International Journal of Mathematics and Computer Science
Volume **20,** Issue no. 4, (2025), 1043–1052
DOI: https://doi.org/10.69793/ijmcs/04.2025/puphasuk

Differential evolution algorithm with adaptive crossover and sorting mutation strategies for high-dimensional optimization

Jeerayut Wetweerapong, Pikul Puphasuk

Department of Mathematics
Faculty of Science
Khon Kaen University
Khon Kaen, Thailand

email: wjeera@kku.ac.th, ppikul@kku.ac.th

(Received September 5, 2025, Accepted October 3, 2025,
Published October 9, 2025)

Abstract

High-dimensional continuous optimization problems have arisen in many modern applications including engineering design, feature learning, and big data analytics. An efficient and scalable optimization method is required to search the solution spaces that exponentially increase in size with the number of decision variables. In this paper, we propose an adaptive differential evolution algorithm using adaptive crossover and sorting mutation strategies (DEASCSM) for solving problems of dimensions up to 1000. The adaptive crossover adapts the search to suit different problem landscapes, while the sorting mutation accelerates the convergence. We investigate the algorithm's scalability by considering both the value-to-reach and limited-budget approaches and evaluate its performance on several benchmark functions. Experimental results show that the DEASCSM algorithm can find optimal solutions for high-dimensional problems and scales differently, depending on the problem characteristics. The performance comparison with well-known methods demonstrates its superiority.

Key words and phrases: Optimization, high-dimensional problems, scalability, adaptive differential evolution algorithm.
AMS (MOS) Subject Classifications: 65K10, 90C26.
ISSN 1814-0432, 2025, https://future-in-tech.net

1 Introduction

High-dimensional optimization poses significant challenges due to large search spaces and nonlinearity, multimodality, and nonseparability of objective functions. As dimensionality increases, the search space grows exponentially, reducing algorithm performance under limited evaluation budgets. Population-based optimization methods, which utilize multiple agents inspired by natural processes, offer improved exploration and interaction compared to traditional single-point approaches. However, assessing algorithm scalability is essential, as linearly scaling evaluation budgets with dimensionality often proves insufficient. A more effective metric, proposed by Storn and Price [1, 2], is the number of evaluations needed to reach a predefined value-to-reach (VTR), which better captures an algorithm's speed, convergence, and reliability.

Piccand et al. [3] investigated the performance and scalability of two variants of PSO (LBest and GBest) on Ackley, Griewank, Rastrigin, and Rosenbrock functions with problem dimensions varied up to 500. The maximum number of function evaluations is 10^7, and the performance measure is the median of the number of iterations required to reach the $VTR = 0.01, 0.05, 100$, and 100 for each function, respectively. The experiment varies the swarm size to $25, 50, 75, 100, 150, 200, 250, 300, 400$, and 500. The results show a scalability issue for Ackley, Rastrigin, and Rosenbrock. It is necessary to increase the swarm size, and the number of evaluations required to solve the problem increases exponentially. Li et al. [4] proposed a hybrid algorithm called PS–ABC, which combines the local search phase in PSO with two global search phases in ABC. They validated the PS–ABC algorithm on high-dimensional benchmark functions with $D = 60, 100$, and 500 and compared it with ABC, PSO, HPA, ABC–PS, and OXDE algorithms using the maximum number of function evaluations equal to $10000D$. Comparison results show that the PS-ABC outperforms the compared methods. Gong et al. [5] introduced a hybrid DE with biogeography-based optimization (BBO) that combines the exploration of DE with the exploitation of BBO, called DE/BBO. They verified the algorithm's performance on test functions with dimensions $D = 10, 30, 50, 100$, and 200. The VTR is 10^{-8} for most functions, and the maximum number of function evaluations is $10000D$. The influence of the population size and different mutation schemes is also studied. The overall performance of DE/BBO is superior to DE and BBO. Tuo et al. [6] presented an improved strategy for DE that integrates crossover and mutation operators with a new local variable adjustment strategy. The problem dimensions D are $200, 500$, and 1000, and the maximum number

