HVC-Net: Deep Learning based Hypervolume Contribution Approximation

Ke Shang, Weiduo Liao, and Hisao Ishibuchi[⋆]

Guangdong Provincial Key Laboratory of Brain-inspired Intelligent Computation,
Department of Computer Science and Engineering,
Southern University of Science and Technology, Shenzhen, China
kshang@foxmail.com, {liaowd@mail.,hisao@}sustech.edu.cn

Abstract. In this paper, we propose HVC-Net, a deep learning based hypervolume contribution approximation method for evolutionary multi-objective optimization. The basic idea of HVC-Net is to use a deep neural network to approximate the hypervolume contribution of each solution in a non-dominated solution set. HVC-Net has two characteristics: (1) It is permutation equivalent to the order of solutions in the input solution set, and (2) a single HVC-Net can handle solution sets of various size (e.g., solution sets with 20, 50 and 100 solutions). The performance of HVC-Net is evaluated through computational experiments by comparing it with two commonly-used hypervolume contribution approximation methods (i.e., point-based method and line-based method). Our experimental results show that HVC-Net outperforms the other two methods in terms of both the runtime and the ability to identify the smallest (largest) hypervolume contributor in a solution set, which shows the superiority of HVC-Net for hypervolume contribution approximation.

Keywords: Hypervolume contribution · Approximation · Evolutionary multi-objective optimization · Deep learning.

1 Introduction

Hypervolume [15, 11] is a popular performance indicator in the field of evolutionary multi-objective optimization (EMO). It possesses rich theoretical properties (e.g., Pareto compliance [16], submodularity [13]), which make it attractive to use in practice. For example, it has been used to design EMO algorithms (e.g., SMS-EMOA [6, 2]) and subset selection algorithms (e.g., greedy hypervolume subset selection [7, 4]).

In SMS-EMOA, the population evolves in a steady-state manner. In each generation, one offspring is generated and added to the population, then one solution is removed from the population so that the hypervolume of the remaining population is maximized. In greedy hypervolume subset selection, in each step, one solution is selected from a candidate set and added to the subset so that the hypervolume of the subset is maximized. In these two cases, in order to remove

[⋆] Corresponding author.

(select) the correct solution, we need to calculate the hypervolume contribution of each solution. Hypervolume contribution is an important concept which describes the amount of hypervolume contributed by one solution to a solution set. In SMS-EMOA, we need to identify the solution with the smallest hypervolume contribution to the population. In greedy hypervolume subset selection, we need to identify the solution with the largest hypervolume contribution to the subset.

However, the calculation of the hypervolume contribution is #P-hard [3], which limits its applicability in many-objective optimization. In order to overcome this drawback, some hypervolume contribution approximation methods have been proposed [1, 12, 5]. Two representative methods are the point-based method [1] and the line-based method [12]. The point-based method is also known as the Monte Carlo sampling method. In this method, in order to approximate the hypervolume contribution of a solution, a sampling space is firstly determined. After that, a large number of samples are uniformly drawn in the sampling space to do the approximation. The line-based method is also known as the R2 indicator variant. In this method, a large number of line segments are uniformly drawn in the hypervolume contribution region of a solution to do the approximation.

In this paper, we propose a new hypervolume contribution approximation method. The proposed method, named HVC-Net, uses a deep neural network to do the approximation. The input of HVC-Net is a non-dominated solution set, and the output of HVC-Net is the approximated hypervolume contribution of each solution in the input solution set. HVC-Net has two characteristics. One is that it is permutation equivalent [14]. That is, a change of the order of solutions in the input solution set will cause the same change of the order of the outputs (i.e., the same results are obtained for any permutation of solutions in the input solution set). The other is that it can handle solution sets of different size (e.g., 20, 50, 100 solutions). That is, a single HVC-Net is trained using solution sets of various size. These two characteristics guarantee high applicability of HVC-Net for hypervolume contribution approximation.

