An Alternative Pareto-based Approach to Multi-objective Neural Architecture Search

Meyssa Zouambi, Clarisse Dhaenens, Julie Jacques

An Alternative Pareto-based Approach to Multi-objective Neural Architecture Search

1st Meyssa Zouambi
Univ. Lille, CNRS, Centrale Lille
UMR 9189 CRIStAL, F-59000
Lille, France
meyssa.zouambi@univ-lille.fr

2nd Clarisse Dhaenens
Univ. Lille, CNRS, Centrale Lille
UMR 9189 CRIStAL, F-59000
Lille, France
clarisse.dhaenens@univ-lille.fr

3rd Julie Jacques
Univ. Lille, CNRS, Centrale Lille
UMR 9189 CRIStAL, F-59000
Lille, France
julie.jacques@univ-lille.fr

***Abstract*—Neural architecture search (NAS) is a field that automates the architecture design of neural networks. NAS can be modeled as an optimization problem. It describes a space of possible architectures and looks for the most performing one. NAS, however, is not limited to finding the most task-accurate neural network. It can consider other objectives during the search to meet different demands and requirements (model size, latency, energy consumption, etc.). Several methods were proposed for multi-objective NAS. Most of them are either based on the scalarization of objectives, or use a Pareto-based approach. Methods based on scalarization require preference weighting between objectives and can suffer from suboptimality, while the Pareto-based methods found in NAS usually require complex operators and many parameters to tune. The goal of our work is to offer an alternative Pareto-based method that solves the above issues. In this paper, we first present a formulation of the NAS problem as a multi-objective optimization (MO) problem. We then design a dominance-based multi-objective local search (DMLS) to solve it. Unlike other NAS methods, our work uses a simple encoding, few parameters, and does not require preference weighting of objectives. To assess its performance, we evaluate this algorithm on a specialized multi-objective NAS benchmark to optimize both accuracy and network complexity. We compare it to state-of-the-art MO methods of this benchmark. Results show that our method finds significantly more Pareto optimal solutions than NGSA-II and overpasses single-objective local search for the same evaluation budget. We conclude that DMLS provides a more practical MO approach for NAS while providing superior performances.**

***Index Terms*—Neural architecture search, Multi-objective local search, Multi-objective NAS.**

I. Introduction

Designing the architecture of a neural network is arguably one of the most crucial steps in developing deep learning models. Different tasks and datasets require tailored architectures to obtain good performances. For this reason, researchers and machine learning engineers spend considerable time trying different architecture configurations for their models. This step is conducted by trial and error and was done manually for decades. Neural architecture search (NAS) solves the time-consuming task of architecture design. It has drawn notable attention in recent years for its ability to produce state-of-the-art models.

979-8-3503-1458-8/23/$31.00 ©2023 European Union

NAS is also not limited to finding the neural network with the best predictive performance. It can take into account other considerations as well, such as complexity, inference time, energy consumption, etc. Multi-objective NAS includes approaches that optimize more than one objective simultaneously. Several methods have been proposed in this context, including Pareto methods that are largely population-based [5] [9] [3], or methods using a scalarization of objectives [1] [20] [6]. Population-based methods are complex to implement for NAS and require a lot of parameter tuning. On the other hand, the scalarization of objectives is considered sub-optimal for multi-objective optimization according to [2].

The goal of our work is to provide an alternative Pareto-based approach that solves the above issues. We design a dominance-based multi-objective local search (DMLS) for NAS. Our method is easy to implement, is not based on the scalarization of objectives, and does not require many parameters. It extends the local search (LS) approach that has proved its effectiveness in single objective NAS [19]. It can naturally exploit techniques based on local considerations that speed up the global search time, such as weight inheritance [16] or network morphism [18]. DMLS has also already been successfully applied in a machine learning context in [8].

The rest of the paper is organized as follows: Section II gives background knowledge about neural architecture search, its formulation in a multi-objective optimization context, and an introduction to the dominance-based multi-objective local search. Section III describes the proposed approach, explains the different parts used for DMLS (solution encoding, neighborhood, and evaluation function), and defines its process. Section IV describes the evaluation protocol to compare the proposed DMLS approach with NSGA-II and LS on a specialized multi-objective benchmark for NAS and discusses the results. Section V concludes this work and gives future research directions.

II. Background

A. Neural Architecture Search

Deep learning models power most modern applications nowadays. Their predictive performances rely on both data and neural network design. When designing neural networks, several choices have to be made, ranging from the number of

