Efficiency Enhancement of Evolutionary Neural Architecture Search via Training-Free Initialization

Quan Minh Phan*†, Ngoc Hoang Luong*†
*University of Information Technology, Ho Chi Minh City, Vietnam
†Vietnam National University, Ho Chi Minh City, Vietnam

Abstract—**In this paper, we adapt a method to enhance the efficiency of multi-objective evolutionary algorithms (MOEAs) when solving neural architecture search (NAS) problems by improving the initialization stage with minimal costs. Instead of sampling a small number of architectures from the search space, we sample a large number of architectures and estimate the performance of each one without invoking the computationally expensive training process but by using a zero-cost proxy. After ranking the architectures via their zero-cost proxy values and efficiency metrics, the best architectures are then chosen as the individuals of the initial population. To demonstrate the effectiveness of our method, we conduct experiments on the widely-used NAS-Bench-101 and NAS-Bench-201 benchmarks. Experimental results exhibit that the proposed method achieves not only considerable enhancements on the quality of initial populations but also on the overall performance of MOEAs in solving NAS problems. The source code of the paper is available at https://github.com/ELO-Lab/ENAS-TFI.**

Index Terms—**Evolutionary computation, Neural architecture search, Multi-objective optimization**

I. Introduction

The world is witnessing rapid developments of artificial intelligence (AI) with machine learning models that can recognize objects with almost absolute accuracy or generate conversations that make it impossible for us to recognize that they are not human-products [3]. Deep neural networks (DNNs) are one of the key components to those successes. However, manually designing an efficient neural network requires a lengthy trial-and-error process and the involvement of domain experts. Neural architecture search (NAS), aiming to automatically design DNN models, is thus an interesting research topic of automated machine learning. The use of evolutionary algorithms (EAs) is a suitable approach for solving NAS because of the similarity between NAS and optimization problems. When addressing NAS as an optimization problem, the goal is to find high-performance architectures that satisfy certain constraints on the characteristics (e.g., the number of network parameters, the number of floating point operations per second). A brief procedure of EAs on solving NAS problems is presented as follows. At first, the initial population is produced with each individual corresponding to an architecture sampled from the search space. In next generations, the population evolves from generation to generation by producing new individuals, removing worse individuals and keeping the better ones. After the stopping criteria are satisfied, the individuals in the final population are considered as the resulting architectures. The main purpose of EAs' procedures is to make the quality of the population better over generations. Realizing the sampling process in the initialization stage is usually performed randomly, we suppose that the performance of EAs on NAS can be improved further if the quality of the initial population is enhanced. This assumption is partially made clear in [1], [12]. However, these methods are only conducted on single-objective evolutionary algorithms (SOEAs). In this study, we not only verify the mentioned results on SOEAs, but also modify and extend these methods for improving the efficiency of multi-objective evolutionary algorithms (MOEAs) in solving NAS problems.

Our contributions in this paper are:

- We propose a method for enhancing the performance of MOEAs in solving multi-objective NAS problems via improving the quality of the initial population with a negligible cost.
- We provide the detail of employing available NAS benchmarks (e.g., NAS-Bench-101, NAS-Bench-201) for creating different NAS problems that can be used to evaluate the performance of EAs in solving NAS.

The remainder of this paper is organized as follows. Section II briefly introduces related works. The detail of our proposed method is presented in Section III. The experimental settings are shown in Section IV. The obtained results are exhibited and discussed in Section V. Finally, conclusions and future directions are given in Section VI.

II. Related Works

Neural Architecture Search (NAS) is an automated machine learning subfield that involves automatically designing network architectures with top performance on specific tasks (e.g., classification, object detection, visual question answering). In essence, NAS can be formulated as an optimization problem. The aim is to find one (or many) architectures that optimize some desired objectives. In the context of solving NAS as a single-objective optimization problem, the optimization objective is usually one of the performance metrics (e.g., the accuracy, the error rate) [14], [16]. Otherwise, when solving NAS as a multi-objective optimization problem, besides the priority of optimizing the performance metrics, the effort is also put into optimizing simultaneously efficiency metrics (e.g., the number of network parameters, the number of floating point operations per second) [10], [13].