The rest of this paper is organized as follows. Section 2 presents the preliminaries of the study. Section 3 introduces a new hypervolume contribution approximation method, HVC-Net. Section 4 conducts experimental studies. Section 5 concludes the paper.

2 Preliminaries

In this section, we present the preliminaries of this paper, including the definitions of hypervolume and hypervolume contribution, and two representative hypervolume contribution approximation methods.

2.1 Hypervolume and Hypervolume Contribution

Hypervolume The hypervolume is a widely used performance indicator in the field of evolutionary multi-objective optimization. Formally, for a solution set S

in the objective space, the hypervolume of S is defined as

$$HV(S,\mathbf{r}) = \mathcal{L}\left(\bigcup_{\mathbf{s}\in S}\{\mathbf{s}'|\mathbf{s}\prec\mathbf{s}'\prec\mathbf{r}\}\right), \tag{1}$$

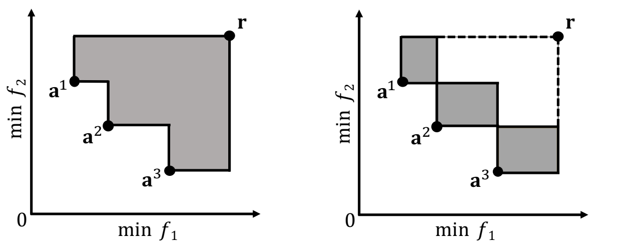
where $\mathcal{L}(\cdot)$ is the Lebesgue measure of a set, $\mathbf{r}$ is a reference point which is dominated by all solutions in S, and $\mathbf{s}\prec\mathbf{s}'$ denotes that $\mathbf{s}$ dominates $\mathbf{s}'$ (i.e., $s_i \leq s'_i$ for all $i = 1, ..., m$ and $s_j < s'_j$ for at least one $j = 1, ..., m$ in the minimization case, where m is the number of objectives).

Fig. 1 (a) gives an illustration of the hypervolume of a solution set $\{\mathbf{a}^1, \mathbf{a}^2, \mathbf{a}^3\}$ in a two-dimensional objective space, where each objective is to be minimized.

Hypervolume Contribution The hypervolume contribution is an important concept based on the hypervolume indicator. It describes the amount of the hypevolume value contributed by a solution to the solution set. Formally, for a solution $\mathbf{s}\in S$, the hypervolume contribution of $\mathbf{s}$ to S is defined as

$$HVC(\mathbf{s}, S, \mathbf{r}) = HV(S,\mathbf{r}) - HV(S\setminus\{\mathbf{s}\},\mathbf{r}). \tag{2}$$

Fig. 1 (b) gives an illustration of the hypervolume contribution of each solution to the solution set $\{\mathbf{a}^1, \mathbf{a}^2, \mathbf{a}^3\}$.



(a) Hypervolume (b) Hypervolume Contribution

Fig. 1. Illustrations of the hypervolume and the hypervolume contribution. The shaded area in (a) is the hypervolume of the solution set $\{\mathbf{a}^1, \mathbf{a}^2, \mathbf{a}^3\}$, and each shaded area in (b) is the hypervolume contribution of the corresponding solution to the solution set $\{\mathbf{a}^1, \mathbf{a}^2, \mathbf{a}^3\}$.

2.2 Hypervolume Contribution Approximation

Two representative hypervolume contribution approximation methods are the point-based method and the line-based method. These two methods are briefly explained as follows.

Point-based Method The point-based method is also known as the Monte Carlo sampling method [1]. Fig. 2 (a) illustrates this method. The basic idea is as follows. To approximate the hypervolume contribution of a solution $\mathbf{s} \in S$, a sampling space (i.e., a hyperrectangle) which contains $\mathbf{s}$'s hypervolume contribution region is firstly determined (e.g., the rectangle bounded by $\mathbf{a}^2$ and $\mathbf{p}$ in Fig. 2 (a)). Then a large number of samples are uniformly drawn in the sampling space (e.g., k samples). Suppose there are k' samples uniquely dominated by $\mathbf{s}$ (e.g., the red samples in Fig. 2 (a)), then the hypervolume contribution of $\mathbf{s}$ is approximated as

$$HVC(\mathbf{s}, S, \mathbf{r}) \approx \frac{k'}{k} V, \tag{3}$$

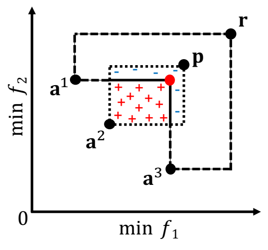
where V is the volume of the sampling space (i.e., the hyperrectangle).

