

A Multi-objective Optimization Benchmark Test Suite for Real-time Semantic Segmentation

Yifan Zhao

Southern University of Science and Technology
Shenzhen, Guangdong, China
AlpAcA0072@gmail.com

Zhichao Lu

City University of Hong Kong
Hong Kong, China
luzhichao.cn@gmail.com

Zhenyu Liang

Southern University of Science and Technology
Shenzhen, Guangdong, China
zhenyuliang97@gmail.com

Ran Cheng*

Southern University of Science and Technology
Shenzhen, Guangdong, China
ranchengcn@gmail.com

ABSTRACT

As one of the emerging challenges in Automated Machine Learning, the Hardware-aware Neural Architecture Search (HW-NAS) tasks can be treated as black-box multi-objective optimization problems (MOPs). An important application of HW-NAS is real-time semantic segmentation, which plays a pivotal role in autonomous driving scenarios. The HW-NAS for real-time semantic segmentation inherently needs to balance multiple optimization objectives, including model accuracy, inference speed, and hardware-specific considerations. Despite its importance, benchmarks have yet to be developed to frame such a challenging task as multi-objective optimization. To bridge the gap, we introduce a tailored streamline to transform the task of HW-NAS for real-time semantic segmentation into standard MOPs. Building upon the streamline, we present a benchmark test suite, CitySeg/MOP, comprising fifteen MOPs derived from the Cityscapes dataset. The CitySeg/MOP test suite is integrated into the EvoXBench platform to provide seamless interfaces with various programming languages (e.g., Python and MATLAB) for instant fitness evaluations. We comprehensively assessed the CitySeg/MOP test suite on various multi-objective evolutionary algorithms, showcasing its versatility and practicality. Source codes are available at <https://github.com/EMI-Group/evoxbench>.

CCS CONCEPTS

• **Theory of computation** → **Evolutionary algorithms**; • **Applied computing** → **Multi-criterion optimization and decision-making**.

KEYWORDS

Multi-objective optimization, benchmarking, real-time semantic segmentation

*Corresponding Author

Permission to make digital or hard copies of part or all of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for third-party components of this work must be honored. For all other uses, contact the owner/author(s).

GECCO '24 Companion, July 14–18, 2024, Melbourne, VIC, Australia

© 2024 Copyright held by the owner/author(s).

ACM ISBN 979-8-4007-0495-6/24/07.

<https://doi.org/10.1145/3638530.3654389>

ACM Reference Format:

Yifan Zhao, Zhenyu Liang, Zhichao Lu, and Ran Cheng. 2024. A Multi-objective Optimization Benchmark Test Suite for Real-time Semantic Segmentation. In *Genetic and Evolutionary Computation Conference (GECCO '24 Companion)*, July 14–18, 2024, Melbourne, VIC, Australia. ACM, New York, NY, USA, 8 pages. <https://doi.org/10.1145/3638530.3654389>

1 INTRODUCTION

Neural Architecture Search (NAS), a pivotal component of Automated Machine Learning [7], has traditionally focused on single-objective optimization problems, aiming to minimize prediction errors. This paradigm involves three key processes: search space design, search strategy formulation, and architecture performance evaluation.

Recent advancements have introduced NAS with multi-objective optimization, particularly in Hardware-aware Neural Architecture Search (HW-NAS). HW-NAS automates the search for deep neural network architectures, aligning them with specific applications [9], which is crucial for resource-constrained platforms. Real-time semantic segmentation emerges as a key challenge in this domain, meeting the need for high computational efficiency with accurate real-time processing.

Semantic segmentation, a cornerstone of computer vision, assigns semantic labels to each image pixel. However, the shift towards real-time applications, such as autonomous driving, has underscored the necessity of considering multiple objectives in model design, including not only hardware-unrelated metrics such as model accuracy (measured as mean Intersection over Union, *mIoU*), the scale of models but also hardware-related inference speed. This emerging demand highlights the significance of HW-NAS in devising architectures that successfully balance these diverse optimization objectives.

Due to the multiple objectives in designing and optimizing the DNN architectures, the HW-NAS tasks can be fundamentally viewed as a Multi-objective Optimization Problem (MOP). Meanwhile, since the black-box optimization characteristics of the HW-NAS, conventional optimization methods encounter significant challenges in addressing such problems.

Therefore, it is intuitive to adopt representative Multi-Objective Evolutionary Algorithms (MOEAs) [12] for HW-NAS. However, the

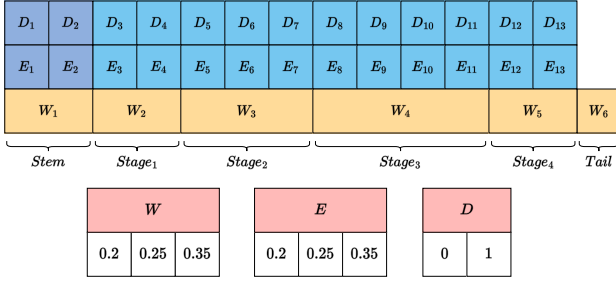


Figure 1: Architecture encoding. The search space is encoded as a 32-bit fixed-length integer-valued string.

application of MOEAs within the HW-NAS field is relatively under-explored. Existing NAS work on semantic segmentation[10] lacks capabilities of hardware awareness or constraints on real-time.

