International Journal of Innovative Computing, Information and Control
Volume **8**, Number **5(A)**, May **2012**
ICIC International ©2012 ISSN 1349-4198
pp. 3049–3060

A NOVEL DIFFERENTIAL EVOLUTION USING MULTIPLE-DEME BASED MUTATION

Dong-Hyun Lim[1], Hoang N. Luong[2] and Chang Wook Ahn[1,*]

[1]Department of Computer Engineering
Sungkyunkwan University
No. 300, Cheonchoen-dong, Jangan-gu, Suwon, Gyeonggi-do 440-746, Korea
*Corresponding author: cwan@skku.edu

[2]Centrum Wiskunde & Informatica (CWI)
P.O. Box 94079, 1090 GB Amsterdam, The Netherlands

Received January 2011; revised June 2011

Abstract. *Mutation strategy has been acknowledged to significantly influence the performance of differential evolution (DE). The popular DE/rand/1 can relatively maintain the diversity of population, but the algorithm often distributes its search effort widely among search space without any focus. On the other hand, greedy strategies, such as DE/best/1 or DE/current-to-best/1, have faster convergence speed, but they are likely to fall into local optima due to insufficient population diversity. In this paper, we propose a novel DE variant which is able to sustain the balance between exploration and exploitation. We introduce the concept of parallel computing into DE and design a multiple-deme based mutation operator (MDM) to enable information exchange among processing units. Experimental results over a number of numerical optimization problems prove that our proposed differential evolution, DE-MDM, outperforms the traditional DE approach in terms of the quality of the final achieved solutions.*
Keywords: Differential evolution, Evolutionary algorithm, Numerical optimization, Population diversity, Parallel processing, Multiple-deme based mutation

1. **Introduction.** Differential Evolution (DE) [1, 2] achieves its popularity in the realm of evolutionary algorithms (EAs) due to its efficiency and efficacy in solving continuous parameter optimization. Growing beyond theoretical research, where DE outperforms its direct competitors (e.g., genetic algorithms [3, 4] and particle swarm optimization [5]) over a number of test benchmarks [6], DE is shown to have a wide applicability to various problem domains from academia to industry [7-13]. Employing typical genetic operators (i.e., selection, crossover, and mutation), DE variants have simpler implementations when compared with advanced EAs, such as Estimation of Distribution Algorithms (EDAs) [14], but are still competent in tackling difficult optimization tasks. Furthermore, the variants of DE have also been proposed to solve many discrete optimization problems [15]. Since its early days when proposed by Storn and Price [1, 2], DE has a thriving research community with continuous developments in terms of both theory and practice.

A robust DE variant would be able to sustain an appropriate balance between exploration and exploitation during the optimization progress [16]. In fact, all evolutionary computation practitioners need to take such requirement into consideration for designing their optimizers [17]. Exploration is defined as the enhancement of population diversity to discover larger regions of search space. On the contrary, exploitation is the reduction of population diversity to make the algorithm quickly converge to an optimum [16, 17]. While exploration guides DE through different areas of the search space, exploitation ensures that DE population be evolved toward promising regions (based on the fitness values of current candidate solutions). Solely excessive usage of any of these two procedures would prevent DE from achieving the global optimal solution. *Greedy* exploitation is ineffective because it would result in premature convergence, where DE population, losing its necessary diversity, is trapped into some misleading local optima. On the other hand, overly *extensive* exploration would compromise on the efficiency of resource consumption when unnecessarily searching over immense landscapes. In addition, DE is also reported to suffer from occasional stagnation [18], i.e., the algorithm hardly progresses even though the population has not converged yet. Therefore, various researches have been conducted into efficiency enhancement for DE by taking full advantage of both global and local information. Noman and Iba [19] proposed a memetic variant of DE coupled with the Fittest Individual Refinement, a crossover-based local search, capable of exploring the neighborhood of the best solutions through successive generations. The algorithm obtained acceptable solutions with fewer numbers of evaluations. Yang et al. [20] proposed a neighborhood search DE (NSDE), a hybridization of DE with a neighborhood search, whose mutation operator would randomly add some amounts of perturbation, normally distributed, to solution vectors. Rahnamayan et al. [21] presented an opposition-based DE (ODE) employing opposition-based learning for population initialization and generation jumping. Das et al. [22] defined a ring topology of neighborhood in DE, where a hybrid model would linearly combine the local mutation component with the global mutation component by a weight factor. A succinct review of previous works on improving DE could also be found in Das et al. [22].

In this paper, inspired by the parallel processing mechanism [23-25], we propose a new DE variant employing a mutation operation which is based on the multiple-deme topology, termed *multiple-deme based mutation* (MDM). Evolutionary algorithms tend to evolve toward elite individuals (i.e., vectors with high fitness values), and in this paper, we refer to the best individual as the *attractor* of the population. While traditional DE might prematurely converge into a local optimum due to its single attractor in the population, our proposed DE variant, referred to as DE-MDM, effectively preserves its population diversity. We run several instances of DE separately to create and maintain multiple attractors in the population; the algorithm can, thus, operate in different promising regions over the search space. Furthermore, independently operating DEs would periodically share their own exploration results with each other to generate new mutant vectors. In other words, the MDM operator helps local information be globally consulted during the optimization progress.

