2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Gaussian Adaptation based Parameter Adaptation for Differential Evolution

R. Mallipeddi, Guohua Wu, Minho Lee and P. N. Suganthan

***Abstract*— Differential Evolution (DE), a global optimization algorithm based on the concepts of Darwinian evolution, is popular for its simplicity and effectiveness in solving numerous real-world optimization problems in real-valued spaces. The effectiveness of DE is due to the differential mutation operator that allows DE to automatically adjust between the exploration/exploitation in its search moves. However, the performance of DE is dependent on the setting of control parameters such as the mutation factor and the crossover probability. Therefore, to obtain optimal performance preliminary tuning of the numerical parameters, which is quite timing consuming, is needed. Recently, different parameter adaptation techniques, which can automatically update the control parameters to appropriate values to suit the characteristics of optimization problems, have been proposed. However, most of the adaptation techniques try to adapt each of the parameter individually but do not take into account interaction between the parameters that are being adapted. In this paper, we introduce a DE self-adaptive scheme that takes into account the parameters dependencies by means of a multivariate probabilistic technique based on Gaussian Adaptation working on the parameter space. The performance of the DE algorithm with the proposed parameter adaptation scheme is evaluated on the benchmark problems designed for CEC 2014.**

I. Introduction

Diffrential Evolution (DE) [1], is one of the most successful stochastic search technique for numerical optimization. Similar to most of the stochastic algorithms that are based on the principles of Darwinian evolution, DE makes use of genetic operators such as mutation, crossover and selection. However, the effectiveness of DE can be attributed to the differential mutation operator through which the algorithm can self-adapt its search to suit the landscape of the optimization problem at hand. The effectiveness of DE has been demonstrated in many application fields such as mechanical engineering [1], communication [2], optics [3], pattern recognition [4], signal processing [5] and power systems [6].

R. Mallipeddi and Minho Lee are with the School of Electronics Engineering, Kyungpook National University, Daegu, South Korea 702 701 (phone: +8253-950-7223; fax: +8253-950-5505; e-mail: mallipeddi.ram@gmail.com, mholee@gmail.com).

Guohua Wu is with the National University of Defense Technology, Changsha, Hunan, China 410073 (e-mail: guohuawu.nudt@gmail.com)

P. N. Suganthan is with School of Electrical and Electronics Engineering, Nanyang Technological University, Singapore, 639 798 (e-mail: epnsugan@ntu.edu.sg)

However, experimentally [7, 8] and theoretically [9] it has been demonstrated that the performance of DE is sensitive to the mutation strategy, crossover strategy and control parameters such as population size (*NP*), crossover rate (*CR*) and scale factor (*F*). In other words, the best combination of strategies and their associated control parameters can be different for different optimization problems. In addition, for the same optimization problem the best combination can vary depending on the available computational resources and accuracy requirements [10]. Therefore, to successfully solve a specific optimization problem, it is necessary to perform trial-and-error search for the most appropriate combination of strategies and their associated parameter values. However, the trial-and-error search process is time-consuming and incurs high computational costs. Therefore, to overcome the time consuming trial-and-error procedure different adaptation schemes [11-15] have been proposed in the literature.

From the different adaptive or self-adaptive parameter control techniques proposed [11-15], it can be observed that a well-designed parameter adaptation scheme can enhance the robustness of an algorithm by dynamically adapting the parameters to the characteristic of different fitness landscapes. In other words, a well-designed parameter adaptation technique can effectively solve various optimization problems without the need for the trial and error process of parameter tuning. In addition, the convergence rate can be improved if the control parameters are adapted to appropriate values at different evolution stages of a specific problem.

Most of the DE parameter adaptation techniques [11-15] employ explorative mutation strategies to obtain better performance. However, in [16], the authors proposed a parameter adaptation method with a greedy mutation strategy and binomial crossover strategy as search basis. The greedy mutation strategy "DE/current-to-*p*best" utilizes the information of multiple best solutions to balance the greediness of the mutation and diversity of the population. The parameter adaptation technique is implemented by evolving the mutation factors and crossover probabilities based on their historical record of success. The authors claim the parameter adaptation to increase the convergence rate while maintaining the reliability of the algorithm at a high level.

From the experimental [7, 8] and theoretical [9], it has been found that the performance of the DE algorithm depends on the appropriate combination of the mutation scale factor and the crossover probability. However, most of the parameter adaptation techniques proposed in the literature [11-16], consider the adaptation of the two different parameter individually but do not consider the interaction between the two parameters. In other words, they do not take into account

the side effects introduced by changing the values of the parameters individually.

Estimation of Distribution Algorithms (EDAs) such as Covariance Matrix Adaptation (CMA) [17] and Gaussian Adaptation (GaA) [18], based on probabilistic models, consider the dependencies between the different variables during the evolution. In other words, CMA and GaA are good at handling the inter-correlations among the problem variables.

In this paper, we propose a parameter adaptation technique based on GaA to manage the dependencies between the two parameters (mutation scale factor, *F* and the crossover probability, *CR*) considered. The proposed adaptation technique is used to improve the performance of DE algorithm and is referred to as GaAPADE.

