数理解析研究所講究録
第1773巻 2012年 198-206

Differential Evolution Combined with Automatic Termination

Bun Theang Ong and Masao Fukushima *

Abstract

Evolutionary Algorithms (EAs) provide a very powerful tool for solving optimization problems. In the last decades, numerous studies have been focusing on improving the performance of EAs. However, there is a lack of studies that tackle the question of the termination criteria. Indeed, EAs still need termination criteria prespecified by the user. In this paper, we propose to combine the Differential Evolution (DE) method with the "Gene Matrix" (GM) in order to equip DE with an automatic termination criterion without resort to predefined conditions. A local search method is also used to refine the best solution obtained upon termination and to determine, via the GM, the depth of the search accordingly to the problem at hand. We name this algorithm "Automatically Terminated Differential Evolution" (ATDE). Numerical experiments show that the solutions obtained with the automatic termination are competitive and reliable, under a reduced number of objective function evaluations.

Keywords—Global Optimization, Differential Evolution, Termination Criteria, Gene Matrix.

1 Introduction

Differential Evolution (DE) is a very competitive evolutionary algorithm for solving real-parameter optimization problems that first appeared in 1995 in a technical report written by R. Storn and K. Price (Storn and Price, 1995). Since then, DE has attracted particular attention and yielded a significant number of research articles.

Practitioners particularly appreciate the relative simplicity to implement and efficiency for many optimization problems in real-world applications (Price et al., 2005; Joshi and Sanderson, 1999; Zhang et al., 2008). Another advantage of DE compared with other EAs is the reduced number of control parameters (three for the classical DE, namely, the population size NP, the crossover rate CR and the scaling factor F). A number of papers in the literature extensively study the influence of these parameters on the performance of the algorithm (Gämperle et al., 2002).

As other EAs, DE is population-based and uses common features of EAs such as recombination and selection operators. However, one of the distinctive features of DE lies in the fact that it exploits the information about differences between trial solutions, the latter being identified as *parameter vectors*, to explore the search space. Basically, in DE, the mutation operator considers two parameter vectors and adds a weighted difference vector to create a third parameter vector. Different flavors of the mutation operator have been investigated. Coupled with different kinds of crossover operators, numerous DE schemes can be designed (Das and Suganthan, 2011).

In this work, we emphasize how to let the search determine a proper termination instant. There is not much work yet in the research of EAs dealing with the question of the termination criteria. Nevertheless, it is recognized that in many real world applications, saving computational resources is extremely important. There are many applications that can benefit from an automatic termination criterion. For instance, in *evolutionary testing* (O'Sullivan et al., 1998; McMinn, 2004), temporal errors are detected whenever outputs are produced too early or if the computational time is too lengthy. This shows the importance of keeping a good balance between exploration and exploitation before deciding a proper termination instant. Multi-start methods may

*The authors are with the Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto University, Kyoto 606-8501, JAPAN (tel: +81-75-753-5519; fax: +81-75-753-4756; email: ong_bt@amp.i.kyoto-u.ac.jp; fuku@amp.i.kyoto-u.ac.jp).

also benefit from automatic termination criteria. Indeed, by using adequate termination conditions, it is possible to reduce the cost of generating candidate solutions and allow more trials with a fixed budget. There are only a few recent works on termination criteria for EAs (Giggs et al., 2006; Kwok et al., 2007; Jain et al., 2001). In (Giggs et al., 2006), the problem characteristics are empirically studied in an attempt to determine the maximum number of generations. In (Kwok et al., 2007), statistics are used to terminate the search when it is estimated that no further improvement in terms of solution quality can be expected. In (Jain et al., 2001), eight termination criteria are studied with clustering techniques that examine the distribution of individuals in the search space at a given generation.