The rest of this paper is organized as follows. The conventional DE is briefly described in Section 2. Section 3 presents our DE-MDM. Section 4 simulates the proposed algorithm and shows the experimental results. Finally, Section 5 concludes the paper.

2. **Differential Evolution.**

Generally, DE has the same fundamental mechanism and genetic operators as other evolutionary algorithms. First, an initial population is randomly generated according to a uniform distribution. Variation operators, such as mutation and crossover, generate new offspring vectors from the current population. There exist various mutation strategies which define the behaviors and characteristics of DE. Selection operator then obtains a set of promising individuals (i.e., candidate solutions having better fitness values) from the previous population and newly created offspring. Unlike traditional genetic algorithms, however, DE performs the mutation operator first, and then the crossover operator, with the selection phase coming last. Next, we describe briefly the essential operators of DE.

An initial population $\{x_{i,0} = (x_{1,i,0}, x_{2,i,0}, \cdots, x_{D,i,0}) \mid i = 1, 2, \cdots, N_P\}$ is uniformly generated over the search space, in which $x_{j,i,0} \in [x_j^{low}, x_j^{high}]$ for $j = 1, 2, \cdots, D$, where

N_P is the population size, D is the dimension of the problem, and $[x_j^{low}, x_j^{high}]$ is the feasible domain of the jth parameter. After the initialization of the population, the genetic operators, such as *Mutation*, *Crossover* and *Selection*, are run iteratively. The population X_g, consisting of N_P candidate solutions, at the generation g is represented as

$$X_g = \{x_{i,g} \mid i = 1, 2, \cdots, N_p\} \tag{1}$$

2.1. **Mutation.** At each generation, the mutation operator constructs the mutant vectors from the current population. There exist various mutation strategies used to create mutant vectors. Some popular mutation strategies are formulated as follows:

1. DE/rand/1

$$v_{i,g} = x_{r0,g} + F(x_{r1,g} - x_{r2,g}) \tag{2}$$

2. DE/current-to-best/1

$$v_{i,g} = x_{i,g} + F(x_{best,g} - x_{i,g}) + F(x_{r1,g} - x_{r2,g}) \tag{3}$$

3. DE/best/1

$$v_{i,g} = x_{best,g} + F(x_{r1,g} - x_{r2,g}) \tag{4}$$

Three mutation strategies above are the most widely used in implementing DE. We have $v_{i,g}$ as the mutant vector; $x_{r0,g}$, $x_{r1,g}$ and $x_{r2,g}$ are distinct individuals randomly chosen from the current population X_g; $x_{best,g}$ is the best individual in X_g, and F is the scaling factor that controls the difference vectors. A visual representation on the above notations can be found in an illustrative example of DE/best/1 as shown in Figure 1.

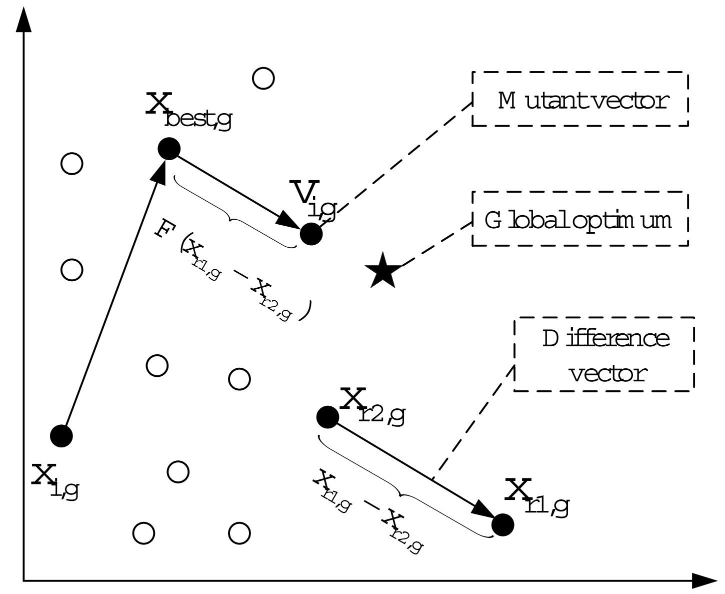

FIGURE 1. An example of the mutation strategy DE/best/1

2.2. **Crossover.** After mutant vectors are generated, DE reproduced new offspring vectors $u_{i,g} = (u_{1,i,g}, u_{2,i,g}, \cdots, u_{D,i,g})$ by the binomial crossover operations as follows:

$$u_{j,i,g} = \begin{cases} v_{j,i,g} & \text{if } rand_j(0,1) \le C_R \\ x_{j,i,g} & \text{otherwise} \end{cases} \tag{5}$$

where $rand_j(0,1)$ generates a random number between 0 and 1 for every i and j. The crossover operator tries to exchange the value of each variable in the target vector $x_{i,g}$ with the mutant vector $v_{i,g}$ in the same manner as the uniform crossover of genetic algorithm with the crossover probability $C_R \in [0,1]$.