The reminder of this paper is organized as follows: Section II presents a literature survey on 1) DE and different adaptive DE variants, and 2) Gaussian Adaptation. Section III presents the proposed GaAPADE algorithm. Section IV presents the experimental results and discussions while Section V concludes the paper.

II. LITERATURE REVIEW

A. Differential Evolution

Differential Evolution (DE) is a real-coded global optimization algorithm over continuous spaces [19]. DE, a parallel direct search method, utilizes *NP* *D*-dimensional parameter vectors, $\mathrm{X}_{i,G} = \{x^1_{i,G},...,x^D_{i,G}\}, i = 1,...,NP$ to encode the candidate solutions. The initial set of parameter vectors referred to as population are uniformly sampled within the search space constrained by the minimum and maximum parameter bounds $\mathrm{X}_{\min} = \{x^1_{\min},...,x^D_{\min}\}$ and $\mathrm{X}_{\max} = \{x^1_{\max},...,x^D_{\max}\}$.

1) Mutation operation: During every generation *G*, corresponding to each individual $\mathbf{X}_{i,G}$ in the current population, referred to as target vector, a mutant vector $\mathbf{V}_{i,G}$ is produced by the mutation operation. The most commonly employed mutation strategies are:

"DE/best/1" [20] $\mathrm{V}_{i,G} = \mathrm{X}_{best,G} + F.(\mathrm{X}_{r^i_1,G} - \mathrm{X}_{r^i_2,G})$ (1)

"DE/best/2 [20] $\mathrm{V}_{i,G} = \mathrm{X}_{best,G} + F.(\mathrm{X}_{r^i_1,G} - \mathrm{X}_{r^i_2,G}) + F.(\mathrm{X}_{r^i_3,G} - \mathrm{X}_{r^i_4,G})$ (2)

"DE/rand/1" [20]: $\mathrm{V}_{i,G} = \mathrm{X}_{r^i_1,G} + F.(\mathrm{X}_{r^i_2,G} - \mathrm{X}_{r^i_3,G})$ (3)

"DE/rand/2"[13] $\mathrm{V}_{i,G} = \mathrm{X}_{r^i_1,G} + F.(\mathrm{X}_{r^i_2,G} - \mathrm{X}_{r^i_3,G}) + F.(\mathrm{X}_{r^i_4,G} - \mathrm{X}_{r^i_5,G})$ (4)

"DE/current-to-rand/1" [21]

$\mathrm{U}_{i,G} = \mathrm{X}_{i,G} + K.(\mathrm{X}_{r^i_1,G} - \mathrm{X}_{i,G}) + F.(\mathrm{X}_{r^i_2,G} - \mathrm{X}_{r^i_3,G})$ (5)

The indices $r^i_1, r^i_2, r^i_3, r^i_4, r^i_5$ are mutually exclusive integers randomly generated within the range [1, *NP*] and are different from the index *i*. The scale factor *F* is a positive value for scaling the difference vector while *K* is randomly chosen within the range [0, 1]. $\mathbf{X}_{best,G}$ is the individual vector with the best fitness value in the population at generation *G*.

2) Crossover operation: After mutation, crossover operation is applied to each pair of the target vector $\mathbf{X}_{i,G}$ and its corresponding mutant vector $\mathbf{V}_{i,G}$ to generate a trial vector $\mathbf{U}_{i,G}$. In DE, the most commonly used crossover is the binomial (uniform) crossover defined as follows [19]:

$$u^j_{i,G} = \begin{cases} v^j_{i,G} & \text{if } (rand_j[0,1] \le CR) \text{ or } (j = j_{rand}) \\ x^j_{i,G} & \text{otherwise} \end{cases} \quad j = 1,\ 2,\ \ldots,\ D \quad (6)$$

In (6), the crossover rate *CR* is a user-specified constant within the range [0, 1] while j_{rand} is a randomly chosen integer in the range [1, *D*].

3) Selection operation: After the crossover, the trial vectors are evaluated to obtain the objective function and selection operation is performed. The objective function value of each trial vector $f(\mathbf{U}_{i,G})$ is compared to that of its corresponding target vector $f(\mathbf{X}_{i,G})$ in the current population. If the trial vector is better than the corresponding target vector, the trial vector will replace the target vector and enter the population of the next generation. In a minimization problem, the selection operation can be expressed as follows:

$$\mathrm{X}_{i,G+1} = \begin{cases} \mathrm{U}_{i,G}, & \text{if } f(\mathrm{U}_{i,G}) \le f(\mathrm{X}_{i,G}) \\ \mathrm{X}_{i,G}, & \text{otherwise} \end{cases} \quad (7)$$

In DE, mutation, crossover and selection are repeated generation after generation until a termination criterion is satisfied. The algorithmic description of DE is summarized in Table I.

TABLE I: Differential Evolution Algorithm

Step 1 Set the generation number $G = 0$, and randomly initialize a population of *NP* individuals.