In this paper, we combine DE with a mechanism that allow the search to accelerate its convergence and to determine automatically an adequate termination instant without prior knowledge about the problem. The mechanism is called the Gene Matrix (GM) (Hedar et al., 2007; Ong and Fukushima, 2011). The GM is a matrix that represents subranges of the possible values of each variable. It gives an indication on the distributions of the variables over the search range. This information is used to provide the search with new diverse solutions and let the search know how far the exploration process has been performed. The number of subranges determines the depth of the search and the cost of the algorithm. This parameter is automatically determined by performing a preliminary "estimation" of the landscape of the problem to be solved. This is done at the beginning of the search by a local search method. To further accelerate the search process, the local search method is also used to improve the best candidate solution obtained upon GM termination.

The performance of ATDE is evaluated through numerical experiments on a set of 15 test problems, taken from the CEC 2005 real-parameter optimization contest (Suganthan et al., 2005), and compared against the classical version of DE as well as the "DE algorithm with ensemble of parameters and mutation strategies" (EPSDE) (Mallipeddi et al., 2011).

The rest of this paper is organized as follows. In Section 2, the basic concepts of DE are reviewed. In Section 3, the new mechanisms of ATDE are described before stating its formal algorithm. Section 4 provides the experiments and comparative study conducted and discusses the results. A summary with conclusions and future work is provided in Section 5.

2 Basic Concepts of Differential Evolution

ATDE seeks an optimal or near-optimal solution of the nonconvex optimization problem

$$\min_{x \in X} f(x), \tag{1}$$

where f is a real-valued function defined on the search space $X \subseteq R^D$ with variables $x \in X$. A lot of EAs and other heuristics have been proposed to deal with this problem, see for instance (Hansen, 2006; Hedar and Fukushima, 2006; Ong and Fukushima, 2011) and references therein. As an EA, ATDE is meant to deal with any type of problems because we do not make any assumption about the landscape of the problem being optimized. Hence, we do not assume the differentiability, or even the continuity of the objective function.

In DE, a population is represented by D-dimensional vectors $x^i, i \in \{1, \ldots, NP\}$, where D is the dimension of the problem and NP is the population size. Like other EAs, DE starts with an initial population that is generated randomly. It is followed by the mutation, crossover and selection operators. We emphasize that, in DE, the mutation, crossover and selection operations are mutually dependent and are performed for every individual x^i in the population sequentially.

2.1 Mutation

At each generation, the parameter vectors undergo mutation to create new vectors v^i according to the formula

$$v^i = x^{r0} + F \cdot (x^{r1} - x^{r2}), \tag{2}$$

where $r0$, $r1$, $r2$ are distinct integers taken from the set $\{1, 2, \ldots, NP\} \backslash \{i\}$ and F is a positive real constant that represents the mutation factor used to control the effect of the difference vector $(x^{r1} - x^{r2})$. The parameter F is commonly referred to as the amplification factor or scale factor. Depending on the number of difference vectors being considered, different strategies can be implemented.

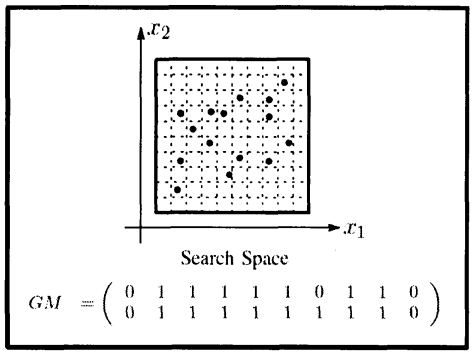

Figure 1: An example of the *Gene Matrix* in R^2.

2.2 Crossover

The crossover operator is called right after the mutation operation. During this process, the vector $u^i = (u_1^i, u_2^i, \ldots, u_D^i) \in R^D$ is generated by

$$u_j^i = \begin{cases} v_j^i, & \text{if } U_j \leq CR \text{ or } j = j_{rand}, \\ x_j^i, & \text{otherwise,} \end{cases} \tag{3}$$

where U_j is a uniform random number from the interval $(0, 1)$, and $CR \in [0, 1]$ is a parameter that represents the crossover rate. For each j and each i, the number U_j is independently generated, and for each i, j_{rand} is a random integer from $[1, D]$.

