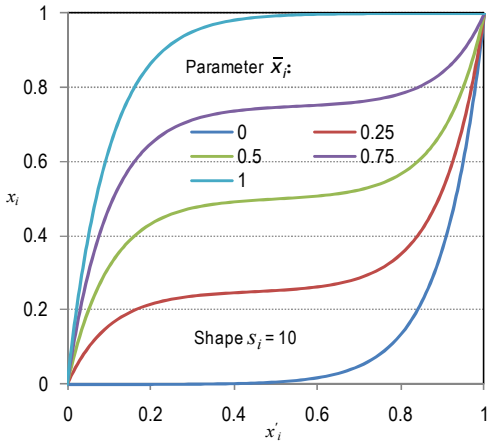

Fig. 4. Effects of mean of dynamic population on the transformation function *h*

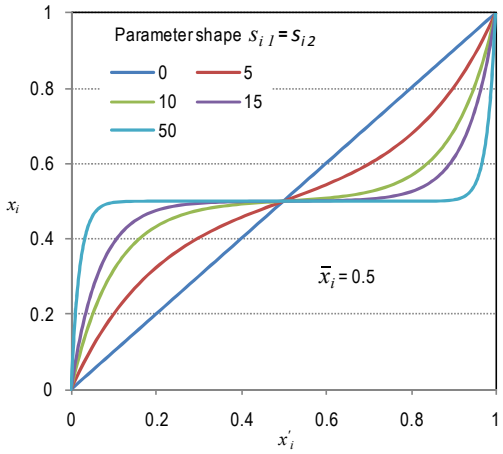

Fig. 5. Effects of shaping scaling factor on the transformation function *h*.

As can be seen, with decreasing variance and thus increasing shaping factor, the corresponding curve is getting more flat so that the space to be searched is focused on the region near to the mean value. However, all curves start and end at zero and one, respectively. Therefore, the search still covers the whole space between the min/max limitations, even though the probability to be selected near to the border is less.

The effect of different shape factors $s_{i1} \neq s_{i2}$ is demonstrated in Fig. 6. Obviously, by choosing different shape factors the space to be searched can be controlled by shifting emphasis to preferred side of the curve around the mean value. The algorithm implemented by the authors for utilizing this control alternative is as follows:

if $x_i^{best} < \bar{x}_i \rightarrow s_{i2} = s_i \cdot AF \; ; \; s_{i1} = s_i$

else if $x_i^{best} > \bar{x}_i \rightarrow s_{i1} = s_i \cdot AF \; ; \; s_{i2} = s_i$

where *AF* represents the asymmetry factor in the range of [1-10].

Crossover -For the remaining un-mutated dimensions the genes of the parent, $\boldsymbol{x}_{best}$, are inherited. In other words, the values of these un-mutated dimensions are clones of the parent. Here, crossover is by direct cloning of certain genes. In this way the offspring is created by combining the vector $\boldsymbol{x}_{best}$, and vector of *m* mutated dimensions.

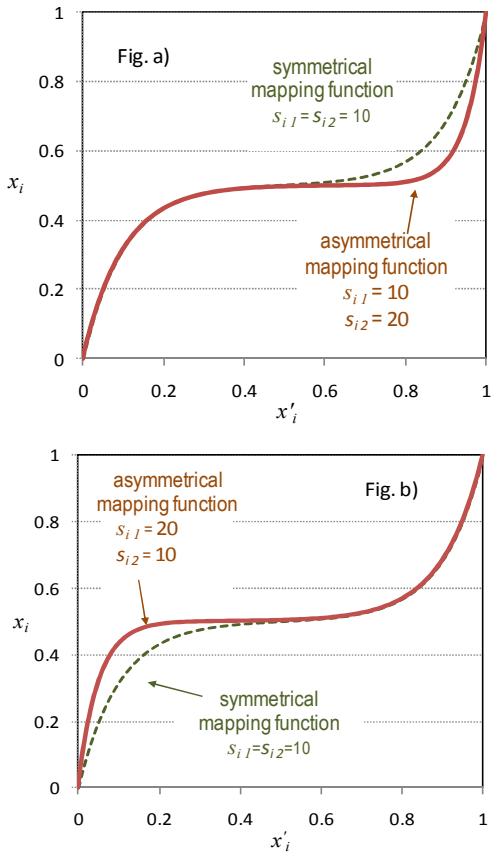

Fig. 6: Effect of different shape factors $s_{i1} \neq s_{i2}$

Treating zero variance - Zero variance can occur when all individuals stored in the archive do not differ with respect to particular optimization variables. In consequence, the shape factor of the mapping curve tends to zero and the optimization does not progress. The probability of zero variance is increasing as the number of mutated variables *m* is small.