Step 2 WHILE stopping criterion is not satisfied DO

Step 2.1 ***Mutation***

Step 2.2 ***Crossover***

Step 2.3 ***Selection***

Step 2.4 ***Increment the generation count*** $G = G + 1$

Step 3 **END WHILE**

Recently, DE has attracted much attention and has been successfully applied to solve many real-world problems [1-6]. However, the performance of the conventional DE algorithm depends on the chosen mutation strategy and the associated control parameters. In addition, as complexity of the optimization problem increases the performance of DE algorithm becomes more sensitive to the strategy and the associated parameter values [7]. Therefore, inappropriate choice of mutation and crossover strategies and their associated parameters may lead to premature convergence, stagnation or wastage of computational resources [7, 14, 22-24]. In literature, various empirical guidelines were suggested for choosing the appropriate strategies and control parameter settings depending on the characteristics of the

optimization problems [7, 19, 20, 25, 26]. However, depending on the complexity of the optimization problem, choosing an appropriate mutation strategy and control parameters is not straight forward due to the complex interaction of control parameters with the DE's performance [11]. In addition, the manual setting and/or tuning of DE strategies and parameters based on the guidelines result in various conflicting conclusions, which lack sufficient justifications. Therefore, to avoid the tuning of parameters by trial-and-error procedure, various adaptive techniques have been proposed [14, 22, 27-29].

Among the three parameters (*NP*, *F* and *CR*), most of the parameter adaptive techniques except [30], set the population size (*NP*) to a predefined value based on the dimensionality of the problem. In [13], a self-adaptive DE algorithm (SaDE) was proposed in which the mutation strategies and the respective control parameter (*CR*) are self-adapted based on their previous experiences of generating promising solutions. The scale factor, *F* was randomly generated with a mean and standard deviation of 0.5 and 0.3 respectively. In [12], the authors introduced a self-adaptation scheme (SDE) in which *CR* is generated randomly for each individual using a normal distribution *N*(0.5,0.15), while scale factor *F* is adapted analogous to the adaptation of crossover rate *CR* in [27]. Therefore, SaDE [13] considers the adaptation of crossover probability (*CR*) only while SDE [12] considers the adaptation of scale factor (*F*) only.

In [28], the authors proposed FADE in which the control parameters *F* and *CR* are adapted based on fuzzy logic controllers whose inputs are the relative function values and individuals of successive generations. FADE outperformed the conventional DE on higher dimensional problems. In [11], a self-adaptation scheme (JDE) was proposed, in which control parameters *F* and *CR* are encoded into the individuals and are adjusted depending on the parameters τ_1 and τ_2. In JDE, *F* and *CR* are initially assigned to 0.5 and 0.9, respectively. In the consecutive iterations, *F* and *CR* are reinitialized if a uniformly generated random number *rand* is less than τ_1 and τ_2 respectively.

Among the different adaptive DE variants, adaptive differential evolution proposed in [16], referred to as JADE, is good in terms of convergence speed and robustness on a variety of optimization problems. JADE [16] implements a mutation strategy "DE/current-to-*p*best" as a generalization to the classic "DE/current-to-best" strategy. Unlike the classic mutation strategy which uses the current best individual, "DE/current-to-*p*best" utilizes the information present in *p* fitter individuals of the current population. The use of multiple solutions helps in balancing the greediness of the mutation and the diversity of the population. In JADE, the control parameters (*F* and *CR*) are updated in an adaptive manner in order to alleviate the trial and error search. In JADE, using the "DE/current-to-*p*best", a mutation vector corresponding to the individual X_i in generation *G* is generated as:

$$V_{i,G} = X_{i,G} + F_i(X^{p}_{best,G} - X_{i,G}) + F_i(X_{r1,G} - X_{r2,G}) \quad (8)$$

where $X_{r1,G}$, $X_{r2,G}$ and $X^{p}_{best,G}$ are selected from the current population. At each generation, the scale factor F_i and crossover probability CR_i of each individual X_i is independently generated as

$$F_i = \text{randc}_i(\mu_F, 0.1) \quad (9)$$

$$CR_i = \text{randn}_i(\mu_{CR}, 0.1) \quad (10)$$

As shown in eqns. (9) and (10), the parameters *F* and *CR* corresponding to each individual are sampled using Cauchy and Normal distributions, respectively. Then mean values μ_F and μ_{CR} are initialized to 0.5 and are updated at the end of each generation as

$$\mu_F = (1-c).\mu_F + c.\text{mean}_L(S_F) \quad (11)$$

$$\mu_{CR} = (1-c).\mu_{CR} + c.\text{mean}_A(S_{CR}) \quad (12)$$

where *c* is a positive constant between 0 and 1. The terms $\text{mean}_A(.)$ and $\text{mean}_L(.)$ denote the arithmetic mean and Lehmer mean [16], respectively. S_F and S_{CR} denote the sets of mutation factors and crossover probabilities, respectively that produced successful trial vectors in the previous generation.