Because NAS is considered as an optimization problem, using evolutionary algorithms (EAs) is an effective approach for solving it. In fact, there are many studies that have applied EAs for solving NAS problems [10], [14], [16]. In [16], Genetic Algorithm (GA) was used to find the convolutional neural network that achieves the highest recognition accuracy on the CIFAR-10 dataset. In another study [14], a different algorithm named Aging Evolution was used to explore the architecture that has the lowest error rate on the CIFAR-10 dataset and transfer it to the ImageNet dataset. Non-dominated Sorting Genetic Algorithm II (NSGA-II) was used to minimize both the error rate and the number of floating point operations per second (FLOPs) on the CIFAR-10 dataset [10].

After analyzing the procedures of many different EAs which are applied for solving NAS problems, we notice that the initial population is usually created at random and thus we can make an impact on this process to improve the quality of the initial population. A given question is that if we improve the quality of the initial population, whether we also have an improvement on the performance of EAs. Moreover, because of the problem related to computational resources cost, this question is not easy to answer. In NAS, reducing the computational resources cost (e.g., the number of GPU hours used) is a challenge, so it is inefficient if we overpay for resources to obtain some enhancements on the quality of initial populations but achieve insignificant improvements on the overall performance. To overcome this obstacle, zero-cost (or training-free) NAS proxies have been applied for improving the quality of the initial population [1], [12].

In brief, a zero-cost proxy is an indicator used to estimate the performance of network architectures without the training process. These metrics are computed based on the characteristics of the neural network (e.g., the parameters, the activation). Mellor et al. [12] proposed a zero-cost proxy for the network accuracy based on the activation patterns in a network when different mini-batches of data are passed through the network. In [4], [7], the condition number of neural tangent kernels and the number of linear regions are used to estimate the performance of networks. Because the computation cost to compute these training-free performance indicators is often negligible (or considerably less than the actual network training), we are able to consider a larger number of candidate architectures during a NAS run. However, the drawback of zero-cost proxies is that if there exists a low correlation between the zero-cost proxy value and the ground-truth accuracy, the search effectiveness of rank-based algorithms would be affected.

The previous studies [1], [12] have taken the advantage of zero-cost proxies and proposed a method called *Warmup* for enhancing the quality of the EAs' initialization stage. In this method, a large number of architectures in the search space are randomly sampled and their performance is estimated by using a zero-cost proxy. The high-performance architectures are then chosen to become the individuals of the initial population. Nevertheless, the effect of the Warmup method has only been demonstrated on single-objective EAs. In this study, we first validate the effectiveness of the Warmup method in single-objective NAS problems and then adapt it for enhancing the performance of MOEAs on solving multi-objective NAS problems. Note that all multi-objective NAS problems which we mention in this study are bi-objective minimization problems.

III. Warmup for Multi-objective Evolutionary Neural Architecture Search

The central idea in the Warmup approach is to sample a large amount of architectures T from the search space and then pick the n best architectures out of them as the individuals in the initial population of an EA, where n is the population size [1], [12]. Overall, the Warmup method has two major steps: *sampling* and *selection*. In this method, the value of T is usually much greater than the value of n ($T \gg n$) because of the strong impact of the value of T on the performance. The greater the value of T is set, the more search space is explored. In other words, the more architectures that we sample, the more chance of finding high-quality architectures. During a typical NAS run, a performance metric (e.g., the accuracy) is commonly used for ranking architectures and thus the training process is required. However, training and evaluating all T architectures would cost too much computational resource. Hence, it is not suitable to utilize an exact performance metric for comparing architectures in this context. To address this problem, a feasible approach is to use a zero-cost proxy for estimating the performance of architectures.

In the previous studies [1], [12], the Warmup method is mainly designed for single-objective NAS problems. At the *selection* step, architectures are ranked through their zero-cost proxy values. Depending on the requirement of the problem is to minimize or maximize, the sort order is ascending or descending respectively. Then, n architectures that have the smallest (or largest) zero-cost proxy value are chosen as the individuals of the initial population. For solving multi-objective NAS problems, besides the performance of architecture, the efficiency of architecture is the remaining objective. Therefore, it is inefficient if we just simply apply the original Warmup method on MOEAs. As a result, we propose some modifications to make the Warmup method work effectively for MOEAs. The details of our proposed Warmup method for MOEAs are presented as follows.