In practice, the sampling space should be as tight as possible. In [1], the tightest sampling space is theoretically derived. The lower bound of the sampling space is the solution itself (e.g., $\mathbf{a}^2$ in Fig. 2 (a)). The upper bound (e.g., $\mathbf{p}$ in Fig. 2 (a)) is determined as follows:

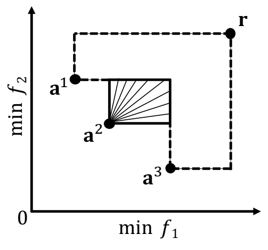
$$p_i = \min \left\{ s'_i | \mathbf{s}' \in S \setminus \{\mathbf{s}\} \text{ and } \mathbf{s}' \prec_i \mathbf{s} \right\}, i = 1, ..., m, \tag{4}$$

where $\mathbf{s}' \prec_i \mathbf{s}$ denotes that $\mathbf{s}'$ dominates $\mathbf{s}$ in all but the ith objective.

Therefore, the tightest sampling space in Fig. 2 (a) is exactly the hypervolume contribution region of $\mathbf{a}^2$ (i.e., $\mathbf{p} = (a_1^3, a_2^1)$). We did not put $\mathbf{p}$ exactly at position (a_1^3, a_2^1) (i.e., the red point) in Fig. 2 (a) for easy illustration.



(a) Point-based method



(b) Line-based method

Fig. 2. Illustrations of the point-based and line-based methods for hypervolume contribution approximation.

Line-based Method The line-based method is also known as the R2 indicator variant [12]. Fig. 2 (b) illustrates this method. The basic idea is as follows. To approximate the hypervolume contribution of a solution $\mathbf{s} \in S$, a set of line segments starting from $\mathbf{s}$ and with different directions are drawn in its hypervolume contribution region. Suppose there are n line segments and the length of each line segment is $l_i, i = 1, ..., n$, then the hypervolume contribution of $\mathbf{s}$ is approximated as

$$HVC(\mathbf{s}, S, \mathbf{r}) \approx \frac{1}{n}\sum_{i=1}^{n}(l_i)^m, \tag{5}$$

where m is the number of objectives.

The directions of the line segments can be defined using a direction vector set $\Lambda = \{\boldsymbol{\lambda}^1, ..., \boldsymbol{\lambda}^n\}$ where each direction vector satisfies $\left\|\boldsymbol{\lambda}^i\right\|_2 = 1$, $\lambda_j^i \geq 0$, $i = 1, ..., n$, $j = 1, ..., m$. The length of each line segment can be calculated as

$$l_i = \min\left\{\min_{\mathbf{s}' \in S \setminus \{\mathbf{s}\}}\left\{g^{*2\text{tch}}(\mathbf{s}'|\boldsymbol{\lambda}^i, \mathbf{s})\right\}, g^{\text{mtch}}(\mathbf{r}|\boldsymbol{\lambda}^i, \mathbf{s})\right\}, i = 1, ..., n, \tag{6}$$

where $g^{*2\text{tch}}(\cdot)$[1] and $g^{\text{mtch}}(\cdot)$ are defined as follows:

$$g^{*2\text{tch}}(\mathbf{s}'|\boldsymbol{\lambda}^i, \mathbf{s}) = \max_{j \in \{1,...,m\}}\left\{\frac{s'_j - s_j}{\lambda_j^i}\right\}, \tag{7}$$

$$g^{\text{mtch}}(\mathbf{r}|\boldsymbol{\lambda}^i, \mathbf{s}) = \min_{j \in \{1,...,m\}}\left\{\frac{|s_j - r_j|}{\lambda_j^i}\right\}. \tag{8}$$