During the past decade, hybridization of EAs has gained significance, due to ability to complement each other's strengths and overcome the drawbacks of the individual algorithms. In [31], the authors proposed a DE parameter adaptation technique based on harmony search (HS) algorithm in which a group of DE control parameter combinations are randomly initialized. The randomly initialized DE parameter combinations form the initial harmony memory (HM) of the HS algorithm. Each combination of the parameters present in the HM is evaluated by testing on the DE population during the evolution. Based on the effectiveness of the DE parameter combinations present in HM, the HS algorithm evolves the parameter combinations. At any given point of time during the evolution of the DE population, the HM contains an ensemble of DE parameters that suits the evolution process of the DE population.

B. Gaussian Adaptation

Estimation of Distribution Algorithms (EDAs) such as Covariance Matrix Adaptation (CMA) [17] and Gaussian Adaptation (GaA) [18] belong to the class of optimization algorithms that rely on probabilistic models and do not use variation operators such as crossover or mutation. In EDAs, the most promising solutions of the last generation are selected and a probability distribution model is built using the information provided by the promising solutions. The solutions in the next solution are sampled from the built model.

EDAs rely on the iterative random sampling and updating the probability distribution model in order to approximate the desired result. Therefore, the process in which the random samples are generated plays a crucial role. In continuous spaces, typical EDAs employ a multivariate Gaussian distribution as the probability density model [32]. Continuous optimization methods, such as, GaA [18], and Evolution

Strategies (ES) [17] use Gaussian sampling to generate candidate solutions from the target distribution and evaluates the target distribution at these sample points.

Covariance Matrix Adaptation (CMA-ES) [6] and GaA algorithm [18] constantly adapt the covariance matrix of the sampling distribution based on the previously accepted samples. In CMA-ES covariance adaptation is employed to increase the likelihood of generating successful mutations while GaA adapts the covariance to maximize the entropy of the search distribution under the constraint that acceptable search points are found with a predefined, fixed hitting probability.

Gaussian Adaptation (GaA) is a stochastic process that adapts a Gaussian distribution to a region or set of feasible points in parameter space. As a result of the adaptation, GaA becomes a maximum dispersion process extending the sampling over the largest possible volume in parameter space while keeping the probability of finding feasible points at a suitable level. GaA is based on the principle of maximum entropy and tries to maximize the entropy H of a multivariate Gaussian distribution $N(\mathbf{m},\mathbf{C})$ given the mean ($\mathbf{m}$) and the covariance ($\mathbf{C}$) information.

$$H = \log(\sqrt{(2\pi e)^D \det(C)}) \tag{13}$$

From eq. (13), it can be observed that the entropy can be maximized by maximizing the determinant of the covariance matrix.

The GaA algorithm starts with mean of a multivariate Gaussian distribution ($\mathbf{m}^{(0)}$) and an initial point ($\mathbf{x}^{(0)}$). In iteration (g + 1), a new solution is sampled as:

$$x^{(g+1)} = m^{(g)} + r^{(g)} Q^{(g)} \eta^{(g)} \tag{14}$$

where $\eta^{(g)} \sim N(\mathbf{0}, \mathbf{I})$. $\mathbf{Q}^{(g)}$ is the normalized square root of $\mathbf{C}^{(g)}$ and is obtained by following decomposition.

$$C^{(g)} = (r.Q^{(g)})(r.Q^{(g)})^T = r^2 (Q^{(g)})(Q^{(g)})^T \tag{15}$$

where r is the scalar *step size*.

In order to minimize a real-valued objective function $f(\mathbf{x})$, GaA uses a fitness dependent acceptance threshold c_T which is monotonically lowered until some convergence criteria are met. If the objective value of the newly sampled solution in eq. (14) is less than c_T, then the mean ($\mathbf{m}$), covariance ($\mathbf{C}$) and the scale factor ($\boldsymbol{r}$) are updated as follows

$$m^{(g+1)} = (1 - \frac{1}{N_m}) m^{(g)} + \frac{1}{N_m} x^{(g+1)} \tag{16}$$

$$C^{(g+1)} = (1 - \frac{1}{N_C}) C^{(g)} + \frac{1}{N_C} (\Delta x)(\Delta x)^T \tag{17}$$

$$r^{(g+1)} = f_e \,.\, r^{(g)} \tag{18}$$

where
$f_e > 1$ is the expansion factor.
N_m and N_C are the weighting factors
$\Delta\mathbf{x} = (\mathbf{x}^{(g+1)} - \mathbf{x}^{(g)})$

If the objective value of the newly sampled solution $x^{(g+1)}$ is greater than the threshold then the mean and covariance are not adapted but the step size is reduced as

$$r^{(g+1)} = f_c \,.\, r^{(g)} \tag{19}$$

where $f_c < 1$ is the contraction factor.

In order to use GaA for optimization, the acceptance threshold cT is continuously lowered as follows.

$$c_T^{(g+1)} = (1 - \frac{1}{N_T}) c_T^{(g)} + \frac{1}{N_T} f(x^{(g+1)}) \tag{20}$$

where N_T is the weighting factor. The fitness-dependent update of c_T makes the algorithm invariant to the linear transformations in the objective function.

TABLE III: Gaussian Adaptation Algorithm

Step 1 Set generation number $G = 0$. Initialize **m**, **C**, ***r*** and c_T.