of function evaluations is 5×10^6 for all dimensions. Experimental results demonstrate that the proposed algorithm is more effective in terms of solution quality. Yildiz and Topal [7] investigated the scalability of microdifferential evolution. The algorithm uses a small population size and a directional local search. The problem dimensions D are $1000, 2000$, and 5000, and the maximum function evaluations are $3 \times 10^6, 5 \times 10^6$, and 10×10^6, respectively. They showed the scalability through the convergence graphs, where the graphs for difficult problems still indicate very high objective function values. Puphasuk and Wetweerapong [8] proposed an enhanced differential evolution algorithm with an adaptive switching crossover strategy (DEASC) for continuous optimization. The algorithm randomizes a scaling factor in $[0.5, 0.7]$ and adaptively chooses a crossover rate value in $[0, 0.1]$ and $[0.9, 1]$ according to the success in creating better trial vectors for the selection process. Experiment results on test problem dimensions up to 100 show that it outperforms several well-known enhanced DE variants. It also gives outstanding results for high-dimensional test functions.

In this paper, we propose an improved DE algorithm, called DEASCSM, that combines sorting mutation and an adaptive switching crossover strategy to balance exploration and exploitation in high-dimensional optimization.

2 The proposed method

We propose a differential evolution algorithm with an adaptive switching crossover strategy and sorting mutation (DEASCSM) to address continuous high-dimensional optimization problems. The adaptive crossover mechanism dynamically switches between low and high crossover rates to better suit the objective functions of various characteristics. The sorting mutation selects the best individual among three randomly chosen vectors as the base vector for mutant generation, aiming to accelerate convergence. The detailed procedure of DEASCSM is presented in Algorithm 1.

3 Experimental Results and Discussion

Two experiments are conducted for testing the performance of the DEASCSM method and the compared methods on eight benchmark functions from the CEC2013 [9].

Algorithm 1 DEASCSM method

Set the input and control parameters: population size (NP), problem dimension (D), lower and upper bounds of the problem (LB,UB), value-to-reach (VTR), maximum number of function evaluations ($maxnf$)
Initialize NP population vectors x_i with real values between LB and UB
Find the best vector x_{best} and its best function value f_{best}
Set the initial probabilities for using low and high crossover rates $pc1 = pc2 = 0.5$
Set the initial counters corresponding to $pc1$ and $pc2$ as $nc1 = nc2 = 0$
Set number of function evaluations $nf = 0$
while $(nf < maxnf)$ and $(f_{best} > VTR)$ **do**
    for $i = 1 : NP$ **do**
        Randomly select distinct population vectors $x_{r_1}, x_{r_2}, x_{r_3}$ different from x_i. Generate a mutant vector x_m using the sorting mutation as

$$x_m = x^*_{r_1} + F(x^*_{r_2} - x^*_{r_3})$$

where $x^*_{r_1}, x^*_{r_2}, x^*_{r_3}$ are sorted vectors as $f(x^*_{r_1}) \le f(x^*_{r_2}) \le f(x^*_{r_3})$, and F is a scaling factor randomed in $[0.5, 0.7]$.
        Generate a uniform random number u in $(0, 1)$
        if $u < pc1$ **then**
            Random a crossover rate CR in $[0, 0.1]$
        else
            Random a crossover rate CR in $[0.9, 1]$
        end if
        Construct the trial vector x_c as

$$x_{c_j} = \begin{cases} x_{m_j} & ; rand_j < CR \text{ or } j = IC \\ x_{ij} & ; \text{otherwise} \end{cases}$$

where $j = 1, 2, \ldots, D$, each $rand_j$ is a uniformly random number in $[0, 1]$, and IC is a randomly fixed index from 1 to D for this crossover.
        Evaluate $f(x_c)$ and increase $nf := nf + 1$
        if $f(x_c) < f(x_i)$ **then**
            Replace x_i with x_c and $f(x_i)$ with $f(x_c)$
            if x_c is generated with $CR \in [0, 0.1]$ **then**
                Increase $nc1 := nc1 + 1$
            else
                Increase $nc2 := nc2 + 1$
            end if
            if $f(x_c) < f_{best}$ **then**
                Replace x_{best} with x_c and f_{best} with $f(x_c)$
            end if
        end if
        if $nc1 + nc2 \ge 100$ **then**
            Adjust $nc1 := nc1 + 10$ and $nc2 := nc2 + 10$ to prevent them from 0
            Update the probabilities using the weighted formula

$$pc1 = 0.9pc1 + 0.1(nc1)/(nc1 + nc2) \text{ and } pc2 = 1.0 - pc1$$

            Reset the counters $nc1 = 0$ and $nc2 = 0$
        end if
    end for
end while
Report the obtained x_{best}, f_{best}, and nf

