Dynamic Perturbation for Population Diversity Management in Differential Evolution

Le Van Cuong
Hanoi University of Science and Technology
Hanoi, Vietnam
cuong.lv212179m@sis.hust.edu.vn

Nguyen Ngoc Bao
Hanoi University of Science and Technology
Hanoi, Vietnam
bao.nn193989@sis.hust.edu.vn

Nguyen Khanh Phuong
Hanoi University of Science and Technology
Hanoi, Vietnam
phuongnk@soict.hust.edu.vn

Huynh Thi Thanh Binh
Hanoi University of Science and Technology
Hanoi, Vietnam
binhht@soict.hust.edu.vn

ABSTRACT

The performance of Differential Evolution (DE) is closely related to the population diversity since its mechanism of generating offspring depends wholly on the differences between individuals. This paper presents a simple perturbation technique to maintain the population diversity in which the noise intensity is adjusted dynamically during the search. A modification of the well-known L-SHADE adaptation method is also introduced to manipulate the convergence behaviour of DE. By incorporating these techniques, we develop a new variant of DE called S-LSHADE-DP. Experiment results conducted on the benchmark suite of CEC '22 competition show that S-LSHADE-DP is highly competitive with current state-of-the-art DE-based algorithms. The implementation of S-LSHADE-DP is available at *https://github.com/cuonglvsoict/S-LSHADE-DP*.

CCS CONCEPTS

• **Theory of computation → Bio-inspired optimization**; • **Computing methodologies → Continuous space search**.

KEYWORDS

evolutionary computation, differential evolution, dynamic perturbation, numerical optimization

ACM Reference Format:
Le Van Cuong, Nguyen Ngoc Bao, Nguyen Khanh Phuong, and Huynh Thi Thanh Binh. 2022. Dynamic Perturbation for Population Diversity Management in Differential Evolution. In *Genetic and Evolutionary Computation Conference Companion (GECCO '22 Companion), July 9–13, 2022, Boston, MA, USA.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3520304.3529075

1 INTRODUCTION

Differential Evolution (DE) [5] is one of the most effective stochastic population-based algorithms for solving continuous optimization problems. Like many other evolutionary algorithms, diversity is a critical issue in Differential Evolution. Since the mechanism of producing offspring in DE completely depends on the deviations between population members, it can be seen that the performance of DE is even more sensitive to the population diversity than other algorithms. Once premature convergence and stagnation occur, the algorithm loses the capability of exploration. The search then comes to a standstill even though the budget of function evaluations has still not been used up.

In this paper, we focus on addressing the population diversity management problem in DE. A simple perturbation technique is proposed in which the perturbation intensity is adjusted dynamically during the course of the search. In this technique, if stagnation occurs with an individual in the population, the perturbation will be invoked, helping the individual escape from the local optima and continue to explore the search space. Besides, we also modify the L-SHADE adaptation [6] by a deterministic rule to control the convergence behaviour of the population. By incorporating these techniques, a new DE variant called Semi-LSHADE with Dynamic Perturbation (S-LSHADE-DP) is then introduced.

Comprehensive experiments are conducted over the latest set of challenging benchmarks from the IEEE CEC and GECCO 2022 competition on real parameter single-objective optimization [1]. The comparison of S-LSHADE-DP against winners of 2020 and 2021 competitions demonstrates that the proposed algorithm is highly competitive with the state-of-the-art algorithms. The sensitivity analysis also shows that each proposed technique has its contribution to the performance of S-LSHADE-DP.

2 DYNAMIC PERTURBATION

The perturbation method is called whenever stagnation occurs with an individual. For each individual $\overline{x}_i$, an indicator S_i is used to indicate its stagnant state, i.e., S_i is the number of consecutive generations passed without fitness improvement of $\overline{x}_i$ or change in its position in the search space. The value of S_i is initialized as zero and is updated at each generation as follows:

$$S_i = \begin{cases} S_i + 1, & \text{if } f(\overline{u}_i) > f(\overline{x}_i) \text{ or } ||\overline{u}_i - \overline{x}_i|| = 0 \\ 0, & \text{otherwise,} \end{cases} \quad (1)$$

where $\overline{u}_i$ is the trial solution generated from parent $\overline{x}_i$. Here, moves without fitness improvement are accepted to reset the indicator S_i to zero, helping the population to pass over flat fitness landscapes.

