A Genetic Algorithm with Tree-structured Mutation for Hyperparameter Optimisation of Graph Neural Networks

Yingfang Yuan  
School of Mathematical and Computer Sciences  
Heriot-Watt University  
Edinburgh, UK  
yyy2@hw.ac.uk

Wenjun Wang  
School of Mathematical and Computer Sciences  
Heriot-Watt University  
Edinburgh, UK  
wenjun.wang@hw.ac.uk

Wei Pang*  
School of Mathematical and Computer Sciences  
Heriot-Watt University  
Edinburgh, UK  
w.pang@hw.ac.uk

Abstract—In recent years, graph neural networks (GNNs) have gained increasing attention, as they possess excellent capability of processing graph-related problems. In practice, hyperparameter optimisation (HPO) is critical for GNNs to achieve satisfactory results, but this process is costly because the evaluations of different hyperparameter settings require excessively training many GNNs. Many approaches have been proposed for HPO which aims to identify promising hyperparameters efficiently. In particular, genetic algorithm (GA) for HPO has been explored, which treats GNNs as a black-box model, of which only the outputs can be observed given a set of hyperparameters. However, because GNN models are extremely sophisticated and the evaluations of hyperparameters on GNNs are expensive, GA requires advanced techniques to balance the exploration and exploitation of the search and make the optimisation more effective given limited computational resources. Therefore, we proposed a tree-structured mutation strategy for GA to alleviate this issue. Meanwhile, we reviewed the recent HPO works which gives the room to the idea of tree-structure to develop, and we hope our approach can further improve these HPO methods in the future.

Index Terms—Generic Algorithm, Tree-structured Mutation, Graph Neural Network, Hyperparameter Optimisation

I. INTRODUCTION

Graph is used to describe the structured objects consisting of nodes and edges, such as citation networks, molecules, and road networks. In graph-related learning and prediction problems, traditional machine learning methods require using feature engineering to process graph data prior to training or prediction [1]. In contrast, graph neural networks (GNNs) are proposed to directly operate on graphs to solve graph-related problems in an end-to-end manner [1]. GNNs exploit neural networks to model graphs and perform representation learning, which requires that GNNs are set with appropriate hyperparameters (e.g., learning rate, the number of neural network layers) to control the learning process. From our perspective, more complicated neural architectures are required to process structured graph data, which may result in the HPO tasks for GNNs being more challenging. Falkner et al. [2] pointed out that deep learning algorithms are very sensitive to many hyperparameters. The work presented in [3] demonstrated that HPO for GNNs is vital for achieving satisfactory results in practice. Therefore, research on effective HPO approaches for GNNs is critical and of great value for GNN applied to various real-world problems.

In brief, HPO is a trial-and-error process to obtain optimal solutions via iteratively generating and evaluating hyperparameter settings until the preset stopping condition(s) are satisfied. Commonly, the objective function of GNNs is selected as the fitness function of HPO to evaluate hyperparameters. However, the evaluation is often very expensive because training GNN models would take a lot of computational resources. On the other hand, with the development of deep learning, neural networks have the increasing number of layers or other optional various hyperparameters (e.g., activation functions, optimisation methods), which results in a large search space and increasing the difficulty of HPO. So, existing works of HPO have been conducted based on the issues of how to find the best solutions with less trials, reducing evaluation cost, and improving the capability of exploration. In this paper, a trial denotes a complete process of evaluating a hyperparameter setting (solution) on the fitness function (objective function).