2.3. **Selection.** The selection operator compares the fitness values of each target vector $x_{i,g}$ with its corresponding offspring vector $u_{i,g}$, which is generated by the crossover operator as described above. The vector having better fitness value is then selected as a promising individual for the next generation. The following rule is used in the task of minimization.

$$x_{i,g+1} = \begin{cases} u_{i,g} & \text{if } f(u_{i,g}) < f(x_{i,g}) \\ x_{i,g} & \text{otherwise} \end{cases} \tag{6}$$

3. **Proposed DE Algorithm.** DE mutation operation is usually implemented as equations (2), (3) or (4). DE/rand/1 is considered as the norm in implementing mutation strategy for DE because it can operate relatively well in various problem types. However, DE/rand/1 spends most of the computational resources on exploring the search space without focusing on any particular promising points. Such extensively explorative strategy, thus, would have slow convergence speed and would take a considerable amount of resources to reach an acceptable solution. On the other hand, intensively exploitative mutation strategies, such as DE/current-to-best/1 and DE/best/1, are shown to be faster in obtaining a final result. However, both of current-to-best/1 and best/1 share common drawbacks: *premature convergence* and *low-quality solution*. These are due to the rapid loss of population diversity as new solutions are generated mainly by mutating the best individual currently available in the population. Since these exploitative strategies carry out their exploration towards only a single attractor (i.e., the single best individual), they miss the opportunity to discover other potential promising regions. For problems with simple landscape, DE/best/1 or DE/current-to-best/1 can quickly locate around the global optimal point. However, when running on rugged landscapes, they easily fall into some misleading local optima due to their inherent *greedy search*-like characteristic. Also, it is most unlikely that they would be able to escape from a local optimum because the population diversity is not maintained wide enough to generate any better solutions lying in a different region. In this regard, we would like to combine the advantages of both explorative strategies and exploitative strategies: *wide population diversity* and *fast convergence speed*. In order to achieve these two goals, we maintain multiple attractors in the population by incorporating the multiple-deme parallel processing concept into DE. The idea is to separately operate several DEs at the same time, and then allow them to exchange information about their best individual vectors at certain intervals.

On the basis of the above structure, we develop the multiple-deme based mutation operator (MDM) to overcome the drawbacks of DE. Especially, we consider the fully-connected demes topology as shown in Figure 2. At first, the whole population is divided into M demes (i.e., groups or sub-populations). At each deme, a DE is executed independently; then, after every certain number of generations, each deme would *consult* other groups to share information about their *attractors*. In this paper, we employ the term *attractor* to refer to the best individual of a group. A local attractor would be the best individual of a sub-population, while the global attractor would be the best individual of the whole population. Here, instead of using its own best individual, at a certain interval, a deme would use attractors of other different demes to construct its mutant vectors. Our MDM modifies the DE/best/1 mutation strategy as follows.

$$v_{i,g}^k = x_{best,g}^{\gamma(k)} + F(x_{r1,g}^k - x_{r2,g}^k), \tag{7}$$

$$\gamma(k) = \begin{cases} rand(1, M) \notin k & \text{if } \bmod (g, \zeta) = 0 \\ k & \text{otherwise} \end{cases}$$

where the superscript of each vector is the deme index number, $rand(1, M)$ generates a random integer from $[1, M]$, and ζ denotes the interval, in terms of the number of generations, at which a deme would exchange information with other demes to create its mutant vectors.

By incorporating the MDM into the DE framework, the proposed algorithm, called DE-MDM, is outlined as follows:

1. Initialize a population with the size of N_P.
2. Partition the population into M demes (i.e., group). Each deme consists of N_P/M individuals.
3. Run the traditional *DE* independently on each deme.
4. After every ζ-generation interval, each deme evolves by referring to the attractors of different demes randomly selected from $M-1$ demes. (See Equation (7) and Figure 2).
5. Iterate from **Step 3** to **Step 4** until the termination criteria are met (e.g., the allowable number of generations or function evaluations is reached).

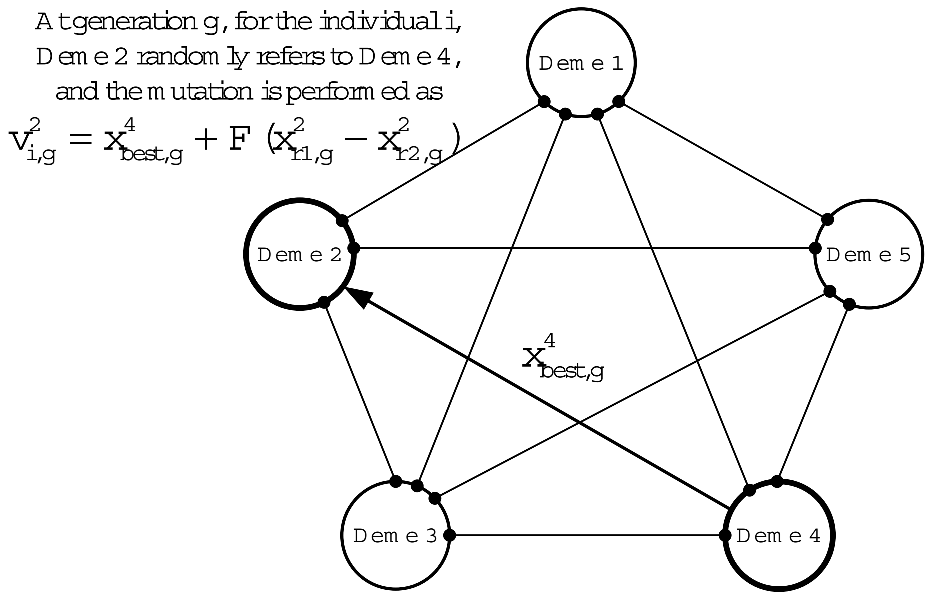
At generation g, for the individual i,
Deme 2 randomly refers to Deme 4,
and the mutation is performed as
$V_{i,g}^2 = X_{best,g}^4 + F(X_{r1,g}^2 - X_{r2,g}^2)$
Deme 1
Deme 2
Deme 3
Deme 4
Deme 5
$X_{best,g}^4$