One of the approaches to solve this problem is to use the last no-zero variance further. The authors have seen considerable improvements with the following insertion to the algorithm:

$s_{i1} = s_i \; ; \quad s_{i2} = s_i$
where s_i calculated from the last nonzero variance v_i
if $s_i < s_d \rightarrow s_d = s_d . k_d \; ; \; s_{i1} = s_d$
else if $s_i > s_d \rightarrow s_d = s_d / k_d \; ; \; s_{i1} = s_d$

Assuming that the variables s_d and k_d are common for all optimization variables and are initialized to:

$s_d = 75$
$k_d = 0.0505/k + 1.0$

For s_d, any value between 10 and 90 is fine but 75 was chosen for the benchmark function optimizations in this

paper. The variable s_d allows tracking the mean of all s_i resulting from the last no-zero variance.

III. Benchmark Function Results

For evaluating and comparing the MVO algorithm with best results reported in literature using variants of PSO [7, 8], a test suite of standard benchmark optimization functions consisting of various unimodal and multimodal problems are used. Five of these functions are presented in Table I. The parameter settings for the benchmark functions are presented in Table II. The parameter settings for the MVO algorithm are as follows:

- dynamic population size, $n = 2$,
- randomly selected number of variables for mutation is 1, and
- the shape scaling factor, f_s = 1.0 and asymmetry factor, AF=1.0.

TABLE I
Benchmark Functions

Function Name	Benchmark Function	f_{min}
f_1	$\sum_{i=1}^{n}(100(x_{i+1}-x_i^2)^2+(x_i-1)^2)$	0
f_2	$\sum_{i=1}^{D}[y_i^2-10\cos(2\pi y_i)+10]$ $y_i = \begin{cases} x_i, \lvert x_i\rvert < \frac{1}{2} \\ \frac{round(2x_i)}{2}, \lvert x_i\rvert \geq \frac{1}{2} \end{cases}$ $i = 1,2,.....,D$	0
f_3	$\sum_{i=1}^{D} ix_i^4 + N(0,1)$	0
f_4	$\frac{\pi}{D}\{10\sin^2(\pi y_i)+\sum_{i=1}^{D-1}(y_i-1)^2[1+10\sin^2(\pi y_{i+1})]$ $+(y_D-1)^2\}+\sum_{i=1}^{D}u(x_i,10,100,4),$ $y_i = 1+\frac{1}{4}(x_i+1),$ $u(x_i,a,k,m) = \begin{cases} k(x_i-a)^m, x_i > a, \\ 0, -a \leq x_i \leq a \\ k(-x_i-a)^m, x_i < -a, \end{cases}$	0
f_5	$0.1\{\sin^2(3\pi x_1)+\sum_{i=1}^{D-1}(x_i-1)^2[1+\sin^2(3\pi x_{i+1})]$ $+(x_D-1)^2[1+\sin^2(2\pi x_D)]\}+\sum_{i=1}^{D}u(x_i,5,100,4)$	0

All function optimization are carried for 50 trials and are shown in Table III. The fitness graphs for five benchmark function optimization with the MVO algorithm over 50 trials are shown in Figs. 7 to 11. It can be observed from Table III, that MVO algorithm produces the best result for function f_1 and similar results to those reported in literature for f_2 to f_5 functions. It should be noted that these results are based on a single parent-offspring pair.