Equation (3) is in fact the scheme used for what is called the *binomial* crossover. In DE, there are mainly two types of crossovers that can be used; the *binomial* and the *exponential* crossovers (Price et al., 2005). ATDE uses the binomial crossover.

2.3 Selection

During the selection process, the parameter vector u^i generated by the crossover operator competes with the vector x^i according to its value of the function $f(\cdot)$. Specifically, the offspring of x^i is determined by

$$\tilde{x}^i = \begin{cases} u^i, & \text{if } f(u^i) \leq f(x^i), \\ x^i, & \text{otherwise.} \end{cases} \tag{4}$$

The population size thus remains constant and its fitness is assured to never decline.

Those three operators, mutation, crossover and selection, can have many different variants, and when assembled together, they form different kinds of DE that can be classified by the following commonly used notation: DE/x/y/z, where "DE" stands for "Differential Evolution", x denotes the base parameter vector to be perturbed, y represents the number of difference vectors to consider, and z is the type of crossover employed. In this work, we consider the DE/rand/1/bin, which is the variant of DE most commonly used in practice (Qin and Suganthan, 2005). In this case, the mutation operator deals with a randomly selected parameter vector and one weighted difference vector, coupled with the binomial crossover operator.

3 Automatically Terminated Differential Evolution

3.1 Gene Matrix and Termination

The range of each component of the parameter vector is divided into several subranges in order to check the diversity of the variable. The GM (Hedar et al., 2007; Ong and Fukushima, 2011) is a matrix initialized as the $D \times m$ zero matrix, where D is the dimension of the problem and m is the number of subranges for each variable. Namely, each entry of the i-th row refers to a subrange of the i-th variable. While the parameter vectors

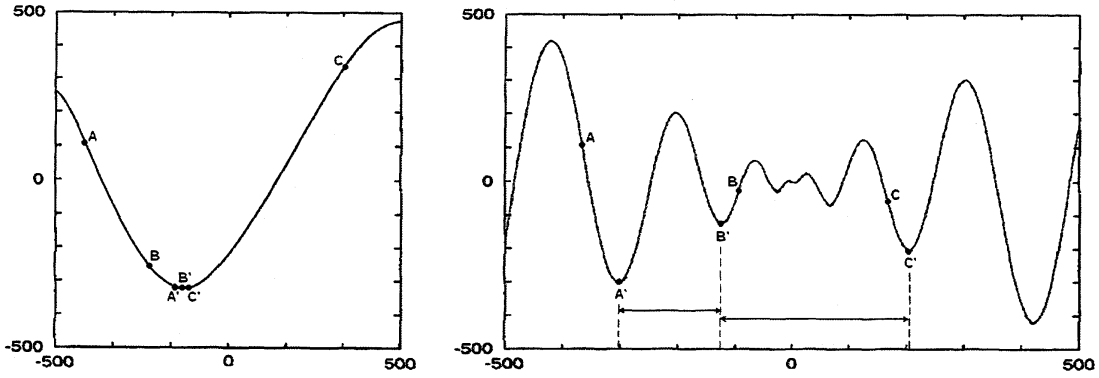

Figure 2: Landscape estimation on an unimodal function (a) and a multimodal function (b).

are exploring the search space, their variables are checked and the corresponding subranges are detected. The visited subranges are then granted with a non-null value in the GM.

Figure 1 shows an example of GM in two dimensions. In the figure, the range of each variable is divided into ten subranges, thus partitioning the search space into a hundred sectors. For the first variable x_1, no parameter vector has been generated inside the subranges 1, 7 and 10. Consequently, GM's values in the $(1, 1)$, $(1, 7)$ and $(1, 10)$ entries are still equal to zero. For the second variable x_2, only the first and the last subranges are still unexplored, hence GM's values in entries $(2, 1)$ and $(2, 10)$ are null. With a full GM, i.e., with no zero entry, the search is considered to have achieved an advanced exploration process and can be stopped. We define the GM completion ratio, referred to as CP, as the number of non-null entries divided by the total number of entries of GM.

