Dynamic Graph Neural Evolution: An Evolutionary Framework Integrating Graph Neural Networks with Adaptive Filtering

Kaichen Ouyang[1], Shengwei Fu[2], Yi Chen[3,4], Huiling Chen[3,*]
[1]Department of Mathematics, University of Science and Technology of China, Hefei, China
Email: oykc@mail.ustc.edu.cn
[2]Key Laboratory of Advanced Manufacturing Technology, Ministry of Education,
Guizhou University, Guiyang, Guizhou, China
Email: shengwei_fu@163.com
[3]Department of Computer Science, Wenzhou University, Wenzhou, China
Email: kenyoncy2016@gmail.com, chenhuiling.jlu@gmail.com
[4]School of Engineering, Westlake University, Hangzhou 310030, China

***Abstract*—This paper proposes an innovative optimization framework, Dynamic Graph Neural Evolution (DGNE), integrating Graph Neural Networks (GNNs) with Evolutionary Algorithms (EAs). Building on the foundation of Graph Neural Evolution (GNE), DGNE introduces a dynamic filtering mechanism and adaptive Gaussian sampling functions to dynamically adjust the population distribution during the optimization process, achieving a balance between global exploration and local exploitation. By emphasizing high-frequency information in the early stages to enhance population diversity and low-frequency information in the later stages to promote convergence, DGNE effectively optimizes the search process. Experiments were conducted on the CEC2017 benchmark suite across 30, 50, and 100 dimensions, comparing DGNE with advanced algorithms (LSHADE, LSHADE_cnEpSin, MadDE, SaDE, EA4eig) and classic algorithms (DE and CMA-ES). Statistical analyses using the Wilcoxon rank-sum test and Friedman mean rank test demonstrate that DGNE achieves the best average ranking in 50 and 100 dimensions and ranks third in 30 dimensions. However, it achieves the highest overall average performance ranking across all dimensions, showcasing its stability and significant advantages in different scenarios. While its performance in low-dimensional tasks is slightly less competitive compared to high-dimensional ones, DGNE still exhibits strong competitiveness. Additionally, we explored the impact of population size. DGNE was evaluated across population sizes of 20, 30, 50, and 100. The results highlight DGNE's robustness, maintaining competitive rankings across all population sizes, with top rankings for smaller population sizes (20 and 30) and strong results at larger sizes (50 and 100). These findings confirm DGNE's adaptability to varying configurations and further validate its effectiveness as a robust optimization framework. Overall, DGNE demonstrates great potential as an optimization method, offering a promising direction for further research and applications in artificial intelligence and optimization fields. Its ability to remain competitive across diverse tasks and configurations underscores its versatility and scalability.**

*Corresponding author: Huiling Chen (chenhuiling.jlu@gmail.com)

***Index Terms*—Machine Learning, Graph Neural Networks (GNNs), Evolutionary Algorithms (EAs), Dynamic Graph Neural Evolution (DGNE), CEC2017 Benchmark Suite**

I. Introduction

Evolutionary Computation (EC) and Machine Learning (ML) represent two cornerstone paradigms in artificial intelligence, each offering unique strengths for solving complex problems. EC, inspired by Darwinian evolution, employs operators such as selection, crossover, and mutation to iteratively evolve solutions within vast search spaces. Originating in the mid-20th century [1]–[3], EC is particularly adept at addressing high-dimensional, nonlinear, and multi-objective optimization challenges, owing to its minimal reliance on mathematical assumptions like differentiability or continuity. Foundational EC algorithms include the Genetic Algorithm (GA) [4], which simulates natural selection to refine populations; Differential Evolution (DE) [5], prized for its simplicity and efficacy in continuous optimization; Particle Swarm Optimization (PSO) [6], which mimics social behavior in flocks to guide search; and Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [7], which leverages adaptive covariance matrices for robust performance in high-dimensional settings.

Advancements in DE have significantly broadened its applicability. For instance, Success-History Based Adaptive DE (SHADE) [8] uses historical success data to dynamically tune control parameters, accelerating convergence. LSHADE [9] extends SHADE with a linear population size reduction mechanism to balance exploration and exploitation throughout the evolutionary process. Further refinements, such as LSHADE_cnEpSin [10], incorporate sinusoidal parameter adjustments and Euclidean neighborhood knowledge for precise search, while LSHADE_SPACMA [11] hybridizes DE with CMA-ES components to optimize the exploration-exploitation trade-off. Other notable variants include JADE [12], which introduces an adaptive mutation strategy with optional external archives, and SaDE [13], which employs self-adaptive mechanisms to dynamically select among multiple mutation strategies, enhancing flexibility across diverse optimization landscapes.

In contrast, ML emphasizes learning patterns from data to facilitate prediction and decision-making. Early ML techniques, including linear regression [14], support vector machines (SVMs) [15], and decision trees [16], relied heavily on hand-crafted features and statistical assumptions. The advent of neural networks revolutionized this field by enabling automated feature extraction through hierarchical layers. Deep learning has since propelled ML forward with architectures such as Convolutional Neural Networks (CNNs) [17], [18] for image processing, Recurrent Neural Networks (RNNs) [19], [20] for sequential and time-series data, Long Short-Term Memory networks (LSTMs) [21] for improved memory in sequences, Transformers [22], [23] for natural language processing, and Graph Neural Networks (GNNs) [24]–[27] for structured and non-Euclidean data. These advancements have catalyzed breakthroughs in domains like computer vision, speech recognition, and graph-based reasoning.