A. Sampling

The sampling process is performed as follows. First, T architectures are randomly sampled from the search space. Instead of architecture evaluation via the validation error rate, which requires many epochs of actual model training, we estimate the performance of these architecture via the *Synaptic Flow* metric [15], which is a zero-cost proxy. The Synaptic Flow metric is used to evaluate the importance of a parameter in a network by approximating the change in the product of all parameters when removing that parameter [15]. To employ the Synaptic Flow metric for estimating the performance of an architecture, we employ the mechanism as in [1], i.e., we calculate the Synaptic Flow metric value of each parameter in the network architecture and take the sum over all parameters.

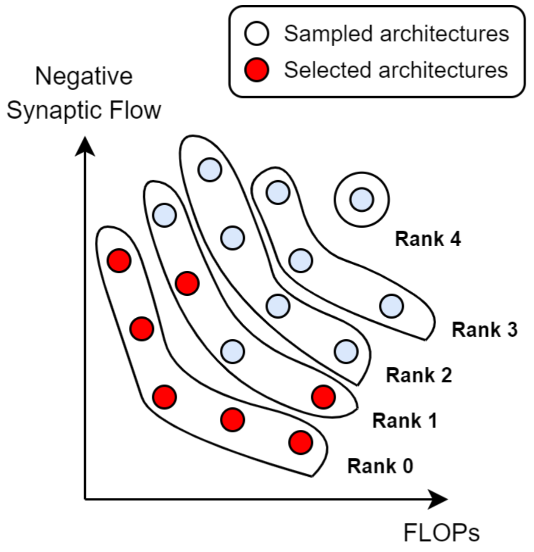

Fig. 1. Example of the *selection* step in our proposed Warmup for MOEAs. In this example, the number of sampled architectures and selected architectures are 17 and 7, respectively. We use the negative Synaptic Flow to have the minimization formulation on both objectives.

B. Selection

After all architectures are evaluated, we rank them via their Synaptic Flow values and efficiency metric values by using the non-dominated sorting [6]. This sorting mechanism is performed based on the definition of the Pareto dominance [5], i.e., an architecture $\boldsymbol{x}$ is said to Pareto dominate another architecture $\boldsymbol{y}$ if $\boldsymbol{x}$ is not worse than $\boldsymbol{y}$ in any objective and $\boldsymbol{x}$ is strictly better than $\boldsymbol{y}$ in at least one objective. In a population of candidate architectures, the architectures which are not Pareto dominated by any architectures are put into rank 0. Rank i consists of architectures which are only Pareto dominated by some architectures in lower ranks ($< i$) and not dominated by any others. The sorting process is finished when all architectures in the population are assigned the ranks that they belong to. The architectures in the lower ranks are considered better than the ones in the higher ranks and vice versa. After ranking architectures, we select the architectures from low ranks to high ranks as the individuals for the initial population. In case the number of remaining slots is fewer than the number of architectures with the same rank (e.g., there are five architectures but only having the last two slots), we fill up these slots by choosing randomly. Fig. 1 illustrates an example of the *selection* step in the Warmup method for MOEAs.

IV. Experimental Settings

A. NAS Problems

We utilize NAS-Bench-101 [17] and NAS-Bench-201 [8] benchmarks to produce NAS problems. A NAS benchmark can be seen as a dictionary in which the keys and the values are architectures and the information on architectures, respectively. All architectures of each NAS benchmark are in the same search space and being trained and evaluated on the same datasets, training strategies, and configuration settings. The usage of NAS benchmarks is not to find state-of-the-art architectures, but to ensure the equity when conducting performance comparisons between algorithms on NAS. The details of NAS benchmarks used in our experiments are presented as follows.