FIGURE 2. An example of the mutation strategy DE/best/1

The operating mechanism of DE-MDM can be explained as follows. First, the whole population is partitioned into several demes (i.e., sub-populations). Each deme then runs an independent DE separately. After every ζ-generation interval, the multiple deme-based mutation is performed by allowing each deme to consult the best individuals of other randomly selected demes. In other words, instead of evolving toward its own *local* attractor, each sub-population uses different local attractors of other demes to generate new offspring. Our MDM strategy has dual effects: 1) fast convergence speed and 2) diversity preservation. The DE/best/1 *greedy* nature of our MDM helps the algorithm converge fast, while the parallel-like operating mechanism assures a diversified population. Running DE simultaneously on different demes will suggest more potential search directions for succeeding exploration. It is also more effective than the traditional mutation strategies DE/current-to-best/1 and DE/best/1 because it is able to explore a wider region of search space. Furthermore, the quality of the optimal solution returned by our approach has higher quality than the results of existing methods. The following section presents experimental results that support our claims.

4. **Experiments and Results.**

We employed 10 benchmark optimization functions (as described in [26, 27]) for empirically verifying the effectiveness of our approach.

4.1. **Test functions.**

1. Sphere function

$$f_1(x) = \sum_{i=1}^{D} x_i^2 \tag{8}$$

$$\text{Range}: -100 \le x_i \le 100, \quad \min(f_1) = f_1(0, \cdots, 0) = 0$$

2. Rotated Hyper-ellipsoid function

$$f_2(x) = \sum_{i=1}^{D} \left(\sum_{j=1}^{i} x_j \right)^2 \tag{9}$$

$$\text{Range}: -100 \le x_i \le 100, \quad \min(f_2) = f_2(0, \cdots, 0) = 0$$

3. Step function

$$f_3(x) = \sum_{i=1}^{D} \Big(\lfloor x_i + 0.5 \rfloor \Big)^2 \tag{10}$$

$$\text{Range}: -100 \le x_i \le 100, \quad \min(f_3) = f_3(0, \cdots, 0) = 0$$

4. Noisy Quartic function

$$f_4(x) = \sum_{i=1}^{D} i x_i^4 + rand[0, 1) \tag{11}$$

$$\text{Range}: -1.28 \le x_i \le 1.28, \quad \min(f_4) = f_4(0, \cdots, 0) = 0$$

5. Rosenbrock's function

$$f_5(x) = \sum_{i=1}^{D-1} (100(x_{i+1} - x_i^2)^2 + (x_i - 1)^2) \tag{12}$$

$$\text{Range}: -30 \le x_i \le 30, \quad \min(f_5) = f_5(1, \cdots, 1) = 0$$

6. Generalized Schwefel's problem 2.26

$$f_6(x) = \sum_{i=1}^{D} \left(-x_i \sin\left(\sqrt{|x_i|} \right) \right) \tag{13}$$

$$\text{Range}: -500 \le x_i \le 500, \quad \min(f_6) = f_6(420.9687, \cdots, 420.9687) = -12569.5$$

7. Penalized function

$$\begin{aligned} f_7(x) =& \frac{\pi}{D}\left\{10\sin^2(\pi y_1) + \sum_{i=1}^{D-1}(y_i-1)^2\left[1+10\sin^2(\pi y_{i+1})\right] + (y_D-1)^2\right\} \\ &+ \sum_{i=1}^{D} u(x_i, 10, 100, 4) \end{aligned} \tag{14}$$

where $y_i = 1 + \frac{1}{4}(x_i+1)$ and $u(x_i, a, k, m)) = \begin{cases} k(x_i-a)^m & x_i > a \\ 0 & -a \le x_i \le a \\ k(-x_i-a)^m & x_i < -a \end{cases}$

$$\text{Range} : -50 \le x_i \le 50, \quad \min(f_7) = f_7(1, \cdots, 1) = 0$$

8. Griewank's function

$$f_8(x) = \frac{1}{4000}\sum_{i=1}^{D} x^2 - \prod_{i=1}^{D}\cos\left(\frac{x_i}{\sqrt{i}}\right) + 1 \tag{15}$$

$$\text{Range} : -600 \le x_i \le 600, \quad \min(f_8) = f_8(0, \cdots, 0) = 0$$

9. Rastrigin's function

$$f_9(x) = \sum_{i=1}^{D}\left(x_i^2 - 10\cos(2\pi x_i) + 10\right) \tag{16}$$

$$\text{Range} : -5.12 \le x_i \le 5.12, \quad \min(f_9) = f_9(0, \cdots, 0) = 0$$