There are many state-of-the-art HPO approaches with different search strategies. For example, TPE [4] and BOHB [2] are proposed based on Bayesian optimisation (BO) which are suitable for the functions that are expensive to evaluate. However, the former makes use of density functions to model two sets (good and bad) of hyperparameters, instead of using Gaussian process as in [5], [6] as a standard surrogate model to approximate the distribution over the objective function. Furthermore, BHBO combines Bayesian optimization and bandit-based methods to ensure good performance at anytime and speed up the process towards optimal solutions. In general, BO-based methods balance the exploration and exploitation by acquisition function. On the contrary, CMA-ES [7] and HESGA [3] are evolutionary approaches which exploit high-quality individuals to guide the generation of promising off-
spring, while they rely on evolutionary operators to explore search space. CMA-ES employs an adaptive multivariate distribution to sample individuals, and HESGA uses a genetic algorithm which relies on crossover and mutation operators to generate better individuals. To deepen the work of HESGA [3], our research focuses on improving the exploration capability of the mutation operator.

Genetic algorithm with hierarchical evaluation strategy (HESGA) [3] utilises the elite archive to preserve excellent individuals, from which a parent can be selected to generate offspring. Meanwhile, the fast evaluation strategy has been introduced in HESGA to speed up optimisation and save computational resources. Furthermore, HESGA uses the single point mutation as in [8]–[10] which is one of the factor for preventing GA from getting stuck in local optima and maintains the diversity of population [11]. However, we believe that the mutation operation in HESGA can be further explored as only classical mutation operations were used in HESGA. Therefore, in this research we propose a tree-structured mutation (TSM) which employs the tree structure to store hierarchical historical information to guide mutation. In this way, GA can explore the whole search space more adaptively and effectively. Meanwhile, the idea of TSM is also compatible and scalable to other HPO algorithms that hierarchize history information for future discovery.

Our contributions are summarised as below:

- We have developed a tree-structured mutation operation for GA to adaptively maintain the balance between exploration and exploitation for searching the hyperparameter space of GNNs, meanwhile retaining the randomness of mutation operator.
- We conducted a review of advanced HPO methods, and we found that the tree-structured approach has the potential to be integrated into other approaches.
- Our research contributes to the development of molecular machine learning [12] as well as HPO for GNNs in general.

The rest of this paper is organized as follows. Section II introduces relevant work on HPO. In Section III, the details of our tree-structured mutation is presented. The experiments are reported and the results are analysed in Section IV. Finally, Section V concludes the paper and explores some directions in future work.

II. RELATED WORK

In this section, we will first review some popular HPO methods. Then, the genetic algorithm with hierarchical evaluation strategy (HESGA) [3] will be introduced because our tree-structured mutation is proposed upon it. Finally, some mutation strategies will be reviewed.

A. Hyperparameter Optimisation Methods

There have been a lot of state-of-the-art HPO methods, and these methods can be classified into black-box optimization and multi-fidelity optimization approaches [13]. The former focus on the effectiveness of the search algorithms and the latter focuses on efficiently using computational resources in search algorithms [14]–[16].

Black-box optimisation [17], [18] implies that the details of objective function and its gradients information are unknown during searching optimal solutions. In HPO, hyperparameters can be evaluated on the objective function but we have not access for any information about the model. Random search [19] and grid search are two simple and commonly used methods for HPO. Bergstra et al. [4] holds the point that random search is more efficient than grid search given a fixed limited computation budget. However, facing the problems with expensive evaluations, it is very challenging for random search and grid search to achieve satisfactory performance.

The use of Bayesian optimisation (BO) can alleviate the issue of high computational cost [14], [18]. The framework of BO is concluded in sequential model-based algorithm [20] that iteratively exploits historical data to fit surrogate models or other transformations, while the most promising candidate is drawn according to a predefined criterion (i.e. acquisition function). The work presented in [4]–[6], [21] makes use of Gaussian Processes (GPs) to approximate the distribution over the objective function, and the candidate is selected by the acquisition function: probability of improvement (PI) or expected improvement (EI). In contrast, TPE [4] employs Parzen estimators as a surrogate to directly model promising and unpromising hyperparameters, and the candidate solutions are drawn according to EI which is proportional to the ratio of two density functions. During searching optimal solutions, the acquisition function plays a significant role for balancing exploration and exploitation.