- NAS-Bench-101 is a benchmark consisting of information on 423,624 unique architectures. The information on each architecture including the encoded structure, the performance metrics (i.e., the train accuracy, the validation accuracy, the test accuracy), and the efficiency metrics (i.e., the number of trainable model parameters). The performance metrics and efficiency metrics of architectures are obtained on the CIFAR-10 dataset.
- NAS-Bench-201 contains 15,625 architectures and their performance on three different datasets (i.e., CIFAR-10, CIFAR-100, and ImageNet16-120). The information of each architecture consists of two performance metrics (i.e., the validation accuracy, the test accuracy) and three efficiency metrics (i.e., the number of trainable model parameters, the latency of model, and FLOPs).

By using two above NAS benchmarks, we have created four single-objective NAS problems and four multi-objective NAS problems. All problems are minimization ones. Table I provides the details of each NAS problem.

TABLE I
THE DETAILS OF EACH NAS PROBLEM

Single-objective NAS problems			
Problem Name	**NAS Benchmark**	**Dataset**	**Objective**
SO-NAS-101	NAS-Bench-101	CIFAR-10	Validation error
SO-NAS-201-1	NAS-Bench-201	CIFAR-10	
SO-NAS-201-2		CIFAR-100	
SO-NAS-201-3		ImageNet16-120	
Multi-objective NAS problems			
Problem Name	**NAS Benchmark**	**Dataset**	**Objective**
MO-NAS-101	NAS-Bench-101	CIFAR-10	Validation error #Params
MO-NAS-201-1	NAS-Bench-201	CIFAR-10	Validation error FLOPs
MO-NAS-201-2		CIFAR-100	
MO-NAS-201-3		ImageNet16-120	

B. Experimental Setup

a) Single-objective NAS problems: We employ Genetic Algorithm (GA) [9] as the baseline SOEA for solving single-objective NAS problems. The search procedure of GA on NAS is presented as follows. The search is started by randomly sampling a number of individuals (i.e., candidate architectures) from the search space to make the initial population. After initializing, the population is divided into pairs of individuals and crossover is then performed for each pair to create new offspring (i.e., new candidate architectures). The crossover of architectures is the exchange of layers or operations between architectures. During mutation, each architecture is then altered with a small probability by randomly changing one (or several) of its structural characteristics (e.g., the layers, the operations, the connections). Next, the current population and the offspring are combined to create a selection pool. A selection mechanism is performed on the selection pool to pick out the better architectures as the individuals for the new population. The tournament selection method is usually utilized as the selection mechanism. By using the tournament selection, the selection pool is divided into tournaments with each tournament having k architectures and the winners (i.e., the best architectures) of tournaments are added to the new population. The selection process is executed until the new population is filled up. The crossover, mutation, and selection process are performed continuously in a sequence until satisfying certain termination conditions.

The configuration settings of GA are $2-$point crossover with probability 0.9, bit-string mutation with probability $1/l$ (l is the length of the solution), and tournament selection with $k = 4$. For the Warmup method, the number of sampled architectures T is set equal to 500 for all problems. As a result, we perform two variants for each problem: baseline GA, and GA with Warmup. For each problem, we perform each algorithm variant 21 runs independently. About the population size n and the maximum number of evaluations (i.e., the number of architectures are truly trained and evaluated), we set 100 and $5,000$ respectively for SO-NAS-101 problem; 40 and $1,000$ respectively for three problems which are created on NAS-Bench-201 benchmark (i.e., SO-NAS-201-1, SO-NAS-201-2, SO-NAS-201-3).

During an NAS run, the optimization objective is to minimize the validation error rate. After all experiments finish, we assess the performance of the baseline GA and the GA with Warmup. For this evaluation purpose, for every algorithm, we compute the average test error rate of the best architectures over 21 runs in each generation. The test error values are used here to show the convergence of each algorithm toward the optimal architecture in each benchmark (assuming to be the one with the best test error rate), and they are not employed by any algorithm during the search process. Note that all the training, validation, and test error values can be queried from the benchmark databases.

b) Multi-objective NAS problems: We choose NSGA-II [6] as the baseline MOEA for solving multi-objective NAS problems. The workflow of NSGA-II is almost similar to GA ones. There is a main difference in the selection mechanism. For NSGA-II, the individuals of the new population are selected by using the non-dominated sorting and the crowding distance. The non-dominated sorting is used for ranking architectures and the crowding distance is used for comparing architectures in the same rank.

