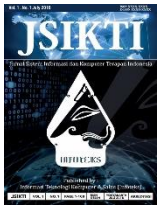
7 (3) March 2025 127-137
JSIKTI: *Jurnal Sistem Informasi dan Komputer terapan Indonesia*
ISSN 2460-7258 (online) | ISSN 1978-1520 (print)

Contents lists available at www.infoteks.org

JSIKTI

Journal Page is available to https://infoteks.org/journals/index.php/jsikti



Research article

Adaptive Operator and Scaling Factor Selection in Differential Evolution using Parametrized Reinforcement Learning

Kadek Gemilang Santiyuda [a], Putu Sugiartawan [b], Gede Agus Santiago [c], Ni Nengah Dita Ardriani [d], Moch Ilham Nur Kafiyanna [e],*

[a,b] *Master of Informatics, Institut Bisnis dan Teknologi Indonesia, Denpasar, Indonesia*
[c,d,e] *Department Informatics Engineering, Institut Bisnis dan Teknologi Indonesia, Denpasar, Indonesia*
email: [a,*] Gemilang.santiyuda@instiki.ac.id, [b] putu.sugiartawan@instiki.ac.id, [c] agus.santiago@instiki.ac.id, [d] dita.ardriani@instiki.ac.id, [e] ilham.nur.kafiyana@instiki.ac.id
* *Correspondence*

ARTICLE INFO

Article history:
Received 7 December 2024
Revised 21 January 2025
Accepted 01 March 2025
Available online 27 March 2025

Keywords:
Differential Evolution, Reinforcement Learning, Parameter Control, Vehicle Routing Problem, 3D Loading

Please cite this article in IEEE style as:
K. G. Santiyuda, P. Sugiartawan, G. A. Santiago, N. N. D. Ardriani, and M. I. N. Kafuyanna, "Adaptive Operator and Scaling Factor Selection in Differential Evolution using Parametrized Reinforcement Learning," *JSIKTI: Jurnal Sistem Informasi dan Komputer Terapan Indonesia*, vol. 7, no. 3, pp. 127–157, 2025.

ABSTRACT

Mutation strategy selection along with parameter settings are well known challenges in enhancing the performance of differential evolution (DE). In this paper, we propose to solve these problems as a parametrized action Markov decision process. A multi-pass deep Q-network (MP-DQN) is used as the reinforcement learning method in the parametrized action space. The architecture of MP-DQN comprises an actor network and a Q-network, both trained offline. The networks' weights are trained based on the samples of states, actions and rewards collected on every DE iterations. We use 99 features to describe a state of DE and experiment on 4 reward definitions. A benchmark study is carried out with functions from CEC2005 to compare the performance of the proposed method to baseline DE methods without any parameter control, with random scaling factor, and to other DEs with adaptive operator selection methods, as well as to the two winners of CEC2005. The results show that DE with MP-DQN parameter control performs better than the baseline DE methods and obtains competitive results compared to the other methods.

1. Introduction

Differential evolution (DE) [40, 41] is one of the well-known evolutionary algorithms along with others, such as genetic algorithm [16], cuckoo search [49], and particle swarm optimization [26]. Differential evolution has also been successfully adopted to solve design and optimization problems, ranging from electrical power system [6, 43], neural network training [4, 42], to logistic [31, 51], and finance [1, 20].

The appeal of DE is its explicit mutation scheme, which makes it straightforward to implement and open for variation of the used mutation operators. This variation plays an important role in DE performance as it has been shown that it can theoretically guarantee global convergence in DE, which the classical DE previously cannot guarantee [23]. However, the global convergence property of DE variant is not always reflected by its performance in practice. In practice, the performance of different DE mutation operators varies from problem to problem [30]. In addition, it's also been shown that using different mutation operators in different optimization stages of DE can improve its performance [14]. Therefore, the challenge is in selecting the mutation operators that yield the best improvement to the solution population at different stages of optimization and in different states of the population.

http://doi.org/10.26594/register.v8i2.2541

ISSN 2460-7258 (online) | ISSN 1978-1520 (print)
7 (3) March

Other influential components of DE are the crossover rate $\boldsymbol{CR}$ and the scaling factor $\boldsymbol{F}$ parameters. Similar to mutation operators, a wide variety of $\boldsymbol{CR}$ and $\boldsymbol{F}$ parameter settings are recommended for different problems [30, 41, 47]. If well designed, adaptive CR and F values can enhance the robustness and the convergence rate of DE [12].

The merit of having tunable and influential operators and parameters in evolutionary algorithm is the flexibility to perform relatively well on a wide range of problems by fine-tuning the operator and the parameters based on the problems or even based on the problem instances at hand. Several parameter tuning methods for evolutionary algorithm are available [11]. However, in some cases with elaborate function evaluations, parameter tuning can be expensive, time consuming, and even impractical for some real-world applications [10].

Based on those findings, there have been great interest in the study of adaptive operator selection (AOS) in selecting discrete parameters (often reproduction operators) and adaptive parameter control (APC) in selecting the appropriate value of continuous parameters. By adopting the strategy of adaptive control, the algorithm can be utilized to solve new problems, and adapted to different stage of problem optimization without tuning the parameters and the operator choice beforehand.

In the case of DE, several AOS and APC methods have been proposed to be incorporated in DE. One well known example is self-adaptive DE (SaDE) [35], which adaptively adjusts the probability of choosing between two mutation operators based on the count of successful and failed use of the operator, while the values of $\boldsymbol{CR}$ and $\boldsymbol{F}$ are randomly drawn from a Gaussian distribution, with adaptive Gaussian mean for the value of $\boldsymbol{CR}$. Another example is an adaptive DE proposed in [19], which uses probability matching (PM) [17] and adaptive pursuit [46] to select the appropriate mutation operator. In this case, the value of $\boldsymbol{CR}$ is drawn from a Gaussian distribution with adaptive mean, and $\boldsymbol{F}$ is drawn from a Cauchy distribution with adaptive location parameter, as proposed in [50].