3 HVC-Net

Motivated by DeepSets [14], which is a deep neural network for dealing with a set as its input, we design HVC-Net to approximate the hypervolume contribution of each solution in a solution set. The architecture of HVC-Net is shown in Fig. 3. The input of HVC-Net is a non-dominated solution set $S = \{\mathbf{s}^1, \mathbf{s}^2, ..., \mathbf{s}^N\}$. The output of HVC-Net is the hypervolume contribution approximation of each solution in S. The working mechanism of HVC-Net can be described in the following three steps.

- **Step 1**: Each of N solutions $\mathbf{s}^i$ $(i = 1, ..., N)$ is presented to the network ϕ and transformed to $\mathbf{a}^i = \phi(\mathbf{s}^i)$.
- **Step 2**: N vectors $\mathbf{a}^i$ are averaged as one vector $\mathbf{b} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{a}^i$. Vector $\mathbf{b}$ is further presented to network η and transformed to $\mathbf{c} = \eta(\mathbf{b})$. Vector $\mathbf{c}$ is added to each of N vectors $\mathbf{a}^i$ and N vectors $\mathbf{d}^i = \mathbf{c} + \mathbf{a}^i$ are obtained.
- **Step 3**: Each of N vectors $\mathbf{d}^i$ is presented to network ρ and N outputs $\widetilde{HVC}_{\boldsymbol{\theta}}(\mathbf{s}^i, S, \mathbf{r}) = \rho(\mathbf{d}^i)$ are obtained, where $\boldsymbol{\theta}$ is the parameter vector of HVC-Net.

It should be noted that **Step 2** can be stacked multiple times (i.e., K) as shown in Fig. 3. **Step 2** is used to learn the relation between a solution and the whole solution set. The two main characteristics of HVC-Net are as follows:

[1] The $g^{*2\text{tch}}(\cdot)$ function defined in Eq. (7) is used in minimization case. For maximization case, s'_j and s_j should be swapped in Eq. (7). Please refer to [12] for more detailed explanations.

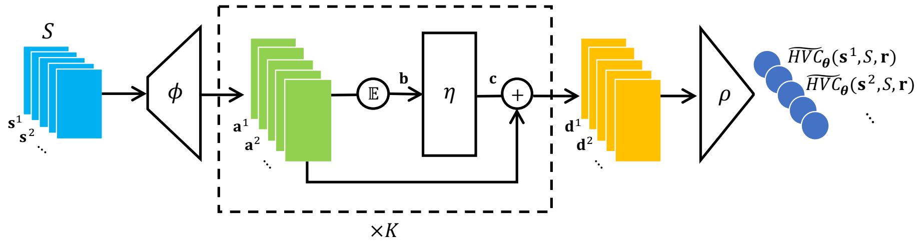


Fig. 3. The architecture of HVC-Net.

1. It is permutation equivalent. That is, for any permutation π of the input solutions (i.e., $S = \{\mathbf{s}^{\pi(1)}, ..., \mathbf{s}^{\pi(N)}\}$), the outputs of HVC-Net are $\widetilde{HVC}_{\boldsymbol{\theta}}(\mathbf{s}^{\pi(1)}, S, \mathbf{r}), ..., \widetilde{HVC}_{\boldsymbol{\theta}}(\mathbf{s}^{\pi(N)}, S, \mathbf{r})$. This means that the approximated hypervolume contribution value for each solution is not affected by the order of the input solutions. This is because the change of the order of the input solutions will lead to the change of the order of the approximated values consistently. This characteristic guarantees the robustness of HVC-Net.
2. A single HVC-Net can handle solution sets of various size. For example, we can use a trained HVC-Net to handle a solution set with 10 solutions. We can also use the same HVC-Net to handle a solution set with 100 solutions. This characteristic guarantees high applicability of HVC-Net.