To address the challenges, this paper proposes a multi-objective optimization benchmark test suite for real-time semantic segmentation, dubbed CitySeg/MOP. The benchmark introduces an architecture search space for real-time semantic segmentation and tailors two categories of performance evaluators covering the multiple objectives. The benchmark test suite is seamlessly integrated into the platform EvoXBench [12] and provides interfaces with various programming languages (e.g., Python and MATLAB) without the requirement for any machine learning accelerating libraries. This fills the gap left by the lack of a multi-objective optimization benchmark test suite for real-time semantic segmentation in the HW-NAS.

2 METHOD

2.1 Problem Formulation

The HW-NAS problem can be formulated as a MOP, which can be defined as follows [12]:

$$\begin{aligned} \min_{\mathbf{x}} F(\mathbf{x}) &= (f^e(\mathbf{x}; \omega^*(\mathbf{x})), f^c(\mathbf{x}), f^H(\mathbf{x})) \\ \text{s.t. } \omega^*(\mathbf{x}) &= \operatorname{argmin}_{\omega} \mathcal{L}(\mathbf{x}; \omega), \quad \mathbf{x} \in \Omega_{\mathbf{x}} \end{aligned} \quad (1)$$

where

$$f^c(\mathbf{x}) = (f_1(\mathbf{x}) \quad \cdots \quad f_m(\mathbf{x})), \quad (2)$$

and

$$f^H(\mathbf{x}) = \begin{pmatrix} f_1^{h_1}(\mathbf{x}) & \cdots & f_n^{h_1}(\mathbf{x}) \\ \vdots & \ddots & \vdots \\ f_1^{h_{|\mathcal{H}|}}(\mathbf{x}) & \cdots & f_n^{h_{|\mathcal{H}|}}(\mathbf{x}) \end{pmatrix}, \quad (3)$$

$$\mathcal{H} = \{h_1, \dots, h_{|\mathcal{H}|}\},$$

where $\Omega_{\mathbf{x}}$ is the architecture search space, $\mathbf{x}$ is an encoded network architecture, $\omega(\mathbf{x})$ is the weights vector of the network architecture $\mathbf{x}$. For a dataset $\mathcal{D}$, there is $\mathcal{D} = \{\mathcal{D}_{\text{train}}, \mathcal{D}_{\text{valid}}, \mathcal{D}_{\text{test}}\}$. $\omega^*(\mathbf{x})$ is the optimal weights vector of the network architecture $\mathbf{x}$ that minimizes loss $\mathcal{L}$ on $\mathcal{D}_{\text{valid}}$. $F(\mathbf{x})$ is the objective vector, which consists of three categories: prediction error (f^e), complexity-related objectives (f^c), and hardware-related objectives (f^H), in which $\mathcal{H}$ denotes the set of hardware platforms. In $f_n^{h_{|\mathcal{H}|}}(\mathbf{x})$, the $h_{|\mathcal{H}|}$ denotes the $|\mathcal{H}|$ -th hardware platform and n denotes the n -th objective.

2.2 Search Space

The search space adopted is inspired by MoSegNAS [11]. The search space's overall structure comprises one stem layer, four stage layers, and one tail layer. The stem and stage layers contain cells that need to be searched, while the tail layer remains constant to be a global average pooling layer. Each cell comprises a sequence of convolutions, which is a typical residual bottleneck block.

In addition, the search space encodes essential attributes, including depth, expansion ratio, and width (denoting the channels). The specific form of the coded architecture is shown in Figure 1.

2.3 Fitness Evaluator

We use different fitness evaluators to cope with the diversity of the objectives. There are two categories of fitness evaluators and three categories of objectives, where the surrogate model is used for f^e , and the look-up table is used for f^c and f^H .

The fitness evaluator of inference accuracy objective (f^e) is measured in $(1 - mIoU)$. The $mIoU$ is defined as follows:

$$mIoU = \frac{1}{N} \sum_{n=1}^N IoU = \frac{1}{N} \sum_{n=1}^N \frac{TP}{TP + FP + FN}, \quad (4)$$

where TP , FP , and FN are the numbers of true positive, false positive, and false negative pixels of the segmentation result in the given image, and N is the number of the defined classes over the whole test set.