"Mutagenesis" is a mechanism that allows some characteristic parameter vectors to "jump" in the search space by modifying the values of some of their variables. Information contained within the GM is used to direct and accelerate the search by specifically generating parameter vectors so that their variables take values within an unvisited area of the search space. ATDE sorts the members of the population of size NP, and then selects the worst N_w ($< NP$) vectors that will participate in the operation. The formal procedure for Mutagenesis is given in Procedure 3.1.

Procedure 3.1 *Mutagenesis*

1. *If there is no zero-position in GM, then return; otherwise, go to Step 2.*
2. *Choose a zero-position (j, k) in GM randomly.*
3. *Update the parameter vector x by setting $x_j = l_j + (k - r)\frac{u_j - l_j}{m}$, with all x_t's $(t \neq j)$ being unchanged, where r is a random number from $(0, 1)$, and l_j, u_j are the lower and upper bounds of the variable x_j, respectively.*
4. *Update GM and return.*

3.2 Landscape Estimation

"Landscape estimation" is a mechanism that allows the search to get an approximative idea about the complexity of the landscape of the problem at hand. To do so, before the search starts, L parameter vectors are generated randomly within the search space. Then, local search steps with limited effort are performed starting from each of the L vectors. The local search steps will "move" the L vectors to new locations in the search space with equal or better objective function values. Afterwards, the distances between each of the new points are computed and the mean distance is calculated. Finally, the obtained value is used to determine the number of subranges m of GM. Figure 2 depicts the processus on an unimodal function and a multimodal function in one dimension. In this figure, three points A, B and C were randomly generated. A local search starting from each of these points led to positions indicated by points A', B' and C', respectively. In the case of Figure 2(a), the example represents an easy function with a local optimum which is also the global optimum in the search space.

Table 1: Parameter setting in ATDE

Parameter	Definition	Value
F	Scale factor	0.3
CR	Crossover rate	0.5
NP	Population size	30
N_w	No. of vectors used by Mutagenesis	4
L	Landscape estimation points	4
CP	GM Completion ratio	90%

The mean distance between points A', B' and C' is relatively small, thus the algorithm deduces that GM will not require a high number of subranges. On the other hand, in the case of Figure 2(b), many optima exist and the local search gets stuck in some local optima. The mean distance between A', B' and C' is larger than in Figure 2(a) and the resulting m will be bigger.

3.3 Intensification

To further enhance the convergence speed, a final intensification process calls a local search method based on the Kelley's modification (Kelley, 1999) of the Nelder-Mead (NM) method (Nelder and Mead, 1965). The local search is initiated from the best parameter vector obtained at the end of the search. ATDE thus behaves like a "Memetic Algorithm" (Moscato, 1999; Le et al., 2009) in order to achieve faster convergence (Ong et al., 2006; Kramer, 2010).

3.4 Formal ATDE Algorithm

The algorithmic parameters with their assigned values are summarized in Table 1. The formal description of ATDE is given in Algorithm 3.2.

Algorithm 3.2 *ATDE Algorithm*

1. ***Initialization.*** *Set values of N_w and (l_j, u_j) for $j = 1, \ldots, D$. Set the scale factor F, crossover rate CR, population size NP and completion ratio CP.*
2. *Generate L points randomly and apply Landscape estimation to infer m.*
3. *Generate an initial population P_0 of size NP. Initialize Gene Matrix GM as the $D \times m$ zero matrix.*
4. *For every individual, repeat sequentially Steps 4.1 to 4.4.*
   - ***4.1.*** *Set the considered parameter vector x^i as the target vector.*
   - ***4.2. Mutation.*** *Generate a vector v^i using Equation (2).*
   - ***4.3. Crossover.*** *Generate a vector u^i using Equation (3). Update GM.*
   - ***4.4. Selection.*** *Compare vector u^i with vector x^i to generate the offspring $\tilde{x}^i$ using Equation (4). Update $x^i := \tilde{x}^i$.*
5. ***Mutagenesis.*** *Alter the N_w worst individuals using Procedure 3.1 and update GM. If GM have reached the completion ratio CP, then go to Step 6. Otherwise, go to Step 4.*
6. ***Intensification.*** *Apply a local search method starting from the best parameter vector.*