3.1 Performance comparison of DEASCSM with other algorithms using VTR

DEASCSM's performance is compared to DE50, DE100, and DEASC across dimensions $D = 50, 100, 200, 500,$ and 1000. Here, DE50 and DE100 refer to the DE algorithm with population sizes $NP = 50$ and $NP = 100$, respectively, both using the classical mutation strategy with parameters $F = 0.5$ and $CR = 0.9$. Each algorithm is tested using $VTR = 10^{-10}$ and $maxnf = 50000D$. Every configuration is executed over 50 independent runs. Table 1 reports $NS/Mean\,nf(\%SD)$ values where NS is the number of successful runs, $Mean\,nf$ is the mean number of function evaluations, and $\%SD$ is the percentage standard deviation of function evaluations, with the best values highlighted in bold. The results demonstrate that DEASCSM consistently delivers the best performance across all tested cases. It achieves $NS = 50$ with the lowest $Mean\,nf$ for all dimensions and functions, except for the Sphere and Sum Square functions at $D = 50$. DEASC also achieves full success ($NS = 50$) but converges more slowly, particularly on the Rosenbrock function. DE50 performs well on the Sphere and Sum Square functions at $D = 50$ but degrades significantly in higher dimensions and fails to solve the Rosenbrock, Rastrigin, and Schwefel functions. DE100, with a larger population, shows improved performance over DE50 on Rosenbrock, Step, Ackley, and Griewank functions, but remains ineffective on Rastrigin and Schwefel and suffers from poor scalability. Overall, the experimental results confirm that DEASCSM outperforms the other methods in both convergence speed and robustness across all tested dimensions and functions.

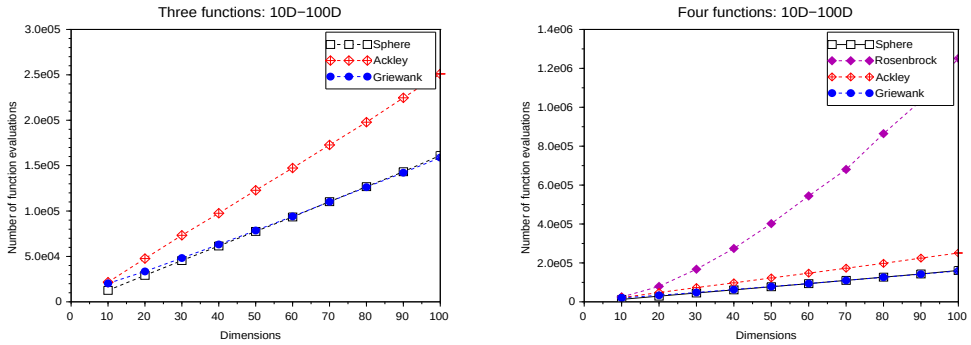
Figure 1 shows the scalability curves of DEASCSM using the $VTR = 10^{-10}$ for (a) low- and (b) high-dimensional functions. The scalability graphs of Sphere, Ackley, and Griewank on the left subfigures show that the Mean nf values increase almost as linear functions for low dimensions $D \leq 100$ but increase exponentially for high dimensions $D > 100$. The DEASCSM requires a very close $Mean\,nf$ for Sphere and Griewank functions and a larger $Mean\,nf$ for Ackley. Subfigures on the right include the scalability curves for the Rosenbrock function. They show that the $Mean\,nf$ values for both low- and high-dimensional Rosenbrock functions increase rapidly as exponential functions, which are significantly higher than other functions. These results suggest that we need more function evaluations on an exponential scale for an optimization method to converge to high-quality solutions when performing on high-dimensional problems.