The intersection of EC and ML has become a vibrant research area, harnessing the strengths of both fields. Evolutionary algorithms enhance ML by optimizing hyperparameters, designing neural architectures [28], [29], and improving performance in tasks such as classification, regression, and feature selection. Conversely, ML augments EC efficiency: surrogate-assisted models [30]–[32] approximate expensive fitness functions to reduce computation, reinforcement learning [33], [34] informs evolutionary search with learned strategies, and clustering techniques [35] refine population diversity. A prominent example is the Graph Neural Evolution (GNE) framework [36]–[38], which represents individuals as graph nodes and applies frequency-domain filters to balance global exploration and local exploitation, yielding exceptional results in complex optimization tasks.

Building on GNE, this paper proposes Dynamic Graph Neural Evolution (DGNE), a novel approach integrating dynamic filtering, adaptive Gaussian sampling, and population size adjustment to enhance optimization dynamics. Evaluated on the CEC2017 benchmark suite across 30, 50, and 100 dimensions, DGNE demonstrates superior performance against state-of-the-art methods, including LSHADE [9], LSHADE_cnEpSin [10], LSHADE_SPACMA [11], MadDE [39], SaDE [13], and EA4eig [40]. Its adaptive mechanisms prove especially effective in high-dimensional settings, improving scalability and solution quality over traditional approaches.

This paper is structured as follows: Section 2 reviews GNE and details the mathematical formulation of DGNE, including its dynamic filtering and sampling strategies. Section 3 presents experimental results, statistical analysis, and comparisons on CEC2017 benchmarks. Section 4 concludes with key insights and outlines future research directions, such as integrating DGNE with emerging ML techniques.

II. Dynamic Graph Neural Evolution(DGNE)

A. Review of GNE

GNE is an innovative approach that combines Evolutionary Algorithms (EAs) with GNNs, representing the population through graph structures and leveraging the characteristics of graphs during the optimization process. In GNE, each individual in the population is modeled as a node in the graph, and the similarities between individuals are encoded via the adjacency matrix. This graph-based representation enables GNE to utilize spectral filtering techniques to dynamically balance exploration and exploitation during optimization, achieving robust and efficient optimization.

The first step of GNE is to represent the population as a graph. Each individual X_i in the population is treated as a node in the graph, and the similarity between any two individuals X_i and X_j is calculated based on cosine similarity. The adjacency matrix A is defined as follows:

$$A_{ij} = \frac{Z_i Z_j}{\|Z_i\|\|Z_j\|} \tag{1}$$

where $Z_i = X_i - X_0$ represents the difference vector between individual X_i and the population center X_0, and A_{ij} represents the similarity between nodes i and j. To normalize the graph, the normalized Laplacian matrix is constructed as:

$$L_{\text{norm}} = I - D^{-\frac{1}{2}} A D^{-\frac{1}{2}} \tag{2}$$

where D is the degree matrix, A is the adjacency matrix, and I is the identity matrix. By performing spectral decomposition on the Laplacian matrix L_{norm}, its eigenvalues Λ and eigenvectors U can be obtained:

$$L_{\text{norm}} = U \Lambda U^T \tag{3}$$

These spectral components allow the definition of a filter function $g(\Lambda)$, which selectively amplifies or suppresses specific frequency components. Low-frequency components represent the global stability and shared information of the population, while high-frequency components capture local diversity and individual differences within the population. The filtered population can then be expressed as:

$$X_{\text{new}} = U g(\Lambda) U^T X \tag{4}$$

In spectral filtering, existing spectral-based graph neural networks typically apply nonlinear transformations φ, ϕ to the original feature matrix X or the filtered feature matrix. These transformations are usually functions similar to Multi-Layer Perceptrons (MLPs). During the training process of graph neural networks, the parameters of these nonlinear transformations are continuously updated using the gradient descent algorithm with a specific learning rate. In this process, the changes in φ and ϕ can be viewed as a resampling operation on the feature matrix in its respective feature space. Through this resampling, features are updated along the optimization direction with a certain step size, gradually adjusting their distribution to better fit the requirements of specific problems. This enhances the model's representational capability and optimization performance for the target task.

Similarly, in evolutionary algorithms, the nonlinear transformations φ and ϕ can be implemented as resampling operations of the current population along the optimization direction. This process progressively adjusts the population's distribution through resampling, making it more concentrated in high-fitness regions and thereby improving optimization results. Based on this idea, GNE introduces an evolutionary algorithm combined with graph neural networks, where the overall population update process is described by the following equation:

$$X_{\text{new}} = \phi\left(Ug(\Lambda)U^T\varphi(X)\right) \tag{5}$$

In Eq. (5), φ and ϕ serve as resampling functions applied before and after the filtering operation. By adding resampling functions before and after filtering, GNE further optimizes the population distribution, achieving a more efficient optimization process. This dual transformation leverages the characteristics of graph neural networks, not only extracting global stability and local diversity from the population during spectral filtering but also dynamically adjusting the population distribution to enhance the search capability and adaptability of the evolutionary algorithm. The core idea of GNE is to integrate the spectral filtering and resampling mechanisms of graph neural networks into evolutionary algorithms. Through spectral decomposition and resampling operations, it dynamically balances exploration and exploitation within the population. This approach allows the population to fully leverage global and local information from the graph during the optimization process, achieving robust and efficient optimization.