10. Shifted Rotated Ackley's function

$$f_{10}(x) = -20\exp\left(-0.2\sqrt{\frac{1}{D}\sum_{i=1}^{D}x_i^2}\right) - \exp\left(\frac{1}{D}\sum_{i=1}^{D}cos(2\pi x_i)\right) + 20 + e \tag{17}$$

$$\text{Range} : -32 \le x_i \le 32, \quad \min(f_{10}) = f_{10}(0, \cdots, 0) = 0$$

The descriptions of the above set of benchmark functions are taken from Zhang et al. [26] and Yao et al. [27]. All functions f_1-f_{10} are high dimensional problems. Functions f_1-f_3 are unimodal functions, in which f_1 and f_2 are continous while f_3 is a step function with a discontinous landscape having plateaus. Function f_4 is a noisy quartic function, where $rand[0, 1)$ would randomly generate the noise uniformly distributed in the interval $[0, 1)$. Rosenbrock function f_5 is a unimodal function when $D = 2$ or $D = 3$; for $D \ge 4$, the function f_5 has multiple minima. Functions f_6-f_{10} are difficult optimization problems because they are multimodal functions with many local minima [26, 27].

4.2. **Experimental results.** For experiments, we set up the parameter D (Dimension) to be 30, the total number of evaluations to be 100,000, the maximal population size to be 200, the minimal population size to be 20, and the number of demes (i.e., M) to be 5. Finally, F and C_R are fixed as 0.95 and 0.5, respectively. Population size setting is a crucial parameter of DE, and different settings may produce various results. Therefore, we conduct experiments with a wide ranges of population sizes (from 20 to 200) to prove that our DE-MDM has better performance than the conventional DE regardless of population size settings. The interval ζ for each deme consulting other demes is another parameter that needs to be configured. We also test DE-MDM with different ζ to investigate its effects on the performance of the algorithm ($\zeta = 10, 20, 30, 40$ or 50).