Table 1: Performance comparison of DEASCSM with DE50, DE100, and DEASC at $VTR = 10^{-10}$ over 50 independent runs

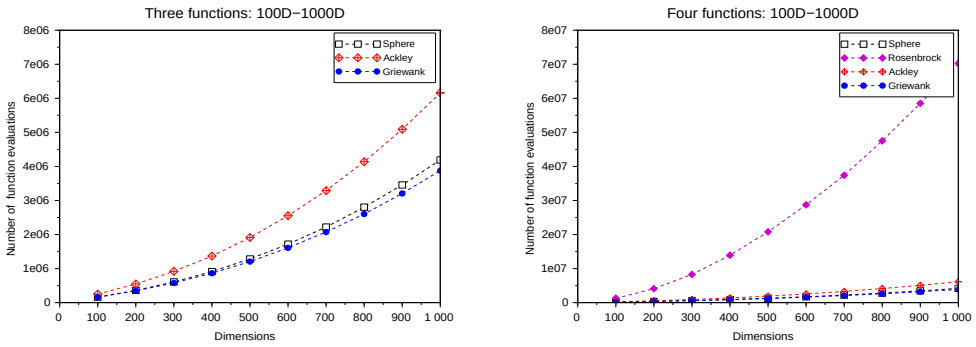
Function	D	DE50	DE100	DEASC	DEASCSM
Sphere	50	**50/69698(5.39)**	50/168468(1.91)	50/85578(0.89)	50/77570(1.23)
	100	50/197348(7.27)	50/309907(1.69)	50/176484(0.74)	**50/161289(1.13)**
	200	50/667742(7.47)	50/693951(3.09)	50/389630(0.68)	**50/359311(1.02)**
	500	50/407652(7.84)	50/3202829(5.12)	50/1387144(0.86)	**50/1283088(0.84)**
	1000	25/14430841(8.32)	50/12923178(3.38)	50/4526815(0.43)	**50/4189740(0.48)**
Sum Square	50	**50/66362(5.83)**	50/160105(2.03)	50/81505(0.95)	50/73639(1.24)
	100	50/195166(8.18)	50/302849(1.98)	50/171708(0.62)	**50/156770(1.01)**
	200	50/668863(8.74)	50/689792(2.54)	50/387946(0.64)	**50/357296(0.82)**
	500	50/4151086(6.64)	50/3271427(4.64)	50/1418784(0.71)	**50/1308668(0.65)**
	1000	17/15704632(7.81)	50/14234970(5.67)	50/4692884(0.43)	**50/4358841(0.45)**
Rosen-brock	50	0/-(-)	47/1041933(4.57)	50/703364(6.72)	**50/402027(3.85)**
	100	0/-(-)	16/4350019(12.63)	50/2139379(5.52)	**50/1251305(4.15)**
	200	0/-(-)	0/-(-)	50/6576131(4.21)	**50/4091852(2.91)**
	500	0/-(-)	0/-(-)	50/3423535(2.18)	**50/20782467(2.36)**
	1000	0/-(-)	0/-(-)	50/124060308(2.10)	**50/70301712(1.96)**
Step	50	4/21826(6.96)	50/54361(4.32)	50/27271(2.81)	**50/23742(3.58)**
	100	0/-(-)	10/124754(6.16)	50/56771(2.13)	**50/50382(3.57)**
	200	0/-(-)	0/-(-)	50/127996(1.89)	**50/67758(5.36)**
	500	0/ -(-)	0/-(-)	50/483104(2.54)	**50/433395(2.68)**
	1000	0/-(-)	0/-(-)	50/1644244(2.59)	**50/1526368(2.24)**
Rastri-gin	50	0/-(-)	0/-(-)	50/133654(1.41)	**50/125264(2.06)**
	100	0/-(-)	0/-(-)	50/357029(1.88)	**50/339482(1.84)**
	200	0/-(-)	0/-(-)	50/1134757(1.41)	**50/1095106(1.81)**
	500	0/-(-)	0/-(-)	50/6262541(1.44)	**50/6015143(1.42)**
	1000	0/-(-)	0/-(-)	50/24953795(1.37)	**50/23887328(1.23)**
Ackley	50	34/113760(4.50)	50/269669(1.89)	50/134557(0.80)	**50/122848(0.81)**
	100	0/-(-)	50/485534(1.51)	50/272059(0.58)	**50/251050(0.82)**
	200	0/-(-)	0/-(-)	50/592502(0.37)	**50/548285(0.64)**
	500	0/-(-)	0/-(-)	50/2063694(0.37)	**50/1911899(0.59)**
	1000	0/-(-)	0/-(-)	50/6608554(0.23)	**50/6160360(0.43)**
Schwe-fel	50	0/-(-)	0/-(-)	50/104962(1.59)	**50/97706(1.13)**
	100	0/-(-)	0/-(-)	50/199978(0.76)	**50/186557(0.82)**
	200	0/ -(-)	0/-(-)	50/499660(1.23)	**50/470518(1.26)**
	500	0/ -(-)	0/-(-)	50/2052446(0.74)	**50/1920986(1.60)**
	1000	0/-(-)	0/-(-)	50/7456315(0.11)	**50/7129359(0.67)**
Grie-wank	50	50/170633(1.88)	49/169460(1.57)	50/87485(1.33)	**50/78549(1.41)**
	100	23/189994(5.93)	44/306575(1.71)	50/174467(0.92)	**50/158863(1.15)**
	200	14/622768(7.57)	42/677068(2.81)	50/378628(0.64)	**50/347458(1.06)**
	500	0/-(-)	20/2857283(3.33)	50/1313024(0.62)	**50/1205510(0.73)**
	1000	0/-(-)	21/10999046(5.10)	50/4176739(0.72)	**50/3876718(0.57)**