GECCO '22 Companion, July 9–13, 2022, Boston, MA, USA

ACM ISBN 978-1-4503-9268-6/22/07.
https://doi.org/10.1145/3520304.3529075

If the value of S_i reaches a predefined value S_{max}, $\overline{x}_i$ is perturbed to produce a new solution $\overline{x}_{new}$. We first generate a random vector $\overline{x}_{rand} = \langle x^1_{rand}, x^2_{rand}, \ldots, x^D_{rand} \rangle$ in the feasible domain using uniform distribution:

$$x^j_{rand} = x^j_{lb} + rand(0,1) \cdot (x^j_{ub} - x^j_{lb}). \tag{2}$$

New solution $\overline{x}_{new}$ is then obtained by combining $\overline{x}_i$ and $\overline{x}_{rand}$ arithmetically. Our experiences show that perturbing stagnant solutions by decaying the intensity works better than using fixed intensity. Thus, we change the perturbation intensity dynamically as follows:

$$x^j_{new} = \begin{cases} \alpha \cdot x^j_i + (1-\alpha) \cdot x^j_{rand}, & \text{if } rand(0,1) \le 0.5 \\ x^j_i, & \text{otherwise,} \end{cases} \tag{3}$$

where α is adjusted according to the number of consumed function evaluations FEs:

$$\alpha = \frac{FEs}{MAX\_FEs}. \tag{4}$$

The lower value of α in the early stages forces higher noise, and therefore, slows down the convergence and enhances the exploration ability. As time continues, the value of α increases resulting in more information of original solutions being retained. Exploitation around current solutions is thus accelerated gradually, leading to convergence. It is also notable that the perturbation is only carried out with the probability of 0.5 at each dimension. This probability helps to prevent the solutions from losing entire results accumulated during the course of the algorithm.

The indicator S_i is reset to zero after the perturbation. New solution $\overline{x}_{new}$ is evaluated and replaces $\overline{x}_i$ regardless of fitness values. This not only preserves the differences between population members but also avoids damaging other solutions since stagnant solutions often lay on local optima and attract other solutions.

3 S-LSHADE-DP ALGORITHM

This section introduces the algorithm S-LSHADE-DP which employs the proposed dynamic perturbation technique and the modified L-SHADE adaptation.

3.1 Mutation operators

We utilize two mutation strategies in S-LSHADE-DP:

1) DE/current-to-$p$best/1 with probability γ:

$$\overline{v}_i = \overline{x}_i + F \cdot (\overline{x}_{p\text{best}} - \overline{x}_i + \overline{x}_{r_1} - \overline{x}_{r_2}) \tag{5}$$

2) DE/target/1 with probability $1 - \gamma$:

$$\overline{v}_i = \overline{x}_i + F \cdot (\overline{x}_{r_1} - \overline{x}_{r_2}) \tag{6}$$

where $\overline{v}_i$ is the donor vector, $\overline{x}_{p\text{best}}$ is selected from $p\%$ top solutions, $\overline{x}_{r_1}$ is randomly selected from the current population, and $\overline{x}_{r_2}$ is randomly selected from the union of the current population and the external archive.

After every G generations, the value of γ is updated based on the effectiveness of mutation strategies in previous generations as follows:

$$\gamma = \begin{cases} 0.7, & \text{if } \frac{\Delta f_1}{FEs_1} > \frac{\Delta f_2}{FEs_2} \\ 0.3, & \text{otherwise,} \end{cases} \tag{7}$$

where Δf_i is the cumulative fitness improvement and FEs_i is the number of function evaluations consumed by the ith operator in previous G generations. The higher performance, the higher probability the operator is carried out. The computational resources are therefore allocated to operators dynamically according to their effectiveness.