It is computationally unacceptable to train from scratch to assess the prediction accuracy. Rather than adopting the proxy methods (with fewer training epochs or gradient approximation[2]), we adopted the surrogate model to approximate the mapping from the network architectures' space to the inference accuracy objective (f^e). In this paper, we adopt a Multi-Layer Perceptron (MLP) with ranking loss and mean square error (MSE) loss as the surrogate model:

$$\begin{aligned} \mathcal{L} &= \mathcal{L}_{\text{rank}} + \mathcal{L}_{\text{mse}}, \\ \mathcal{L}_{\text{mse}} &= \frac{1}{N} \sum_i \|\hat{y}_i - y_i\|^2, \\ \mathcal{L}_{\text{rank}} &= \frac{1}{2N} \sum_{i,j} \max(0, \gamma - \delta(\hat{y}_i, \hat{y}_j)(y_i - y_j)), \end{aligned} \quad (5)$$

$$\delta(\hat{y}_i, \hat{y}_j) = \begin{cases} 1, & \text{if } \hat{y}_i > \hat{y}_j, \\ -1, & \text{otherwise,} \end{cases}$$

where the $\hat{y}_i$ is the output of the MLP when given input x_i . The γ is the margin controller hyperparameter set to 5×10^{-2} .

We have the assumption that complexity-related objectives (f^c) and the hardware-related objectives (f^H) are decomposable, i.e. the objectives can be decomposed to the sum up of the layers' corresponding objectives. Under this assumption, we can enumerate all the values of the stems and the stages individually to gain the look-up table. Thus, we can compose the architectures' stem and stages by summing up the values of f^c and f^H to gain the overall result. The procedure of the whole fitness evaluation is shown in Figure 2. The f^H can be affected by stochastic perturbation with

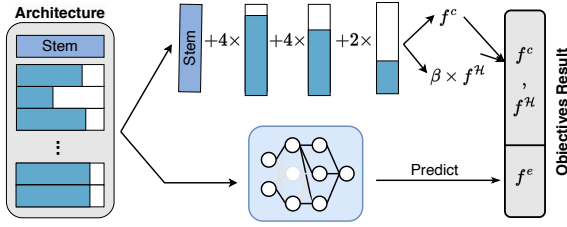


Figure 2: An example process of performance evaluation via fitness evaluator. The upper half and the bottom half indicate an example process of the look-up predictor and the surrogate model predictor respectively.

the coefficient β to simulate the hardware’s performance fluctuation. The β is set to $[0.95, 1.05]$ for f_1^H and $[0.98, 1.02]$ for f_2^H in this paper.

Unlike the stochastic gradient descent process, the approximated optimal weights of the candidate network architecture are inherited from the trained hypernetwork. Hypernetwork essentially introduces a mapping from the network architecture space $\mathbb{Z}^d$ to the weight space $\mathbb{R}^m$, i.e. $\Psi : \mathbb{Z}^d \rightarrow \mathbb{R}^m$. Ψ can be formulated as follows:

$$\Psi(\mathbf{x}) = \underset{\omega}{\operatorname{argmin}} \mathcal{L}(\mathbf{x}; \omega), \quad (6)$$

where d is the dimension of network architecture space $\mathbb{Z}$ and m is the dimension of weights space $\mathbb{R}$.

Table 1: Definition of the CitySeg/MOP test suite.

Problem	D	M	Objectives
CitySeg/MOP1	32	2	$f^e, f_1^{h_1}$
CitySeg/MOP2	32	3	$f^e, f_1^{h_1}, f_1^c$
CitySeg/MOP3	32	3	$f^e, f_1^{h_1}, f_2^c$
CitySeg/MOP4	32	4	$f^e, f_1^{h_1}, f_2^{h_1}, f_1^c$
CitySeg/MOP5	32	5	$f^e, f_1^{h_1}, f_2^{h_1}, f_1^c, f_2^c$
CitySeg/MOP6	32	2	$f^e, f_1^{h_2}$
CitySeg/MOP7	32	3	$f^e, f_1^{h_2}, f_1^c$
CitySeg/MOP8	32	3	$f^e, f_1^{h_2}, f_2^c$
CitySeg/MOP9	32	4	$f^e, f_1^{h_2}, f_2^{h_2}, f_1^c$
CitySeg/MOP10	32	5	$f^e, f_1^{h_2}, f_2^{h_2}, f_1^c, f_2^c$
CitySeg/MOP11	32	3	$f^e, f_1^{h_1}, f_1^{h_2}$
CitySeg/MOP12	32	5	$f^e, f_1^{h_1}, f_1^{h_2}, f_2^{h_1}, f_2^{h_2}$
CitySeg/MOP13	32	6	$f^e, f_1^{h_1}, f_1^{h_2}, f_2^{h_1}, f_2^{h_2}, f_1^c$
CitySeg/MOP14	32	6	$f^e, f_1^{h_1}, f_1^{h_2}, f_2^{h_1}, f_2^{h_2}, f_2^c$
CitySeg/MOP15	32	7	$f^e, f_1^{h_1}, f_1^{h_2}, f_2^{h_1}, f_2^{h_2}, f_1^c, f_2^c$