Step 2 WHILE stopping criterion is not satisfied DO

Step 2.1 *Sample a new solution using eq. (14)*

Step 2.2 *Evaluate and Check if the objective value of the newly sampled solution is less the threshold* c_T

Step 2.3 *Update m, C, r and* c_T

Step 2.4 *Increment the generation count G = G + 1*

Step 3 **END WHILE**

III. Gaussian Adaptation based parameter adaptation for differential evolution (GAPADE)

As highlighted in the previous section, depending on the nature of problem (unimodal or multimodal) and available computation resources, different optimization problems require different mutation and crossover strategies combined with different parameter values to obtain optimal performance. In addition, to solve a specific problem, different mutation and crossover strategies with different parameter settings may be better during different stages of the evolution than a single set of strategies with unique parameter settings as in the conventional DE. Motivated by these observations, many adaptive and self-adaptive parameter adaptive techniques have been proposed [10-16]. However, most of the adaptive techniques try to adapt the control parameters (F and CR) individually. For instance, in JADE [16], the mutation factors and crossover probabilities are evolved based on their historical record of success. F and CR

values corresponding to the individuals in the current generation are generated from a corresponding mean values using Cauchy and Gaussian distributions, respectively. After the selection process, the F and CR values that were able to produce successful trial vectors are collected. Then the respective mean values of F and CR are updated using Lehmer and arithmetic means respectively. In other words, the F and CR are generated (see eq. (9) and (10)) and adapted (see eq. (11) and (12)) individually. Therefore, JADE does not consider the inter-correlation between the two parameters.

However, in [10], it has been demonstrated that performance of DE depends on the combination of F and CR. In other words, the parameters F and CR on which the performance of DE depends are inter-correlated. Therefore, adapting the two parameters individually may not result in the optimal performance of the DE algorithm.

In this paper, we present a parameter adaptation technique which considers the inter-correlation between the two parameters. The parameters evolve based on the Gaussian adaptation process which is used for parameter optimization.

As most of the adaptation algorithms, the proposed GaAPADE adapts the scale factor F and the crossover probability CR which mainly affect the performance of DE. In GaAPADE, we employ GaA on the bi-dimensional continuous space composed by F and CR. Therefore, the data structures employed by GaAPADE are the mean vector **m** and the covariance matrix **C**. The mean vector (**m**) comprises of the mean values of F and CR while the covariance matrix (**C**) comprises the inter-dependencies between the two parameters.

As in JADE [16], every DE individual is assigned with a personal version of the parameters, i.e. there is a couple F_i, CR_i for each individual i sampled using eq. (14). In other words, every time that these parameters are needed (for mutation and crossover in DE), they are sampled from the multivariate Gaussian distribution identified by **m** and **C**. In the current work, the mean vector **m** is initialized to [0.5, 0.5] and covariance matrix (**C**) is set to an identity matrix.

During every generation of the DE evolution, the F_i and CR_i values corresponding to the individuals in the population are generated using the mean (**m**) and the covariance matrix (**C**) using equation (14). Each individual in the DE algorithm uses the F_i and CR_i values to produce the mutation vectors and consequently trial vectors. The combination of F_i and CR_i values that resulted in an offspring that produces maximum improvement is used to update the mean (**m**) and the covariance (**C**). The continuous updating of **m** and **C** by the parameter combinations that produced better solutions will help the parameter search to move to the regions where more suitable combination of the parameters can be generated. The limits of the F and CR are set to be (0, 1.0] and [0, 1.0], respectively.

TABLE III: Outline of GaAPADE

Step 1 Set the generation count $G = 0$, and randomly initialize a population of NP individuals. Initialize **m**, **C**, ***r*** and c_T.

Step 2 WHILE stopping criterion is not satisfied DO

Step 2.1 *Sample new parameter combinations using eq. (14)*

Step 2.2 *Mutation*

Step 2.3 *Crossover*

Step 2.4 *Selection*

Step 2.5 *Check if the improvement by the best parameter combination is greater than the threshold* c_T

Step 2.6 *Update m, C, r and* c_T

Step 2.7 *Increment the generation count* $G = G + 1$

Step 3 **END WHILE**

IV. Experimental Setup and Results

We evaluated the performance of the proposed GaAPADE algorithm on a set of 30 test problems designed for CEC 2014 [33]. Out of the 30 benchmark problems, problems 1-3 are unimodal functions, 4-16 are multimodal functions, 17-22 are hybrid functions and 22-30 are composite functions. Each of the 30 test problems is scalable. The algorithm is tested on the 10D, 30D, 50D and 100D versions of the test problems. However, in the current version the results of 10D and 30D are presented. The evaluation criteria and testing environment are as follows:

Test Problems: 30 Minimization Problems

Dimensions: D = 10, 30, 50, 100

Search Range: $[-100,100]^D$

Runs / problem: 51

Maximum Number of Function Evaluations: 10000*D (100000 for 10D; 300000 for 30D; 500000 for 50D; 1000000 for 100D)

Termination Criteria: When maximum number of function evaluations is reached.