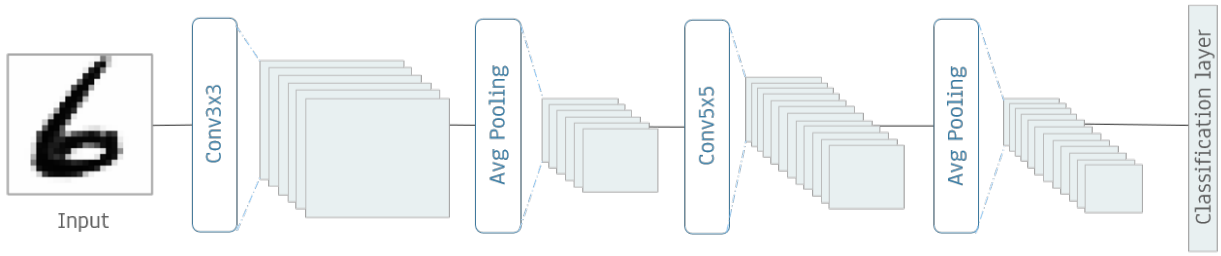


Fig. 1: Example of a simple CNN architecture for image classification. ConvNxN represents a convolutional operation with kernel size N. Each operation results in a set of feature maps. The last layer uses the resulting feature maps for classification

layers to the type of operations used at each layer. The number of architecture parameters grows exponentially with the complexity of the network. This can leave machine learning engineers with many potential models that perform differently based on their structure and configuration.

Figure 1 shows an example of a simple convolutional neural network (CNN) where many architectural parameters need to be defined. For example, the type of operation of each layer (e.g., Convolution, Pooling), the kernel size of the convolutional operations (e.g., 3x3, 5x5), the type of pooling functions (e.g., Max Pooling, Average Pooling), etc.

There is no clear procedure on how to choose the best parameters. Architecture design is a trial-and-error process that is both time and resource-consuming.

The goal of neural architecture search is to automatically find the architectures that maximize the model performance from a large set of possible architecture configurations. This set of possible architectures defines the *NAS search space*, and can contain a very large number of architectures. To efficiently explore it, NAS utilizes *search strategies*. The most popular ones include gradient-based approaches [11], evolutionary algorithms [12], and reinforcement learning [7].

B. Multi-objective Neural Architecture Search

Neural architecture search has been recently applied to find architectures that fulfill several criteria. It optimizes the predictive performance and other objectives as well. This trend meets the current demands of models that can now run on different platforms and for more demanding tasks. Examples of these objectives are latency, energy consumption, size, etc. Following the notation found in [13], the multi-objective NAS problem can be formulated as follows:

$$\text{Minimize } \mathbf{F}(\boldsymbol{a}) = \left(f_1\left(\boldsymbol{a}; \boldsymbol{w}^*(\boldsymbol{a})\right), \ldots, f_k\left(\boldsymbol{a}; \boldsymbol{w}^*(\boldsymbol{a})\right), f_{k+1}(\boldsymbol{a}), \ldots, f_m(\boldsymbol{a})\right)^T$$

$$\text{Subject to } \boldsymbol{w}^*(\boldsymbol{a}) \in \operatorname{argmin} \mathcal{L}(\boldsymbol{w}; \boldsymbol{a}), \quad \boldsymbol{a} \in \boldsymbol{\Omega}_a, \quad \boldsymbol{w} \in \boldsymbol{\Omega}_w$$

$\boldsymbol{a}$ represents a candidate architecture from the search space Ω_a, and $\boldsymbol{w}(\boldsymbol{a})$ its associated weights. The weights are obtained after training the architecture on the training set using the loss function $\mathcal{L}(\boldsymbol{w}; \boldsymbol{a})$.

$\mathbf{F} : \boldsymbol{\Omega} \rightarrow \mathbb{R}^m$ are the m objectives to minimize during the architecture search. These objectives can either depend on both the architecture and its weights (f_1 to f_k), for example, the accuracy. Or they can depend on the design of the architecture only (f_{k+1} to f_m), such as the number of parameters or the latency.

To solve this multi-objective optimization problem, two types of approaches have been used in the literature. The first type is based on the scalarization of objectives. It creates a weighted function of f_i to obtain a single objective optimization problem. Many notable works in multi-objective NAS use this approach. In [1], authors use a reinforcement learning algorithm to search for the optimal network architecture for both predictive performance and latency. The reinforcement agent constructs the network based on a reward signal defined by a weighted sum of the two objectives.

Authors in [20] use a differentiable NAS framework to search for architectures that optimize the same objectives. Their loss function is a weighted product of cross-entropy (prediction metric) and latency.

The work in [6] also proposes a differentiable NAS method in which the search space is composed of densely connected blocks with different widths and spatial resolutions. Here, the loss function is defined as the weighted sum instead of the weighted product.

Techniques based on the scalarization of objectives suffer from two main issues. First, they require using weights that defines the importance of each objective before the search. This needs prior knowledge and experiments. Second, they use an aggregation of objectives which is considered suboptimal [2].