Furthermore, evolutionary computation has demonstrated the capability of solving HPO problems [3], [7], [22], [23]. CMA-ES [7] imitates the biological evolution, assuming that no matter what kind of gene changes, the results (traits) always follow a Gaussian distribution of a variance and zero-mean. Meanwhile, the generated population is evaluated on objective function, and a portion of the individuals with excellent performance is selected to suggest evolution, moving to the area where better individuals would be drawn with higher probability.

Successive halving [24] is an bandit-based multi-fidelity method for efficiently allocating computational resource that give the most budget to the most promising individuals. Meanwhile, successive halving is an iterative process in which the best halve of individuals are retained and half of the individuals with lower performance are discarded when a portion of total computational resource runs out. Hyperband [25] and BOHB [2] are proposed based on successive halve, but the former proposed an modification that frequently performs the successive halving method with different budget allocations to find the best solutions. Li et al. [25] pointed out that because the terminal losses are unknown, the trials of different allocations are helpful to recognise high-quality solutions. For example, two individuals might not have significant difference on fitness in the 30th epoch, but greater difference may be observed in the 50th epoch if more computational resources.
B. Genetic Algorithm

Genetic algorithm (GA) as a class of evolutionary computation which has been applied to HPO \[3\], \[26\], \[27\]. GA evaluates the qualities of individuals in the population by using a fitness function. To search high-quality individuals, the GA relies on crossover, mutation, and selection operators. The selection is inspired by natural selection which means GA selects individuals as parents according to the probabilities (e.g., roulette wheel) to generate next generation by crossover and mutation, and the probabilities are proportional to their quality (fitness). There are a lot of crossover techniques such as uniform crossover \[28\], single-point \[29\], and heuristic crossover \[30\] which all aim to recombine genes from the parents to generate offspring. In the evolutionary process, mutation operators as the last step are generally assigned with a low probability to be invoked. However, it plays a significant role for preventing GA from getting stuck in local optima. Similar to the probabilistic sampling of solutions in BO-based optimisation, mutation is employed to endure GA with the exploration capability of solutions.

Variable length genetic algorithm \[31\] has been proposed to solve neural network HPO problem by increasing the length of chromosome (i.e., enlarge search space) if the fitness does not satisfy preset conditions. Moreover, HESGA \[3\] introduced elite archive mechanism which is used to store a number of current optimal individuals. Parents are selected respectively from elite archive and the population according to roulette wheel (Fig. 1). In this way, new offspring always own at least a portion of the strong genes from the parent in the elite archive if crossover occurs. Meanwhile, HESGA provides the strategy of using fast and full evaluation methods, to alleviate the issue of the expensive evaluations. Fast evaluation is applied to the whole population. The individuals having the most difference of fitness in the early training stage (10% ∼ 20% of the total epochs) are considered promising, and it assumes that individuals with steeper learning curve have greater potential to achieve satisfactory results. Meanwhile, full evaluation is employed to measure a small portion of individuals selected by fast evaluation. The strategy of fast and full evaluation can be considered as multi-fidelity optimisation, which focuses on decreasing the evaluation cost by combining a large number of low cost evaluations with low-fidelity and a small number of costly evaluation with high-fidelity.

C. Mutation

Many mutation strategies have been proposed for GA, and we only present some of them which are related to our research. Random (uniform) mutation and non-uniform mutation have been proposed in \[32\]. In random mutation, where a gene is replaced with a random value drawn from a defined range. On the other hand, in non-uniform mutation, the strength of mutation decreases along with increase in the number generations. In adaptive mutation \[33\], \[34\], each solution is assigned with a mutation rate, in order to maintain the diversity of the population without affecting algorithms performance. Based on Gaussian mutation, self-adaptive mutation \[35\] allows GA to vary the mutation strength during the evolution, while self-adaptive Gaussian mutation based adaptation of population size is proposed in \[36\]. In Pointed directed (PoD) mutation \[37\], each gene is tightly associated with a single bit that guides the direction of mutation which the gene may follow. Compared with Gaussian mutation, Deep et al. \[38\] reported that PoD mutation is more advantageous in unconstrained benchmark problems. The above mentioned mutation operators exploit the historical information which is generated during evolution to adjust the mutation strategy dynamically, which inspired our TSM to improve the naive single point mutation used in \[3\].