3.2 Parameter adaptation

We modify L-SHADE adaptation technique [6], where the scale factor F is still adapted based on the success history, and the population size NP is also reduced linearly as in original works. The difference in this study is the setting of the crossover rate CR. Therefore, we name our version Semi-LSHADE (S-LSHADE).

The influences of crossover rate CR on the convergence behaviour of DE have been well-studied [7]. Pointing at an observation that premature convergence is not a concern at low CR, the crossover rate CR_i for each individual x_i at each generation is generated as follows:

$$CR_i = \begin{cases} 0, & \text{if } FEs \le 0.5 \cdot MAX\_FEs, \\ rand(0,1) & \text{otherwise.} \end{cases} \tag{8}$$

Incorporating with the binomial crossover, S-LSHADE-DP conducts independent searches at each axis during the first half of the search process. Small exploratory moves in this stage lead to a low convergence speed and prevent the population from crowding. Separable problems can be effectively solved in this time. In the later stage, CR is randomly generated in range (0, 1), resulting in a variety of search behaviours. High values of CR generated in this time conduct large moves on multiple axes, thus favoring non-separable problems and speeding up the convergence.

We now come to the overall framework of S-LSHADE-DP as shown in Algorithm 1.

Algorithm 1: The pseudocode of S-LSHADE-DP

```
t ⟵ 0;
Randomly initialize the population P_0;
Initialize S_i ⟵ 0 for each individual x̄_i in P_0;
while termination conditions are not satisfied do
    P_{t+1} ← Perform mutation, crossover, and selection
    on P_t;
    for each individual x̄_i in P_{t+1} do
        if S_i = S_max then
            x̄_new ⟵ Execute perturbation on x̄_i;
            Evaluate x̄_new and replace x̄_i by x̄_new;
            S_i ⟵ 0;
        end if
    end for
    Update γ as given in Eq. (7);
    Update NP and memory M_F; // L-SHADE adaptation
    t ⟵ t + 1;
end while
Return best solution found;
```

4 EXPERIMENTAL SET-UP AND RESULTS

To evaluate the performance of S-LSHADE-DP algorithm, we conduct different experiments on the benchmark suite of the CEC 2022

competition on real parameter single objective bound constrained optimization [1]. There are 12 test problems with two scales $D = 10$ ($D10$) and $D = 20$ ($D20$). Most of the parameters of S-LSHADE-DP are set to default values from the literature: the population sizes $N_{max} = 100$ and $N_{min} = 4$, $p = 10\%$. Besides, S_{max} is set to 100, γ is initialized as 0.3, and G is set to 20. The computational budget is set as the competition guideline: MAX_FEs = 200,000 for $D10$ and 1,000,000 for $D20$ [1]. The error value $|f(\overline{x}_{best}) - f(\overline{x}^*)|$ is used as the evaluation criteria, where $\overline{x}^*$ is the global optimal solution and $\overline{x}_{best}$ is the best solution obtained by an algorithm.

4.1 Comparison with state-of-the-art DE-based algorithms

In the first experiment, the overall performance of S-LSHADE-DP is compared with the winners of IEEE CEC competitions on real parameter single objective bound constrained optimization, including IMODE [3] (2020), APGSK_IMODE [2], and NL-SHADE-RSP [4] (2021). All configurations of these algorithms are retained as in original papers. The best, mean, and standard deviation of error values for 30 independent runs are shown in Table 1 and Table 2. A non-parametric test called Wilcoxon's rank-sum test (WRST) is conducted at 0.05 significance level to assess the significance of the outcomes. The Friedman test is also used to calculate the ranking of algorithms based on the mean error value. The results of the statistical test are summarized in the last two rows of the same tables, where "+", "=", and "-" indicate that the proposed algorithm exhibits superior, similar, and inferior performance, respectively.