4 Numerical Results

4.1 Methodology

A set of 15 benchmark functions, listed in Table 2, provided by the special session on real-parameter optimization in the IEEE Congress on Evolutionary Computations CEC 2005 (Suganthan et al., 2005) was used to compare the performance of the proposed method ATDE against the classical DE (Storn and Price, 1997) and the EPSDE (Mallipeddi et al., 2011) methods in 10 dimensions. They are based on widely used benchmark functions and aim at covering a diverse set of problem properties and difficulties. They are all scalable and most

Table 2: Benchmark functions

f	Function name	Bounds	Global min	Error Tol
f_1	Shifted Sphere	[-100,100]	-450	10^{-6}
f_2	Shifted Schwefel's 1.2	[-100,100]	-450	10^{-6}
f_3	Shifted rotated high conditioned elliptic	[-100,100]	-450	10^{-6}
f_4	Shifted Schwefel's 1.2 with noise in fitness	[-100,100]	-450	10^{-6}
f_5	Schwefel's 2.6 with global optimum on bounds	[-100,100]	-310	10^{-6}
f_6	Shifted Rosenbrock's	[-100,100]	390	10^{-2}
f_7	Shifted rotated Griewank's without bounds	[0,600]	-180	10^{-2}
f_8	Shifted rotated Ackley's with global optimum on bounds	[-32,32]	-140	10^{-2}
f_9	Shifted Rastrigin's	[-5,5]	-330	10^{-2}
f_{10}	Shifted rotated Rastrigin's	[-5,5]	-330	10^{-2}
f_{11}	Shifted rotated Weierstrass	[-0.5,0.5]	90	10^{-2}
f_{12}	Schwefel's 2.13	[-100,100]	-460	10^{-2}
f_{13}	Expanded extended Griewank's + Rosenbrock's	[-3,1]	-130	10^{-2}
f_{14}	Expanded rotated extended Scaffe's	[-100,100]	-300	10^{-2}
f_{15}	Hybrid composition 1	[-5,5]	120	10^{-2}

Table 3: Error mean obtained by each methods after the same Feval as ATDE

		Error mean (SRate)		
f	Feval mean	ATDE	DE	EPSDE
f_1	1.68e+03	9.96e-13 (100%)	7.12e+00 (0%)	3.85e+02 (0%)
f_2	1.91e+03	1.73e-12 (100%)	2.06e+03 (0%)	2.69e+03 (0%)
f_3	6.03e+03	9.47e+01 (84%)	3.15e+06 (0%)	1.58e+06 (0%)
f_4	3.70e+04	2.03e+00 (28%)	2.44e+01 (28%)	6.82e-15 (100%)
f_5	3.64e+04	7.37e+01 (0%)	1.21e+02 (0%)	6.14e+01 (0%)
f_6	6.87e+03	9.60e-01 (72%)	3.26e+01 (0%)	2.22e+02 (0%)
f_7	9.71e+03	1.49e+00 (0%)	7.14e+01 (0%)	7.87e-01 (0%)
f_8	3.66e+04	2.04e+01 (0%)	2.04e+01 (0%)	2.04e+01 (0%)
f_9	1.57e+04	6.78e-01 (64%)	6.48e-01 (52%)	1.08e-03 (96%)
f_{10}	1.28e+04	1.03e+01 (0%)	2.89e+01 (0%)	2.73e+01 (0%)
f_{11}	3.05e+04	7.09e+00 (0%)	8.45e+00 (0%)	8.03e+00 (0%)
f_{12}	3.66e+04	8.76e+01 (60%)	4.84e+02 (0%)	9.51e+02 (0%)
f_{13}	1.70e+04	6.55e-01 (0%)	1.34e+00 (0%)	7.62e-01 (0%)
f_{14}	3.56e+04	3.27e+00 (0%)	3.62e+00 (0%)	3.81e+00 (0%)
f_{15}	3.39e+04	4.64e+01 (76%)	2.04e+02 (4%)	2.17e+02 (0%)