III. Method

In this section, the design of tree-structured mutation (TSM) will be described. Meanwhile, we will introduce the use of TSM in HESGA.

A. HESGA

The framework of HESGA \[3\] is shown in Fig. 1. The TSM is proposed as an further improvement of HESGA. The original HESGA uses the binary encoding scheme for hyperparameters on HPO: each hyperparameter is assigned with a fixed-length binary code with a step size (i.e., resolution). For example, if the batch size ranges from 8 to 512 with a step size of 8, and then a batch size of 16 is represented by \([0, 0, 0, 0, 0, 1, 0]\). Single point crossover and mutation are employed in HESGA.
In this research, based on the framework of HESGA, we use TSM to replace the single point mutation, considering that a more sophisticated mutation may lead to better performance. During the experiments, we observed that in elite archive and populations, there may exist the same individuals which waste computational resources for evaluating them. Many reasons may lead to this issue, for example, an extremely excellent individual in the population will be selected to be included in elite archive meanwhile outperforms other individuals in elite archive, and it means this individual has higher probability to be selected as both Parent A and B in the next generation. Two identical parents will invalid crossover by which the same individuals will be generated. To solve this issue, we set the condition in Steps (5.2) (Fig. 4) to prohibit duplicates. If the duplication is detected, the mutation is forcibly invoked. Based on TSM, we use two strategies: TSM with an given individual and TSM without an given individual respectively for the situation: mutation is invoked with preset probability in Steps (5.1 and 5.2) and force mutation when duplicates are detected in Steps (5.3 and 5.4), as shown in Fig. 4.

1) TSM with An Given Individual: HESGA operates the single point mutation with the probability of 0.2, as indicated by Step (5) in Fig. 1. TSM is also invoked with the same probability in Step (5.1) in Fig. 4. Compared with the change of a bit of binary encoding under a uniform distribution, TSM uses historical information recorded in the tree to generate a probability distribution to guide the mutation. Specifically, when mutation is activated, for the individual concerned, its corresponding ranges of hyperparameters (batch size, learning rate, the size of convolutions layers, and the number of neurons in fully-connected layer), which forms a subspace of the whole search space and can be represented by a binary string 0000. Meanwhile, the left and right child nodes respectively define the lower and higher ranges of hyperparameters by thresholds. For example, if $s$ ranges from 8 to 512, the median value 256 is chosen as the threshold to give two subspace with same size. $s_0$ indicates a range from 8 to 255 and $s_1$ ranges from 256 to 512. In this way, the whole search space is divided into many subspaces, the number of subspaces equals to the total number of leaf nodes $c$ (i.e., the maximum width $n_a$), or the squared of the dimension of hyperparameter settings $n_h^2$, where $n_h$ is also the maximum depth of the tree without leaf nodes.

To use the tree to suggest mutations, the internal and leaf nodes take the responsibility to record historical information. The internal nodes are used to store the times of mutation occurred on different dimensions in a subspace. Concurrently, the leaf nodes $c$ are used to record the times of subspace has been visited. For example, when a hyperparameter setting $\lambda$ sampled from the subspace $(s_0, l_0, f_0, n_0)$ has been evaluated on an objective function, the value of node $c_0$ will be added with 1. If $\lambda$ underwent a mutation on batch size, then the counter in the $s_0$ node will also be added with 1. In this way, the whole search space is hierarchized which increase the ability of exploration for mutation.