The second type of approach used in the literature is multi-objective optimization using a Pareto-front. This approach does not require defining a trade-off between predictive performance and other objectives a-priori. Since, typically, there is no feasible solution that minimizes all objective functions at once, this approach chooses a set of solutions that are not dominated by one another. In a minimization problem, a solution a is said to dominate another solution b if $\forall i \in \{1, \ldots, m\}, f_i(a) \leq f_i(b)$ and $\exists i \in \{1, \ldots, m\}, f_i(a) < f_i(b)$

The Pareto-front contains all solutions that are not domi-

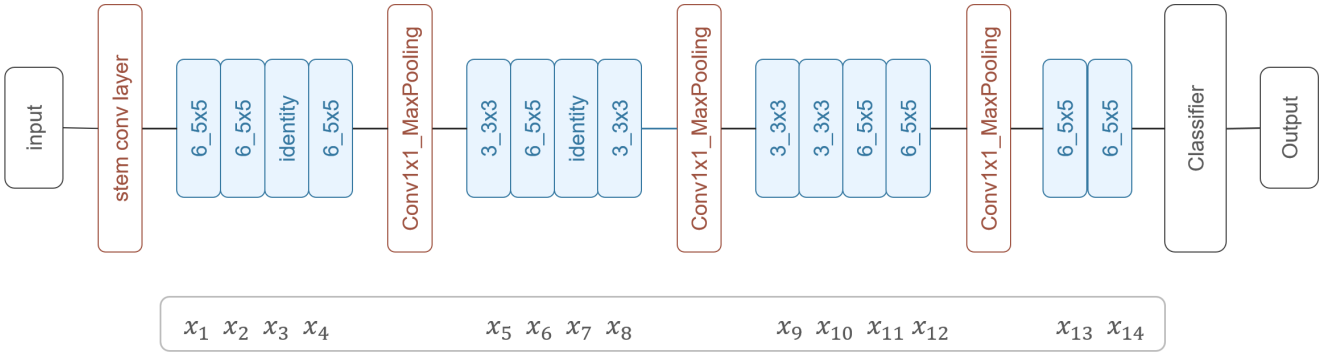


Fig. 2: Solution encoding for MacroNASBenchmark. Blue cells are the operations that are chosen for each architecture. Red cells are pre-fixed operations that are common in all architectures (best seen in color).

nated by any other solution of the search space.

Works using this approach for NAS include population-based methods like NSGA-II or other variations. In [5], authors propose a population-based method that renders, at each generation, a set of architectures generated using local transformation. They use Lamarckian inheritance [15] to permit faster architecture evaluation. Their work, however, uses complex operators which makes it difficult to adapt to other tasks.

In [14], authors propose a genetic algorithm based on NSGA-II to improve both classification performance and the number of floating-point operations during the architecture search. They initialize their population from hand-crafted architectures and explore new ones using crossovers and mutations.

Authors in [22] use the cuckoo search algorithm and incorporate the fast non-dominated sorting approach for optimizing classification accuracy, inference latency, number of parameters, and number of floating-point operations. New architectures are constructed via Lévy flight mutations.

Population-based methods generally require a considerable amount of parameter-tuning for their design (operators, size of the population, etc.). They are slower to converge compared to non-population-based methods but explore more of the search space, which could lead to better solutions.

C. Dominance-based Multi-objective Local Search

In a mono-objective context, local search methods are known to provide good-quality solutions for many hard combinatorial optimization problems. They start from an initial solution, selected at random or using another heuristic, and then generate neighbors of this solution by applying a neighborhood function N. This function creates adjacent solutions by applying small transformations to the current solution. These neighbors are explored (and evaluated) following one of the available exploration strategies. For example, in the first-improve strategy, the first evaluated neighbor that has a better quality is selected and replaces the current solution. After this update, the process is repeated with this new solution. The search stops when no neighbor is better than the current solution, so there is no possibility to improve it (the heuristic reaches a local optimum).

This method has proven its effectiveness for the NAS problem in a mono-objective context [19]. Unlike most of its search strategies, LS is easy to implement and does not require complex encoding or parameter tuning, which makes it easily applicable to newer NAS tasks. Local search also has the advantage of naturally exploiting certain methods that speed up the search time of the NAS process. Such as weight inheritance [16] or network morphism [18], which reduce the training time for architecture evaluation.

Dominance-based multi-objective local search (DMLS) is an adaptation of the single-objective local search for multi-objective optimization problems [10]. Unlike classical LS, instead of manipulating one solution, DMLS maintains a set of solutions in an archive. This archive contains elements that are non-dominated by one another. Meaning that if we compare two solutions from this archive, none of them is superior. The archive is progressively improved by exploring the neighborhood of the solutions it contains. When it meets a neighbor that is non-dominated by the elements of the archive, it is added to the set. When solutions from the archive become dominated during the exploration of the neighborhood, they are removed automatically.

DMLS has already been successfully applied in machine learning contexts [8]. Since it is a variation to LS that is already state-of-the-art for many NAS benchmarks [19], it is a good candidate for the multi-objective NAS problem.

III. Proposed Approach

Considering the importance of multi-objective neural network design, we expanded the use of local search to multi-objective NAS. While previous work has explored the use of LS in this context [3], their method is based on the scalarization of objectives. Our approach, on the other hand,

uses a Pareto approach with dominance-based multi-objective local search to perform multi-objective resolution.