2.4 Benchmark Generation

The network architectures are trained and tested on the Cityscapes dataset [4], which is a large-scale dataset for urban city street semantic segmentation. Input images with a low expand ratio can

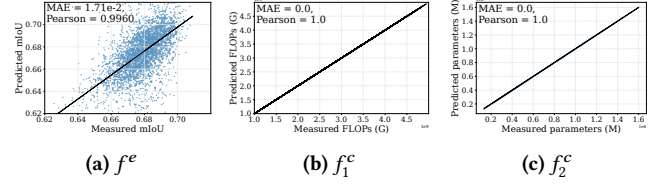


Figure 3: Correlations between the measured and the predicted metrics. The Mean Absolute Error (MAE) and Pearson correlation coefficient (ρ) are shown in the corresponding subfigures.

significantly reduce memory usage and computation resource overhead. Thus, we provide images at different expansion ratios in the search space.

Based on EvoXBench [12], we construct a tailored benchmark test suite CitySeg/MOP for real-time semantic segmentation. As summarized in Table 1, there are fifteen instances in total, D indicates the number (dimension) of decision variables, and M indicates the number of objectives in the problems. For deployment, there are two hardware devices involved: the NVIDIA GeForce RTX 4090 (h_1) and Huawei Atlas 200 DK (h_2), where the former is a representative GPU device and the latter is a typical ARM-based edge computing device with only 9.5W energy consumption. The CitySeg/MOP test suite’s problems are divided into three categories. The first five problems contain h_1 ’s objectives, and the sixth to the tenth problems contain h_2 ’s objectives. The last five problems enable the hardware-aware features by combining h_1 ’s and h_2 ’s objectives. Table 2 in Appendix A provides the definition of objectives in CitySeg/MOP test suite.

As summarized in Table 1, the test instances are listed for hardware and number of objectives in ascending order, from two to seven objectives.

3 EXPERIMENTS

This section provides the experimental results for comprehensively assessing the proposed benchmark test suite CitySeg/MOP. Firstly, we conducted experiments to assess the prediction accuracy, sample diversity, and evaluation efficiency of the predictors; afterward, we conducted benchmark tests running on the proposed test suite. The results of evaluation efficiency and benchmark tests are obtained by running each algorithm 31 times on EvoX [8] or PlatEMO [13] independently, with 10,000 fitness evaluations.

3.1 Prediction Accuracy

The correlations between the measured and the predicted metrics are shown in Figure 3. For Figure 3b and Figure 3c, we measured the number of floating point operations (FLOPs) (f_1^c) and the number of parameters (f_2^c) of the smallest unit. Network architectures’ FLOPs (f_1^c) and parameters (f_2^c) are weighed against the corresponding parameters of the smallest units. The Mean Absolute Error (MAE) of the FLOPs and the parameters is 0.0. For Figure 3a, the MAE and ρ of the prediction error (f^e) are 1.71×10^{-2} and 0.9960 respectively. Figure 3 shows the accuracy of the predictions for f^e , f_1^c and f_2^c .

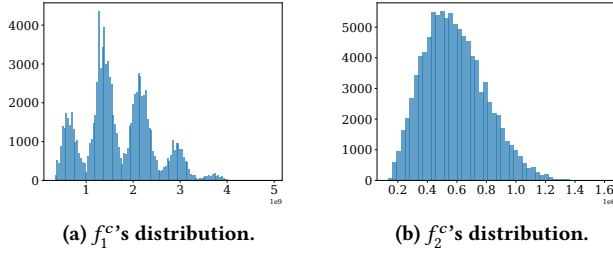


Figure 4: Distribution of f_1^c and f_2^c under randomly sampled architectures.

3.2 Sample Diversity

Figure 4 shows the randomly sampled architectures' distribution of FLOPs (f_1^c) and number of parameters (f_2^c). The FLOPs (f_1^c) has multiple peaks, which is normally distributed, and the values within each peak are also normally distributed. The short tail on the left occurs because network architectures with the depth of all stages being zero are not permitted. Conversely, the long-tailed peaks on the right represent scenarios where no stage has zero depth.

Figure 6 in Appendix B shows the randomly sampled architectures' correlation between hardware-related consumption and FLOPs, the number of parameters. We can see from Figure 6a that data is distributed in multiple clusters, resulting from the stages in the search space. This is consistent with the data distribution in Figure 4a.

3.3 Evaluation Efficiency

Table 6 in Appendix B provides the evaluation efficiency when called by Python or MATLAB's interface. The evaluation is performed on a local machine with one CPU core using PlatEMO [13] for MATLAB, and NVIDIA GeForce RTX 4090 using EvoX [8] for Python respectively.

3.4 Benchmark Tests

As a demonstration, we conducted benchmark tests on the proposed CitySeg/MOP test suite using six representative MOEAs falling into three different categories: (1) NSGA-II [5] and NSGA-III [6] (dominance-based); (2) MOEA/D [14] and RVEA¹ [3] (decomposition based); (3) IBEA [15] and HypE [1] (indicator based).