3.1 How to Train HVC-Net

In HVC-Net, we implicitly assume the minimization case where the reference point for hypervolume contribution calculation is set to $\mathbf{r} = (1, ..., 1)$ and all solutions in S are located in $[0, 1]^m$. For the training of HVC-Net, we prepare the training data as follows. First, we prepare L non-dominated solution sets $\{S_1, S_2, ..., S_L\}$ where each solution set is located in $[0, 1]^m$. Then, we calculate the hypervolume contribution of each solution in each solution set based on the reference point $\mathbf{r} = (1, ..., 1)$. That is, we obtain the target output $HVC(\mathbf{s}^i, S_j, \mathbf{r})$ for each solution $\mathbf{s}^i \in S_j$ $(i = 1, ..., |S_j|, j = 1, ..., L)$.

Based on the training data (i.e., the pairs of the solution sets and the corresponding hypervolume contributions), we define the loss function of HVC-Net as follows:

$$\mathcal{L}(\boldsymbol{\theta}) = \frac{1}{L}\sum_{j=1}^{L}\frac{1}{|S_j|}\sum_{i=1}^{|S_j|}\left(\log \widetilde{HVC}_{\boldsymbol{\theta}}(\mathbf{s}^i, S_j, \mathbf{r}) - \log HVC(\mathbf{s}^i, S_j, \mathbf{r})\right)^2. \quad (9)$$

The loss function defined in Eq. (9) is similar to the mean squared error (MSE) loss function. The only difference is that we add log function to the hypervolume contribution (approximation) values. This is because the hypervolume contribution values are usually very small (e.g., in the magnitude of 10^{-4}). Using log values can make the training easier and better. More details about the training of HVC-Net are described in Section 4.1.

3.2 How to Use HVC-Net

After we train a HVC-Net, we can use it to approximate the hypervolume contribution if the solution set is in $[0,1]^m$ and the reference point is $\mathbf{r}=(1,...,1)$. The question is how to use it for hypervolume contribution approximation when the solution set and the reference point are both arbitrarily given. Before answering this question, we need the following properties.

Property 1. For any positive vector $\boldsymbol{\alpha} \in \mathbb{R}^m_{>0}$, $HVC(\mathbf{s},S,\mathbf{r}) = \frac{1}{\prod_{i=1}^m \alpha_i} HVC(\boldsymbol{\alpha} \odot \mathbf{s}, \boldsymbol{\alpha} \odot S, \boldsymbol{\alpha} \odot \mathbf{r})$, where $\odot$ denotes the element-wise multiplication[2].

Property 2. For any real vector $\boldsymbol{\beta} \in \mathbb{R}^m$, $HVC(\mathbf{s},S,\mathbf{r}) = HVC(\mathbf{s}+\boldsymbol{\beta}, S+\boldsymbol{\beta}, \mathbf{r}+\boldsymbol{\beta})$.

Property 3. $HVC(\mathbf{s},S,\mathbf{r}) = HVC(-\mathbf{s},-S,-\mathbf{r})$ where $HVC(-\mathbf{s},-S,-\mathbf{r})$ is calculated for maximization problems whereas $HVC(\mathbf{s},S,\mathbf{r})$ is calculated for minimization problems.

The above three properties can be easily obtained from the properties of the hypervolume indicator obtained in [10]. Based on these properties, we can first transform the solution set and the reference point so that the reference point is $\mathbf{r}=(1,...,1)$ and the solution set is located in $[0,1]^m$. Then we use HVC-Net to approximate the hypervolume contribution for the transformed solution set. Lastly, we calculate the hypervolume contribution approximation for the original solution set based on the output of HVC-Net. The last step is not needed when our task is to find the solution with the smallest or largest hypervolume contribution in a solution set.

Thus, although HVC-Net is trained based on solution sets in $[0,1]^m$ and reference point $\mathbf{r}=(1,...,1)$, it can be used for any solution set with any reference point.