In general, various AOS methods have been developed [39]. Several of them are based on reinforcement learning (RL), e.g., PM [17], bandit-based [13], Q(λ)-learning [34], and SARSA [7, 9, 24]. A recent addition to the RL-based AOS is a double deep Q-network (DDQN) [21] AOS in DE (DE-DDQN) [37], in which Q-value approximation function for each mutation operator is trained offline by solving several problem instances from CEC2005 problems [44]. Afterwards, DE-DDQN is used to solve new unseen problems with the learned Q-value approximation function used to select the appropriate mutation operator using a greedy policy. An earlier study is also carried out using RL as AOS method with offline training using FL-FALCON as the function approximator [45]. Common state features extracted from the population are population diversity both in decision and solution spaces, as well as stagnation count and parents' solution quality [24, 45]. The history of operators' performance can also be used as the state feature [37].

To the best of our knowledge, there has been only one study proposed to use RL to control both discrete, including reproduction operators, and continuous parameters at the same time, namely [24]. However, in this study a continuous parameter is still discretized into several ranges beforehand, and the number of available ranges becomes the new user-defined parameter.

In this paper, we propose an adaptive mutation operator and scaling factor selection in DE using multi-pass deep Q networks (MP-DQN) [5]. The proposed method is called DE-MPDQN in brief. The crossover rate $\boldsymbol{CR}$ however is set to 1 so that the effect of the choice of mutation operator and the scaling factor $\boldsymbol{F}$ can be fully observed. Four of the most frequently used mutation operators are used in the mutation operator pool. The scaling factor is treated as the parameter of the chosen mutation operator. Thus, the problem of choosing the mutation operator and the scaling factor can be treated as solving a parameterized action Markov decision process (PAMDP) [29].

MP-DQN has a similar architecture to its predecessor, parameterized deep Q-network (P-DQN) [48]. A P-DQN comprises an actor network that determines the values of the continuous action-parameters given the state, and a Q-network to approximate the Q-values of the actions given the state padded with all the previously determined parameters. The action that has the maximum approximated Q-values will then be chosen, alongside with its corresponding parameters. However,

http://doi.org/10.26594/register.v8i2.XXX

ISSN 2460-7258 (online) | ISSN 1978-1520 (print)
7 (3) March

passing the state and all the parameters once to the Q-network results in non-zero gradients with respect to the parameters of the non-chosen actions, which is termed as false gradients [5]. This is prevented in MP-DQN by decomposing the pass to the Q-network into multiple pass. The main difference between our proposed method and previously RL-based AOS methods is the ability of MP-DQN to control both discrete and continuous parameters without discretization of the continuous parameters. The MP-DQN model can also be trained offline and online.

In this paper, 99 state features, as proposed in [37], are used as information about population diversity, current algorithm performance, and parent solutions' performance. They are extracted from the population as the input to the MP-DQN networks. As other AOS methods, MP-DQN also faces the problem of designing the suitable reward definition, also known as the credit assignment problem. A different reward definition can affect the learning behavior of the RL agent for AOS and ultimately affects the performance of the evolutionary algorithm, as shown in [25, 37]. In this paper, we experiment on two different reward functions and each one is tested with and without negative rewards, making a total of 4 reward definitions.

An experimental study is carried out with five functions of CEC2005 special session on real-parameter optimization [44]. Each function is tested for dimensions of 10 and 30, which in total makes 10 test functions. The performance of MP-DQN is compared to five baseline DE methods. The baseline DE methods consists of DE that chooses the mutation operator randomly and DE that only uses one single mutation operator for each of the four available mutation operators. The performance of DE-MPDQN is also compared to other DE with other AOS methods (PM-AdapSS [15], F-AUC [18], RecPM-AOS [38], and DE-DDQN [37]), and the two winners of CEC2005 competition LR-CMAES [2] and IPOP-CMAES [3].

The rest of this paper is organized as follows. A brief introduction of DE and MP-DQN is given in the Section 2. The proposed method, including the state features and the reward definition, is described in Section 3. The experimental study, including the experimental settings and the results are given in Section 4. Lastly, the findings of the paper are summarized in Section 5.

2. Research Methods

This research proposes an adaptive approach for selecting mutation operators and scaling factors in Differential Evolution (DE) using a reinforcement learning technique called Multi-Pass Deep Q-Network (MP-DQN). The method treats the selection of operators and associated parameters as a **parameterized action Markov decision process (PAMDP)**, as introduced in [29].

2.1. MP-DQN Architecture

The MP-DQN framework is an extension of the Parameterized Deep Q-Network (P-DQN) [48], which uses two neural networks:

1. An **actor network** that generates continuous parameter values (scaling factors).
2. A **Q-network** that estimates the action-value $\mathbf{Q(s,(a,X_a))}$ for each pair of discrete action aaa and its continuous parameters $\mathbf{X_a}$.

A core limitation of P-DQN is that it computes gradients for non-selected actions, introducing **false gradients**. To address this, MP-DQN uses **multiple forward passes**, each isolating one action and its parameter vector. This decomposition ensures that Q-values are correctly influenced only by the associated parameter, eliminating false gradients and improving training stability.