Table 7 in Appendix B summarizes the statistical results of the HV values of the test instances on the CitySeg/MOP test suite. Of these test instances, in terms of solving test instances with MOPs and MaOPs (Problems with more than three objectives), algorithms NSGA-II, NSGA-III (dominance-based methods) have an advantage. IBEA, HypE (indicator based methods) perform better on certain problems.

Figure 7 in Appendix B shows the non-dominated solutions obtained by algorithms. By performing MOEAs on CitySeg/MOP15, algorithms favor low energy-consumption, low-latency network architectures. However, these algorithms are inconsistent in the choice of prediction error objective and model scale objectives.

¹We use the version with the reference vector regeneration strategy, which is also known as the RVEA*.

4 CONCLUSION

In this paper, we introduce CitySeg/MOP, a multi-objective optimization benchmark test suite tailored for real-time semantic segmentation. By considering HW-NAS tasks real-time semantic segmentation as standard MOPs, CitySeg/MOP encompasses various objectives such as accuracy, computational efficiency, and real-time constraints. The experimental results demonstrate the efficiency, accuracy and diversity of CitySeg/MOP. Moreover, by providing seamless integration to the EvoXBench [12] platform, it enables efficient interfaces to general MOEAs implemented in various programming languages. As a standard multi-objective optimization benchmark test suite tailored for real-time semantic segmentation, CitySeg/MOP contributes to filling the gap between training EMO research and its applications in complex HW-NAS tasks.

REFERENCES

- [1] Johannes Bader and Eckart Zitzler. 2011. HypE: An Algorithm for Fast Hypervolume-Based Many-Objective Optimization. *IEEE Transactions on Evolutionary Computation* 19, 1 (2011), 45–76.
- [2] Bowen Baker, Otkrist Gupta, Ramesh Raskar, and Nikhil Naik. 2018. Accelerating Neural Architecture Search using Performance Prediction. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Workshop Track Proceedings*.
- [3] Ran Cheng, Yaochu Jin, Markus Olhofer, and Bernhard Sendhoff. 2016. A Reference Vector Guided Evolutionary Algorithm for Many-Objective Optimization. *IEEE Transactions on Evolutionary Computation* 20, 5 (2016), 773–791.
- [4] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. 2016. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*. 3213–3223.
- [5] Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and T. Meyarivan. 2002. A Fast and Elitist Multiobjective Genetic Algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation* 6, 2 (2002), 182–197.
- [6] Kalyanmoy Deb and Himanshu Jain. 2014. An Evolutionary Many-Objective Optimization Algorithm Using Reference-Point-Based Nondominated Sorting Approach, Part I: Solving Problems With Box Constraints. *IEEE Transactions on Evolutionary Computation* 18, 4 (2014), 577–601.
- [7] Thomas Elsken, Jan Hendrik Metzen, and Frank Hutter. 2019. Neural Architecture Search: A Survey. *Journal of Machine Learning Research* 20 (2019), 55:1–55:21.
- [8] Beichen Huang, Ran Cheng, Zhuozhao Li, Yaochu Jin, and Kay Chen Tan. 2024. EvoX: A Distributed GPU-accelerated Framework for Scalable Evolutionary Computation. *IEEE Transactions on Evolutionary Computation* (2024). <https://doi.org/10.1109/TEVC.2024.3388550>
- [9] Chaojian Li, Zhongzhi Yu, Yonggan Fu, Yongan Zhang, Yang Zhao, Haoran You, Qixuan Yu, Yue Wang, Cong Hao, and Yingyan Lin. 2021. HW-NAS-Bench: Hardware-Aware Neural Architecture Search Benchmark. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*.
- [10] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L. Yuille, and Li Fei-Fei. 2019. Auto-DeepLab: Hierarchical Neural Architecture Search for Semantic Image Segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2019, Long Beach, CA, USA, June 16-20, 2019*. 82–92.
- [11] Zhichao Lu, Ran Cheng, Shihua Huang, Haoming Zhang, Changxiao Qiu, and Fan Yang. 2023. Surrogate-Assisted Multiobjective Neural Architecture Search for Real-Time Semantic Segmentation. *IEEE Transactions on Artificial Intelligence* 4, 6 (2023), 1602–1615.
- [12] Zhichao Lu, Ran Cheng, Yaochu Jin, Kay Chen Tan, and Kalyanmoy Deb. 2024. Neural Architecture Search as Multiobjective Optimization Benchmarks: Problem Formulation and Performance Assessment. *IEEE Transactions on Evolutionary Computation* 28, 2 (2024), 323–337.
- [13] Ye Tian, Ran Cheng, Xingyi Zhang, and Yaochu Jin. 2017. PlatEMO: A MATLAB platform for evolutionary multi-objective optimization. *IEEE Computational Intelligence Magazine* 12, 4 (2017), 73–87.
- [14] Qingfu Zhang and Hui Li. 2007. MOEA/D: A Multiobjective Evolutionary Algorithm Based on Decomposition. *IEEE Transactions on Evolutionary Computation* 11, 6 (2007), 712–731.
- [15] Eckart Zitzler and Simon Künzli. 2004. Indicator-Based Selection in Multiobjective Search. In *Parallel Problem Solving from Nature - PPSN VIII, 8th International Conference, Birmingham, UK, September 18-22, 2004, Proceedings*. 832–842.