3.2 Performance comparison of DEASCSM with other algorithms using $maxnf$

To compare the performance of DEASCSM with other algorithms using $maxnf$, we conduct two sub-experiments with different settings according to the original papers. The first sub-experiment compares DEASCSM with DE and DE/BBO over 50 independent runs using $maxnf = 10000D$ where $D = 50, 100$, and 200. Table 2 reports the mean of the best function values



(a) Low-dimensional functions



(b) High-dimensional functions

Figure 1: Scalability performance comparison of DEASCSM at $VTR = 10^{-10}$

($Mean\ fb$) and the standard deviations (SD), where the values for DE and DE-BBO are from the original paper [5]. The results of all 18 cases show that DEASCSM gives the best $Mean\ fb$ for 13 cases, while DE and DE/BBO provide the best $Mean\ fb$ for 0 and 9 cases, respectively.

We also use the Welch t-test at a 0.05 significance level to verify the results. The symbols $+, =$, and $-$ denote that DEASCSM performs significantly better than, similarly to, and worse than a compared method. From the last column of the table, the summation values $(+, =, -)$ when compared with DE and DE-BBO are $(18, 0, 0)$ and $(9, 4, 5)$, respectively. They indicate that DEASCSM significantly outperforms DE and DE/BBO. The second sub-experiment compares the performance of DEASCSM and those of PS-ABC, ABC-PS, and OXDE using $maxnf = 10000D$ where $D = 60, 100$, and 500. Each algorithm performs 20 independent runs. Table 3 reports the $Mean\ fb$, SD, and the Welch t-test at a 0.05 significance level. The values for the compared methods are from the original reference [4]. The results of

Table 2: Performance comparison of DEASCSM with DE and DE/BBO using $maxnf = 10000D$ over 50 independent runs