of them are non-separable. The error tolerance levels defined in Table 2 (Error Tol) are used to evaluate if a run is successful or not. A run is considered successful when the fixed error tolerance level is achieved at the end of the search. The success rate, abbreviated as SRate, is defined as the number of successful runs divided by the total number of independent runs, here equal to 25 for all methods. The error is equal to $f(x^{best}) - f(x^*)$, where $f(x^{best})$ represents the best function value obtained by the algorithm and $f(x^*)$ is the known exact global minimum value.

4.2 Comparison with DE

To evaluate the performance of ATDE against the canonical DE, we first let ATDE run until it terminates automatically and then compute its success rates for each function. Afterward, the same amount of objective function evaluations are used as the termination condition. Results are shown in Table 3 in 10 dimensions where the numbers of function evaluations that was required by ATDE and DE are reported as well as the mean of the error and the success rate for each function.

By comparing the results, reported in Table 3, obtained by ATDE and DE after the same amount of Feval, which was here determined by the automatic termination of ATDE, we can see that the performance of ATDE in terms of both accuracy and reliability is much better than the performance of DE. For all problems, ATDE could obtain a solution closer to a global optimum. Actually, DE could not detect a global optimum in most cases, except for problems f_4, f_9 and f_{15}. And yet, the SRate is relatively low. ATDE, on the other hand, could reach the global optima in 8 functions out of 15 with decent SRate.

To further assess whether the new mechanisms of ATDE (GM, Mutagenesis, intensification) improve the

Table 4: Comparison between the error obtained by ATDE after automatic termination and the error obtained by DE after double the Feval required by ATDE

	ATDE		DE	
f	Feval mean	Error mean (SRate)	Feval mean	Error mean (SRate)
f_1	1.68e+03	9.96e-13 (100%)	3.39e+03	6.92e-03 (0%)
f_2	1.91e+03	1.73e-12 (100%)	3.84e+03	3.51e+02 (0%)
f_3	6.03e+03	9.47e+01 (84%)	1.21e+04	1.73e+06 (0%)
f_4	3.70e+04	2.03e+00 (28%)	7.40e+04	1.59e+01 (28%)
f_5	3.64e+04	7.37e+01 (0%)	7.28e+04	8.27e+01 (0%)
f_6	6.87e+03	9.60e-01 (72%)	1.38e+04	8.76e+00 (0%)
f_7	9.71e+03	1.49e+00 (0%)	1.94e+04	7.05e+01 (0%)
f_8	3.66e+04	2.04e+01 (0%)	7.32e+04	2.04e+01 (0%)
f_9	1.57e+04	6.78e-01 (64%)	3.14e+04	3.24e-01 (68%)
f_{10}	1.28e+04	1.03e+01 (0%)	2.56e+04	2.33e+01 (0%)
f_{11}	3.05e+04	7.09e+00 (0%)	6.10e+04	7.67e+00 (0%)
f_{12}	3.66e+04	8.76e+01 (60%)	7.32e+04	1.52e+02 (0%)
f_{13}	1.70e+04	6.55e-01 (0%)	3.40e+04	1.11e+00 (0%)
f_{14}	3.56e+04	3.27e+00 (0%)	7.12e+04	3.45e+00 (0%)
f_{15}	3.39e+04	4.64e+01 (76%)	6.78e+04	2.08e+02 (8%)

canonical DE, we allowed DE to run with double the Feval that was required by ATDE. Results are shown in Table 4. Comparing the results of DE in Tables 3 and 4 reveal that the convergence rate of DE is relatively slow. Although it could obtain more accurate results in some functions (functions f_1, f_2 and f_6), it is not enough to outperform ATDE.