TABLE II
Parameter Settings For Benchmark Functions in Table I

Function Name	Dimension	Initialization Range	Number of Evaluations
f_1	30	[-2.048 , 2.048]	200000
f_2	30	[[-5.12, 2]	200000
f_3	30	[-1.28, 1.28]	200000
f_4	30	[-50, 50]	200000
f_5	30	[-50, 50]	200000

TABLE III
Results Averaged Over 50 Trials

Function Name	MVO Min	MVO Mean	MVO Max	MVO Std	Best PSO based algorithms
f_1	4.18e-05	**4.68e-02**	1.38e-01	4.12e-02	5.39e+000 (FDR – [8])
f_2	1.00e-99	5.30e-08	2.51e-06	3.55e-07	**4.36e-010** (CLPSO – 8])
f_3	1.20e-03	4.40e-03	8.72e-03	1.76e-3	**3.15e-03** (SADE – [9])
f_4	9.27e-30	6.96e-27	1.67e-25	2.58e-26	**6.60e-30** (SADE – [9])
f_5	3.23e-28	6.21e-25	1.59e-23	2.82e-24	**5.00e-29** (SADE – [9])

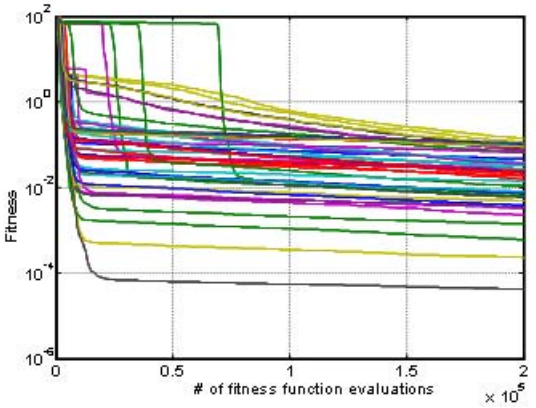

Fig. 7. Fitness graphs over 50 trails for the benchmark function f_1 in Table I.

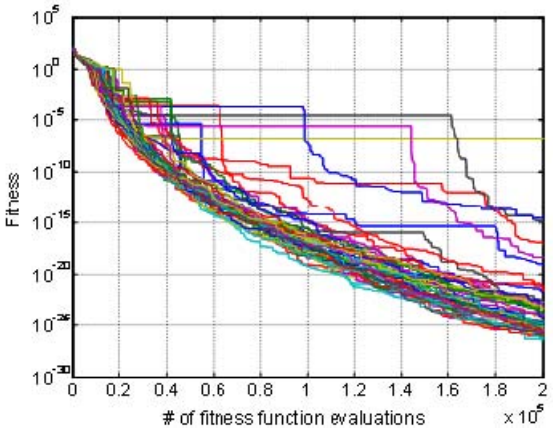

Fig. 8. Fitness graphs over 50 trails for the benchmark function f_2 in Table I.

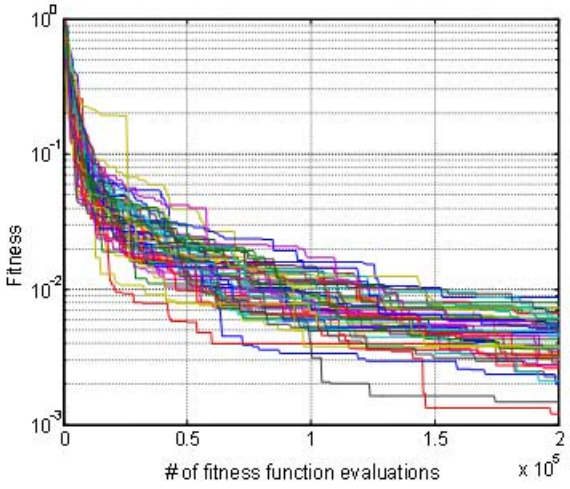

Fig. 9. Fitness graphs over 50 trails for the benchmark function f_3 in Table I.

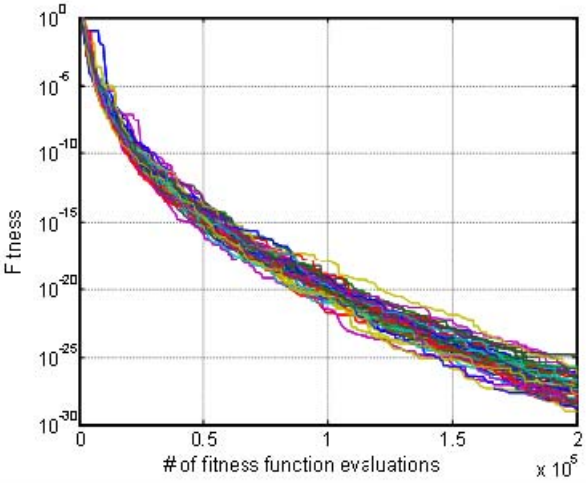

Fig. 10. Fitness graphs over 50 trails for the benchmark function f_4 in Table I.