The multi-pass strategy leverages batched input processing (e.g., via TensorFlow or PyTorch) to evaluate all actions in parallel, which makes the algorithm efficient and scalable

2.2. *Parameterized Action Space (PAMDP)*

A PAMDP is a generalization of standard MDPs that allows each action to be associated with a vector of continuous parameters. Formally, the action space is:

$$A = \bigcup_{a \in \mathcal{A}} \{(a, x_a) \mid x_a \in \mathcal{X}_a\} \quad (1)$$

http://doi.org/10.26594/register.v8i2.XXX

ISSN 2460-7258 (online) | ISSN 1978-1520 (print)
7 (3) March

where $\boldsymbol{A}$ is the set of discrete mutation operators, and $\mathcal{X}_a \subset \mathbb{R}^m$ represents the space of continuous parameters (in our case, scaling factors).

This formulation allows DE to adaptively select not just which mutation strategy to use, but also how to apply it by choosing the most suitable scaling factor in a given optimization context.

2.3. State Features

The RL agent's decision is based on **99 state features** as proposed in [37], which describe:

1. **Population diversity** in decision and objective space.
2. **Fitness performance** of parent and offspring solutions.
3. **Stagnation metrics**, remaining budget.
4. **Operator performance history** including four types of offspring metrics (OM1 to OM4), which measure improvement over current, best, and median solutions.

The final 16 features (features 84–99) maintain a FIFO window of recent improvements to track temporal patterns, enabling better credit assignment for operator performance

2.4. Reward Definitions

Four different reward functions are tested:

1. $\boldsymbol{R_1}$: Fitness improvement normalized by the optimal gap (non-negative).
2. $\boldsymbol{R_2}$: Same as $\boldsymbol{R_1}$, but allows negative rewards.
3. $\boldsymbol{R_3}$: Rule-based reward — high reward for improving best solution.
4. $\boldsymbol{R_4}$: Similar to $\boldsymbol{R_3}$, but gives penalty (-1) if no improvement occurs.

These rewards aim to solve the credit assignment problem, which is crucial in reinforcement learning as they directly impact the learning behavior and policy optimization [25],[7].

Table 1. The Summary of the State Features

http://doi.org/10.26594/register.v8i2.XXX

ISSN 2460-7258 (online) | ISSN 1978-1520 (print)
7 (3) March

Index	Feature	Notes
1	$\frac{f(x_i)-f(x_{bsf})}{f(x_{wsf})-f(x_{bsf})}$	The current considered solution is denoted as x_i. The solution with the worst fitness, and the solution with the best fitness throughout the 25 runs of each problem instance are respectively denoted as x_{wsf} and x_{bsf}.
2	$\frac{\sum_{j=1}^{NP}\frac{f(x_j)}{NP}-f(x_{bsf})}{f(x_{wsf}-f(x_{bsf}))}$	The population size is denoted as NP.
3	$\frac{std_{j=1,\ldots,NP}(f(x_j))}{std^{max}}$	$std(.)$ is the operator to compute the standard operation, and std^{max} is the standard deviation of the population fitness when half of the population fitness is $f(x_{wsf})$ and the other half is $f(x_{bsf})$.
4	$\frac{FE^{max}-t}{FE^{max}}$	The maximum function evolution for each run is denoted as FE^{max}. This feature gives a sense of budget information for the agent.
5	$\frac{dim_f}{dim^{max}}$	The dimension of the decision space of the specific problem instance f is denoted as dim_f while dim^{max} denotes the maximum decision space dimension of all problem instances
6	$\frac{stagcount}{FE^{max}}$	The number of function evaluation without improvement of $f(x_{bsf})$ is recorded and denoted as stagcount (stagnation counter).
7-11	$\frac{d(x_i,x_j)}{d^{max}}, j \in r_1, r_2, r_3, r_4, r_5$	$d(x, y)$ computes the Euclidean distance between solution x and y in the decision space while d^{max} denotes the maximum distance in the decision space of a specific problem instance which is the distance between the upper bound and the lower bound of the decision space. The index of the randomly selected parent solutions is denoted as r_1, r_2, r_3, r_4, and r_5.
12	$\frac{d(x_i,x_rbest)}{d^{max}}$	The index of the best solution among the randomly selected parent is denoted as $rbest$.
13-17	$\frac{f(x_i)-f(x_j)}{f(x_wsf)-f(x_bsf)}, \forall j \in r_1, r_2, r_3, r_4, r_5$	
18	$\frac{f(x_i)-f(x_rbest)}{f(x_wsf)-f(x_bsf)}$	
19	$\frac{d(x_i,x_bsf)}{d^{max}}$	
20-35	$\sum_{g=1}^{gen}\frac{N_m^{succ}(g,op)}{N^{tot}(g,op)}$	The current generation is denoted as gen. For each operator $op \in [1, 4]$, $N_m^{succ}(g, op)$ and $N^{tot}(g, op)$ are respectively the count of successful application operator op in the g-th generation according to the m-th offspring metric OM_m and the total count of application of operator op in the g-th generation.
36-51	$\frac{\sum_{g=1}^{gen}\sum_{k=1}^{N_m^{succ}(g,op)}OM_m(g,k,op)}{\sum_{g=1}^{gen}N^{tot}(g,op)}$	
52-67	$\frac{OM_m^{best}(gen,op)-OM_m^{best}(gen-1,op)}{OM_m^{best}(gen-1,op)\lvert N^{tot}(gen,op)-N^{tot}(gen-1,op)\rvert}$	
68-83	$\sum_{g=1}^{gen}OM_m^{best}(g,op)$	
84-99	$\sum_{w=1}^{W}OM_m(w,op)$	The w-th value of the recorded in the offspring metric window generated operator op is denoted as $OM_m(w, op)$.