Those results demonstrate that the GM, Mutagenesis and intensification mechanisms successfully improve the DE paradigm by considerably increasing the convergence rate, accuracy and reliability of the algorithm.

4.3 Comparison with EPSDE

In this section, ATDE is compared against a recent state-of-the-art algorithm called EPSDE that uses an ensemble of mutation strategies and parameter values with DE (Mallipeddi et al., 2011). This method makes use of a set of mutation strategies with different parameter settings and thus avoids unique parameter settings as in the classical DE. Since different strategies may become more effective during the evolution process than others depending on the problem, EPSDE keeps a pool of mutation strategies and parameters that have diverse characteristics in an attempt to adapt to different stage of the evolution process and to the nature of a particular problem. In (Mallipeddi et al., 2011), EPSDE is favorably compared to several well-known state-of-the-art DE methods.

Comparison with a state-of-the-art method will allow us to ascertain the benefits that result from the new mechanisms of ATDE. It can be observed in Table 3 that under such considerably restricted number of Feval (around 1,700 Feval for function f_1 and 1,900 Feval for function f_2, for example), even EPSDE cannot reach a global optimum in most cases. However, it is interesting to see that for functions f_4 and f_9, more Feval were available and EPSDE could obtain good results. This demonstrates that the Landscape estimation mechanism, coupled with the GM, can determine a termination instant that avoid unnecessary objective function evaluations for easy functions and confer more effort to harder problems. The accuracy and reliability as well are effectively improved by the use of Mutagenesis and Intensification.

5 Conclusion

In this paper, we have proposed to combine DE with new mechanisms and acceleration elements in order to equip DE with an automatic termination strategy that does not require any prescribed criterion such as the maximum number of generations or function evaluations. The added mechanisms are the GM, Mutagenesis, Intensification and Landscape estimation.

Numerical experiments reveal that the results obtained by ATDE are equal or superior to those of the classical DE and the recent EPSDE, without undue objective function evaluations and without negative impact on the quality of the solution obtained. The use of GM effectively assists ATDE to achieve wide exploration and deep exploitation before stopping the search. Furthermore, we could show that GM, Mutagenesis, Intensification and

Landscape estimation closely work in combination to successfully improve the performance of DE in terms of accuracy, reliability and convergence rate.

Future work includes the comparison of ATDE against different types of state-of-the-art algorithms.

References

Das, S. and Suganthan, P. (2011). Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, 15(1):4–31.

Gämperle, R., Müller, S. D., and Koumoutsakos, P. (2002). A parameter study for differential evolution. In Grmela, A. and Mastorakis, N., editors, *Advances in Intelligent Systems, Fuzzy Systems, Evolutionary Computation*, pages 293–298. WSEAS Press, Interlaken, Switzerland.

Giggs, M. S., Maier, H. R., Dandy, G. C., and Nixon, J. B. (2006). Minimum number of generations required for convergence of genetic algorithms. In *Proceedings of the 2006 IEEE Congress on Evolutionary Computation*, pages 2580–2587, Vancouver, BC, Canada.

Hansen, N. (2006). The CMA evolution strategy: A comparing review. In Lozano, J., Larranaga, P., Inza, I., and Bengoetxea, E., editors, *Towards a new evolutionary computation. Advances on estimation of distribution algorithms*, pages 75–102. Springer.

Hedar, A.-R. and Fukushima, M. (2006). Tabu search directed by direct search methods for nonlinear global optimization. *European Journal of Operational Research*, 170:329–349.