Evaluating the quality of a neural architecture is very costly. It requires training the architecture on a training set and assessing its performance on a validation set. Depending on the task, architecture, and hardware used, this step can take several hours for a single evaluation. To simplify experimentation, various benchmarks have been proposed in the literature [4], [17], [21]. They provide pre-computed metrics for all possible architectures for their given task and pre-defined search space. Our work uses the **MacroNASBenchmark** [3], a benchmark designed specifically for evaluating multi-objective NAS methods. It includes a search space of over 200,000 unique architectures, along with their predictive performance on image classification for two popular datasets: CIFAR10 and CIFAR100. In addition, each architecture is assigned a metric for complexity. Our proposed approach can nonetheless be adapted for other search spaces, datasets, or metrics by following the same encoding and procedures.

In the following, we explain the different components of the DMLS for NAS and give a detailed description of our proposed approach.

A. Solution Encoding

In the context of NAS, a solution a defines an architecture of a neural network. The encoding used in our method is a list of values. This list contains the different parameters chosen that uniquely define an architecture in the search space.

For the **MacroNASBenchmark** [3], each solution consists of 14 unrepeated cells (x_1 to x_{14}). Each cell x_i can take one of three options (*Identity*; *3_3x3*: MBConv[1] with expansion rate 3 and kernel size 3; *6_5x5*: MBConv with expansion rate 6 and kernel size 5). Figure 2 shows an example of a solution from this search space. On top, is the representation of the architecture, and on the bottom, is the corresponding encoding, where x_1 to x_{14} contains the values of each of the corresponding cells.

B. Neighborhood Function

The neighborhood function N generates a neighborhood $N(a)$ that contains the architectures close to a. In our work, we define the neighborhood of a solution as the set of solutions that differs by exactly one cell from the current one. The set of neighbors is obtained by selecting each cell and enumerating all the possible values. For the used benchmark, there are 28 possible neighbors for each solution (2 other alternative values for the 14 cells).

Figure 3 shows an example of a neighbor generation. On top, there is the current solution, and on the bottom, its neighbor. Generating one neighbor consists in choosing one operation and changing it by another, for example here, 6_5x5 (MBConv with expansion rate 6 and kernel 5) is replaced with an Identity operation.

[1]Inverted Linear BottleNeck layer with Depth-Wise Separable Convolution

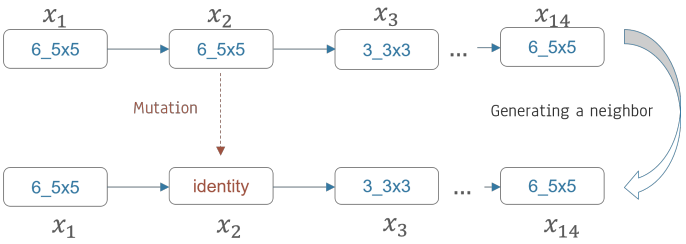


Fig. 3: Neighbor generation with the neighborhood function.

C. Solution Evaluation

To evaluate an architecture, we chose two objectives, one representing the predictive performance of the model (to maximize) and one for the model's complexity (to minimize). Optimizing the model's complexity along with the predictive performance is usually useful when we want a lightweight model or a model that is easier to interpret. MacroNASBenchmark provides for each architecture precomputed accuracies (sum of well-classified images divided by the total number of images) for two image classification datasets (CIFAR10 and CIFAR100). It also provides a metric for complexity (the Million Multiply Accumulate Operations MMACs) that is used as the second objective.

D. Solution archive

The solution archive contains a set of solutions that are non-dominated by one another. In this case, the objectives used are $f_1 = 1 - accuracy$ and $f_2 = MMACs$.

We initialize the archive with a trivial solution (containing all Identity operations) which is the optimal solution for the MMACs metric. We sample different solutions uniformly at random from the search space. When meeting a solution that is not dominated by other solutions from the archive, we add it until we reach the desired initial size. Any solution that becomes dominated is automatically removed.

The initial archive size is the only parameter that needs to be tuned for this approach, the impact of its choice will be studied in the experiment section. The maximum size of the archive is set to infinity (unbounded).

E. Algorithm Process

Algorithm 1 gives the main steps of this method. First, it starts by initializing an archive of size p with a set of solutions. After that, it randomly picks one of the solutions to explore. A solution is explored by modifying a single variable every time and evaluating its resulting neighbor. Once it finds a neighbor that is non-dominated by the current archive, it adds it to the archive and removes any solution that will be dominated by it. To speed up the search and explore more of the search space, we mark the solution that was used to generate this neighbor as *visited* and don't use it again. The algorithm then repeats this process for another random solution in the archive. If we explore all the neighbors of a solution, it is also marked as *visited* and is no longer picked. The algorithm converges when