2.5. Training and Evaluation

The agent is trained offline using 32 benchmark problems from the CEC2005 real-parameter optimization suite [44]. Each problem is tested in 10- and 30-dimensional spaces. During training, the mean reward across episodes is monitored, and the best-performing model is saved. Evaluation is then performed on 10 unseen test functions to assess generalization. Each experiment is repeated 25 times, and performance is measured using error statistics [37].

The neural networks used in MP-DQN have four hidden layers with 100 neurons each and employ the ReLU activation function [33]. An inverting gradient technique is used to constrain scaling factor values, while ε-greedy exploration and Ornstein-Uhlenbeck noise enhance exploration during training [22], [28].

Table 2. The Parameter Settings for the Experimental Study

http://doi.org/10.26594/register.v8i2.XXX

ISSN 2460-7258 (online) | ISSN 1978-1520 (print)
7 (3) March

Parameter	Value
Population size NP	100
Crossover rate CR	1
Maximum number of FE FE^{max}	10^5
Discount rate γ	0.9
Replay buffer maximum size N	10^6
Replay buffer warmup size N_w	10^4
Batch size B	64
Q-network learning rate α	10^{-3}
Actor learning rate β	10^{-4}
Q-network averaging weight τ_Q	0.1
Actor averaging weight τ_x	0.001
Hidden layers	[100,100,100,100]
Hidden layer activation function	ReLU[33]

3. Results and Discussion

Table 3. Mean and standard deviation of function error values obtained by 25 runs for each function on test set. Former five are dimension 10 and last five are dimension 30. We refer DE-DDQN as DDQN. Bold entry is the minimum mean error value found by any method for each function

Problem	Score	Random	DE1	DE2	DE3	DE4	AdapSS	FAUC	RecPM	LR	IPOP	DDQN	MPDQN1	MPDQN2	MPDQN3	MPDQN4
F3-10	Average	2.34E+08	2.78E+08	2.26E+08	2.38E+08	2.63E+08	3.37E+04	3.53E+05	3.08E+04	**4.94E-09**	5.60E-09	7.38E+00	2.14E+06	8.80E+06	4.91E+05	6.98E+02
	Std	1.06E+08	1.30E+08	1.10E+08	1.23E+08	1.42E+08	3.62E+05	1.65E+04	2.64E+04	1.45E-09	1.93E-09	3.59E+00	3.30E+06	3.11E+06	6.28E+05	8.81E+02
F9-10	Average	1.20E+02	1.18E+02	1.22E+02	1.16E+02	1.22E+02	4.10E+01	4.36E+01	3.79E+01	8.60E+01	**6.21E+00**	3.68E+01	1.93E+01	7.38E+01	2.12E+01	2.63E+01
	Std	1.32E+01	1.20E+01	1.88E+01	1.44E+01	1.71E+01	6.36E+00	5.99E+00	6.33E+00	3.84E+01	2.10E+00	4.64E+00	1.21E+01	9.75E+00	8.04E+00	5.77E+00
F16-10	Average	6.46E+02	6.50E+02	6.31E+02	5.91E+02	6.33E+02	1.90E+02	2.05E+02	1.89E+02	1.49E+02	**1.11E+02**	1.79E+02	1.58E+02	2.88E+02	1.54E+02	1.38E+02
	Std	1.02E+02	9.65E+01	1.15E+02	1.07E+02	9.97E+01	2.21E+01	1.41E+01	1.25E+01	8.01E+01	1.66E+01	2.05E+01	1.70E+01	4.19E+01	2.95E+01	1.86E+01
F18-10	Average	1.33E+03	1.36E+03	1.39E+03	1.36E+03	1.36E+03	6.13E+02	6.94E+02	6.48E+02	8.40E+02	6.02E+02	5.81E+02	5.88E+02	6.89E+02	9.40E+02	**5.12E+02**
	Std	1.16E+02	8.81E+01	1.11E+02	1.09E+02	9.67E+01	1.67E+02	1.93E+02	1.82E+02	2.17E+02	2.76E+02	2.47E+02	2.23E+02	1.10E+02	9.01E+01	2.36E+02
F23-10	Average	1.49E+03	1.51E+03	1.51E+03	1.51E+03	1.49E+03	6.66E+02	7.73E+02	6.37E+02	1.22E+03	9.49E+02	6.56E+02	7.82E+02	1.17E+03	1.01E+03	**6.15E+02**
	Std	5.16E+01	6.71E+01	6.03E+01	5.58E+01	4.97E+01	1.99E+02	2.05E+02	1.23E+02	5.16E+02	3.52E+02	1.57E+02	2.12E+02	8.29E+01	2.27E+02	1.11E+02
F3-30	Average	2.48E+09	2.68E+09	2.50E+09	2.65E+09	2.51E+09	1.52E+07	6.44E+07	1.31E+07	**1.28E+06**	6.11E+06	3.06E+06	4.46E+07	9.50E+07	2.13E+08	1.47E+07
	Std	6.60E+08	7.84E+08	9.04E+08	6.69E+08	8.22E+08	5.50E+07	5.88E+06	6.84E+06	7.13E+05	3.79E+06	2.54E+06	2.73E+07	2.94E+07	1.24E+08	3.95E+06
F9-30	Average	5.33E+02	5.27E+02	5.42E+02	5.19E+02	5.41E+02	2.54E+02	2.88E+02	2.53E+02	4.19E+02	**4.78E+01**	2.39E+02	2.72E+02	2.15E+02	2.31E+02	2.31E+02
	Std	3.09E+01	3.40E+01	3.73E+01	4.53E+01	3.43E+01	2.69E+01	1.72E+01	1.26E+01	1.02E+02	1.15E+01	1.52E+01	7.72E+01	2.27E+01	2.60E+01	1.17E+01
F16-30	Average	1.19E+03	1.18E+03	1.18E+03	1.21E+03	1.20E+03	3.11E+02	3.48E+02	2.97E+02	2.52E+02	**1.96E+02**	3.74E+02	2.97E+02	3.92E+02	4.05E+02	3.50E+02
	Std	1.36E+02	1.72E+02	1.16E+02	1.35E+02	1.63E+02	6.26E+01	5.27E+01	3.00E+01	2.08E+02	1.45E+02	9.03E+01	8.40E+01	1.05E+02	1.39E+02	3.32E+01
F18-30	Average	1.41E+03	1.43E+03	1.41E+03	1.42E+03	1.42E+03	9.65E+02	1.02E+03	9.71E+02	9.64E+02	**9.08E+02**	9.45E+02	1.02E+03	1.11E+03	1.10E+03	9.78E+02
	Std	5.70E+01	4.70E+01	6.47E+01	4.59E+01	5.54E+01	5.59E+01	2.37E+01	2.31E+01	1.46E+02	2.76E+00	1.42E+01	6.67E+01	5.40E+01	6.70E+01	2.20E+01
F23-30	Average	1.58E+03	1.57E+03	1.55E+03	1.57E+03	1.57E+03	9.43E+02	1.10E+03	9.67E+02	7.51E+02	**6.92E+02**	9.74E+02	1.17E+03	1.25E+03	1.26E+03	1.17E+03
	Std	4.64E+01	4.05E+01	4.51E+01	4.14E+01	5.15E+01	1.40E+02	1.01E+02	1.30E+02	3.30E+02	2.38E+02	1.69E+02	5.69E+01	2.97E+01	3.99E+01	1.01E+02