In the present work, we employ the DE/current-to-*p*best mutation strategy along with the binomial crossover. As mentioned above, the proposed adaptation scheme works in the bi-dimensional parametric space (F and CR). In the proposed algorithm, we initially sample 20D solution vectors, out of which 100 individuals are selected as the population members at the start of every generation. After the generation the solution vectors are replaced. In addition to the parameters of the DE algorithms, the parameters of the GaA algorithm are set to the same values that are proposed in [18].

Tables IV and V show the performances of the proposed GaAPADE algorithm on 10D and 30D versions of the benchmark problems, respectively. The performance of the algorithm on each instance of the test problem is reported in terms of best, worst, median, mean and standard deviation values of the errors between the true optimal value and the obtained objective value.

From the results, we could observe that, for 10D problems, the proposed GaAPADE algorithm can find the global optimal solution for unimodal functions 1-3 with 100% success rate. The algorithm is also able to solve the multimodal function 8 with 100% success rate. In case of multimodal functions 4, 6 and 7, the algorithm is able to reach the optimal solution in only few instances.

In the case of 30D problems, as well the proposed algorithm can solve the unimodal function with100% success rate. In the case of multimodal problems, functions 4, 6 and 7 can be solved during some of runs. However, the algorithm fails to solve the 30D version of the multimodal function 8.

However, the algorithm fails to find the optimal solution over the 51 runs for the 10D and 30D versions of functions 5 and 9 – 30 due to the high multi-modality and parameter dependencies.

The algorithm complexity is calculated for 10D, 30D, 50D and 100D versions of function 18 based on the criteria provided in the technical report [33]. The algorithmic complexity increases with the increase in the problem dimensionality as shown in Table VI. The proposed algorithm is simulated in the Matlab environment and is implemented on system with the following configuration.

System Configuration
Intel ® Core ™ i5-3570 CPU 3.40 GHz
8.00 GB RAM
Windows 7 Enterprise
Language: Matlab

V. CONCLUSION

In this paper, we proposed a Gaussian Adaptation based parameter adaptation technique for DE. In the proposed GaAPADE, GaA works on the bi-dimensional parameter space (*F* and *CR*). The use of a multivariate probability distribution for the learning/sampling of DE parameters allows discovering and managing the parameters inter-correlations in a natural way. The proposed adaptation technique is evaluated on the benchmark problem set designed for CEC 2014.

ACKNOWLEDGMENT

This research was supported by the Original Technology Research Program for Brain Science through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science and Technology (2013034988)

The research was supported by 'Software Convergence Technology Development Program', through the Ministry of Science, ICT and Future Planning (S1002-13-1014).

REFERENCES

[1] R. Joshi and A. C. Sanderson, "Minimal representation multisensor fusion using differential evolution," *IEEE Transactions on Systems, Man, and Cybernetics Part A:Systems and Humans.,* vol. 29, pp. 63-76, 1999.

[2] R. Mallipeddi, J. P. Lie, S. G. Razul, P. N. Suganthan, and C. M. S. See, "Robust Adaptive Beamforming Based on Covariance Matrix Reconstruction for Look Direction Mismatch," *Progress in Electromagnetic Research Letters,* vol. 25, pp. 37-46, 2011.

[3] M. K. Venu, R. Mallipeddi, and P. N. Suganthan, "Fiber Bragg grating sensor array interrogation using differential evolution," *Optoelectronics and Advanced Materials - Rapid Communications,* vol. 2, pp. 682-685, 2008.

[4] S. Das and A. Konar, " Automatic image pixel clustering with an improved differential evolution " *Applied Soft Computing,* vol. 9, pp. 226-236 JAN 2009

[5] R. Storn, "Differential evolution design of an IIR-filter " presented at the IEEE International Conference on Evolutionary Computation 1996.

[6] R. Mallipeddi, S. Jeyadevi, P. N. Suganthan, and S. Baskar, "Efficient constraint handling for optimal reactive power dispatch problems," *Swarm and Evolutionary Computation,* vol. 5, pp. 28-36, 2012.

[7] R. Gämperle, S. D. Müller, and P. Koumoutsakos, "A Parameter Study for Differential Evolution," presented at the Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation, Interlaken, Switzerland, 2002.

[8] J. Liu and J. Lampinen, "On setting the control parameter of the differential evolution method," in *Proc. 8th Int, Conf. Soft Computing (MENDEL 2002)*, 2002, pp. 11-18.

[9] Z. Jingqiao and A. C. Sanderson, "An approximate gaussian model of Differential Evolution with spherical fitness functions," in *Evolutionary Computation, 2007. CEC 2007. IEEE Congress on*, 2007, pp. 2220-2228.

[10] R. Mallipeddi, P. N. Suganthan, Q. K. Pan, and M. F. Tasgetiren, "Differential evolution algorithm with ensemble of parameters and mutation strategies," *Applied Soft Computing,* vol. 11, pp. 1679-1696, 2011.

[11] J. Brest, S. Greiner, B. Boscovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *IEEE Transactions on Evolutionary Computation,* vol. 10 pp. 646-657 DEC 2006 2006.