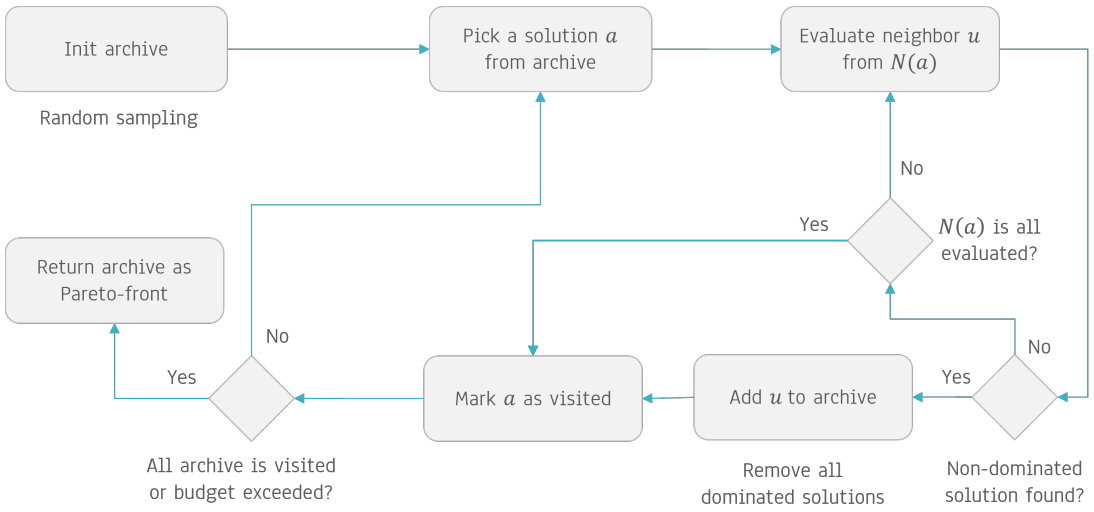


Fig. 4: The dominance-based multi-objective local search process applied for NAS.

there is no more unvisited solution in the archive. The resulting archive is the approximation to the Pareto set of this method.

Figure 4 illustrates the process of this method.

IV. Experiments

A. Experimental Protocol

To assess the performance of the proposed algorithm, we run it for the CIFAR10 and CIFAR100 image classification datasets provided by the MacroNASBenchmark [3]. We chose this benchmark because it is specifically designed for evaluating multi-objective NAS, as stated in their paper. CIFAR10 and CIFAR100 are also the two most popular datasets in NAS benchmarks for this task [4], [17], [21].

We compare our work with two other methods previously tested in [3]. Both of these use different approaches: a population-based method with NSGA-II and a scalarization-based approach using a customized local search[2]. The two methods are described as state-of-the-art for this benchmark [3].

We use accuracy and MMACs as the two objectives for the architecture search. Both of them are provided by the benchmark. We limit the number of evaluations to 6000 evaluated architectures (approx 3% of the size of the search space). We fix this number because in a non-benchmark setting evaluating architectures is very costly. If an algorithm stops before the evaluation budget is reached, it restarts the search.

The benchmark provides the globally-optimal solutions (Pareto optimal). It is the set of solutions that are non-dominated by the whole search space. They are obtained by exhaustive search and will be a point of comparison to the approximate Pareto-front obtained by other methods.

The setup of each method is as follows:

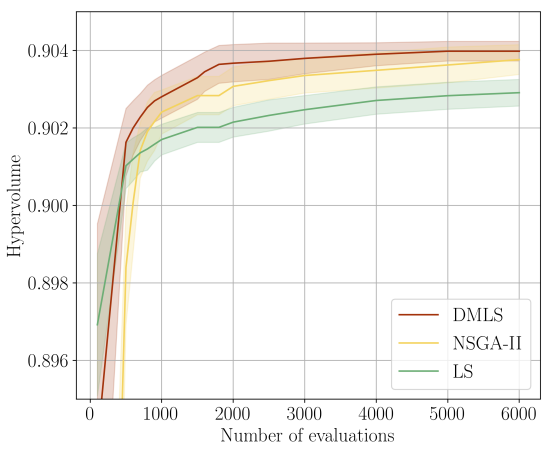
- For the NSGA-II algorithm, it uses a 2-point crossover, a single-variable mutation with probability $p_m = 1/l$,

[2]Two more algorithms are presented for this benchmark. Unfortunately, we were not able to run them for comparison using the provided code.

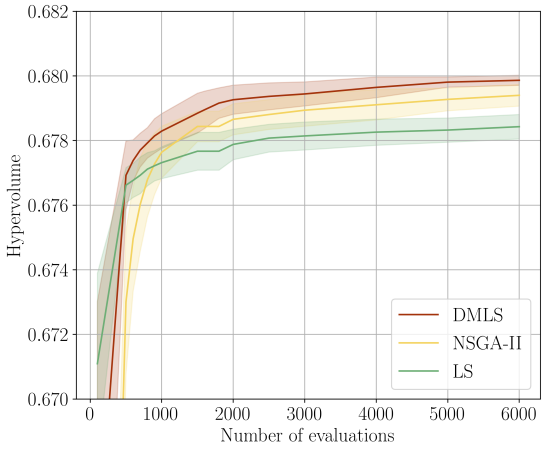
Algorithm 1 Dominance-based Local Search for NAS

Input: Ω_a: search space N: neighborhood function $maxBudget$: max evaluations p: initial size of archive
Output: set of non-dominated solutions