3.1. Experimental Setup

To evaluate the proposed DE-MPDQN approach, a comprehensive set of experiments was conducted using benchmark functions from the CEC2005 suite. A total of 32 functions (with dimensions 10 and 30) were used during training, while 10 unseen functions were selected for evaluation. Each experiment was run 25 times independently, and results were averaged to ensure statistical significance.

http://doi.org/10.26594/register.v8i2.XXX

ISSN 2460-7258 (online) | ISSN 1978-1520 (print)
7 (3) March

In this experiment, the DE-MPDQN was implemented with MP-DQN agents consisting of multi-layer perceptron networks. Each network had four hidden layers of 100 ReLU-activated neurons. The reward functions R1 to R4 were used to create four variants: DE-MPDQN1 through DE-MPDQN4. A total of 32 benchmark functions (16 from the CEC2005 suite, in both 10D and 30D) were used for training, and 10 distinct functions for testing. Non-deterministic and unbounded functions (e.g., F4, F7, F17, F25) were excluded from the experiments.

To ensure robustness and statistical validity, each algorithm was independently run 25 times per test function, and the performance was measured using mean final error and standard deviation. Function evaluations were limited to 10^5 FEs per run, or terminated early if a solution reached an error below 10^{-8}. This setup ensures fair comparison under consistent evaluation budgets.

The VRP-3L problem was also included as a real-world test case, which reflects the algorithm's potential for industrial deployment beyond synthetic benchmarks.

3.2. Comparative Performance

The performance of DE-MPDQN was compared against:

1. Five baseline DE variants: one random operator selection and four fixed mutation strategies.
2. Four state-of-the-art AOS-based DE algorithms: **PM-AdapSS, F-AUC, RecPM-AOS, and DE-DDQN**.
3. Two CEC2005 winners: **LR-CMAES** and **IPOP-CMAES**.

The results (see Table 3 of the source) confirm that DE-MPDQN4, which uses a rule-based reward with penalties (R4), consistently delivers lower mean error values across the majority of test functions. This trend is apparent even when compared to advanced AOS algorithms such as DE-DDQN and RecPM-AOS.

DE-MPDQN variants significantly outperform the baseline DE variants, including those with fixed mutation strategies (DE1–DE4) and random operator selection. This highlights the benefit of dynamic operator and scaling factor adaptation.

Among state-of-the-art methods, DE-MPDQN4's performance is competitive with or better than RecPM-AOS, DE-DDQN, and even the 2005 CEC winners, IPOP-CMAES and LR-CMAES, in multiple problem instances. While DE-DDQN also uses offline training, its action space is limited to discrete mutation operators, whereas DE-MPDQN can control both operator selection and continuous scaling factors jointly — an essential feature for complex optimization problems.

3.3. Statistical Analysis

The Friedman test was used to determine the significance of differences among all tested algorithms. Results indicated a statistically significant difference ($p < 0.01$). A post-hoc Nemenyi test using DE-MPDQN4 as the control method revealed that its performance was significantly better than the five DE baseline variants.

While DE-MPDQN4 outperformed most competitors, its differences compared to RecPM-AOS, DE-DDQN, and IPOP-CMAES were not statistically significant, suggesting comparable optimization capability among the top methods.

The **Friedman test** was conducted to evaluate the overall significance of differences among the 13 compared algorithms. With **$p < 0.01$**, the test confirms that there is a statistically significant performance gap among the methods.

A **post-hoc Nemenyi test** was performed using DE-MPDQN4 as the control algorithm. The results indicate that DE-MPDQN4's superiority is statistically significant when compared to all **five DE baselines**. This affirms the effectiveness of the MP-DQN-based parameter selection mechanism.

However, differences between DE-MPDQN4 and high-performing competitors like **RecPM-AOS**, **DE-DDQN**, and **IPOP-CMAES** were **not statistically significant**, indicating that these methods operate at a similarly high level. Nevertheless, DE-MPDQN offers the **added advantage of joint discrete-continuous adaptation**, a feature not fully exploited by the others.

These statistical results support the conclusion that DE-MPDQN4 is not only effective but also **adaptively reliable**, especially for real-world applications where parameter control must be generalized across problem instances.