B. Mathematical Model of Dynamic Graph Neural Evolution (DGNE)

DGNE builds on the foundation of GNE by introducing dynamic filtering mechanisms and adaptive Gaussian sampling functions. These mechanisms enable DGNE to dynamically adjust the population distribution during the optimization process via spectral filtering and adaptive resampling strategies, thereby balancing exploration and exploitation.

1) Construction of the Dynamic Filter: At each iteration, DGNE constructs a dynamic filter to balance the contributions of high-frequency and low-frequency components in the spectral domain. High-frequency components capture local diversity, while low-frequency components reflect global stability. The weights of the high- and low-frequency components are dynamically adjusted based on the current iteration t, with the weight definitions as follows:

$$w_h(t) = 0.8 - 0.5\left(\frac{t}{T}\right)^2, \quad w_i(t) = 0.2 + 0.5\left(\frac{t}{T}\right)^2 \tag{6}$$

The expressions for the low-frequency filter and high-frequency filter are given as:

$$\Lambda_l = 2e^{-10\lambda^2}, \quad \Lambda_h = 2 - 2e^{-10\lambda^2} \tag{7}$$

The final expression for the dynamic filter $g(\Lambda)$ is:

$$g(\Lambda) = w_i\Lambda_l + w_h\Lambda_h \tag{8}$$

Algorithm 1 DGNE

```
Require: Population size $N$, maximum iterations $T$, objective function $f$, lower bound $LB$, upper bound $UB$, sampling function $\varphi, \phi$, filter $g$
 1: Initialization: Population $X$, optimal solution $x_{best}$, optimal function value $f_{best}$, $t = 0$
 2: while $t <= T$ do
 3:     Update the $\mu_1 = \frac{1}{k}\sum_{i=1}^{k} X_i$ and $\sigma_t^2 = 0.0005e^{-16\frac{t-\frac{T}{2}}{T}}$ of sampling function $\varphi$
 4:     Construct the sampling function $\varphi$
 5:     Apply $\varphi(X)$ to $X$ for nonlinear transformation
 6:     Calculate the similarity matrix $A$ for the population $X$
 7:     Calculate the normalized Laplacian matrix $L$ using $L = I - D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$
 8:     Calculate the Diagonal eigenvalue matrix $\Lambda$ and the eigenvector matrix $U$ of the $L$
 9:     Update the weights $w_h(t) = 0.8 - 0.5\left(\frac{t}{T}\right)^2, \quad w_l(t) = 0.2 + 0.5\left(\frac{t}{T}\right)^2$
10:     Construct the Filter $g(\Lambda) = w_1\Lambda_l + w_h\Lambda_h$
11:     Obtain the updated population $X_{\text{new}}$ through $X_{\text{new}} = Ug(\Lambda)U^TX$
12:     Obtain the combined population $X^c = [X, X_{\text{new}}]$
13:     Update the $\mu_2 = \frac{1}{k}\sum_{i=1}^{k} X_i^c$, and $\sigma_t^2 = 0.0005e^{-16\frac{t-T/2}{T}}$ of sampling function $\phi$
14:     Construct the sampling function $\phi$
15:     Apply $\phi(X_{new})$ to $X_{new}$ for nonlinear transformation
16:     $t = t + 1$
17:     Update the optimal solution $x_{best}$ and the optimal function value $f_{best}$
18: end while
```

By amplifying or suppressing specific frequency components, this filter dynamically adjusts the population distribution to balance global exploration and local exploitation. During the early iterations of the algorithm, the weight of the high-frequency components is higher, enhancing the local diversity of the population and promoting broad exploration of the solution space. In the later iterations, the weight of the low-frequency components increases, strengthening global stability and guiding the population to converge toward the optimal solution. **Fig. 1** illustrates the schematic diagram of the dynamic filter, showcasing its adaptation over iterations at steps 0, 100, 200, 300, 400, and 500. The diagram depicts the transition of the filter's shape, where high-frequency components are assigned greater weights in the early stages, gradually shifting to emphasize low-frequency components in later stages. This transition reflects the dynamic balance between exploration and exploitation, effectively guiding the optimization process through different phases.

2) Adaptive Sampling Functions: DGNE introduces two sampling functions, φ and ϕ, which optimize the population distribution before and after spectral filtering, respectively. Before spectral filtering, φ is used to generate new candidate solutions around the better-performing individuals via a

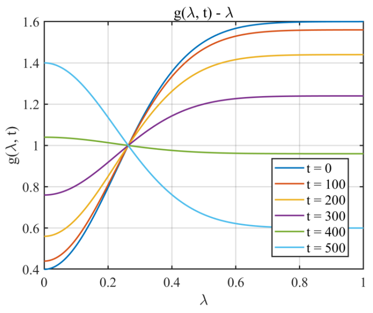


Fig. 1: Schematic Diagram of the Dynamic Filter.