Initialization: $archive \leftarrow \{\}$ *//empty archive*
while size($archive$) $< p$ and $maxBudget$ not reached **do**
  Randomly pick an architecture $a \in \Omega_a$
  $Evaluate(a)$
  if a is non-dominated by solutions in $archive$ **then**
    Add a to $archive$
    Remove solutions dominated by a.
  end if
end while
Randomly pick an unvisited solution a from $archive$
while $maxBudget$ not reached and not all archive solutions are visited **do**
  randomly pick $u \in N(a)$
  $Evaluate(u)$
  if u is non-dominated by solutions in $archive$ **then**
    Add u to $archive$
    Remove solutions dominated by u.
    Mark a as visited
    Randomly pick a new unvisited solution a from $archive$
  end if
  if $N(a)$ is all evaluated **then**
    Mark a as visited
    Randomly pick an unvisited solution a from $archive$
  end if
end while
return $archive$

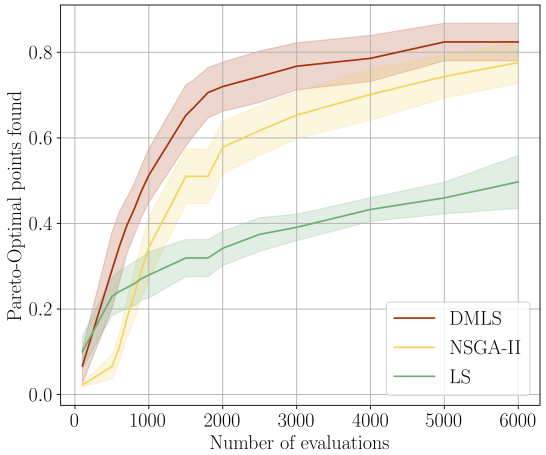


(a) Hypervolume (CIFAR10)

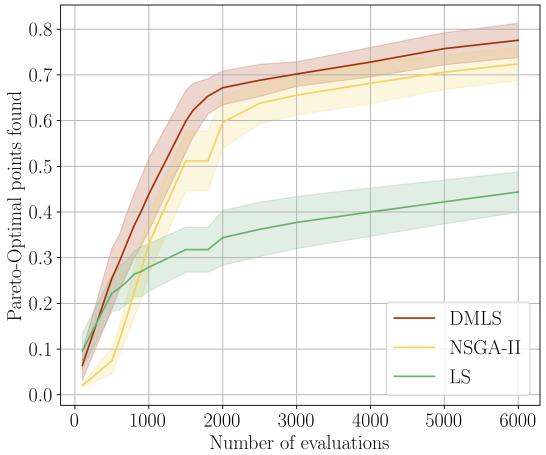


(b) Hypervolume (CIFAR100)

Fig. 5: Evolution of the hypervolume across the number of evaluated architectures for CIFAR10 and CIFAR 100. Results are averaged for 30 runs for each algorithm.



(a) % of Pareto optimal solutions found (CIFAR10)



(b) % of Pareto optimal solutions found (CIFAR100)

Fig. 6: Evolution of the percentage of Pareto optimal solutions found across the number of evaluated architectures for CIFAR10 and CIFAR 100. Results are averaged for 30 runs for each algorithm.

tournament size of 2, and population size n_{pop} = 100. As suggested in [3] for this problem.

- For the local search algorithm, a random scalarization of the objectives is used in this single-objective search algorithm. This compensates for having to choose weights for each objective a-priori. An archive is maintained to keep the non-dominated solutions from each iteration.
- For our method (DMLS), the only parameter to choose is the initial size of the archive. We run our tests for multiple archive sizes ranging from 2 to 20 and compare the results. All archive sizes seem to reach the same point after a certain number of iterations. However, bigger sizes of the archive tend to give the method a slower start (part of this is explained by the time required to find the expected number of initial solutions). We choose to work with an archive size of 7 as it finds more Pareto optimal solutions in the early stages of the search.
- Following the protocol in [3], we also add the trivial solution to the archive (or population) to each method before starting the search. This solution contains all identity cells and represents the lower bound for the second objective.

For each method, all experiments are averaged over 30 runs. We compare the algorithm from several standpoints: the hypervolume that should be maximized (we normalized the objectives, turned them into a minimization problem, and calculated the hypervolume with (1,1) as a reference point) and the percentage of Pareto optimal solutions found.

We use the Mann-Whitney U rank test for assessing the statistical significance of our results.

B. Results and Discussion

	Evaluations	DMLS	NSGA-II	LS
CIFAR10	500	**90.16±0.08**	89.84±0.14	90.10±0.05
	1000	**90.28±0.05**	90.24±0.05	90.17±0.03
	3000	**90.37±0.039**	90.33±0.04	90.24±0.03
	6000	**90.39±0.02**	90.37±0.03	90.29±0.03
CIFAR100	500	**67.69±0.10**	67.30±0.22	67.66±0.05
	1000	**67.82±0.05**	67.76±0.08	67.73±0.04
	3000	**67.94±0.03**	67.89±0.04	67.81±0.04
	6000	**67.98±0.01**	67.93±0.03	67.84±0.03

TABLE I: Table reporting the hypervolume (multiplied by 10^2 for better readability) for each method on CIFAR10 and CIFAR100. The results at presented for 500, 1000, 3000, and 6000 evaluations. Values in bold are statistically better (Mann-Whitney U rank test).