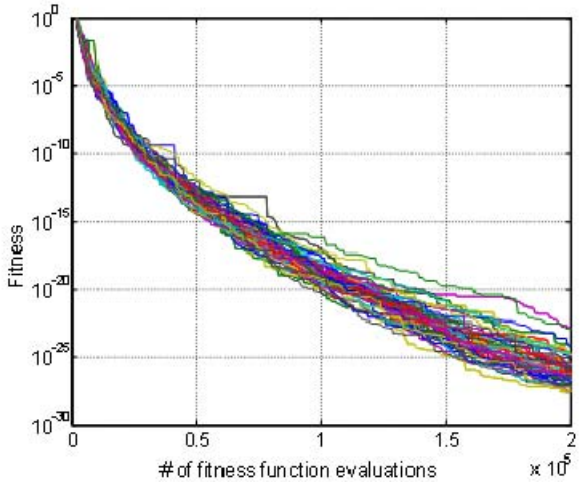

Fig. 11. Fitness graphs over 50 trails for the benchmark function f_5 in Table I.

IV. Conclusion

A new optimization algorithm – the mean-variance optimization - has been presented. The main characteristics of the MVO algorithm is based on the unique transformation used for mutating genes in the offspring based on the mean-variance of a dynamic population. Both inputs and outputs of the mapping function used to transform uniformly distributed random variables covers the range [0,1]. Therefore the optimization is performed in the same range. Variables are de-normalized only for any function evaluation. In the basic MVO presented in this paper, a single parent-offspring pair per concept is implemented. The performance of MVO algorithm has been demonstrated on standard optimization benchmark functions. It is shown that MVO algorithms finds the near optimal solution and is simple to implement.

Currently, MVO algorithm is evaluated on constrained optimization problems. The performance of the MVO algorithm needs to be investigated with respect to population size, other selection and crossover strategies and methods treating the problem of zero variance. Furthermore, the authors are working towards a swarm MVO algorithm (SMVO). Another promising approach comprises dynamic parameter control during the iteration which allows adaptation to the specific optimization problem and iteration progress.

References

[1] J. Kennedy and R. C. Eberhart, "Particle swarm optimization," in *Proc. IEEE Int. Conf. Neural Networks*, pp. 1942–1948, 1995.

[2] Y. del Valle Y, G. K. Venayagamoorthy, S. Mohagheghi, J. C. Hernandez and R. G. Harley, "Particle swarm optimization: Basic concepts, variants and applications in power systems", *IEEE Trans. on Evolutionary Computation*, Vol. 12, Issue 2, April 2008, pp. 171-195.

[3] A. Engelbrecht, *Computational Intelligence: An Introduction*, John Wiley & Sons, Ltd, England. ISBN 0-470-84870-7.

[4] D. Dasgupta, "Advances in Artificial Immune systems", *IEEE Computational Intelligence Magazine,* Vol. 1, No. 4, Nov. 2006, pp. 40-49.

[5] T. K. Das, G. K. Venayagamoorthy, U. O. Aliyu, "Bio-inspired Algorithms for the Design of Multiple Optimal Power System Stabilizers: SPPSO and BFA", *IEEE Transactions on Industry Applications*, Vol. 44, Issue 5, September/October 2008, pp. 1445-1457.

[6] A. Saber, G. K. Venayagamoorthy, "Intelligent Unit Commitment with V2G - A Cost-Emission Optimization", *Journal of Power Sources*, Vol. 195, 2010, pp. 898-911.

[7] G. K. Venayagamoorthy, "A Successful Interdisciplinary Course on Computational Intelligence", *IEEE Computational Intelligence Magazine – A special issue on Education*, Vol. 4, No. 1, February 2009, pp. 14-23.

[8] J. J. Liang, A. K. Qin, P. N. Suganthan and S. Baskar, "Comprehensive Learning Particle Swarm Optimizer for Global Optimization of Multimodal Functions," *IEEE Transactions on Evolutionary Computation,* vol. 10, no. 3, pp. 281- 295, June 2006.

[9] J. Brest, S. Greiner, B. Bo˘skovic´, M. Mernik, and V. Žumer, "Self-Adapting Control Parameters in Differential Evolution: A Comparative Study on Numerical Benchmark Problems," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 6, pp 646-657 Dec. 2006.