On $D10$ test problems, the first impression when looking at the Table 1 is that out of the best results derived from all algorithms, 11 out of 12 are found by the proposed algorithm S-LSHADE-DP, except for only problem f_4. Among these best results, 8 out of 11 are global optimal solutions. In terms of statistical test results, Table 1 shows that S-LSHADE-DP performs better than or equal to IMODE and NL-SHADE-RSP over all 12 problems. Compared to APGSK_IMODE, the proposed algorithm S-LSHADE-DP statistically outperforms in two problems (f_8 and f_{10}), loses in three problems (f_4, f_6, and f_{12}), and achieves similar results in the remaining problems. On $D20$ problems, once again, Table 2 shows that S-LSHADE-DP derives 11 out of 12 best results found by all algorithms, except for only problem f_{12}. For statistical test results, S-LSHADE-DP is better or equal to all competitors in all 12 test problems. These achievements are much better than results in $D10$ problems and suggest the scalability of the proposed algorithm. Regarding to the results of Friedman tests, S-LSHADE-DP is the best performing algorithm with the lowest ranks for both dimensions $D10$ and $D20$.

4.2 Sensitivity analysis

In this subsection, we conduct sensitivity analysis to assess the contribution of each component to the overall performance of S-LSHADE-DP. We design two variants of S-LSHADE-DP, including S-LSHADE-DP without using dynamic perturbation method (S-LSHADE) and S-LSHADE-DP using the original L-SHADE adaptation without the modification in the setting of CR (LSHADE-DP). It can be seen from Table 3 that S-LSHADE-DP is the best performing version with the lowest ranking compared to other variants.

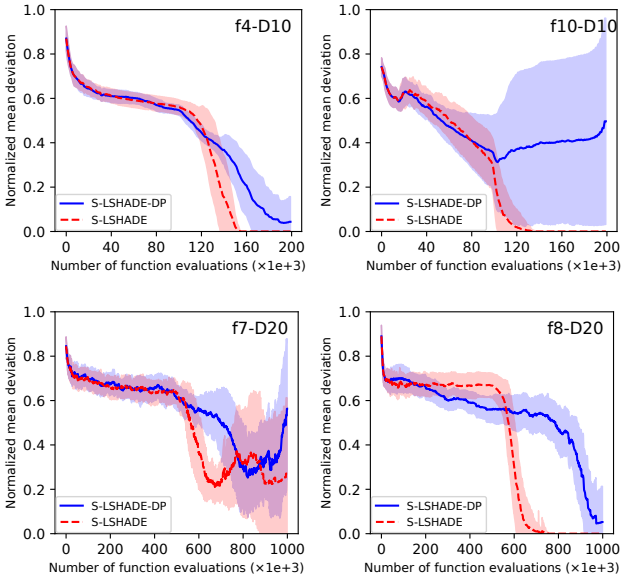

Figure 1: The normalized mean deviation of each solution from the best solution derived from S-LSHADE-DP and S-LSHADE on test problems of CEC '22 competition.

These results confirm the significant efficiency of the dynamic perturbation technique and the modification in the setting of CR of L-SHADE. Up to this point, we can conclude that the overall performance of S-LSHADE-DP is not contributed by a single component but the combination of all of them.

4.3 Further analysis of the population diversity

In this experiment, the influences of the perturbation technique and the configuration of CR on the population diversity are further examined. The mean deviation of all individuals from the best individual in the population that was used in IMODE [3] is used in this experiment to measure the population diversity. We study the mean deviation obtained by S-LSHADE-DP and its variant, S-LSHADE. The normalized mean deviation values of several test problems are plotted in Figure 1. It can be seen that in the early stage of the search process, both algorithms maintain a high deviation, i.e., high population diversity due to the setting of CR. However, the deviation decreases rapidly in S-LSHADE in the later stage suggesting that the populations have crowded in small regions around the best solutions. The shrinking is likely to lead to premature convergence, and this is where the perturbation technique comes in handy. Figure 1 shows a significant difference between the population diversity of two algorithms in the later stage.