Gaussian distribution. Its sampling formula is as follows:

$$\varphi(X) \sim N(\mu_1, \sigma_t^2) \tag{9}$$

where $\mu_1 = \frac{1}{k}\sum_{i=1}^{k} X_i$ represents the mean of the top k individuals sorted by fitness, and $\sigma_t^2 = 0.0005e^{-16\frac{t-\frac{T}{2}}{\frac{T}{2}}}$ is the variance of the distribution, which decreases dynamically with the number of iterations to focus the search. Here, we set k as [0.2N],where N represents the number of individuals in the population.

$$X^f = Ug(\Lambda)U^T\varphi(X) \tag{10}$$

where U and Λ are the eigenvectors and eigenvalues of the normalized Laplacian matrix L_{norm}, and $g(\Lambda)$ is the dynamic filter. After spectral filtering, ϕ generates new candidate solutions by sampling around the better-performing individuals from the combined population before and after filtering. Its sampling formula is as follows:

$$\phi(X) \sim N(\mu_2, \sigma_t^2) \tag{11}$$

where the combined population $X^c = [X, X^f]$ and $\mu_2 = \frac{1}{k}\sum_{i=1}^{k} X_i^c$ represents the mean of the top k individuals in the combined population sorted by fitness. Here, we set k as [0.2N],where N represents the number of individuals in the population.

DGNE introduces dynamic filtering mechanisms and adaptive Gaussian sampling functions, providing an innovative optimization framework for evolutionary algorithms. The dynamic filtering mechanism adjusts the weights of high- and low-frequency components, enhancing exploration and local diversity in the early stages of the algorithm while improving exploitation and global stability in the later stages. Through two-stage sampling before and after spectral filtering, DGNE ensures that the population distribution maintains diversity while gradually focusing on high-fitness regions. Additionally, by leveraging the spectral properties of graphs, DGNE dynamically adjusts the population structure to capture both global stability and local diversity. **Fig. 2** illustrates the overall framework of the proposed DGNE, which includes two sampling processes and a dynamic filter. The filter transitions from high-frequency to low-frequency filtering, reflecting the shift from enhancing exploration capabilities in the early stages to strengthening exploitation capabilities in the later stages. This structure allows the model to effectively balance exploration and exploitation throughout the iterative process. The pseudocode provided in **Algorithm 1** outlines the implementation of DGNE, detailing how the population is dynamically updated through spectral filtering, resampling functions, and sampling operations. This algorithm serves as a comprehensive guide to the procedural steps involved in DGNE, ensuring clarity and reproducibility of the method.

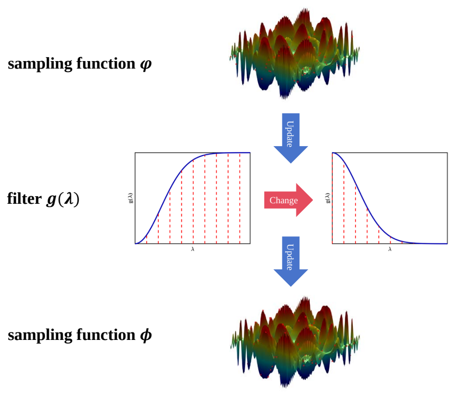


Fig. 2: An illustration of the proposed DGNE .

III. Experimental Results and Analysis

To comprehensively evaluate the performance of DGNE, extensive experiments were conducted on the CEC2017 benchmark suite, which includes 30 test functions with dimensionalities of 30, 50, and 100. The proposed method was compared against several state-of-the-art and classical evolutionary algorithms, including LSHADE, LSHADE_cnEpSin, MadDE, SaDE, EA4eig, DE, and CMA-ES. To ensure fairness, all algorithms were configured with a population size of 30 and a maximum iteration count of 500. Each algorithm was independently executed 30 times to ensure statistical reliability. The experimental results were analyzed using the Wilcoxon rank-sum test and the Friedman mean rank test to assess the significance and consistency of DGNE's performance. This section is organized as follows: Section 3.1 presents the overall performance comparison of DGNE against competing algorithms; Section 3.2 investigates the impact of varying population sizes on DGNE's performance; and Section 3.3 explores the behavior of DGNE beyond 500 iterations.

A. Overall Performance Comparison

To validate the performance of the DGNE, experiments were conducted on 30 benchmark functions from the CEC2017 test suite. The experiments were performed with dimensionalities of 30, 50, and 100 to comprehensively evaluate the scalability and optimization effectiveness of DGNE. The proposed method was compared with various high-performance and classical evolutionary algorithms, including LSHADE, LSHADE_cnEpSin, MadDE, SaDE, EA4eig, DE, and CMA-ES. To ensure fairness, the population size was set to 30, and the number of iterations was fixed at 500 for all algorithms.

Each algorithm was independently run 30 times to ensure the statistical reliability of the results. Additionally, the Wilcoxon rank-sum test and Friedman mean rank test were employed to perform statistical analyses on the experimental results [41], verifying the significance and consistency of the algorithm's performance.