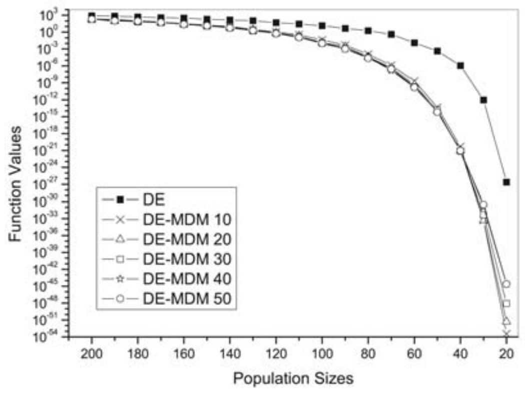

FIGURE 3. Sphere function

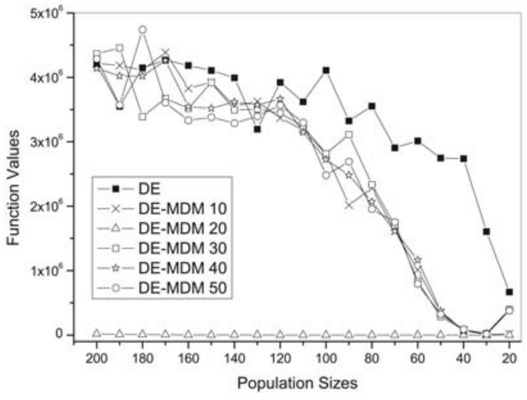

FIGURE 4. Rotated hyper ellipsoid function

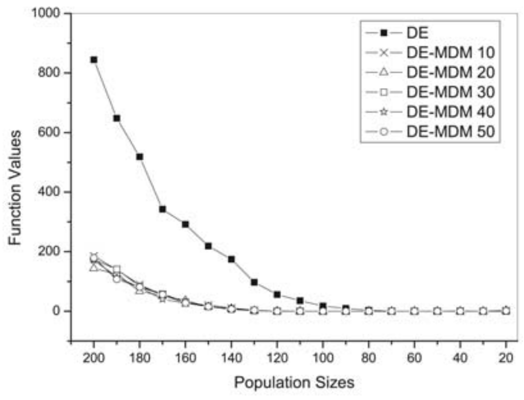

FIGURE 5. Step function

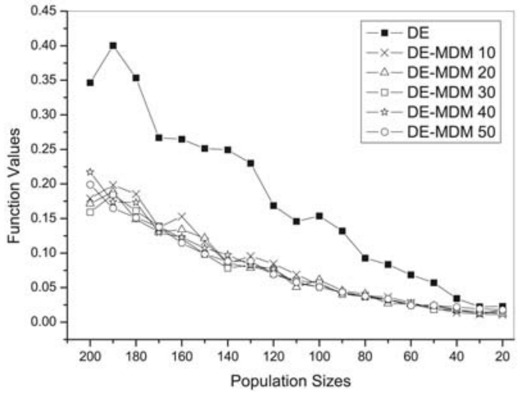

FIGURE 6. Noisy quartic function

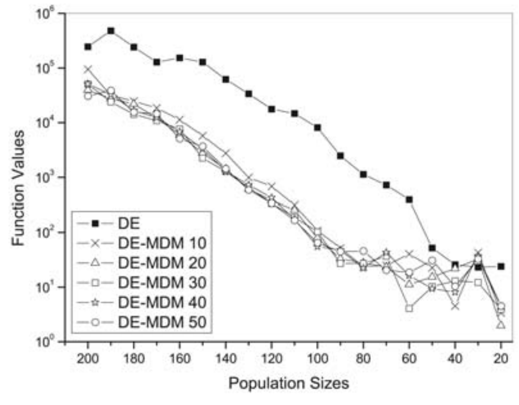

FIGURE 7. Rosenbrock's function

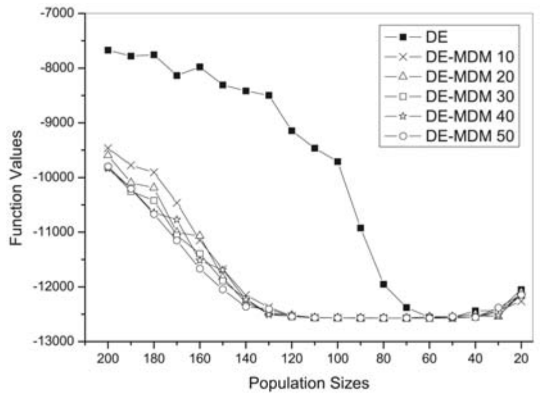

FIGURE 8. Schwefel's problem 2.26

Figures 3-12 present the function values obtained after running DE and five variants of DE-MDM (with different ζ values) on 10 benchmark functions f_1-f_{10} with varying population sizes (from 20 to 200). Excluding a few exceptions (e.g., Figure 4), all five DE-MDM variants have similar performance and show the same tendencies in final results with respect to the population size settings. Therefore, we can refer to our approach as DE-MDM regardless of the ζ setting to discuss experimental results.

Figures 3-12 exhibit significant gaps of differences in performance of algorithms when changing the population size settings. DE and DE-MDM with small population sizes

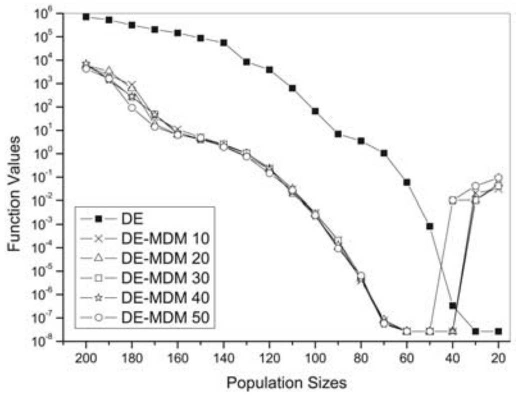

FIGURE 9. Penalized function

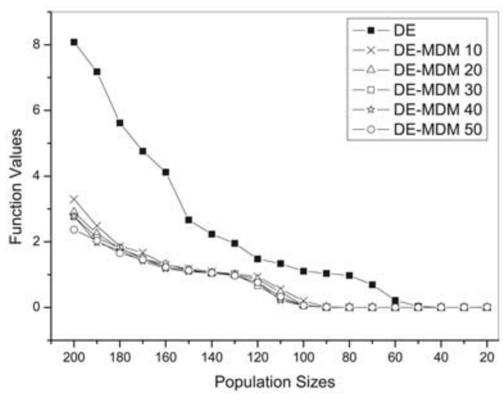

FIGURE 10. Griewank's function

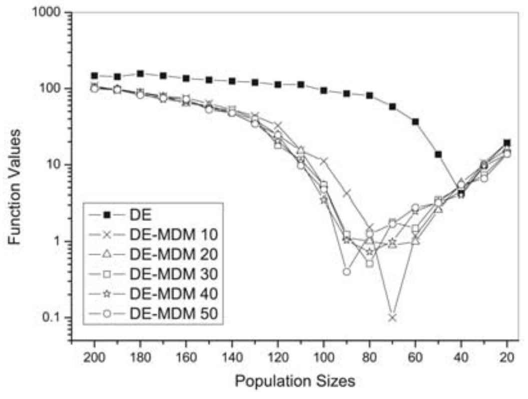

FIGURE 11. Rastrigin's function

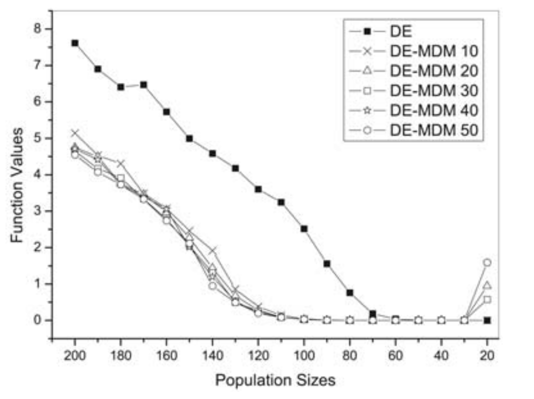

FIGURE 12. Shifted rotated Ackley's function

obtain better results. It can be concluded from the graphs that small population sizes (from 20 to 60) is preferable in solving this set of benchmark functions (all with $D = 30$). Another observable remark is that under same population size settings our DE-MDM can outperform the conventional DE in terms of final function values.