Table I reports the hypervolume of each method for CIFAR10 and CIFAR100 at different evaluation steps (500, 1000, 1500, 3000, and 6000). We notice the three approaches showing very close values at each step, with DMLS seemingly taking the lead. In the early stage of the search, at 500 evaluations, the NSGA-II seems to give a slightly lower hypervolume than LS before quickly catching in the 1000 evaluations mark. Figures 5.a and 5.b show the quick evolution of the hypervolume across the number of evaluations for CIFAR10 and CIFAR100, respectively. Here we see that in the first evaluations, LS briefly gives the highest hypervolume. This can be explained by the time it takes to initialize the archive of DMLS and the population for NSGA-II. But since NSGA-II requires a larger population, it takes it more time to surpass LS.

The very close hypervolume results suggest that the solutions composing the approximate Pareto-front are spread in an equivalent manner. For this reason, another metric should be used to better compare these methods. Here we will use the number of optimal Pareto points found for each of them.

Table II presents the percentage of Pareto optimal solutions found for each method on CIFAR10 and CIFAR100. For this metric, there is a more significant difference between each method, and this is true for both datasets as well. When the search budget is reached, DMLS finds statistically more Pareto optimal solutions. In CIFAR10, for example, DMLS ends the search with more than 82% of the Pareto optimal solutions, followed by NSGA-II with 77.6% of solutions, and lastly, LS with only 49.7% of optimal solutions. Similarly, for CIFAR100, DMLS takes the lead with 77.6% Pareto optimal solutions found at the end of the search, against 74.6% and 44.4% for NSGA-II and LS, respectively. It is clearly shown in Figure 6 as well. Here we can see that DMLS performs much better since the beginning of the search, even if it has a lower hypervolume than LS, as seen previously.

	Evaluations	DMLS	NSGA-II	LS
CIFAR10	500	**29.07±8.89**	6.52±2.79	22.90±4.45
	1000	**51.27±6.29**	34.39±7.31	27.94±5.37
	3000	**76.73±5.49**	65.31±5.49	39.07±3.08
	6000	**82.41±4.39**	77.58±4.78	49.71±6.17
CIFAR100	500	**25.49±6.67**	7.45±2.83	22.22±3.99
	1000	**43.92±7.93**	33.07±6.49	27.90±5.22
	3000	**70.19±2.69**	65.55±4.31	37.71±5.69
	6000	**77.58±3.74**	72.41±3.64	44.37±4.42

TABLE II: Table reporting the percentage of Pareto optimal solutions found for each method on CIFAR10 and CIFAR100. The results at presented for 500, 1000, 3000, and 6000 evaluations. Values in bold are statistically better (Mann-Whitney U rank test)

To sum up, results show that even if DMLS is only slightly higher in hypervolume than other methods, it can find significantly more Pareto optimal solutions compared to them. Which gives more architecture choices at the end of the search for a limited evaluation budget. On top of that, it is easy to implement and requires only one parameter to adjust (the size of the archive). This method also has the potential to exploit local properties to speed up network training (which is the most time-consuming task), such as weight inheritance [16] or network morphism [18]. This would further reduce the time needed to search for architectures in a non-benchmark setting.

V. Conclusion and Future Directions

The goal of this paper is to propose an alternative multi-objective resolution method for NAS using dominance-based multi-objective local search (solution encoding, neighborhood, and evaluation function). Unlike other multi-objective NAS strategies, it does not require defining weighted preferences between objectives like scalarization methods, nor is it complex to tune like other Pareto-front-based methods. It requires only one parameter to choose (the initial archive size), which makes it easily reusable for new tasks without much tuning. Since it is based on local search, it has the potential to exploit the local property-based evaluation strategies for NAS that will speed up the global search time. The experiments on a specialized MO benchmark show that our approach obtains statistically better Hypervolume and finds significantly more Pareto optimal solutions than NSGA-II and LS. In future works, we will compare this method using datasets for other types of tasks, such as natural language processing or image segmentation. We will also study the effect of using it in conjunction with the weight inheritance strategy to assess the speed up it can obtain.

References

[1] Han Cai, Ligeng Zhu, and Song Han. Proxylessnas: Direct neural architecture search on target task and hardware. *arXiv preprint arXiv:1812.00332*, 2018.

[2] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE transactions on evolutionary computation*, 6(2):182–197, 2002.

[3] Tom Den Ottelander, Arkadiy Dushatskiy, Marco Virgolin, and Peter AN Bosman. Local search is a remarkably strong baseline for neural architecture search. In *Evolutionary Multi-Criterion Optimization: 11th International Conference, EMO 2021, Shenzhen, China, March 28–31, 2021, Proceedings 11*, pages 465–479. Springer, 2021.