3.4. Effectiveness of Reward Functions

http://doi.org/10.26594/register.v8i2.XXX

ISSN 2460-7258 (online) | ISSN 1978-1520 (print)
7 (3) March

The four variants of DE-MPDQN were differentiated by the reward functions used:

1. **DE-MPDQN1 and DE-MPDQN2**: Based on normalized fitness improvement.
2. **DE-MPDQN3 and DE-MPDQN4**: Based on rule-based scoring (improvement vs. best or current solution), with DE-MPDQN4 using negative penalties for non-improvement.

Among them, **DE-MPDQN4 consistently delivered the best results**, confirming that incorporating negative rewards improved the agent's learning efficiency by encouraging exploration and penalizing non-contributive actions.

The results from the benchmark experiments clearly show that reward design has a significant impact on the learning performance of the MP-DQN agent. In particular, DE-MPDQN4, which incorporates **negative rewards for non-improving offspring**, consistently outperforms other variants. This aligns with reinforcement learning theory, where the presence of **punitive signals** helps the agent avoid unproductive actions, thus accelerating convergence and improving decision quality.

While DE-MPDQN1 and DE-MPDQN2 are based on normalized fitness improvements, they often suffer from a lack of penalty, making the learning process slower and more prone to premature convergence. On the other hand, the reward functions R3 and R4 (used in DE-MPDQN3 and DE-MPDQN4 respectively) provide a **more robust credit assignment**, distinguishing between regular improvements and breakthroughs toward the global optimum.

This confirms that **reward shaping**—especially through the inclusion of negative signals—plays a crucial role in guiding learning behavior and enhancing long-term performance in parameterized reinforcement learning setups.

3.5. Adaptability and Generalization

An important finding is that the proposed method successfully generalizes to unseen problem instances. The agent was trained offline on benchmark problems and was then applied to solve different test functions without further training. This shows the robustness and adaptability of DE-MPDQN in handling diverse optimization challenges.

One of the major strengths of the DE-MPDQN approach is its demonstrated **ability to generalize across problem instances**. The training process only needs to be conducted once, using a broad selection of benchmark functions that represent a wide range of landscape characteristics, including unimodal, multimodal, and hybrid functions.

Once trained, the agent's policy can be **directly applied to unseen test functions without further tuning**, and yet still achieve state-of-the-art performance. This property significantly increases its value in real-world optimization scenarios, where re-training is often impractical due to time, computational, or data constraints.

In comparison to traditional DE methods or even other AOS approaches, which often require **problem-specific adjustments**, DE-MPDQN proves to be a **truly adaptive controller** that understands context and adjusts operator usage and parameter scaling accordingly.

3.6. Computational Cost

Although training DE-MPDQN requires considerable computational resources, this is a one-time cost. Once trained, the model can be reused for multiple problems with no need for re-tuning, making it suitable for real-world applications where optimization speed and adaptability are critical.

The training phase of DE-MPDQN is computationally intensive, involving thousands of training episodes and millions of function evaluations. However, this cost is incurred only once, and can be significantly reduced through the use of GPU acceleration and parallel processing.

In the long run, the amortized cost of training becomes negligible compared to the benefits of rapid, generalizable optimization in deployment. During the evaluation phase, the agent operates in a lightweight inference mode, where no network updates are needed. As a result, the optimization process becomes much more efficient while retaining high accuracy.

This makes DE-MPDQN highly suitable for applications in engineering design, logistics, and other fields where optimization must be fast, robust, and adaptive without incurring new learning costs.

http://doi.org/10.26594/register.v8i2.XXX

ISSN 2460-7258 (online) | ISSN 1978-1520 (print)
7 (3) March

Table 4. Averagerankingofallmethods

Algo	Random	DE1	DE2	DE3	DE4	AdapSS	FAUC	RecPM	LR	IPOP	DDQN	MPDQN1	MPDQN2	MPDQN3	MPDQN4
Rank	12.5	13.55	12.75	12.85	13.35	5.5	7.4	4.8	5.1	2.2	4.1	5.8	8.4	7.6	4.1

Table 4 presents the average ranking of all compared algorithms across the benchmark functions. A lower rank indicates better overall performance. As shown, DE-MPDQN4 achieves the best rank (4.1) among all 13 evaluated methods, confirming its superior performance in most test cases.

Traditional DE variants such as DE1–DE4 and the Random strategy perform poorly, with ranks consistently above 12, highlighting the limitations of using fixed or stochastic operator selection. Among the AOS-based methods, RecPM-AOS (rank 4.8) and FAUC (4.8) demonstrate competitive performance, but still fall short of DE-MPDQN4.

Notably, DE-MPDQN variants show a clear progression of effectiveness with reward design:

1. DE-MPDQN1 and DE-MPDQN2 rank 5.8 and 8.4 respectively,
2. DE-MPDQN3 improves further to 7.6,
3. and DE-MPDQN4 tops the list at 4.1.

This trend reaffirms the earlier analysis in Section 3.4: reward design plays a pivotal role in enabling effective learning. Moreover, the table illustrates how DE-MPDQN, particularly with R4, outperforms or rivals even advanced algorithms like IPOP-CMAES (rank 2.2) and DE-DDQN (rank 4.1).

Overall, this table provides strong empirical support for the conclusion that DE-MPDQN, especially with well-designed reward functions, delivers state-of-the-art performance in adaptive differential evolution.

4. Conclusion

In this research proposed a novel method called DE-MPDQN (Differential Evolution with Multi-Pass Deep Q-Network) to adaptively select mutation operators and scaling factors in Differential Evolution (DE) by leveraging reinforcement learning in parameterized action spaces. The method was designed to address the limitations of static parameter settings and operator selection, which are commonly found in traditional DE algorithms.