[12] M. G. H. Omran, A. Salman, and A. P. Engelbrecht, "Self-adaptive Differential Evolution," in *Computational Intelligence and Security, PT 1, Proceedings Lecture Notes in Artificial Intelligence*, 2005, pp. 192-199.

[13] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential Evolution Algorithm with Strategy Adaptation for Global Numerical Optimization," *IEEE Transactions on Evolutionary Computation,* vol. 13, pp. 398-417, April 2009.

[14] D. Zaharie, "Control of Population Diversity and Adaptation in Differential Evolution Algorithms," *In: Proceedings of the 9th International Conference on Soft Computing, Brno 41-46,* 2003.

[15] J. Tvrdik, "Adaptation in differential evolution: A numerical comparison," *Applied Soft Computing,* vol. 9, pp. 1149-1155 JUN 2009.

[16] J. Zhang, "JADE: Adaptive Differential Evolution with Optional External Archive," *IEEE Transactions on Evolutionary Computation,* vol. 13, pp. 945-958, October 2009.

[17] A. Auger and N. Hansen, "A Restart CMA Evolution Strategy With Increasing Population Size," presented at the IEEE Congress on Evolutionary Computation, 2005.

[18] C. Müller and I. Sbalzarini, "Gaussian Adaptation Revisited – An Entropic View on Covariance Matrix Adaptation," in *Applications of Evolutionary Computation*. vol. 6024, C. Chio, S. Cagnoni, C. Cotta, M. Ebner, A. Ekárt, A. Esparcia-Alcazar, C.-K. Goh, J. Merelo, F. Neri, M. Preuß, J. Togelius, and G. Yannakakis, Eds., ed: Springer Berlin Heidelberg, 2010, pp. 432-441.

[19] R. Storn and K. Price, "Differential Evolution - A Simple and Efficient Heuristic for Global Optimization over Continuous Spaces," *Journal of Global Optimization,* vol. 11, pp. 341-359, 1997.

[20] R. Storn, "On the Usage of Differential Evolution for Function Optimization " presented at the Biennial Conference of the North American Fuzzy Information Processing Society (NAFIPS), Berkeley, 1996.

[21] A. Iorio and X. Li, "Solving Rotated Multi-objective Optimization Problems Using Differential Evolution," presented at the Australian Conference on Artificial Intelligence, Cairns, Australia., 2004.

[22] S. Das, A. Konar, and U. K. Chakraborty, "Two Improved Differential Evolution Schemes for Faster Global Search," *Proceedings of the 2005 conference on Genetic and evolutionary computation,* pp. 991-998, 2005.

[23] J. Lampinen and I. Zelinka, "On Stagnation of the Differential Evolution Algorithm," *Proceedings of MENDEL 2000, 6th International Mendel Conference on Soft Computing,* pp. 76-83, 2000.
[24] K. V. Price, R. M. Storn, and J. A. Lampinen, Eds., *Differential Evolution : A Practical Approach to Global Optimization* (Natural Computing Series. Berlin: Springer, 2005, p.^pp. Pages.
[25] R. Storn and K. Price, "Differential Evolution : A Simple Evolution Strategy for Fast Optimization," *Dr. Dobb's Journal,* vol. 22, pp. 18-24, April 1997.
[26] J. Rönkkönen, S. Kukkonen, and K. V. Price, "Real-parameter optimization with differential evolution," in *2005 IEEE Congress on Evolutionary Computation, IEEE CEC 2005. Proceedings*, 2005, pp. 506-513.
[27] H. A. Abbass, "The Self-Adaptive Pareto Differential Evolution Algorithm," *Proceedings of the IEEE Congress on Evolutionary Computation (CEC2002),* vol. 1, pp. 831-836, 2002.
[28] J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algorithm," *Soft Computing,* vol. 9, pp. 448-462, Jun 2005.
[29] D. Zaharie and D. Petcu, "Adaptive Pareto differential evolution and its parallelization," in *Proc. of 5th International Conference on Parallel Processing and Applied Mathematics*, Czestochowa, Poland, 2003, pp. 261-268.
[30] J. Teo, "Exploring dynamic self-adaptive populations in differential evolution," *Soft Computing,* vol. 10, pp. 673-686, 2006.
[31] R. Mallipeddi, "Harmony Search Based Parameter Ensemble Adaptation for Differential Evolution," *Journal of Applied Mathematics,* vol. 2013, p. 12, 2013.
[32] B. Yuan and M. Gallagher, "Experimental Results for the Special Session on Real-Parameter Optimization at CEC 2005: A Simple, Continuous EDA," presented at the IEEE Congress on Evolutionary Computation, 2005.
[33] J. J. Liang, B. Y. Qu, and P. N. Suganthan, "Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization," Nanyang Techological University, Singapore2014.