In this research, based on the framwork of HESGA, we use TSM to replace the single point mutation, considering that a more sophisticated mutation may lead to better performance. During the experiments, we observed that in elite archive and populations, there may exist the same individuals which waste computational resources for evaluating them. Many reasons may lead to this issue, for example, an extremely excellent individual in the population will be selected to be included in elite archive meanwhile outperforms other individuals in elite archive, and it means this individual has higher probability to be selected as both Parent A and B in the next generation. Two identical parents will invalid crossover by which the same individuals will be generated. To solve this issue, we set the condition in Steps (5.2) (Fig. 4) to prohibit duplicates. If the duplication is detected, the mutation is forcibly invoked. Based on TSM, we use two strategies: TSM with an given individual and TSM without an given individual respectively for the situation: mutation is invoked with preset probability in Steps (5.1 and 5.2) and force mutation when duplicates are detected in Steps (5.3 and 5.4), as shown in Fig. 4.

1) TSM with An Given Individual: HESGA operates the single point mutation with the probability of 0.2, as indicated by Step (5) in Fig. 1. TSM is also invoked with the same probability in Step (5.1) in Fig. 4. Compared with the change of a bit of binary encoding under a uniform distribution, TSM uses historical information recorded in the tree to generate a probability distribution to guide the mutation. Specifically, when mutation is activated, for the individual concerned, its corresponding subspace (i.e., pathway) along the tree will be...
identified. In that pathway, the times $t$ of previous occurred mutation respectively according to the mutated position stored in the internal nodes will be given to Equation (1) to compute probabilities. In Equation (1), $i$ is the depth of an internal node given pathway $p$, and $t_i$ represents the times recorded in that node. $f$ is a reciprocal function, $j$ are the depth of all internal nodes (e.g., $n_1$ with depth 4, $f_0$ with depth 3, $l_0$ with depth 2, and $s_0$ with depth 1) in pathway $p$, and $t_j$ is the times stored in those nodes respectively. The probabilities will determine where the mutation happens following the roulette wheel method, and the mutated individual will be used to update the tree to affect future mutation. In this way, in a subspace, the dimension that has been previously explored frequently will have less probability to be explored in the future. When the dimension is sampled, the new value is drawn from the search space defined in that node by uniform distribution. For example, when $s_0$ is selected, a new value from 8 to 255 with the resolution of 8 will be drawn and converted into binary coding to replace the orginal fragment. The TSM with an given individual is summarised in Algorithm 1.

$$p(t_i) = \frac{f(t_i)}{\sum_{j=1}^{n_h} f(t_j)} \quad i,j \in \{1, 2, ..., n_h\}, \text{given } p.$$  

(1)

2) **TSM without An Given Individual:** To avoid duplicated individuals existing in the elite archive, in Fig. 1 Step (9) is added with a duplication detection mechanism. The duplicates will be deleted except the one with the most fitness value (in experiments, there exist fluctuations when evaluating the same hyperparameter settings due to the randomness in the GNN training and evaluation process). Meanwhile, if the duplicates are found in the population, we will take TSM without an given individual (shown in Fig. 4 Steps (5.3 and 5.4)) to generate a new individual to replace them. In fact, TSM with an given individual is also feasible to deal with the duplication issue, but here we proposed the TSM without an given individual to increase the population diversity.

Similar to the Algorithm 1, TSM without an given individual (Algorithm 2) also exploits the information stored in the tree and follow the roulette wheel to select points of mutation. However, the latter on selecting a subspace to sample new individuals (mutated ones), rather than given a subspace to compute the probabilities of each subspace by Equation (2) where $c$ are the indexes of leaf nodes, $n_s$ is the total number of leaf nodes (i.e., maximum width), $f$ is the reciprocal function. Each leave node corresponds a subspace, and the times of a subspace being explored is recorded on it. Eq. (2) indicates that the subspace with a large number of $t$ will have less probability to be explored by mutation. In this way, exploitation can be achieved by crossover and elite archive, and exploration is facilitate by mutations. The pseudo-code is shown in Algorithm 2.