Function	D	DE	DE/BBO	DEASCSM	Significance
		$Mean\ fb(SD)$	$Mean\ fb(SD)$	$Mean\ fb(SD)$	
Sphere	50	1.87E-32(1.64E-32)	**0**(**0**)	4.57E-91(6.74E-91)	+,–
	100	2.30E-31(1.37E-31)	6.16E-34(1.97E-33)	**1.27E-88**(**1.16E-88**)	+,+
	200	1.41E-24(3.14E-24)	3.07E-32(2.48E-32)	**4.35E-80**(**2.26E-80**)	+,+
Rosen-brock	50	5.33E+01(2.89E+01)	4.43E+01(1.39E+01)	**4.74E-17**(**3.07E-16**)	+,+
	100	1.76E+02(4.22E+01)	1.19E+02(3.38E+01)	**9.42E-01**(**1.33E+00**)	+,+
	200	4.39E+02(1.11E+02)	2.95E+02(4.48E+01)	**9.99E+01**(**2.03E+01**)	+,+
Step	50	5.57E+04(1.64E+04)	2.74E+04(6.22E+02)	**0**(**0**)	+,+
	100	3.30E+05(1.34E+05)	4.93E+04(1.11E+03)	**0**(**0**)	+,+
	200	3.00E+00(7.75E+00)	**0**(**0**)	**0**(**0**)	+,=
Rastrigin	50	2.33E+01(8.27E+01)	**0**(**0**)	**0**(**0**)	+,=
	100	7.28E+01(1.07E+01)	7.36E-01(8.48E-01)	**0**(**0**)	+,+
	200	2.12E+02(2.12E+01)	1.76E+01(2.89E+00)	**2.18E-13**(**4.50E-14**)	+,+
Ackley	50	3.52E-02(1.74E-01)	**5.92E-15**(**1.79E-15**)	7.54E-15(0)	+,–
	100	1.87E+00(5.72E-01)	**7.84E-15**(**7.03E-16**)	1.91E-14(3.31E-15)	+,–
	200	5.30E+00(8.47E-01)	**1.09E-14**(**1.12E-15**)	3.79E-14(3.49E-15)	+,–
Griewank	50	5.08E-03(1.62E-02)	**0**(**0**)	**0**(**0**)	+,=
	100	8.43E-03(1.71E-02)	**0**(**0**)	**0**(**0**)	+,=
	200	1.33E-01(2.50E-01)	**1.11E-16**(**0**)	1.95E-16(7.27E-17)	+,–
Summary	(+,=,–)	(18,0,0)	(9,4,5)		

all 15 cases show that DEASCSM gives the best results for 11 cases, while PS-ABC, ABC-PS, and OXDE provide the best results for 9, 1, 0, and 0 cases, respectively. The summation values $(+, =, --)$ when compared with PS-ABC, ABC-PS, and OXDE are $(6, 6, 3)$, $(13, 0, 2)$, and $(15, 0, 0)$, respectively. They indicate that DEASCSM significantly outperforms ABC-PS, and OXDE and slightly outperforms PS-ABC.

4 Conclusion

We proposed the DEASCSM algorithm and evaluated its performance on high-dimensional optimization problems up to $D = 1000$. It employs an adaptive crossover strategy to balance exploration and exploitation, and a sorting mutation to enhance convergence. Tests using both VTR and $maxnf$ criteria show that DEASCSM reliably finds optimal solutions, even as evaluation demands grow with dimensionality. Its scalability varies with problem characteristics but, overall, it outperforms several DE variants and the compared methods in speed and accuracy.

Table 3: Performance comparison of DEASCSM with PS-ABC, ABC-PS, and OXDE using $maxnf = 10000D$ over 20 independent runs