TABLE I: Wilcoxon rank sum test statistical results

DGNE VS.	CEC 2017		
	(Dim=30)	(Dim=50)	(Dim=100)
LSHADE	14/6/10	17/4/9	21/2/7
LSHADE_cnEpSin	14/3/13	18/1/11	19/4/7
MadDE	18/3/9	21/4/5	25/1/4
SaDE	14/5/11	19/2/9	22/2/6
EA4eig	13/2/15	16/3/11	22/1/7
DE	23/5/2	27/0/3	28/0/2
CMA-ES	20/6/4	23/0/7	24/0/6
Overall (+/=/–)	116/30/64	141/14/55	161/10/39

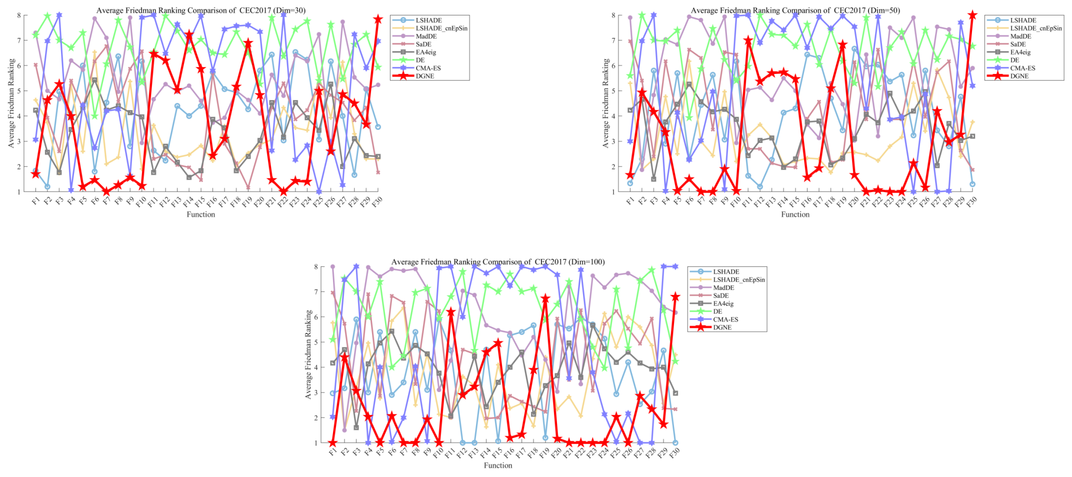
TABLE II: Friedman mean rank test for all test suites

Suites	CEC 2017					
	30		50		100	
	Ave. Rank	Overall Rank	Ave. Rank	Overall Rank	Ave. Rank	Overall Rank
LSHADE	4.29	5	4.17	4	3.92	3
LSHADE_cnEpSin	3.38	2	3.25	2	3.71	2
MadDE	5.41	7	5.65	7	6.11	7
SaDE	3.98	4	4.32	5	4.38	5
EA4eig	3.27	1	3.55	3	3.94	4
DE	6.72	8	6.65	8	6.30	8
CMA-ES	5.27	6	5.30	6	5.12	6
DGNE	3.68	3	3.13	1	2.52	1

Table 1 presents the results of the Wilcoxon rank-sum test, which evaluates the statistical performance of DGNE against several state-of-the-art and classical evolutionary algorithms on the CEC2017 benchmark suite (dimensions 30, 50, and 100). **Table 2** complements this by providing the Friedman mean rank test results, which rank the algorithms based on their average performance across the same test suite, offering a comprehensive evaluation of overall competitiveness. The results indicate that DGNE performs exceptionally well in medium- and high-dimensional tasks. In **Table 1**, DGNE achieves results of 141/14/55 and 161/10/39 at 50 and 100 dimensions, respectively, demonstrating its strong performance in complex optimization tasks. This trend is further validated in **Table 2**, where DGNE secures average rankings of 3.13 and 2.52 at 50 and 100 dimensions, respectively, ranking first overall in terms of comprehensive performance across different dimensions. For 30-dimensional tasks, DGNE still performs impressively with results of 116/30/64 in **Table 1**. Although it ranks third in **Table 2** with an average rank of 3.68, it is only behind LSHADE_cnEpSin and EA4eig, the champion algorithms of the CEC2017 and CEC2022 test suites, respectively. Notably, the Wilcoxon rank-sum test results show that DGNE's performance at 30 dimensions is very close to these two champion algorithms, highlighting its competitiveness in low-dimensional problems. Among the competitors, LSHADE_cnEpSin, the champion of the CEC2017 test suite, demonstrates reliability and robustness across all dimensions, particularly excelling in high-dimensional tasks. For example, in **Table 1**, it achieves a result of 19/4/7 at 100 dimensions, and in **Table 2**, it consistently ranks second overall. Similarly, EA4eig, the champion of the CEC2022 test suite, shows its unique strengths in low-dimensional tasks. For instance, in 30-dimensional tasks, it ranks first in **Table 2** with an average rank of 3.27 and achieves 13/2/15 in **Table 1**. However, EA4eig's performance declines as the dimensionality increases, ranking third and fourth in Table 2 at 50 and 100 dimensions, respectively, indicating its suitability for low-dimensional problems. LSHADE demonstrates adaptability across dimensions, achieving a result of 21/2/7 at 100 dimensions in **Table 1**, and steadily improving its rank in **Table 2**, moving from fifth at 30 dimensions to third at 100 dimensions, showcasing its scalability. SaDE also maintains stable performance, with results such as 14/5/11 at 30 dimensions in **Table 1** and consistently ranking fourth or fifth in **Table 2**. In conclusion, DGNE achieves undisputed first place in comprehensive performance at 50 and 100 dimensions and demonstrates strong competitiveness at 30 dimensions, only slightly behind LSHADE_cnEpSin and EA4eig, the two champion algorithms. Based on the combined results of Table 1 and **Table 2**, DGNE exhibits outstanding scalability and adaptability, particularly excelling in high-dimensional tasks. Meanwhile, other algorithms such as LSHADE_cnEpSin in high-dimensional tasks and EA4eig in low-dimensional tasks also demonstrate their strengths in specific scenarios. **Fig. 3**, combined with the analysis of the CEC2017 test suite, highlights DGNE's performance across different function types and dimensionalities. DGNE performs exceptionally well on unimodal and multimodal functions (F1–F10) as well as on certain composition functions (F21–F26), particularly in high-dimensional problems, where its dynamic graph structure excels in exploitation and scalability. However, its performance declines on hybrid functions (F11–F15) and the highly complex composition function F30, reflecting its limitations in handling intricate, non-separable landscapes with diverse characteristics. Additionally, DGNE's effectiveness in low-dimensional tasks (e.g., 30 dimensions) is relatively weaker, where simpler algorithms like EA4eig and LSHADE are more efficient in navigating smaller search spaces. Overall, DGNE demonstrates strong potential for high-dimensional and specific problem types, but it requires further enhancement in exploration capabilities and adaptability to tackle complex hybrid functions and low-dimensional tasks effectively.