[4] Xuanyi Dong and Yi Yang. Nas-bench-201: Extending the scope of reproducible neural architecture search. *arXiv preprint arXiv:2001.00326*, 2020.

[5] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. Efficient multi-objective neural architecture search via lamarckian evolution. *arXiv preprint arXiv:1804.09081*, 2018.

[6] Jiemin Fang, Yuzhu Sun, Qian Zhang, Yuan Li, Wenyu Liu, and Xinggang Wang. Densely connected search space for more flexible neural architecture search. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10628–10637, 2020.

[7] Yesmina Jaafra, Jean Luc Laurent, Aline Deruyver, and Mohamed Saber Naceur. Reinforcement learning for neural architecture search: A review. *Image and Vision Computing*, pages 57–66, 2019.

[8] Julie Jacques, Julien Taillard, David Delerue, Clarisse Dhaenens, and Laetitia Jourdan. Conception of a dominance-based multi-objective local search in the context of classification rule mining in large and imbalanced data sets. *Applied Soft Computing*, 34:705–720, 2015.

[9] Ye-Hoon Kim, Bhargava Reddy, Sojung Yun, and Chanwon Seo. Nemo: Neuro-evolution with multiobjective optimization of deep neural network for speed and accuracy. In *ICML 2017 AutoML Workshop*, pages 1–8, 2017.

[10] Arnaud Liefooghe, Jérémie Humeau, Salma Mesmoudi, Laetitia Jourdan, and El-Ghazali Talbi. On dominance-based multiobjective local search: design, implementation and experimental analysis on scheduling and traveling salesman problems. *Journal of Heuristics*, 18(2):317–352, 2012.

[11] Hanxiao Liu, Karen Simonyan, and Yiming Yang. Darts: Differentiable architecture search. *arXiv preprint arXiv:1806.09055*, 2018.

[12] Yuqiao Liu, Yanan Sun, Bing Xue, Mengjie Zhang, Gary G Yen, and Kay Chen Tan. A survey on evolutionary neural architecture search. *IEEE Transactions on Neural Networks and Learning Systems*, 2021.

[13] Zhichao Lu, Kalyanmoy Deb, Erik Goodman, Wolfgang Banzhaf, and Vishnu Naresh Boddeti. Nsganetv2: Evolutionary multi-objective surrogate-assisted neural architecture search. In *European Conference on Computer Vision*, pages 35–51. Springer, 2020.

[14] Zhichao Lu, Ian Whalen, Vishnu Boddeti, Yashesh Dhebar, Kalyanmoy Deb, Erik Goodman, and Wolfgang Banzhaf. Nsga-net: neural architecture search using multi-objective genetic algorithm. In *Proceedings of the genetic and evolutionary computation conference*, pages 419–427, 2019.

[15] Jonas Prellberg and Oliver Kramer. Lamarckian evolution of convolutional neural networks. In *Parallel Problem Solving from Nature–PPSN XV: 15th International Conference, Coimbra, Portugal, September 8–12, 2018, Proceedings, Part II 15*, pages 424–435. Springer, 2018.

[16] Esteban Real, Sherry Moore, Andrew Selle, Saurabh Saxena, Yutaka Leon Suematsu, Jie Tan, Quoc V Le, and Alexey Kurakin. Large-scale evolution of image classifiers. In *International Conference on Machine Learning*, pages 2902–2911. PMLR, 2017.

[17] Julien Siems, Lucas Zimmer, Arber Zela, Jovita Lukasik, Margret Keuper, and Frank Hutter. Nas-bench-301 and the case for surrogate benchmarks for neural architecture search. *arXiv preprint arXiv:2008.09777*, 2020.

[18] Tao Wei, Changhu Wang, Yong Rui, and Chang Wen Chen. Network morphism. In *International Conference on Machine Learning*, pages 564–572. PMLR, 2016.

[19] Colin White, Sam Nolen, and Yash Savani. Exploring the loss landscape in neural architecture search. *arXiv:2005.02960*, 2020.

[20] Bichen Wu, Xiaoliang Dai, Peizhao Zhang, Yanghan Wang, Fei Sun, Yiming Wu, Yuandong Tian, Peter Vajda, Yangqing Jia, and Kurt Keutzer. Fbnet: Hardware-aware efficient convnet design via differentiable neural architecture search. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10734–10742, 2019.

[21] Chris Ying, Aaron Klein, Eric Christiansen, Esteban Real, Kevin Murphy, and Frank Hutter. Nas-bench-101: Towards reproducible neural architecture search. In *International Conference on Machine Learning*, pages 7105–7114. PMLR, 2019.

[22] Nan Zhang, Jianzong Wang, Jian Yang, Xiaoyang Qu, and Jing Xiao. Multi-objective cuckoo algorithm for mobile devices network architecture search. In *Artificial Neural Networks and Machine Learning–ICANN 2020: 29th International Conference on Artificial Neural Networks, Bratislava, Slovakia, September 15–18, 2020, Proceedings, Part I 29*, pages 312–324. Springer, 2020.