$$p(t_c) = \frac{f(t_c)}{\sum_{c=1}^{n_s} f(t_c)} \quad i,c \in \{1, 2, 3...n_s\}, \text{given leaf nodes.}$$  

(2)

IV. EXPERIMENTS

To assess the effectiveness of TSM, we conducted the experiments to compare HESGA with single-point mutation [3] and HESGA equipped with TSM. The HESGA in our experiments is modified to reject duplication, and as mentioned in Section III-A the duplicates found in population will be forcibly re-sampled by TSM without an given individual. The settings of HESGA are as follows: the max generation is set to 10, probabilities of crossover and mutation are 0.8 and 0.2, which are the same to the original experimental settings [3]. Meanwhile, the proportion of size of elite archive to the size of population is 0.5, as considering that 0.1 may lead to the select pressure.

Our experiments have been conducted on three benchmark datasets: ESOL, FreeSolv, and Lipophilicity from [12]. They
There are four hyperparameters selected for HPO, including the neural architectures to model circular fingerprint for posed to be used for molecular machine learning. GC exploits graph model (GC) has been selected because it was proven that there are many types of GNN algorithms, but the neural fingerprint used to assess the performance of GNNs. Meanwhile, there are three molecular property prediction datasets which are used to assess the performance of HPO. Meanwhile, in the HPO experiments, the best hyperparameter setting found as the fitness function for HPO, and it is also used to measure the performance of HPO. The best hyperparameter settings are completely different and come from different two subspaces, which indicates the complex and multi-modal nature of the underlying hyperparameter search space, and the discovery in different subspaces is meaningful.

In the experiments on FreeSolv (Table II), HESGA presents better performance than HESGA+TSM. FreeSolv is the smallest dataset used in our experiments. Small size of dataset is intrinsically challenging for GC to learn and keep a stable performance, which may lead the underperformance of HESGA+TSM since its strength is exploration, stable and accurate feedback (i.e., the results of evaluation on GC) is essential for conducting efficient exploration.

Furthermore, when duplicates appear in population, HESGA uses single-point mutation which retains most of the genes from parents, and relatively speaking the strength of the mutation is small, while HESGA+TSM is guided by the tree-based mutation, which explores the search space more broadly compared with HESGA. Therefore, we considered that when the size of a dataset is relatively small, global exploration may bring more fluctuation and unknown factors upon the instability of GC, which bring more difficulties to the problems, while “local” exploration might be more effective since it reduces the underlying unknown risks from global area.

In contrast, Lipophilicity is the largest dataset in our experiments, and as shown in Table III, HESGA+TSM outperforms HESGA in this dataset with significant difference, and lower standard deviation means the result of HESGA+TSM is more stable. When the size of dataset increased, the full evaluation becomes more accurate, and it provides more helpful information for GA to generate strong offspring in terms of exploitation. Meanwhile, more effective exploration becomes necessary, and HESGA+TSM shows better performance because TSM makes use of historical information to search hyperparameter space rather than search the each part of the hyperparameter space with equal probability.

V. CONCLUSION AND FUTURE WORK

In this paper, we proposed TSM for genetic algorithm as applied to HPO of GNNs, which improved the performance of HESGA on the largest dataset in our experiments. Compared with operating HPO on a relatively small dataset, the achieved improvement of HPO performance on the larger dataset is more meaningful since the evaluations are more expensive. Meanwhile, the research of GA-based HPO for GNN is in

Algorithm 2 TSM without An Given Individual

```
1: Input tree, t = []
2: for c = 1 → ns do
3:   t_total = f(t_c) ⊢ t_c denotes the times recorded in node c, f is reciprocal function
4:   t.append(f(t_c))
5: end for
6: t’ ← each element in t is divided by t_total
7: i = roulette wheel(n_s, t’,) ⊢ subspace i
8: v = []
9: for j = 1 → n_h do
10:   r_j ← the defined value range in node with degree j
11:   v_j = uniform(r) ⊢ mutated value v_j by sampling from a uniform distribution given range r_j
12: v.append(binary(v_j))
13: end for
14: s = v
15: Output individual s
```