B. Impact of Population Size

As shown in Table 3, DGNE was evaluated alongside state-of-the-art algorithms using population sizes of 20, 30, 50, and 100. The experiments were conducted on the CEC2017 benchmark suite, comprising 90 functions across 30, 50, and 100 dimensions (30 functions per dimension). The number



(a)

Fig. 3: Average Friedman Rankings of DGNE and Competitors on CEC2017.

of iterations was fixed at 500 for these experiments. DGNE consistently achieved competitive rankings across all population sizes, with average ranks of 3.09 (population size = 20), 3.11 (population size = 30), 3.79 (population size = 50), and 3.33 (population size = 100). Notably, DGNE ranked 1st for population sizes of 20 and 30, 3rd for a population size of 50, and 2nd for a population size of 100. These results highlight DGNE's robustness to variations in population size. Among the competing algorithms, LSHADE_cnEpSin demonstrated strong performance, particularly at larger population sizes. For instance, it ranked 1st at population sizes of 50 and 100, with average ranks of 2.98 and 2.53, respectively. This suggests that LSHADE_cnEpSin benefits from larger populations, likely due to its enhanced exploration capabilities. EA4eig also performed well, securing 2nd place at population sizes of 20 and 50, with average ranks of 3.41 and 3.41, respectively. However, its performance slightly declined at a population size of 100, where it ranked 3rd. LSHADE and SaDE maintained stable performance across all population sizes, consistently ranking 4th or 5th, indicating their reliability in diverse scenarios. On the other hand, MadDE, DE, and CMA-ES exhibited weaker performance, consistently ranking in the lower tiers across all population sizes. For example, DE ranked 8th in all cases, highlighting its limitations in handling larger populations and more complex optimization tasks. Overall, DGNE's ability to adapt to varying population sizes while maintaining top-tier performance underscores its robustness and scalability. While LSHADE_cnEpSin and EA4eig also demonstrated strong adaptability, DGNE's consistent ranking across different configurations makes it a highly competitive choice for optimization tasks.

C. Analysis of the Exploration and Exploitation

The balance between exploration and exploitation is a crucial concept in evolutionary computation. The exploration phase aims to search a broader solution space, while the exploitation phase focuses on refining the search to better approximate the optimal solution. Properly balancing these two phases is essential for enhancing the robustness of the algorithm [42]. In this section, we calculate the percentages of exploration and exploitation using Eq. (12) and Eq. (13), respectively. Fig.4 illustrates the evolution of exploration and exploitation proportions in the population, calculated using the following equations:

$$\text{Exploration}(\%) = \frac{\text{Div}(t)}{\text{Div}_{\max}} \times 100 \tag{9}$$

$$\text{Exploitation}(\%) = \frac{|\text{Div}(t) - \text{Div}_{\max}|}{\text{Div}_{\max}} \times 100 \tag{10}$$

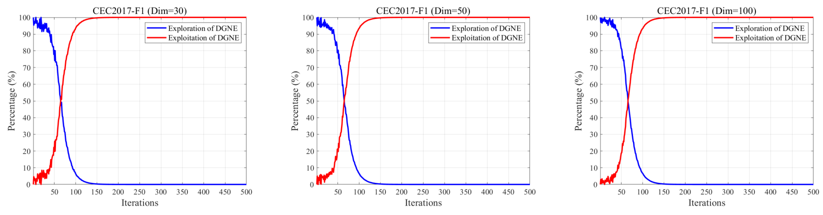
$$\text{Div}(t) = \frac{1}{\text{dim}} \sum_{d=1}^{\text{dim}} \sum_{i=1}^{n} |\text{median}(x_d(t)) - x_{i,d}(t)| \tag{11}$$

The parameter $Div(t)$ denotes the dimension diversity measure and Div_{max} represents the maximum diversity achieved throughout the entire iteration process,which is calculated in Eq.(14).The results show that the exploration proportion is high in the early stages and decreases rapidly after 200 iterations, while the exploitation proportion is low initially and increases significantly after 200 iterations.

Fig.5 illustrates the low-frequency and high-frequency filter functions. It can be observed that the low-frequency filter function decreases as the eigenvalue λ increases, while the high-frequency filter function increases with λ. Fig.1 shows the dynamically weighted filter function. As the iterations progress, the filter gradually transitions from a high-frequency-biased filter to a low-frequency-biased filter. Remarkably, this transition corresponds well with the exploration-to-exploitation shift observed in Fig.4, further validating the design of DGNE.