A EXPERIMENTAL SETUP

The experiments were conducted on EvoX [8] for Python, and PlatEMO [13] for MATLAB.

Table 2: Definition of objectives in CitySeg/MOP test suite.

Objectives	Definition
f^e	prediction error
$f_1^{h_1}$	h_1 's inference latency
$f_1^{h_2}$	h_2 's inference latency
$f_2^{h_1}$	h_1 's inference energy consumption
$f_2^{h_2}$	h_2 's inference energy consumption
f_1^c	# of floating point operations
f_2^c	# of parameters/weights

Table 3: Summary of experimental setup.

Parameter	Setting
# of runs	31
# of evaluations	10,000
D , # of decision variables	32

Table 4: Population size settings. The size of the population N is set corresponding to the number of objectives M .

M	(H_1, H_2)	N
2	(99, 0)	100
3	(13, 0)	105
4	(7, 0)	120
5	(5, 0)	126
6	(4, 1)	132
7	(3, 2)	217

Table 5: The reference point used for calculating the hypervolume of CitySeg/MOP.

Problem	Reference Point
CitySeg/MOP1	[0.0, 1.9741]
CitySeg/MOP2	[0.0, 1.9741, 3.3107e8]
CitySeg/MOP3	[0.0, 1.9741, 1.3251e5]
CitySeg/MOP4	[0.0, 1.9741, 678.0692, 3.3107e8]
CitySeg/MOP5	[0.0, 1.9741, 678.0692, 3.3107e8, 1.3251e5]
CitySeg/MOP6	[0.0, 58.7465]
CitySeg/MOP7	[0.0, 58.7465, 3.3107e8]
CitySeg/MOP8	[0.0, 58.7465, 1.3251e5]
CitySeg/MOP9	[0.0, 58.7465, 734.3339, 3.3107e8]
CitySeg/MOP10	[0.0, 58.7465, 734.3339, 3.3107e8, 1.3251e5]
CitySeg/MOP11	[0.0, 1.9741, 58.7465]
CitySeg/MOP12	[0.0, 1.9741, 58.7465, 678.0692, 734.3339]
CitySeg/MOP13	[0.0, 1.9741, 58.7465, 678.0692, 734.3339, 3.3107e8]
CitySeg/MOP14	[0.0, 1.9741, 58.7465, 678.0692, 734.3339, 1.3251e5]
CitySeg/MOP15	[0.0, 1.9741, 58.7465, 678.0692, 734.3339, 3.3107e8, 1.3251e5]

B SUPPLEMENTARY RESULTS

Table 6: Average time cost (in seconds) of evaluating 100 architectures performance with Python or MATLAB interface. Results are estimated by averaging 31 runs on NVIDIA GeForce RTX 4090 for Python, and one CPU core of a local machine for MATLAB.

Problem	Python	MATLAB
CitySeg/MOP1	1.7776 $\pm$ 0.0383	2.5191 $\pm$ 0.0049
CitySeg/MOP2	1.8190 $\pm$ 0.0253	2.5058 $\pm$ 0.0034
CitySeg/MOP3	1.8336 $\pm$ 0.0214	2.5053 $\pm$ 0.0049
CitySeg/MOP4	1.8774 $\pm$ 0.0274	2.4861 $\pm$ 0.0045
CitySeg/MOP5	1.8752 $\pm$ 0.0271	2.3957 $\pm$ 0.0072
CitySeg/MOP6	1.8445 $\pm$ 0.0230	2.3964 $\pm$ 0.0022
CitySeg/MOP7	1.8719 $\pm$ 0.0274	2.3977 $\pm$ 0.0016
CitySeg/MOP8	1.8563 $\pm$ 0.0345	2.3999 $\pm$ 0.0020
CitySeg/MOP9	1.8563 $\pm$ 0.0323	2.3949 $\pm$ 0.0019
CitySeg/MOP10	1.8473 $\pm$ 0.0226	2.3845 $\pm$ 0.0018
CitySeg/MOP11	1.8357 $\pm$ 0.0272	2.4067 $\pm$ 0.0025
CitySeg/MOP12	1.8362 $\pm$ 0.0182	2.3969 $\pm$ 0.0017
CitySeg/MOP13	1.8343 $\pm$ 0.0172	2.3903 $\pm$ 0.0015
CitySeg/MOP14	1.8340 $\pm$ 0.0247	2.3956 $\pm$ 0.0021
CitySeg/MOP15	2.0100 $\pm$ 0.0622	2.3653 $\pm$ 0.0018