5 CONCLUSION

In this study, we developed a new DE variant called S-LSHADE-DP. The algorithm employed two new techniques: the dynamic perturbation technique and the modification of L-SHADE adaptation. They are all simple yet effective in maintaining diversity and controlling the convergence behaviour of the population. Experiment

Table 1: Computational results derived from S-LSHADE-DP and state-of-the-art algorithms on the $D10$ test suite.

Func.	S-LSHADE-DP		IMODE			APGSK_IMODE			NL-SHADE-RSP		
	Best	Mean (Std.)	Best	Mean (Std.)	WRST	Best	Mean (Std.)	WRST	Best	Mean (Std.)	WRST
f1	**0.00E+00**	**0.00E+00 (0.00E+00)**	**0.00E+00**	**0.00E+00 (0.00E+00)**	=	**0.00E+00**	**0.00E+00 (0.00E+00)**	=	**0.00E+00**	**0.00E+00 (0.00E+00)**	=
f2	**0.00E+00**	**0.00E+00 (0.00E+00)**	**0.00E+00**	**0.00E+00 (0.00E+00)**	=	**0.00E+00**	**0.00E+00 (0.00E+00)**	=	**0.00E+00**	**0.00E+00 (0.00E+00)**	=
f3	**0.00E+00**	**0.00E+00 (0.00E+00)**	**0.00E+00**	**0.00E+00 (0.00E+00)**	=	**0.00E+00**	**0.00E+00 (0.00E+00)**	=	**0.00E+00**	**0.00E+00 (0.00E+00)**	=
f4	1.99E+00	4.72E+00 (1.33E+00)	4.02E+00	7.86E+00 (1.75E+00)	+	**9.95E-01**	**3.98E+00 (1.23E+00)**	-	5.97E+00	1.27E+01 (3.45E+00)	+
f5	**0.00E+00**	**0.00E+00 (0.00E+00)**	**0.00E+00**	1.87E-04 (1.01E-03)	=	**0.00E+00**	**0.00E+00 (0.00E+00)**	=	**0.00E+00**	**0.00E+00 (0.00E+00)**	=
f6	**1.01E-02**	2.60E-01 (1.27E-01)	1.32E-01	3.40E-01 (1.20E-01)	+	3.49E-02	**1.47E-01 (9.53E-02)**	-	2.52E-02	1.99E-01 (1.39E-01)	=
f7	**0.00E+00**	**0.00E+00 (0.00E+00)**	**0.00E+00**	4.67E-05 (1.20E-04)	=	**0.00E+00**	**0.00E+00 (0.00E+00)**	=	**0.00E+00**	1.60E-08 (8.64E-08)	=
f8	**1.52E-06**	1.89E-01 (2.73E-01)	2.98E-01	7.30E-01 (2.82E-01)	+	2.26E-02	2.73E-01 (2.29E-01)	+	1.14E-03	**1.01E-01 (1.63E-01)**	=
f9	**0.00E+00**	**1.91E+02 (8.54E+01)**	2.29E+02	2.29E+02 (0.00E+00)	=	2.29E+02	2.29E+02 (0.00E+00)	=	**0.00E+00**	2.14E+02 (5.72E+01)	=
f10	**0.00E+00**	**1.25E-02 (6.35E-02)**	3.60E+00	5.57E+01 (4.48E+01)	+	1.00E+02	1.00E+02 (3.40E-02)	+	**0.00E+00**	3.96E-02 (6.74E-02)	+
f11	**0.00E+00**	**0.00E+00 (0.00E+00)**	**0.00E+00**	**0.00E+00 (0.00E+00)**	=	**0.00E+00**	**0.00E+00 (0.00E+00)**	=	**0.00E+00**	**0.00E+00 (0.00E+00)**	=
f12	**1.59E+02**	1.62E+02 (1.77E+00)	**1.59E+02**	1.62E+02 (1.27E+00)	=	**1.59E+02**	**1.60E+02 (1.37E+00)**	-	1.60E+02	1.64E+02 (9.77E-01)	+
+/=/-				**4/8/0**			2/7/3			**3/9/0**	
Ranking		**2.04**		3.21			2.25			2.50	

Table 2: Computational results derived from S-LSHADE-DP and state-of-the-art algorithms on the $D20$ test suite.