TABLE III: Friedman Average Rankings of Different Algorithms with Varying Population Sizes

Algorithms	**Population Size**							
	20		30		50		100	
	Ave. Rank	**Overall Rank**	**Ave. Rank**	**Overall Rank**	**Ave. Rank**	**Overall Rank**	**Ave. Rank**	**Overall Rank**
LSHADE	4.23	4	4.13	4	4.39	5	4.21	4
LSHADE_cnEpSin	3.83	3	3.44	2	**2.98**	1	**2.53**	1
MadDE	5.63	7	5.72	7	5.45	7	5.60	7
SaDE	4.55	5	4.23	5	3.91	4	4.26	5
EA4eig	3.41	2	3.59	3	3.41	2	3.74	3
DE	5.89	8	6.56	8	6.95	8	7.24	8
CMA-ES	5.36	6	5.23	6	5.13	6	5.11	6
DGNE	**3.09**	1	**3.11**	1	3.79	3	3.33	2



(a) Dim=30 (b) Dim=50 (c) Dim=100

Fig. 4: Exploration and Exploitation Analysis of DGNE

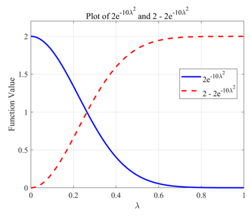


Fig. 5: Illustration of Low-Frequency and High-Frequency Filter Functions

IV. Conclusion

This paper proposes an innovative optimization framework, DGNE, which integrates GNNs with EAs. By introducing a dynamic filtering mechanism and adaptive Gaussian sampling functions, DGNE achieves an effective balance between global exploration and local exploitation during the optimization process. By dynamically adjusting the population distribution, DGNE demonstrates significant advantages in solving complex optimization problems, as validated by experiments on the CEC2017 benchmark suite. The experimental results show that DGNE outperforms or matches state-of-the-art algorithms such as LSHADE, LSHADE_cnEpSin, MadDE, SaDE, EA4eig, as well as classic algorithms like DE and CMA-ES, in terms of convergence speed, robustness, and adaptability. DGNE achieves the best average rankings in 50 and 100 dimensions and ranks third in 30 dimensions, highlighting its scalability and outstanding performance in high-dimensional optimization tasks. Furthermore, DGNE effectively leverages the spectral properties of graphs and dynamically balances exploration and exploitation, making it particularly effective in addressing large-scale and complex problems.

However, DGNE also has some limitations. In low-dimensional tasks, while DGNE remains competitive, there is still room for improvement, especially in enhancing its exploration capability in the early stages of the algorithm. Additionally, DGNE shows weaker performance on certain hybrid and highly complex composition functions, suggesting areas for further refinement in handling highly diverse and rugged landscapes. Moreover, the impact of population size and iteration count further influences DGNE's performance. DGNE has demonstrated robust adaptability across varying population sizes, consistently achieving top rankings with population sizes of 20, 30, 50, and 100. The framework ranked 1st for smaller populations (20 and 30), 2nd for a population size of 100, and 3rd for 50, underscoring its scalability in different configurations.

Future research could focus on enhancing DGNE's exploration capabilities to further improve its performance on challenging multimodal and hybrid functions. This could include exploring the use of alternative filter functions, sampling functions, and adaptive sampling mechanisms. Moreover, optimizing its computational efficiency in low-dimensional tasks could improve its versatility. Exploring other advanced mechanisms, such as multi-objective optimization, parallelization, and integration with other machine learning techniques, could further expand DGNE's applicability in a wider range of optimization and artificial intelligence challenges.

In conclusion, DGNE offers a robust, scalable, and efficient optimization framework, demonstrating significant potential in advancing the integration of evolutionary algorithms and graph neural networks, and providing a promising direction for solving complex real-world problems. Its ability to maintain top-tier performance across varying population sizes and iteration counts, combined with its adaptability and scalability, makes DGNE a highly competitive choice for optimization tasks.

Acknowledgment

The research was supported by National Natural Science Foundation of China under Grant (Grant No. 62301367) and in part by Zhejiang Provincial Clinical Research Center for Pediatric Diseases (Grant No. ZJEK2303Z). The financial contributions are gratefully acknowledged.

References

[1] A. S. Fraser, "Simulation of genetic systems by automatic digital computers i. introduction," *Australian Journal of Biological Sciences*, vol. 10, no. 4, pp. 484–491, 1957.

[2] D. B. Fogel, *Artificial Intelligence Through Simulated Evolution*. Wiley-IEEE Press, 1998.

[3] I. Rechenberg, "Cybernetic solution path of an experimental problem," *Roy. Aircr. Establ., Libr. transl.*, vol. 1122, 1965.

[4] J. H. Holland, "Genetic algorithms," *Scientific American*, vol. 267, no. 1, pp. 66–73, 1992.

[5] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, pp. 341–359, 1997.

[6] V. Stanovov and E. Semenkin, "Success rate-based adaptive differential evolution l-srtde for cec 2024 competition," in *2024 IEEE Congress on Evolutionary Computation (CEC)*, IEEE, 2024.