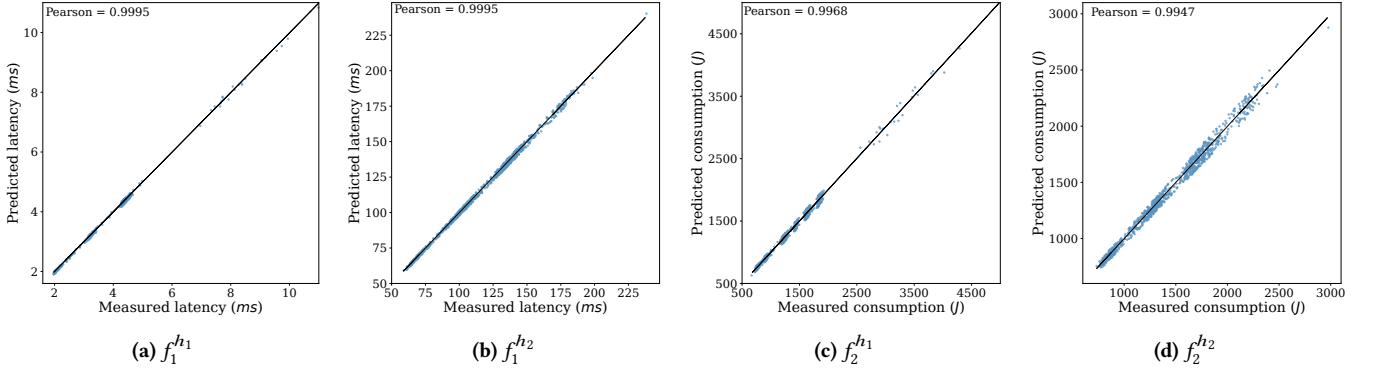


Figure 5: Correlations between the measured and the predicted metrics. The measured and the predicted metrics are obtained from measurements and the benchmark test suite correspondingly.

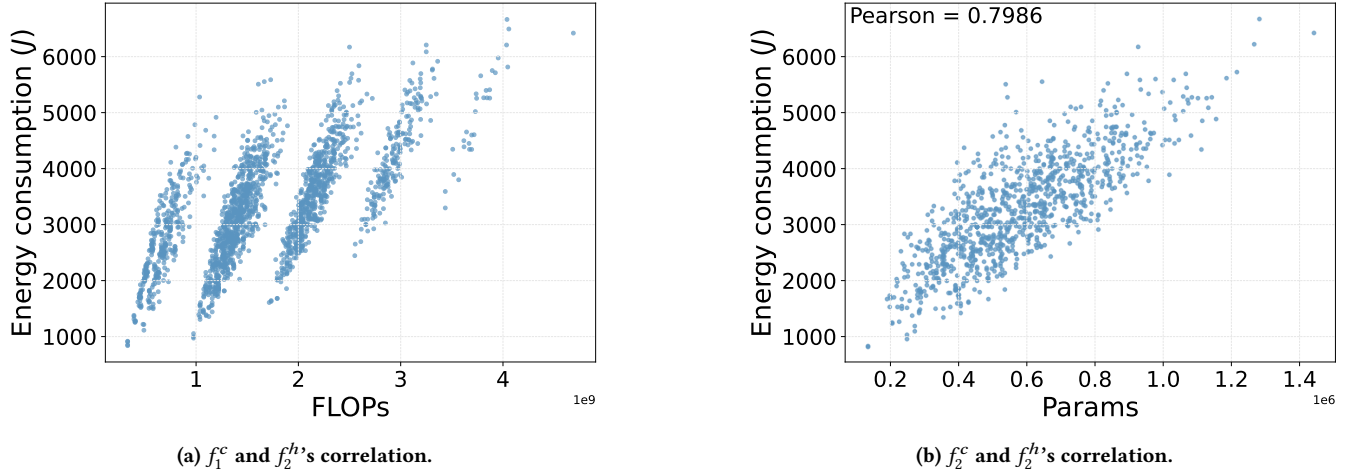


Figure 6: Correlation between f_2^h and f_1^c/f_2^c under randomly sampled architectures.