For experiments on multi-objective NAS problems, we employ an external secondary population, which is called *elitist archive* $\mathcal{A}$ [11]. The elitist archive is used to store the non-dominated architectures obtained during the search. When a new architecture is created, if it is not Pareto dominated by any elite architecture in $\mathcal{A}$, we then insert it into $\mathcal{A}$. Existing architectures in $\mathcal{A}$ that are Pareto dominated by the newly-added architecture would be removed from $\mathcal{A}$. The architectures in $\mathcal{A}$ form the so-called *approximation front* containing non-dominated architectures obtained so far by an MOEA during an NAS run. When an MOEA terminates, the final approximation front in $\mathcal{A}$ can be regarded as the optimization result achieved by the algorithm. Besides using $\mathcal{A}$, we also use a different set $\mathcal{D}$ to store dominated solutions. During the search process, if an architecture is Pareto dominated by another architecture, it will be added to $\mathcal{D}$. Every time a new architecture is generated, if it is already in $\mathcal{D}$, we will discard it and generate a new one. In this study, the size of $\mathcal{A}$ and $\mathcal{D}$ are not limited.

We use the standard configuration settings for NSGA-II ($2-$point crossover with probability 0.9, and bit-string mutation with probability $1/l$). For the Warmup method, we sample $T = 500$ architectures for all problems. We perform two variants, i.e., baseline NSGA-II and NSGA-II with Warmup, for solving multi-objective NAS problems. Similar to experiments on the single-objective NAS problems, we also perform each variant 21 runs independently for each problem. The population size n and the maximum number of evaluations are set to: 100 and $30,000$ for MO-NAS-101 problem; 20 and $3,000$ for MO-NAS-201-1, MO-NAS-201-2, and MO-NAS-201-3 problems.

We use non-dominated fronts formed by architectures in $\mathcal{A}$ over generations of each variant to compare their performance. To assess the quality of the fronts, we utilize two performance indicators: the Inverted Generational Distance (IGD) [2] and the hypervolume [5]. The formula of calculating the IGD value is presented as follows.

$$IGD(\mathcal{S}, \mathcal{P}_F) = \frac{1}{|\mathcal{P}_F|} \sum_{f^0 \in \mathcal{P}_F} \min_{x \in \mathcal{S}} \{d(f(x), f^0)\} \qquad (1)$$

where $\mathcal{P}_F$ is the Pareto-optimal front (or a reference front), $\mathcal{S}$ is an approximation front (i.e., an non-dominated front), and $d(\cdot, \cdot)$ computes the Euclidean distance in the objective space. The IGD value of an approximation front $\mathcal{S}$ indicates how close $\mathcal{S}$ is to the Pareto-optimal front $\mathcal{P}_F$ and how well-spread $\mathcal{S}$ is along the the Pareto-optimal front $\mathcal{P}_F$ [2]. When using IGD value for comparison between two algorithms, the algorithm with the lower IGD value is the better one.

In (1), the Pareto-optimal front $\mathcal{P}_F$ is required to calculate the IGD value. Because of using NAS benchmarks, we can easily determine all non-dominated architectures in the benchmarks to form the Pareto-optimal front for each problem. However, it is impossible to pre-determine the Pareto-optimal front in real-world problems. The hypervolume is a suitable alternative performance indicator in these cases. The hypervolume value is computed as the volume enclosed by the non-dominated front and a reference point (e.g., the point which has the worst possible objective values). When using the hypervolume value to compare the quality of fronts, the one with the larger hypervolume value is the better one. Similar to single-objective NAS experiments, we also use the test performance in computing IGD and hypervolume values.