Hedar, A.-R., Ong, B. T., and Fukushima, M. (2007). Genetic algorithms with automatic accelerated termination. Technical report, Dept. of Applied Mathematics and Physics, Kyoto University.

Jain, B. J., Pohlheim, H., and Wegener, J. (2001). On termination criteria of evolutionary algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, page 768. Morgan Kaufmann Publishers.

Joshi, R. and Sanderson, A. (1999). Minimal representation multisensor fusion using differential evolution. *IEEE Transactions on Systems, Man and Cybernetics, Part A: Systems and Humans*, 29(1):63–76.

Kelley, C. T. (1999). Detection and remediation of stagnation in the Nelder-Mead algorithm using a sufficient decrease condition. *SIAM Journal on Optimization*, 10(1):43–55.

Kramer, O. (2010). Iterated local search with Powell's method: A memetic algorithm for continuous global optimization. *Memetic Computing*, 2:69–83.

Kwok, N. M., Ha, Q. P., Liu, D. K., Fang, G., and Tan, K. C. (2007). Efficient particle swarm optimization: A termination condition based on the decision-making approach. In *Proceedings of the IEEE Congress on Evolutionary Computation*, pages 25–28, Singapore.

Le, M., Ong, Y.-S., Jin, Y., and Sendhoff, B. (2009). Lamarckian memetic algorithms: Local optimum and connectivity structure analysis. *Memetic Computing*, 1:175–190.

Mallipeddi, R., Suganthan, P. N., Pan, Q. K., and Tasgetiren, M. F. (2011). Differential evolution algorithm with ensemble of parameters and mutation strategies. *Applied Soft Computing*, 11:1679–1696.

McMinn, P. (2004). Search-based software test data generation: A survey. *Software Testing Verification and Reliability*, 14(2):105–156.

Moscato, P. (1999). Memetic algorithms: An introduction. In Corne, D., Dorigo, M., Glover, F., Dasgupta, D., Moscato, P., Poli, R., and Price, K. V., editors, *New ideas in optimization*. McGraw-Hill Ltd., UK, Maidenhead, UK, England.

Nelder, J. and Mead, R. (1965). A simplex method for function minimization. *Computer Journal*, 7:308–313.

Ong, B. T. and Fukushima, M. (2011). Genetic algorithm with automatic termination and search space rotation. *Memetic Computing*, 3:111–127.

Ong, Y.-S., Lim, M.-H., Zhu, N., and Wong, K. (2006). Classification of adaptive memetic algorithms: A comparative study. *IEEE Transactions on Systems, Man, and Cybernetics–Part B*, 36(1):141–152.

O'Sullivan, M., Vossner, S., and Wegener, J. (1998). Testing temporal correctness of real-time systems. In *Proceedings of the Sixth International Conference on Software Testing Analysis and Review (EuroSTAR'98)*, Munich, Germany.

Price, K. V., Storn, R. M., and Lampinen, J. A., editors (2005). *Differential Evolution A Practical Approach to Global Optimization*. Natural Computing Series, Springer-Verlag, Berlin, Germany.

Qin, A. and Suganthan, P. (2005). Self-adaptive differential evolution algorithm for numerical optimization. In *Proceedings of the 2005 IEEE Congress on Evolutionary Computation*, volume 2, pages 1785–1791.

Storn, R. and Price, K. (1995). Differential evolution – a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical report.

Storn, R. and Price, K. (1997). Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359.

Suganthan, P. N., Hansen, N., Liang, J. J., Deb, K., Chen, Y. P., Auger, A., and Tiwari, S. (2005). Problem definitions and evaluation criteria for the CEC-2005 special session on real-parameter optimization. Technical report, Singapore: Nanyang Technol. Univ.

Zhang, J., Avasarala, V., Sanderson, A., and Mullen, T. (2008). Differential evolution for discrete optimization: An experimental study on combinatorial auction problems. In *Proceedings of the IEEE World Congress on Computational Intelligence*, pages 2794 –2800.