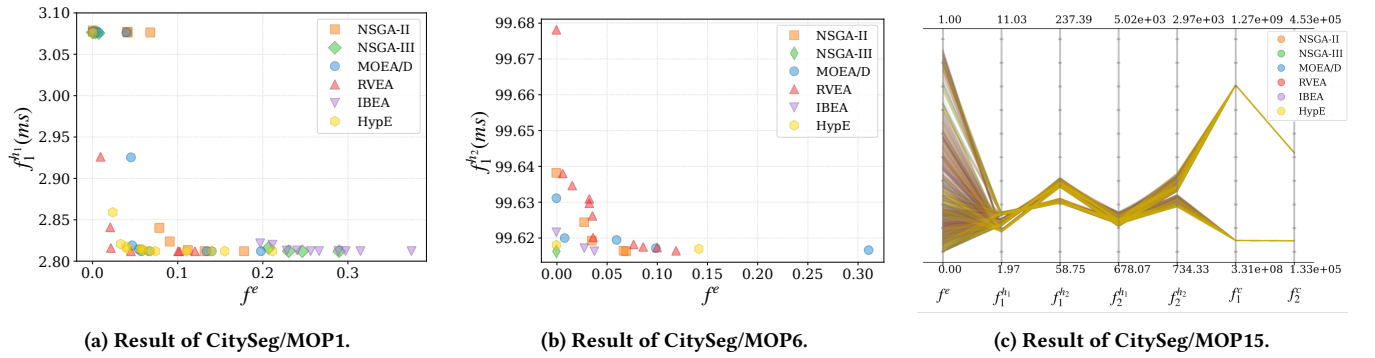


Figure 7: Non-dominated solutions obtained by each algorithm on CitySeg/MOP1, CitySeg/MOP6 and CitySeg/MOP15.

Table 7: Statistical results (mean value and standard deviation) of the hypervolume values of the test instances on CitySeg/MOP test suite. The best results obtained from each instance are highlighted in bold. The superscripts +, −, and ≈ denote the best, the worst, and identically distributed result obtained by the six algorithms. Whether the two sets of data are identically distributed or not is obtained from the Rank Sum Test.

Problem	NSGA-II	NSGA-III	MOEA/D	RVEA	IBEA	HypE
CitySeg/MOP1	0.9001(0.0045) ≈	0.8983(0.0071) ≈	0.8423(0.0505)−	0.8683(0.0229)−	0.8990(0.0043)−	0.8967(0.0111)−
CitySeg/MOP2	0.7995(0.0016) ≈	0.7991(0.0020) ≈	0.7492(0.0415)−	0.6976(0.0705)−	0.7739(0.0174)−	0.7949(0.0050)−
CitySeg/MOP3	0.8238(0.0022) ≈	0.8229(0.0036) ≈	0.7582(0.0333)−	0.7813(0.0320)−	0.7830(0.0076)−	0.8134(0.0181)−
CitySeg/MOP4	0.6979(0.0011) ≈	0.6983(0.0007) ≈	0.5474(0.0854)−	0.5951(0.0695)−	0.6006(0.0503)−	0.6979(0.0014) ≈
CitySeg/MOP5	0.6562(0.0009) ≈	0.6565(0.0006) ≈	0.5576(0.0451)−	0.5315(0.1117)−	0.5612(0.0420)−	0.6563(0.0006) ≈
CitySeg/MOP6	0.7719(0.0001) ≈	0.7718(0.0003) ≈	0.7107(0.0541)−	0.7367(0.0227)−	0.7692(0.0091) ≈	0.7706(0.0073) ≈
CitySeg/MOP7	0.7311(0.0003) ≈	0.7312(0.0002) ≈	0.6745(0.0488)−	0.6810(0.0297)−	0.7090(0.0339)−	0.7303(0.0030) ≈
CitySeg/MOP8	0.7324(0.0002) ≈	0.7324(0.0002) ≈	0.6843(0.0342)−	0.6889(0.0314)−	0.7225(0.0223)−	0.7284(0.0128)−
CitySeg/MOP9	0.5767(0.0004) ≈	0.5768(0.0003) ≈	0.4804(0.0373)−	0.5405(0.0354)−	0.4923(0.0409)−	0.5768(0.0004) ≈
CitySeg/MOP10	0.5473(0.0003) ≈	0.5473(0.0003) ≈	0.4606(0.0386)−	0.5138(0.0325)−	0.4726(0.0315)−	0.5472(0.0004) ≈
CitySeg/MOP11	0.6884(0.0010) ≈	0.6884(0.0009) ≈	0.5699(0.0351)−	0.6620(0.0359)−	0.6634(0.0213)−	0.6840(0.0097)−
CitySeg/MOP12	0.4688(0.0019)−	0.4705(0.0012) +	0.3558(0.0651)−	0.4358(0.0294)−	0.3889(0.0283)−	0.4527(0.0172)−
CitySeg/MOP13	0.4184(0.0051)−	0.4228(0.0006) +	0.3459(0.0212)−	0.3772(0.0217)−	0.3503(0.0349)−	0.4150(0.0093)−
CitySeg/MOP14	0.4299(0.0028)−	0.4339(0.0007) +	0.3406(0.0200)−	0.4118(0.0173)−	0.3569(0.0240)−	0.4214(0.0157)−
CitySeg/MOP15	0.3971(0.0018)−	0.3980(0.0025) +	0.3272(0.0265)−	0.3003(0.0718)−	0.3376(0.0192)−	0.3956(0.0045)−