Func.	S-LSHADE-DP		IMODE			APGSK_IMODE			NL-SHADE-RSP		
	Best	Mean (Std.)	Best	Mean (Std.)	WRST	Best	Mean (Std.)	WRST	Best	Mean (Std.)	WRST
f1	**0.00E+00**	**0.00E+00 (0.00E+00)**	**0.00E+00**	3.33E-07 (1.80E-06)	=	**0.00E+00**	**0.00E+00 (0.00E+00)**	=	**0.00E+00**	**0.00E+00 (0.00E+00)**	=
f2	**0.00E+00**	**4.03E-01 (1.21E+00)**	1.32E+01	4.55E+01 (9.21E+00)	+	1.32E-05	3.36E+01 (2.02E+01)	+	**0.00E+00**	4.53E+01 (1.22E+01)	+
f3	**0.00E+00**	**0.00E+00 (0.00E+00)**	**0.00E+00**	**0.00E+00 (0.00E+00)**	=	**0.00E+00**	**0.00E+00 (0.00E+00)**	=	**0.00E+00**	**0.00E+00 (0.00E+00)**	=
f4	**9.99E+00**	**1.34E+01 (2.94E+00)**	3.76E+01	5.13E+01 (7.36E+00)	+	1.59E+01	2.29E+01 (3.32E+00)	+	3.88E+01	7.12E+01 (1.20E+01)	+
f5	**0.00E+00**	2.98E-03 (1.61E-02)	1.65E+02	4.31E+02 (1.43E+02)	+	**0.00E+00**	**0.00E+00 (0.00E+00)**	=	**0.00E+00**	1.45E-01 (5.94E-01)	=
f6	**1.88E-01**	**2.14E+00 (2.72E+00)**	4.00E+02	9.01E+02 (3.45E+02)	+	1.20E+00	9.46E+00 (6.95E+00)	+	**1.88E-01**	4.48E+00 (3.78E+00)	+
f7	**0.00E+00**	1.31E+01 (8.57E+00)	1.07E+01	2.12E+01 (4.20E+00)	+	9.95E-01	**7.12E+00 (6.11E+00)**	=	2.98E-04	9.42E+00 (6.68E+00)	=
f8	**3.54E+00**	**1.87E+01 (4.88E+00)**	1.85E+01	2.16E+01 (7.07E-01)	+	1.30E+01	2.00E+01 (1.93E+00)	=	1.17E+01	1.97E+01 (1.72E+00)	+
f9	**1.81E+02**	**1.81E+02 (5.68E-14)**	**1.81E+02**	**1.81E+02 (5.68E-14)**	=	**1.81E+02**	**1.81E+02 (5.68E-14)**	=	**1.81E+02**	**1.81E+02 (5.68E-14)**	=
f10	**0.00E+00**	**0.00E+00 (0.00E+00)**	**0.00E+00**	6.24E-01 (1.13E+00)	+	1.00E+02	1.00E+02 (4.64E-02)	+	**0.00E+00**	**0.00E+00 (0.00E+00)**	=
f11	**0.00E+00**	**3.00E+01 (9.00E+01)**	3.00E+02	3.00E+02 (0.00E+00)	+	**0.00E+00**	1.70E+02 (1.49E+02)	+	3.00E+02	3.00E+02 (0.00E+00)	+
f12	2.31E+02	2.34E+02 (2.08E+00)	**2.29E+02**	2.38E+02 (2.82E+00)	+	2.30E+02	**2.33E+02 (1.13E+00)**	=	2.35E+02	2.42E+02 (4.10E+00)	+
+/=/-				**9/3/0**			**5/7/0**			**6/6/0**	
Ranking		**1.71**		3.46			2.17			2.67	

Table 3: Computational results derived from S-LSHADE-DP and its two variants on the $D20$ test suite.