Table IV. Results for 10D

Function	Best	Worst	Median	Mean	Std.
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	3.48E+01	3.48E+01	2.82E+01	1.35E+01
5	7.24E+00	2.00E+01	2.00E+01	1.96E+01	1.85E+00
6	0	1.83E+00	0	1.40E-01	3.64E-01
7	0	2.22E-02	1.92E-03	3.53E-03	4.41E-03
8	0	0	0	0	0
9	1.86E+00	5.18E+00	0	3.42E+00	8.75E-01
10	2.37E-02	1.21E+01	3.40E+00	6.00E-01	2.34E+00
11	2.40E+00	3.85E+02	1.38E+02	1.60E+02	1.10E+02
12	6.43E-02	2.37E-01	1.50E-01	1.47E-01	4.09E-02
13	3.05E-02	9.67E-02	6.57E-02	6.54E-02	1.34E-02
14	3.92E-02	2.06E-01	8.86E-02	9.32E-02	3.22E-02
15	3.78E-01	7.99E-01	5.76E-01	5.83E-01	9.74E-02
16	9.00E-01	2.50E+00	2.01E+00	2.01E+00	3.03E-01
17	2.64E+00	3.01E+01	8.50E+00	9.09E+00	4.79E+00
18	1.50E-02	5.45E-01	1.85E-01	2.09E-01	1.49E-01
19	8.90E-02	1.07E+00	2.20E-01	2.60E-01	1.52E-01
20	1.63E-01	8.44E-01	4.25E-01	4.33E-01	1.48E-01
21	2.89E-02	1.16E+00	4.50E-01	4.52E-01	2.41E-01
22	7.41E-01	5.66E+00	3.32E+00	3.18E+00	1.14E+00
23	3.29E+02	3.29E+02	3.29E+02	3.29E+02	0
24	1.00E+02	1.11E+02	1.09E+02	1.08E+02	2.08E+00
25	1.07E+02	2.01E+02	2.01E+02	1.68E+02	4.11E+01
26	1.00E+02	1.00E+02	1.00E+02	1.00E+02	1.74E-02
27	1.44E+00	4.00E+02	2.10E+00	9.56E+01	1.63E+02
28	3.69E+02	4.81E+02	3.69E+02	3.84E+02	3.36E+01
29	2.22E+02	2.24E+02	2.22E+02	2.22E+02	6.81E-01
30	4.54E+02	5.50E+02	4.62E+02	4.68E+02	1.90E+01

Table V. Results for 30D

Function	Best	Worst	Median	Mean	Std.
1	0	0	0	0	0
2	0	0	0	0	0
3	0	0	0	0	0
4	0	1.95E-02	0	3.82E-04	2.73E-03
5	2.00E+01	2.00E+01	2.00E+01	2.00E+01	4.26E-03
6	0	6.39E+00	1.79E-06	5.92E-01	1.06E+00
7	0	0	0	0	0
8	1.15E-03	4.33E+00	1.32E+00	1.28E+00	8.72E-01
9	1.18E+01	3.68E+01	1.72E+01	1.70E+01	3.90E+00
10	2.47E+00	4.53E+01	7.79E+00	1.00E+01	7.38E+00
11	1.20E+03	2.58E+03	1.99E+03	1.96E+03	3.41E+02
12	2.63E-02	4.99E-01	2.02E-01	2.10E-01	1.08E-01
13	8.55E-02	1.86E-01	1.40E-01	1.43E-01	2.14E-02
14	1.30E-01	2.65E-01	2.04E-01	2.03E-01	3.08E-02
15	2.51E+00	4.11E+00	3.03E+00	3.14E+00	4.14E-01
16	8.18E+00	1.08E+01	9.95E+00	9.83E+00	5.53E-01
17	3.98E+01	5.56E+02	1.60E+02	1.70E+02	1.19E+02
18	3.30E+00	2.31E+01	7.23E+00	8.77E+00	4.11E+00
19	2.36E+00	5.60E+00	3.95E+00	3.96E+00	7.67E-01
20	1.93E+00	1.84E+01	5.45E+00	5.95E+00	3.16E+00
21	1.47E+00	2.83E+02	1.35E+02	1.17E+02	8.26E+01
22	2.24E+01	1.57E+02	3.58E+01	7.40E+01	5.68E+01
23	3.15E+02	3.15E+02	3.15E+02	3.15E+02	0
24	2.22E+02	2.26E+02	2.24E+02	2.24E+02	8.64E-01
25	2.03E+02	2.03E+02	2.03E+02	2.03E+02	7.19E-02
26	1.00E+02	1.00E+02	1.00E+02	1.00E+02	2.40E-02
27	3.00E+02	4.01E+02	3.00E+02	3.19E+02	3.46E+01
28	7.68E+02	8.94E+02	8.46E+02	8.38E+02	2.96E+01
29	7.13E+02	7.35E+02	7.16E+02	7.17E+02	3.99E+00
30	4.55E+02	4.47E+03	1.32E+03	1.52E+03	8.02E+02

Table VI. Computational Complexity

Dimension	T_0	T_1	T'_2	$(T'_2 - T_1)/T_0$
$D = 10$	1.248E-01	2.3400E-01	2.4087E+00	1.7425E+01
$D = 30$	1.248E-01	9.3600E-01	7.2728E+00	5.0776E+01
$D = 50$	1.248E-01	2.0748E+00	1.6514E+01	1.1569E+02
$D = 100$	1.248E-01	6.5676E+00	2.3887E+01	1.3878E+02