V. Results and Discussions

A. Single-objective NAS Problems

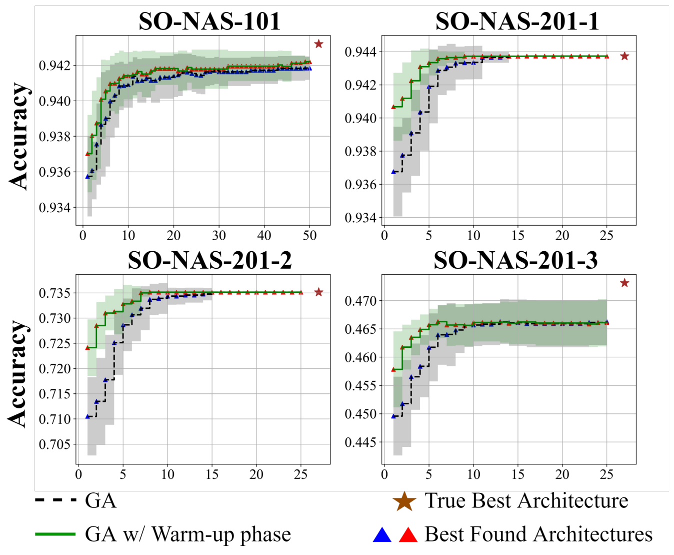

Fig. 2. Average performance of algorithms on single-objective NAS problems. Horizontal axis: the number of generations.

Fig. 2 shows that the quality of initial populations in algorithms with the Warmup method have been improved. This quality improvement helps increase the performance of the algorithms in the early stages (i.e., in early generations). These improvements have high practical value and are relevant to real-world situations because we do not have much computational resources for training and evaluating a large number of architectures in practice. In generations 1-5, statistical tests (i.e., Student's t-tests) at the 99% confidence level show that these enhancements are significant on SO-NAS-201-1, SO-NAS-201-2, and SO-NAS-201-3 problems. Although the number of sampled architectures is relatively smaller than the search space size ($500 \ll 15,625$), the quality of the initial population has improved impressively. We can figure out the architectures with the high performance without any training cost by using the zero-cost proxy Synaptic Flow. For the SO-NAS-101 problem, we do not achieve a significant enhancement on the quality of the initial populations and also on the performance of algorithms in early generations. We consider that the difference in the correlation coefficient is the explanation for the difference between the results in problems which are created on NAS-Bench-101 and NAS-Bench-201. Abdelfattah et al. [1] have shown that the Spearman's rank correlation coefficient between the final validation accuracy and the *Synaptic Flow* metric on NAS-Bench-101 and NAS-Bench-201 are 0.37 and 0.75, respectively. Because of the low correlation between the network accuracy and the proxy value of the architectures in NAS-Bench-101 search space, warm-up initial populations, that are filled with individuals selected based on the proxy, would not be significantly better than randomly initialized populations.