By framing the operator and parameter selection problem as a parameterized action Markov decision process (PAMDP), DE-MPDQN effectively utilized 99 state features extracted from the DE population and learned a selection policy through offline training. The use of multi-pass architecture in MP-DQN ensured accurate learning without introducing false gradients.

Extensive experimental results using benchmark functions from CEC2005 demonstrated that DE-MPDQN, particularly the variant trained with negative reward feedback (DE-MPDQN4), significantly outperformed baseline DE methods and achieved competitive results compared to state-of-the-art adaptive and reinforcement learning-based DE variants.

Furthermore, the method showed strong generalization capability, effectively solving unseen optimization problems without retraining. This makes DE-MPDQN a promising approach for practical, real-world applications where dynamic adaptation and robustness are required.

In summary, DE-MPDQN offers a flexible, efficient, and adaptive enhancement to DE, combining the strengths of evolutionary optimization with modern reinforcement learning techniques. Future work may explore online learning integration, application to multi-objective optimization, or hybridization with other metaheuristics.

5. Suggestion

This research has demonstrated that the integration of reinforcement learning with Differential Evolution (DE) through the DE-MPDQN approach is effective in adaptively selecting mutation operators and scaling factors. However, several aspects could be explored in future studies to further enhance the method. First, it is recommended to investigate the integration of online learning, allowing the model to continuously adapt during the optimization process rather than relying solely

http://doi.org/10.26594/register.v8i2.XXX

ISSN 2460-7258 (online) | ISSN 1978-1520 (print)
7 (3) March

on offline training. This could improve responsiveness to dynamic or real-time optimization problems. Second, the proposed method may be extended to multi-objective optimization, enabling the adaptive mechanism to handle trade-offs between conflicting objectives, which is common in real-world applications.

In addition, future research may explore the development of more refined reward functions that better capture the optimization landscape or incorporate diversity and convergence indicators, potentially improving the learning process. Another suggestion is to test the DE-MPDQN method on real-world problems, such as in engineering design, logistics, or scheduling, to assess its practical effectiveness and adaptability. Lastly, hybridizing DE-MPDQN with other metaheuristic algorithms, like Particle Swarm Optimization or Genetic Algorithms, may offer complementary strengths and further boost optimization performance in complex or constrained problem domains.

References

[1] M. E. Abdual-Salam, H. M. Abdul-Kader, and W. F. Abdel-Wahed, "Comparative study between Differential Evolution and Particle Swarm Optimization algorithms in training of feed-forward neural network for stock price prediction," Proc. 7th Int. Conf. on Informatics and Systems (INFOS), pp. 1–8, 2010.

[2] A. Auger and N. Hansen, "Performance evaluation of an advanced local search evolutionary algorithm," Proc. IEEE Congress on Evolutionary Computation, vol. 2, pp. 1777–1784, 2005.

[3] A. Auger and N. Hansen, "A restart CMA evolution strategy with increasing population size," Proc. IEEE Congress on Evolutionary Computation, vol. 2, pp. 1769–1776, 2005.

[4] M. Baioletti, G. Di Bari, V. Poggioni, and M. Tracolli, "Can Differential Evolution Be an Efficient Engine to Optimize Neural Networks?" in Machine Learning, Optimization, and Big Data, Springer, pp. 401–413, 2018.

[5] C. J. Bester, S. D. James, and G. D. Konidaris, "Multi-pass Q-networks for deep reinforcement learning with parameterised action spaces," arXiv preprint arXiv:1905.04388, 2019.

[6] Y. Chabane and A. Ladjici, "Differential evolution for optimal tuning of power system stabilizers to improve power systems small signal stability," Proc. 5th Int. Conf. on Systems and Control (ICSC), pp. 84–89, 2016.

[7] F. Chen, Y. Gao, Z. Q. Chen, and S. F. Chen, "SCGA: Controlling genetic algorithms with Sarsa(0)," Proc. Int. Conf. on Computational Intelligence for Modelling, Control and Automation (CIMCA), pp. 1177–1182, 2005.

[10] A. E. Eiben et al., "Parameter Control in Evolutionary Algorithms," in Parameter Setting in Evolutionary Algorithms, Springer, pp. 19–46, 2007.

[11] A. E. Eiben and S. K. Smit, "Evolutionary Algorithm Parameters and Methods to Tune Them," in Autonomous Search, Springer, pp. 15–36, 2012.

[12] T. Eltaeib and A. Mahmood, "Differential evolution: A survey and analysis," Applied Sciences, vol. 8, no. 10, 2018.

[13] Á. Fialho et al., "Dynamic Multi-Armed Bandits and Extreme Value-Based Rewards for Adaptive Operator Selection in Evolutionary Algorithms," in Learning and Intelligent Optimization, Springer, pp. 176–190, 2009.

[14] Á. Fialho et al., "Comparison-Based Adaptive Strategy Selection with Bandits in Differential Evolution," in Parallel Problem Solving from Nature (PPSN XI), Springer, pp. 194–203, 2010.

[15] Á. Fialho, M. Schoenauer, and M. Sebag, "Toward Comparison-Based Adaptive Operator Selection," Proc. 12th Genetic and Evolutionary Computation Conf., pp. 767–774, 2010.

[16] D. E. Goldberg, Genetic Algorithms in Search, Optimization and Machine Learning, Addison-Wesley, 1989.

[17] D. E. Goldberg, "Probability matching, the magnitude of reinforcement, and classifier system bidding," Machine Learning, vol. 5, no. 4, pp. 407–425, 1990.

http://doi.org/10.26594/register.v8i2.XXX

ISSN 2460-7258 (online) | ISSN 1978-1520 (print)
7 (3) March

[18] W. Gong, Á. Fialho, and Z. Cai, "Adaptive Strategy Selection in Differential Evolution," Proc. 12th Genetic and Evolutionary Computation Conf., pp. 409–416, 2010.