Func.	S-LSHADE-DP	S-LSHADE		LSHADE-DP	
		Mean (Std.)	WRST	Mean (Std.)	WRST
f1	0.00E+00 (0.00E+00)	0.00E+00 (0.00E+00)	=	4.10E+02 (2.21E+03)	=
f2	4.03E-01 (1.21E+00)	6.18E-01 (1.62E+00)	=	4.75E+01 (2.02E+00)	+
f3	0.00E+00 (0.00E+00)	0.00E+00 (0.00E+00)	=	0.00E+00 (0.00E+00)	=
f4	1.34E+01 (2.94E+00)	8.12E+01 (1.36E+01)	+	1.54E+01 (2.99E+00)	+
f5	2.98E-03 (1.61E-02)	1.85E+02 (3.76E+02)	+	0.00E+00 (0.00E+00)	=
f6	2.14E+00 (2.72E+00)	1.12E+00 (8.41E-01)	=	3.12E+00 (3.15E+00)	+
f7	1.31E+01 (8.57E+00)	1.53E+01 (8.40E+00)	=	1.61E+01 (5.26E+00)	=
f8	1.87E+01 (4.88E+00)	1.95E+01 (3.65E+00)	+	2.04E+01 (6.01E-01)	=
f9	1.81E+02 (5.68E-14)	1.81E+02 (5.68E-14)	=	1.81E+02 (5.68E-14)	=
f10	0.00E+00 (0.00E+00)	0.00E+00 (0.00E+00)	=	9.71E+01 (1.80E+01)	+
f11	3.00E+01 (9.00E+01)	3.00E+01 (9.00E+01)	=	3.33E+02 (4.71E+01)	+
f12	2.34E+02 (2.08E+00)	2.39E+02 (3.55E+00)	+	2.37E+02 (3.35E+00)	+
+/=/-		4/8/0		6/6/0	
Ranking	**1.16**	1.75		2.33	

results on the test suite of CEC '22 competition showed that the proposed algorithm S-LSHADE-DP clearly outperforms in many test problems compared to state-of-the-art algorithms.

ACKNOWLEDGMENTS

Le Van Cuong was funded by Vingroup JSC and supported by the Master, PhD Scholarship Programme of Vingroup Innovation Foundation (VINIF), Institute of Big Data, code VINIF.2021.ThS.BK.10.

This research is supported by the International Technology Center Pacific (ITC-PAC) under Contract No. FA520919PA148 for Huynh Thi Thanh Binh.

REFERENCES

[1] Abhishek Kumar, Kenneth V. Price, Ali Wagdy Mohamed, Anas A. Hadi, and P. N. Suganthan. 2022. Problem Definitions and Evaluation Criteria for the 2022 Special Session and Competition on Single Objective Bound Constrained Numerical Optimization. *Nanyang Technological University, Tech. Rep* (2022).

[2] Ali Wagdy Mohamed, Anas A Hadi, Prachi Agrawal, Karam M Sallam, and Ali Khater Mohamed. 2021. Gaining-sharing knowledge based algorithm with adaptive parameters hybrid with IMODE algorithm for solving CEC 2021 benchmark problems. In *2021 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 841–848.

[3] Karam M Sallam, Saber M Elsayed, Ripon K Chakrabortty, and Michael J Ryan. 2020. Improved multi-operator differential evolution algorithm for solving unconstrained problems. In *2020 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 1–8.

[4] Vladimir Stanovov, Shakhnaz Akhmedova, and Eugene Semenkin. 2021. NL-SHADE-RSP Algorithm with Adaptive Archive and Selective Pressure for CEC 2021 Numerical Optimization. In *2021 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 809–816.

[5] Rainer Storn and Kenneth Price. 1997. Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization* 11, 4 (1997), 341–359.

[6] Ryoji Tanabe and Alex S Fukunaga. 2014. Improving the search performance of SHADE using linear population size reduction. In *2014 IEEE congress on evolutionary computation (CEC)*. IEEE, 1658–1665.

[7] Daniela Zaharie. 2009. Influence of crossover on the behavior of differential evolution algorithms. *Applied soft computing* 9, 3 (2009), 1126–1138.