B. Multi-objective NAS Problems

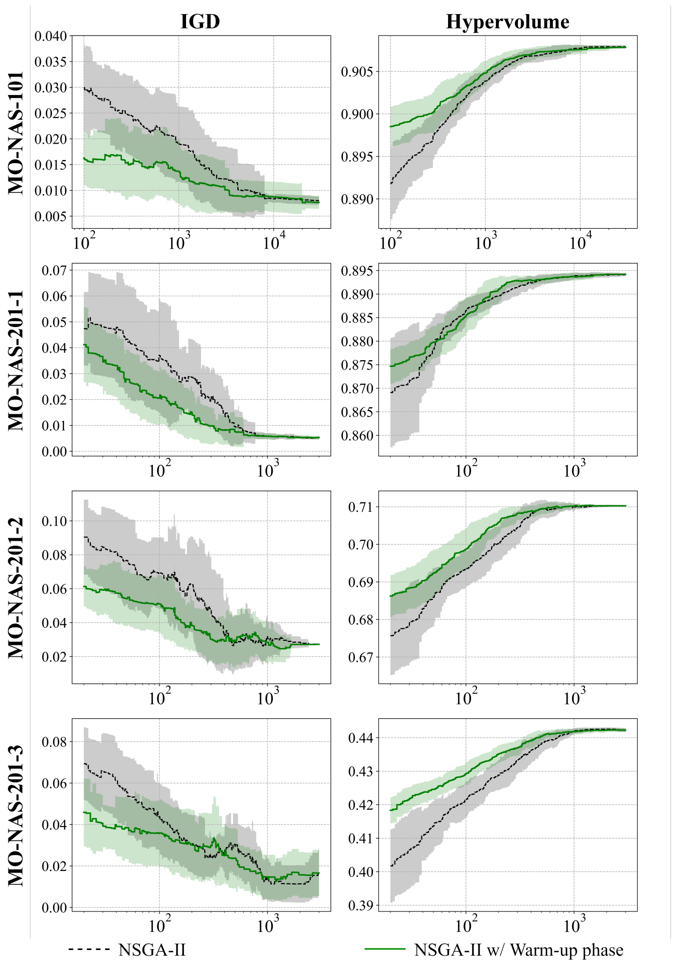

Fig. 3. Average performance of algorithms on multi-objective NAS problems. Horizontal axis: the number of evaluations (logarithm scale).

Similar to the results of single-objective NAS problems, Fig. 3 and Table II show that the Warmup method also improves the quality of initial populations and the performance of NSGA-II in solving multi-objective NAS problems in early generations. However, when the searches are run longer for more generations, the early head start advantage of NSGA-II with warm-up populations gradually diminishes compared to NSGA-II with random initialization, and the performance gaps are not statistically significant anymore. Although the quality of initial populations has been improved, it is not enough to obtain better trade-off fronts at the end of an NAS run when all the computational budget is used up. These results are understandable since the Warmup method only affects the initialization step, and it still depends on whether variation operators (i.e., crossover, mutation) are able to create better offspring. We suppose that the results here can be further improved if we combine the Warmup method with a local search method (as in [13]) that could potentially guide EAs during the search process. Nevertheless, we argue that the improvements of the Warmup method in early generations are

TABLE II
MEANS AND STANDARD DEVIATIONS (IN BRACKETS) OF THE IGD AND THE HYPERVOLUME OF ALGORITHMS ON 21 RUNS ON MO-NAS-201-3. THE BOLD RESULTS INDICATE THE BETTER ALGORITHM BETWEEN THE ORIGINAL ALGORITHM AND THE ALGORITHM INTEGRATING THE WARMUP METHOD (99% CONFIDENCE LEVEL)

IGD		
#Evals	NSGA-II	With warmup phase
20	0.065646(0.020546)	**0.041707(0.017836)**
100	0.033789(0.011013)	0.028145(0.014171)
200	0.024144(0.009621)	0.021379(0.014494)
500	0.017056(0.015256)	0.010007(0.009758)
1000	0.001569(0.000919)	0.002173(0.004673)

Hypervolume		
#Evals	NSGA-II	With warmup phase
20	0.402649(0.010583)	**0.417805(0.004194)**
100	0.422741(0.005287)	**0.430497(0.002575)**
200	0.430366(0.003801)	**0.436583(0.001967)**
500	0.43932(0.001896)	**0.441414(0.000539)**
1000	0.442234(0.000245)	0.442208(0.000297)

not trivial, since in practice a few generations of EAs solving NAS with actual network training could already incur a huge computational cost. It is therefore highly beneficial it we can efficiently and reliably obtain high-quality architectures within limited computing budgets.

VI. CONCLUSIONS

We verified the effectiveness of the Warmup method in improving the performance of SOEAs and adapted it to enhance the performance of MOEAs in solving NAS problems. When employing the Warmup method for MOEAs, a large number of architectures are sampled and their performance are estimated by a zero-cost proxy. Next, the non-dominated sorting is used to rank architectures via their proxy values and efficiency metrics. The best architectures are then chosen as the individuals of the initial population. To evaluate the efficacy of the Warmup method on the quality of initial populations and the overall performance of EAs, we designed single-objective NAS problems and multi-objective NAS problems by using well-known NAS benchmarks for performing experiments. The obtained results confirm the efficacy of the original Warmup method on enhancing the performance of SOEAs. Besides, the results showed that our proposed modification to the Warmup method enhances the quality of initial populations of MOEAs in almost all problems. The effectiveness of training-free initialization is governed by the correlations between the network accuracy performance and the zero-cost proxies. Designing a more sophisticated, but still efficient, initialization mechanism and finding other proxy metrics with higher correlations are potential future works.