For unimodal functions f_1 and f_2, it is apparent that DE-MDM surpasses DE with all different population size settings. A population of 20 individuals yields the best performance for all algorithms when solving the simple sphere function f_1, and five DE-MDM variants achieve significantly better final results than the conventional DE (Figure 3). Similar remarks can be found when solving the rotated hyper ellipsoid function f_2 (Figure 4). The variant DE-MDM with $\zeta = 20$, however, shows an exceptional case, in which its performance is superior to DE and all other DE-MDM variants regardless of population sizes.

For the step function f_3 and the noisy quartic function f_4 (Figure 5 and Figure 6, respectively), DE-MDM performs considerably better than DE with population sizes ranging from 80 to 200. Also, for smaller settings ($20 \leq N_p < 80$), our DE-MDM still performs slightly better than DE. We note that the gap between the worst function values (returned by large populations) and the best function values (returned by small populations) of our DE-MDM is much narrower than that of the conventional DE. It implies that the conventional DE is sensitive to population sizes, and thus requires an appropriate setting to obtain acceptable results; our DE-MDM, on the contrary, is more robust and stable.

Multimodal functions f_5-f_{10} (see Figures 7-12) have been acknowledged to be more difficult than unimodal functions because the optimization algorithms would be challenged by many misleading local optima before locating the global optimum [27]. A robust algorithm should maintain enough population diversity to escape local optima and good exploitation to achieve the best result within its allowable computational resources. In this regard, our DE-MDM is shown to be superior to the conventional DE on most test cases. In general, with the same population size settings (regardless of their values), DE-MDM obtains better final results when compared to DE. The only few exceptions are the cases of Penalized function f_7 (with population size of 20 or 30) and Ackley's function f_{10} (with population size of 20), in which DE can locate better solutions than our DE-MDM. However, for any larger populations, DE-MDM still proves its superiority.

Table 1 reports the best function values obtained by DE and DE-MDM when optimizing the above set of 10 benchmark functions. The best final results of each algorithm are presented with their corresponding population sizes and the interval ζ. The experiments, thus, clearly support the superiority of DE-MDM over the conventional DE.

Table 1. Experimental results

	DE		DE-MDM		
function	N_P	Value	N_P	ζ	Value
f_1	20	2.25586E-27	20	10	**3.55395E-54**
f_2	20	666439.4	20	20	**1.27233E-46**
f_3	30-70	0.0	30-120	10-50	0.0
f_4	30	0.022422	20	10	**0.011015619**
f_5	30	23.03484	20	20	**1.97633998**
f_6	60	−12557.9	50-90	10-50	**−12569.5**
f_7	20-30	2.66575E-08	40-60	10-50	2.66575E-08
f_8	40	6.4133E-06	30-40	10-50	**5.42101E-20**
f_9	40	4.201033	70	10	**0.100012759**
f_{10}	20	0.0	30-40	10-50	0.0

5. Conclusion.

Controlling the budget of computational resource on exploration and exploitation is a fundamental research topic in differential evolution and in other evolutionary algorithms as well. Population diversity preservation and fast convergence speed are two conflicting criteria of a robust optimization algorithm. Insisting on a criterion would require compromising on the other one. In this paper, our DE-MDM was proposed to address the above problem of balancing exploration and exploitation. We designed a parallel processing-inspired DE variant, and incorporated it with a novel multiple-deme based mutation strategy (MDM). The whole population is partitioned into different demes (i.e., sub-populations), and at each deme, an independent DE is run separately in the scope of that deme. This parallel computing-like model can preserve the population diversity due to various DEs operating on different regions of the search space. DE/best/1 is implemented as the mutation strategy so that each DE could converge quickly toward its local best individual. At regular intervals, each deme would consult the best individuals of other demes to construct its mutant vectors. In this way, all demes can propagate their own explorative knowledge and can take advantage of the information obtained from other demes as well.

We performed the experiments on 10 numerical optimization tasks. The performance of our DE-MDM was compared with the conventional DE. Experimental results proved that DE-MDM could outperform the traditional DE, in terms of the quality of the obtained solutions, with a wide range of population size settings. The results supported our assertion that DE-MDM could effectively and efficiently balance its exploration (i.e., various promising regions are discovered) with its exploitation (i.e., making use of exchanged information to obtain better solutions within the limit of computational resouces).

Lastly, we emphasize that our DE-MDM has potentials for being applied to complicated real-world problems. Industrial tasks which consume considerable amounts of computation resources would demand that the optimization algorithm be implemented in a true parallel processing framework. Having a multiple-deme based operating mechanism, our DE-MDM, thus, can be straightforwardly scaled to operate effectively in such system.

Acknowledgment. This research was supported by MKE, Korea under ITRC NIPA-2012-(C1090-1221-0008).

REFERENCES

[1] R. Storn and K. Price, Differential evolution a simple and efficient heuristic for global optimization over continuous spaces, *J. Global Optimization*, vol.11, no.4, pp.341-359, 1997.

[2] K. V. Price, R. M. Storn and J. A. Lampinen, *Differential Evolution: A Practical Approach to Global Optimization*, 1st Edition, Springer-Verlag, New York, NY, 2005.