[19] W. Gong et al., "Adaptive strategy selection in differential evolution for numerical optimization: An empirical study," Information Sciences, vol. 181, no. 24, pp. 5364–5386, 2011.

[21] H. van Hasselt, A. Guez, and D. Silver, "Deep Reinforcement Learning with Double Q-Learning," Proc. 30th AAAI Conf. Artificial Intelligence, pp. 2094–2100, 2016.

[22] M. Hausknecht and P. Stone, "Deep reinforcement learning in parameterized action space," Proc. 4th Int. Conf. on Learning Representations (ICLR), 2016.

[24] G. Karafotias, A. E. Eiben, and M. Hoogendoorn, "Generic parameter control with reinforcement learning," Proc. GECCO 2014, pp. 1319–1326, 2014.

[25] G. Karafotias, M. Hoogendoorn, and A. E. Eiben, "Evaluating Reward Definitions for Parameter Control," in Applications of Evolutionary Computation, Springer, pp. 667–680, 2015.

[26] J. Kennedy and R. Eberhart, "Particle swarm optimization," Proc. IEEE Int. Conf. Neural Networks, vol. 4, pp. 1942–1948, 1995.

[29] W. Masson, P. Ranchod, and G. Konidaris, "Reinforcement Learning with Parameterized Actions," Proc. AAAI Conf. Artificial Intelligence, pp. 1934–1940, 2016.

[30] E. Mezura-Montes, J. Velázquez-Reyes, and C. A. Coello Coello, "A Comparative Study of Differential Evolution Variants for Global Optimization," Proc. 8th Genetic and Evolutionary Computation Conf., pp. 485–492, 2006.

[33] V. Nair and G. E. Hinton, "Rectified Linear Units Improve Restricted Boltzmann Machines," Proc. 27th Int. Conf. Machine Learning, pp. 807–814, 2010.

[34] J. E. Pettinger and R. M. Everson, "Controlling Genetic Algorithms with Reinforcement Learning," Proc. 4th Genetic and Evolutionary Computation Conf., pp. 692, 2002.

[35] A. K. Qin and P. N. Suganthan, "Self-adaptive differential evolution algorithm for numerical optimization," Proc. IEEE Congress on Evolutionary Computation, vol. 2, pp. 1785–1791, 2005.

[36] Y. Sakurai et al., "A method to control parameters of evolutionary algorithms by using reinforcement learning," Proc. 6th Int. Conf. on Signal Image Technology and Internet Based Systems (SITIS), pp. 74–79, 2010.

[37] M. Sharma, A. Komninos, and M. López-Ibáñez, "Deep reinforcement learning based parameter control in differential evolution," Proc. GECCO, 2019.

[38] M. Sharma, M. López-Ibáñez, and D. Kazakov, "Performance Assessment of Recursive Probability Matching for Adaptive Operator Selection in Differential Evolution," in PPSN XV, Springer, pp. 321–333, 2018.

[39] M. Sharma, M. López-Ibáñez, and D. Kazakov, "Unified Framework for the Adaptive Operator Selection of Discrete Parameters," arXiv preprint, 2020.

[40] R. Storn, "On the usage of differential evolution for function optimization," Proc. North American Fuzzy Information Processing, pp. 519–523, 1996.

[41] 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.

[42] B. Subudhi and D. Jena, "A differential evolution based neural network approach to nonlinear system identification," Applied Soft Computing, vol. 11, no. 1, pp. 861–871, 2011.

[43] K. R. Sudha, "Design of differential evolution algorithm-based robust fuzzy logic power system stabiliser using minimum rule base," IET Generation, Transmission & Distribution, vol. 6, no. 2, pp. 121–132, 2012.

[44] P. N. Suganthan et al., "Problem definitions and evaluation criteria for the CEC 2005 special session on real-parameter optimization," KanGAL Report 2005005, 2005.

http://doi.org/10.26594/register.v8i2.XXX

ISSN 2460-7258 (online) | ISSN 1978-1520 (print)
7 (3) March

[45] T. H. Teng, S. D. Handoko, and H. C. Lau, "Self-organizing neural network for adaptive operator selection in evolutionary search," Lecture Notes in Computer Science, vol. 10079, pp. 187–202, 2016.

[46] D. Thierens, "An Adaptive Pursuit Strategy for Allocating Operator Probabilities," Proc. 7th Genetic and Evolutionary Computation Conf., pp. 1539–1546, 2005.

[47] J. Vesterstrom and R. Thomsen, "A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems," Proc. IEEE Congress on Evolutionary Computation, vol. 2, pp. 1980–1987, 2004.

[48] J. Xiong et al., "Parameterized deep Q-networks learning: Reinforcement learning with discrete-continuous hybrid action space," arXiv preprint arXiv:1810.06394, 2018.

[49] X.-S. Yang and S. Deb, "Cuckoo Search via Lévy Flights," Proc. IEEE World Congress on Nature and Biologically Inspired Computing, pp. 210–214, 2009.

[50] J. Zhang and A. C. Sanderson, "JADE: Adaptive differential evolution with optional external archive," IEEE Transactions on Evolutionary Computation, vol. 13, no. 5, pp. 945–958, 2009.

[51] Y.-X. Zhao et al., "An Improved Differential Evolution Algorithm for Maritime Collision Avoidance Route Planning," Abstract and Applied Analysis, vol. 2014, Article ID 614569, 2014.

http://doi.org/10.26594/register.v8i2.XXX