Function	D	PS-ABC	ABC-PS	OXDE	DEASCSM	Significance
		$Mean\ fb$ (SD)	$Mean\ fb$ (SD)	$Mean\ fb$ (SD)	$Mean\ fb$ (SD)	
Sphere	60	**0**	2.4697E-03	2.0111E-20	**0**	=,+,+
		(**0**)	(5.7247E-04)	(1.8100E-20)	(**0**)	
	100	**0**	3.9066E-03	7.1141E-08	**0**	=,+,+
		(**0**)	(1.2206E-02)	(3.5769E-08)	(**0**)	
	500	**0**	4.0020E-04	2.9913E-03	1.9622E-55	–,+,+
		(**0**)	(2.8712E-03)	(4.0907E-03)	(5.0365E-56)	
Rosen-brock	60	5.8672E+01	4.8637E-02	3.9279E+01	**1.5037E-14**	+,+,+
		(3.6833E-02)	(1.5012E-01)	(2.2475E+01)	(**2.4422E-14**)	
	100	9.8219E+01	8.7094E+00	9.2730E+01	**4.1724E-01**	+,+,+
		(9.3594E-01)	(5.3352E+00)	(3.5371E+00)	(**5.4827E-01**)	
	500	4.0667E+02	**9.9222E+01**	1.5340E+03	3.9772E+02	+,–,+
		(8.2411E+01)	(**7.0812E+01**)	(1.9651E+02)	(2.0190E+01)	
Rastrigin	60	**0**	6.2663E-06	5.8873E+01	**0**	=,+,+
		(**0**)	(8.2557E-06)	(1.1657E+01)	(**0**)	
	100	**0**	7.7497E-05	3.2792E+02	**0**	=,+,+
		(**0**)	(5.5547E-04)	(7.8394E+01)	(**0**)	
	500	**0**	2.1558E-05	5.0365E+03	1.7961E+01	–,–,+
		(**0**)	(1.0934E-05)	(7.3559E+02)	(3.9802E-01)	
Ackley	60	3.1771E-09	3.3370E-11	2.0717E+01	**8.2600E-15**	+,+,+
		(2.0798E-09)	(2.5757E-11)	(1.8598E-01)	(**2.1316E-15**)	
	100	6.5777E-11	2.5021E-12	2.1133E+01	**1.7497E-14**	+,+,+
		(3.2900E-11)	(1.2578E-12)	(3.5337E-02)	(**3.4809E-15**)	
	500	3.6480E-10	2.0616E-05	1.9724E+01	**1.1697E-13**	+,+,+
		(1.4129E-10)	(8.0021E-05)	(4.8627E+01)	(**9.4397E-15**)	
Griewank	60	**0**	1.6354E-08	1.4792E-03	**0**	=,+,+
		(**0**)	(3.3446E-07)	(3.1184E-03)	(**0**)	
	100	**0**	1.0229E-08	1.1797E-09	**0**	=,+,+
		(**0**)	(3.6532E-06)	(6.0733E-10)	(**0**)	
	500	**0**	3.9642E-07	3.0348E-06	8.1046E-16	–,+,+
		(**0**)	(3.0278E-08)	(3.5308E-06)	(1.0590E-16)	
Summary	(+,=,–)	(6,6,3)	(13,0,2)	(15,0,0)		

Acknowledgement

The research on differential evolution algorithm with adaptive crossover and sorting mutation strategies for high-dimensional optimization by Khon Kaen University has received funding support from the National Science Research and Innovation Fund (NSRF).

References

[1] R. Storn, Differential evolution research-Trends and open questions. In Advances in Differential Evolution, Berlin, Germany, Springer, **143**, 2008, 1–31.

[2] R. Storn, K. Price, Differential evolution–A simple and efficient heuristic for global optimization over continuous spaces,J. Glob. Optim., **11**, 1997, 341–359.

[3] S. Piccand, M. O' Neill, J. Walker, On the scalability of particle swarm optimisation. In Proc. IEEE CEC2008, Hong Kong, China, Jun. 1–6, 2008, 2505–2512.

[4] Z. Li, W. Wang, Y. Yan, Z. Li, PS-ABC: A hybrid algorithm based on particle swarm and artificial bee colony for high-dimensional optimization problems, Expert Syst. Appl., **42**, no. 22, (2015), 8881–8895.

[5] W. Gong, Z. Cai, C. X. Ling, DE/BBO: a hybrid differential evolution with biogeography-based optimization for global numerical optimization, Soft Comput., **15**, (2010), 645-665.

[6] S. Tuo, J. Zhang, X. Yuan, L. Yong, A new differential evolution algorithm for solving multimodal optimization problems with high dimensionality, Soft Comput., **22**, (2018), 4361–4388.

[7] Y. E. Yildiz, A. O. Topal, Large scale continuous global optimization based on micro differential evolution with local directional search, Inf. Sci., **477**, (2019), 533–544.

[8] P. Puphasuk, J. Wetweerapong, An enhanced differential evolution algorithm with adaptation of switching crossover strategy for continuous optimization, Found. Comput. Decis. Sci., **45**, no. 2, (2020), 97–124.

[9] X. Li, K. Tang, M. N. Omidvar, Z. Yang, K. Qin, Benchmark Functions for the CEC'2013 special session and competition on large-scale global optimization, (2013).