ACKNOWLEDGMENT

This research is funded by Vietnam National University HoChiMinh City (VNU-HCM) under grant number DSC2021-26-06.

REFERENCES

[1] Abdelfattah, M.S., Mehrotra, A., Dudziak, L., Lane, N.D.: Zero-cost proxies for lightweight NAS. In: 9th International Conference on Learning Representations, ICLR 2021. OpenReview.net (2021)

[2] Bosman, P.A.N., Thierens, D.: The balance between proximity and diversity in multiobjective evolutionary algorithms. IEEE Trans. Evol. Comput. **7**(2), 174–188 (2003)

[3] Brown, T.B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, A., et al.: Language models are few-shot learners. In: Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020 (2020)

[4] Chen, W., Gong, X., Wang, Z.: Neural architecture search on imagenet in four GPU hours: A theoretically inspired perspective. In: 9th International Conference on Learning Representations. OpenReview.net (2021)

[5] Deb, K.: Multi-objective optimization using evolutionary algorithms. Wiley-Interscience series in systems and optimization, Wiley (2001)

[6] Deb, K., Agrawal, S., Pratap, A., Meyarivan, T.: A fast and elitist multiobjective genetic algorithm: NSGA-II. IEEE Trans. Evol. Comput. **6**(2), 182–197 (2002)

[7] Do, T., Luong, N.H.: Training-free multi-objective evolutionary neural architecture search via neural tangent kernel and number of linear regions. In: 28th International Conference on Neural Information Processing. ICONIP 2021. Lecture Notes in Computer Science, vol. 13109, pp. 335–347. Springer (2021)

[8] Dong, X., Yang, Y.: Nas-bench-201: Extending the scope of reproducible neural architecture search. In: 8th International Conference on Learning Representations, ICLR 2020. OpenReview.net (2020)

[9] Holland, J.H.: Genetic algorithms. Scientific american **267**(1), 66–73 (1992)

[10] Lu, Z., Whalen, I., Boddeti, V., Dhebar, Y.D., Deb, K., Goodman, E.D., Banzhaf, W.: Nsga-net: neural architecture search using multi-objective genetic algorithm. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2019. pp. 419–427. ACM (2019)

[11] Luong, H.N., Bosman, P.A.N.: Elitist archiving for multi-objective evolutionary algorithms: To adapt or not to adapt. In: Proceedings of the 12th International Conference on Parallel Problem Solving from Nature (PPSN). Lecture Notes in Computer Science, vol. 7492, pp. 72–81. Springer (2012)

[12] Mellor, J., Turner, J., Storkey, A.J., Crowley, E.J.: Neural architecture search without training. In: Proceedings of the 38th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 139, pp. 7588–7598. PMLR (2021)

[13] Phan, Q.M., Luong, N.H.: Enhancing multi-objective evolutionary neural architecture search with surrogate models and potential point-guided local searches. In: International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2021. Lecture Notes in Computer Science, vol. 12798, pp. 460–472. Springer (2021)

[14] Real, E., Aggarwal, A., Huang, Y., Le, Q.V.: Regularized evolution for image classifier architecture search. In: Proceedings of the AAAI conference on artificial intelligence. pp. 4780–4789. AAAI Press (2019)

[15] Tanaka, H., Kunin, D., Yamins, D.L., Ganguli, S.: Pruning neural networks without any data by iteratively conserving synaptic flow. In: Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020 (2020)

[16] Xie, L., Yuille, A.L.: Genetic CNN. In: IEEE International Conference on Computer Vision, ICCV 2017. pp. 1388–1397. IEEE Computer Society (2017)

[17] Ying, C., Klein, A., Christiansen, E., Real, E., Murphy, K., Hutter, F.: Nas-bench-101: Towards reproducible neural architecture search. In: Proceedings of the 36th International Conference on Machine Learning, ICML 2019. Proceedings of Machine Learning Research, vol. 97, pp. 7105–7114. PMLR (2019)