[7] N. Hansen, S. D. Müller, and P. Koumoutsakos, "Reducing the time complexity of the derandomized evolution strategy with covariance matrix adaptation (cma-es)," *Evolutionary Computation*, vol. 11, no. 1, pp. 1–18, 2003.

[8] R. Tanabe and A. Fukunaga, "Success-history based parameter adaptation for differential evolution," in *2013 IEEE Congress on Evolutionary Computation*, IEEE, 2013.

[9] R. Tanabe and A. S. Fukunaga, "Improving the search performance of shade using linear population size reduction," in *2014 IEEE Congress on Evolutionary Computation (CEC)*, IEEE, 2014.

[10] N. H. Awad, M. Z. Ali, and P. N. Suganthan, "Ensemble sinusoidal differential covariance matrix adaptation with euclidean neighborhood for solving cec2017 benchmark problems," in *2017 IEEE Congress on Evolutionary Computation (CEC)*, IEEE, 2017.

[11] A. W. Mohamed *et al.*, "Lshade with semi-parameter adaptation hybrid with cma-es for solving cec 2017 benchmark problems," in *2017 IEEE Congress on Evolutionary Computation (CEC)*, IEEE, 2017.

[12] 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.

[13] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential evolution algorithm with strategy adaptation for global numerical optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, 2008.

[14] Student, "The probable error of a mean," *Biometrika*, pp. 1–25, 1908.

[15] C. Cortes, "Support-vector networks," *Machine Learning*, 1995.

[16] J. R. Quinlan, "Induction of decision trees," *Machine Learning*, vol. 1, pp. 81–106, 1986.

[17] Y. LeCun *et al.*, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.

[18] K. He *et al.*, "Deep residual learning for image recognition," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[19] J. L. Elman, "Finding structure in time," *Cognitive Science*, vol. 14, no. 2, pp. 179–211, 1990.

[20] S. Hochreiter, "Long short-term memory," *Neural Computation MIT-Press*, 1997.

[21] K. Cho *et al.*, "Learning phrase representations using rnn encoder-decoder for statistical machine translation," 2014.

[22] A. Vaswani, "Attention is all you need," in *Advances in Neural Information Processing Systems*, 2017.

[23] J. Devlin, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2018.

[24] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016.

[25] W. Hamilton, Z. Ying, and J. Leskovec, "Inductive representation learning on large graphs," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[26] P. Veličković *et al.*, "Graph attention networks," 2017.

[27] P. Zhang *et al.*, "Transgnn: Harnessing the collaborative power of transformers and graph neural networks for recommender systems," in *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2024.

[28] X. Yan *et al.*, "Neural architecture search based on bipartite graphs for text classification," *IEEE Transactions on Neural Networks and Learning Systems*, 2024.

[29] W. Pan *et al.*, "Brain-inspired multi-scale evolutionary neural architecture search for deep spiking neural networks," *IEEE Transactions on Evolutionary Computation*, 2024.

[30] H. Hao, X. Zhang, and A. Zhou, "Large language models as surrogate models in evolutionary algorithms: A preliminary study," 2024.

[31] W. Li, Q. Su, and Z. Hu, "A grey prediction evolutionary algorithm with a surrogate model based on quadratic interpolation," *Expert Systems with Applications*, vol. 236, p. 121261, 2024.

[32] L. Xie *et al.*, "Surrogate-assisted evolutionary algorithm with model and infill criterion auto-configuration," *IEEE Transactions on Evolutionary Computation*, 2023.

[33] Y. Song *et al.*, "Reinforcement learning-assisted evolutionary algorithm: A survey and research opportunities," *Swarm and Evolutionary Computation*, vol. 86, p. 101517, 2024.

[34] P. Li *et al.*, "Bridging evolutionary algorithms and reinforcement learning: A comprehensive survey on hybrid algorithms," *IEEE Transactions on Evolutionary Computation*, 2024.

[35] Y. Zhang *et al.*, "Diffusion models are evolutionary algorithms," 2024.

[36] K. Ouyang and S. Fu, "Graph neural networks are evolutionary algorithms," 2024.

[37] X. Wang and M. Zhang, "How powerful are spectral graph neural networks," in *International Conference on Machine Learning*, PMLR, 2022.

[38] Z. Chen *et al.*, "Bridging the gap between spatial and spectral domains: A unified framework for graph neural networks," *ACM Computing Surveys*, vol. 56, no. 5, pp. 1–42, 2023.

[39] S. Biswas *et al.*, "Improving differential evolution through bayesian hyperparameter optimization," in *2021 IEEE Congress on Evolutionary Computation (CEC)*, IEEE, 2021.

[40] P. Bujok and P. Kolenovsky, "Eigen crossover in cooperative model of evolutionary algorithms applied to cec 2022 single objective numerical optimisation," in *2022 IEEE Congress on Evolutionary Computation (CEC)*, IEEE, 2022.

[41] J. Carrasco *et al.*, "Recent trends in the use of statistical tests for comparing swarm and evolutionary computing algorithms: Practical guidelines and a critical review," *Swarm and Evolutionary Computation*, vol. 54, p. 100665, 2020.

[42] J. Yu, Y. Zhang, and C. Sun, "Balance of exploration and exploitation: Non-cooperative game-driven evolutionary reinforcement learning," *Swarm and Evolutionary Computation*, vol. 91, p. 101759, 2024.