are three molecular property prediction datasets which are used to assess the performance of GNNs. Meanwhile, there are many types of GNN algorithms, but the neural fingerprint graph model (GC) has been selected because it was proposed to be used for molecular machine learning. GC exploits the neural architectures to model circular fingerprint for learning the molecular representations for prediction tasks. There are four hyperparameters selected for HPO, including s_b (batch size), l_r (learning rate), s_c (the size of graph convolution layer), and n_n (the number of neurons in fully-connect layer); the search space is constructed as follows: the range of s_b is 8 ~ 512 with step size 8, the range of l_r is 0.0001 ~ 0.0032 with step size 0.0001, the range of s_c is 8 ~ 512 with step size 8, and the range of n_n is 32 ~ 1024 with step size 32. As mentioned in Section III-B, the defined value range in node with degree j is divided into many subspace, which indicates the complex and multi-modal nature of the underlying hyperparameter search space, and the discovery in different subspaces is meaningful.

As in [3], the root mean square error (RMSE) of GC is used as the fitness function for HPO, and it is also used to measure the HPO performance. The best hyperparameter setting found by a HPO method (HESGA and HESGA+TSM) will be given to GC to run 30 times, and the mean of the 30 RMSEs are used to measure the performance of HPO. Meanwhile, in order to make the comparisons more convincing, t-test with significance level of two tailed 5% (i.e. 10%) is employed to compare the performance of HESGA and HESGA+TSM and see if there exists significant difference. In t-test, negative t value and the hypothesis h = 1 indicates that HESGA outperforms HESGA+TSM; positive t value and the hypothesis h = 1 indicates that HESGA+TSM significantly outperform HESGA; or h = 0 (the null hypothesis) means that there is no significant difference between HESGA and HESGA+TSM.

In the experiments on ESOL (Table I), HESGA+TSM has smaller mean and standard deviation of RMSEs than HESGA. However, there is no significant difference between HESGA and HESGA+TSM at the significance level α = 5%. We considered that the size of ESOL dataset constrains the performance of HESGA+TSM, because a sufficient amount of data is a prerequisite for GC possessing better performance with a more complex architecture. In Table II, HESGA+TSM found larger s_c and n_n which means the GC with more parameters, and they require more training data to tune them for a better performance. Furthermore, if we set the significance level α = 10%, HESGA+TSM outperforms HESGA in the validation data set. Meanwhile, it is obvious that two hyperparameter settings are completely different and come from different two subspaces, which indicates the complex and multi-modal nature of the underlying hyperparameter search space, and the discovery in different subspaces is meaningful.

In the experiments on FreeSolv (Table II), HESGA presents better performance than HESGA+TSM. In contrast, Lipophilicity is the largest dataset in our experiments, and as shown in Table III, HESGA+TSM outperforms HESGA in this dataset with significant difference, and lower standard deviation means the result of HESGA+TSM is more stable. When the size of dataset increased, the full evaluation becomes more accurate, and it provides more helpful information for GA to generate strong offspring in terms of exploitation. Meanwhile, more effective exploration becomes necessary, and HESGA+TSM shows better performance because TSM makes use of historical information to search hyperparameter space rather than search the each part of the hyperparameter space with equal probability.
TABLE I
THE RESULTS ON ESOL DATASET

| Hyperparameters | Train | Validation | Test |  |
|-----------------|-------|------------|------|---|
| HESGA | s_b=64 | 0.2847 | 0.8843 | 0.8862 | Mean RMSE |
|             | t_r=0.0028 | | | |
|             | s_c=304 | 0.0424 | 0.0509 | 0.0506 | Mean STD |
|             | n_r=64 | | | |
| HESGA + TSM | s_b=48 | 0.2734 | | 0.86585 | Mean RMSE |
|             | t_r=0.0012 | | | |
|             | s_c=320 | 0.0333 | 0.0357 | 0.0463 | Mean STD |
|             | n_r=320 | | | |
| T-test on results with significance level of \( \alpha = 5\% \) | \( t = 1.1249, h = 0 \) | \( t = 1.5963, h = 0 \) | \( t = 0.9311, h = 0 \) |  |