[3] M. Mitchell, *An Introduction to Genetic Algorithms*, MIT Press, Cambridge, MA, 1996.

[4] D. E. Goldberg, *Genetic Algorithms in Search, Optimization, and Machine Learning Reading*, Addsion Wesley, MA, 1989.

[5] J. Kennedy and R. Eberhart, Particle swarm optimization, *Proc. of IEEE Int. Conf. Neural Netw.*, pp.1942-1948, 1995.

[6] J. Vesterstrom and R. Thomsen, A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems, *Proc. of the 6th Congr. Evol. Comput.*, pp.1980-1987, 2004.

[7] M. Omran, A. P. Engelbrecht and A. Salman, Differential evolution methods for unsupervised image classification, *Proc. of the 7th Congr. Evol. Comput.*, pp.966-973, 2005.

[8] S. Das, A. Abraham and A. Konar, Adaptive clustering using improved differential evolution algorithm, *IEEE Trans. Syst., Man, Cybern. A*, vol.38, no.1, pp.218-237, 2008.

[9] T. Rrogalsky, R. W. Derksen and S. Kocabiyik, Differential evolution in aerodynamic optimization, *Proc. of the 46th Annu. Conf. Can. Aeronautics Space Inst.*, pp.29-36, 1999.

[10] F. S. Wang and H. J. Jang, Parameter estimation of a bio-reaction model by hybrid differential evolution, *Proc. of the 2nd IEEE Congr. Evol. Comput.*, pp.410-417, 2000.

[11] R. Joshi and A. C. Sanderson, Minimal representation multi-sensor fusion using differential evolution, *IEEE Trans. Syst., Man, Cybern., Part A*, vol.29, no.1, pp.63-76, 1999.

[12] C.-J. Lin, C.-F. Wu and C.-Y. Lee, Design of a recurrent functional neural fuzzy network using modified differential evolution, *International Journal of Innovative Computing, Information and Control*, vol.7, no.2, pp.669-683, 2011.

[13] F. T. Lin, Application of differential evolution for fuzzy linear programming, *ICIC Express Letters*, vol.5, no.6, pp.1851-1856, 2011.

[14] M. Pelikan, D. E. Goldberg and F. G. Lobo, A survey of optimization by building and using probabilistic model, *Comput. Optim. Appl.*, vol.21, no.5, pp.5-20, 2002.

[15] V. Vegh, G. K. Pierens and Q. M. Tieng, A variant of differential evolution for discrete optimization problems requiring mutually distinct variables, *International Journal of Innovative Computing, Information and Control*, vol.7, no.2, pp.897-914, 2011.

[16] M. G. Epitropakis, V. P. Plagianakos and M. N. Vrahatis, Balancing the exploration and exploitation capabilities of the differential evolution algorithm, *Proc. of the 10th IEEE Congr. Evol. Comput.*, pp.2686-2693, 2008.

[17] L. Hanshen and K. Lishan, Balance between exploration and exploitation in genetic search, *Wuhan University Journal of Natural Sciences*, vol.4, no.1, pp.28-32, 1999.

[18] J. Lampinen and I. Zelinka, On stagnation of the differential evolution algorithm, *Proc. of the 6th Int. Mendel Conf. Soft Computing*, Brno, Czech Republic, pp.76-83, 2000.

[19] N. Noman and H. Iba, Enhancing differential evolution performance with local search for high dimensional function optimization, *Proc. of the Genetic Evol. Comput. Conf.*, pp.967-974, 2005.

[20] Z. Yang, J. He and X. Yao, Making a difference to differential evolution, *Advances in Metaheuristics for Hard Optimization*, pp.415-432, 2007.

[21] S. Rahnamayan, H. R. Tizhoosh and M. M. A. Salama, Opposition-based differential evolution algorithms, *IEEE Trans. Evol. Comput.*, vol.12, no.1, pp.64-79, 2008.

[22] S. Das, A. Abraham, U. K. Chakraborty and A. Konar, Differential evolution using a neighborhood-based mutation operator, *IEEE Trans. Evol. Comput.*, vol.13, no.3, pp.526-553, 2009.

[23] D. K. Tasoulis, N. G. Pavlidis, V. P. Plagianakos and M. N. Vrahatis, Parallel differential evolution, *Proc. of the 6th IEEE Congr. Evol. Comput.*, 2004.

[24] V. P. Plagianakos and M. N. Vrahatis, Parallel evolutionary training algorithms for 'hardware friendly' neural networks, *Natural Computing*, vol.1, pp.307-322, 2002.

[25] A. Geist, A. Beguelin, J. Dongarra, W. Jiang, R. Manchek and V. Sunderam, *PVM: Parallel Virtual Machine – A User's Guide and Tutorial for Networked Parallel Computing*, MIT Press, Cambridge, 1994.

[26] J. Zhang and A. C. Sanderson, *Adaptive Differential Evolution – A Robust Approach to Multimodel Problem Optimization*, Springer-Verlag, New York, NY, 2009.

[27] X. Yao, Y. Liu, K. H. Liang and G. Lin, Fast evolutionary algorithms, in *Advances in Evolutionary Computing: Theory and Applications*, G. Rozenberg, T. Back and A. Eiben (eds.), New York, NY, Springer, 2003.