TABLE II
THE RESULTS ON FREE SOLV DATASET

| Hyperparameters | Train | Validation | Test |  |
|-----------------|-------|------------|------|---|
| HESGA | s_b=24 | 0.5498 | 1.0970 | 1.0370 | Mean RMSE |
|             | t_r=0.00340 | | | |
|             | s_c=224 | 0.1251 | 0.1199 | 0.1246 | Mean STD |
|             | n_r=64 | | | |
| HESGA + TSM | s_b=8 | 1.1181 | 1.4083 | 1.3508 | Mean RMSE |
|             | t_r=0.0010 | | | |
|             | s_c=480 | 0.2692 | 0.2408 | 0.2874 | Mean STD |
|             | n_r=576 | | | |
| T-test on results with significance level of \( \alpha = 5\% \) | \( t = -10.3063, h = 1 \) | \( t = -6.2298, h = 1 \) | \( t = -5.3940, h = 1 \) |  |

TABLE III
THE RESULTS ON LIPOPHILICITY DATASET

| Hyperparameters | Train | Validation | Test |  |
|-----------------|-------|------------|------|---|
| HESGA | s_b=136 | 0.3656 | 0.7402 | 0.7293 | Mean RMSE |
|             | t_r=0.00331 | | | |
|             | s_c=232 | 0.0881 | 0.0476 | 0.0462 | Mean STD |
|             | n_r=1024 | | | |
| HESGA + TSM | s_b=168 | 0.2390 | | 0.6962 | Mean RMSE |
|             | t_r=0.0011 | | | |
|             | s_c=304 | 0.0676 | 0.0284 | 0.0269 | Mean STD |
|             | n_r=864 | | | |
| T-test on results with significance level of \( \alpha = 5\% \) | \( t = 6.1358, h = 1 \) | \( t = 4.2642, h = 1 \) | \( t = 2.7381, h = 1 \) |  |

an early stage, our work will help other researchers to better explore this area. On the other hand, our research contributes to the development of molecular machine learning, which may facilitate the research from the relevant domain such as materials science, biology and drug discovery. Furthermore, the idea of utilising tree structure to hierarchize search space and record the historical information can be applied to other HPO methods. Finally, in future, we believe more research can be done along further developing TSM for evolutionary HPO. We list some potential directions as follows

A. Reward and Rejection Mechanism

Currently, the inner and leaf nodes are used to store the times of previous exploration. Inspired by reinforcement learning [41], the results of evaluations on fitness function can be recorded as rewards with the times together to affect future discovery, by reconstructing Eqs. (1) and (2) in this way, TSM is not only for increasing the exploration ability, but also for improving the exploitation in the later stage by introducing reward mechanism. Meanwhile, the times and rewards can also work together with in a surrogate model to reject those unpromising trials according to historical information.

B. Adaptive Threshold

In TSM, the binary tree requires thresholds to be specified to generate child nodes. In our experiments, we choose median values as thresholds. However, it is easy to observe that \( s_b \) in all experiments are from the lower range (i.e. the left child node). If the thresholds are adaptive and can be updated by the rewards referring the percentile in TPE [4], TSM may be further improved to better focus on adaptively adjusted subspace.

REFERENCES

[1] J. Zhou, G. Cui, Z. Zhang, C. Yang, Z. Liu, L. Wang, C. Li, and M. Sun, “Graph neural networks: A review of methods and applications,” arXiv preprint arXiv:1812.08434, 2018.

[2] S. Falkner, A. Klein, and F. Hutter, “Bohb: Robust and efficient hyperparameter optimization at scale,” in International Conference on Machine Learning. PMLR, 2018, pp. 1437–1446.