4 Experiments

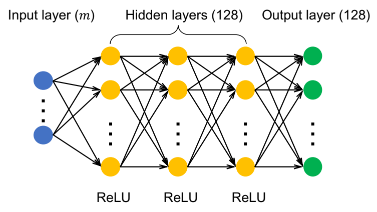
In this section, we conduct computational experiments to examine the performance of HVC-Net by comparing it with the point-based and line-based methods for hypervolume contribution approximation.

4.1 Experimental Settings

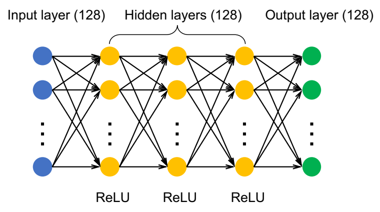
HVC-Net Specifications In HVC-Net in Fig. 3, three networks ϕ, η, and ρ need to be specified. In our experiments, all three networks are specified as feedforward neural networks. Fig. 4 shows the structures of ϕ, η, and ρ used in our experiments. All the networks have three hidden layers. In the hidden layers of the three networks, we use ReLU activation function for efficient training.

[2] For two vectors $\mathbf{a}=(a_1,...,a_m)$ and $\mathbf{b}=(b_1,...,b_m)$, $\mathbf{a}\odot\mathbf{b}=(a_1b_1,...,a_mb_m)$. For a set B, $\mathbf{a}\odot B$ means $\mathbf{a}\odot\mathbf{b}$ for all $\mathbf{b}\in B$.

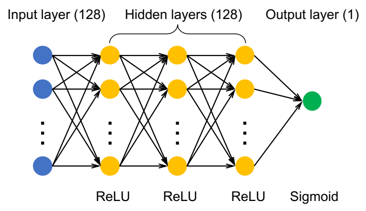
In the output layer of network ρ, we use Sigmoid activation function since the hypervolume contribution values are in $[0, 1]$ for the training solution sets. In HVC-Net, **Step 2** can be stacked K times as shown in Fig. 3. In our experiments, we set $K = 10$. That is, we have 10 different η networks in HVC-Net.



(a) Network ϕ



(b) Network η



(c) Network ρ

Fig. 4. Networks ϕ, η, and ρ in HVC-Net. The number in the parentheses indicates the number of neurons in each layer. The activation function used in each layer is shown under each layer.

Training and Testing Data Generation We examine 5, 8, and 10-objective cases (i.e., $m = 5, 8, 10$). We generate training data with L solution sets $\{S_1, ..., S_L\}$ for each case where $L = 1,000,000$. Each solution set is generated using the following procedure:

- Step 1: Randomly sample an integer $num \in [1, 100]$ where num denotes the number of solutions in the solution set.
- Step 2: Randomly sample 1000 solutions in $[0, 1]^m$ as candidate solutions.
- Step 3: Apply non-dominated sorting to these 1000 solutions and obtain different fronts $\{F_1, ..., F_l\}$ where F_1 is the first front (i.e., the set of non-dominated solutions in the 1000 solutions) and F_l is the last front.
- Step 4: Identify all the fronts F_i with $|F_i| \geq num$. If no front satisfies this condition, go back to Step 2.
- Step 5: Randomly pick one front F_i with $|F_i| \geq num$ and randomly select num solutions from this front to construct one solution set.

This procedure is used in order to select a wide variety of non-dominated solution sets for training. On average, about 10,000 solution sets with the same size are generated (1,000,000 solution sets in total for 100 different sizes).

We generate two types of testing solution sets. Type-I testing solution sets are generated using exactly the same procedure as described above. We generate 10,000 Type-I testing solution sets for each case (i.e., $m = 5, 8, 10$). These 10,000 solution sets form one group. We generate 10 different groups of Type-I testing solution sets for each case of m. Type-II testing solution sets are generated using a similar procedure. The only difference is that an integer $num \in [101, 200]$ is randomly sampled in Step 1 and 10,000 solutions are randomly sampled in Step 2. We generate 10,000 Type-II testing solution sets for each case of m. These 10,000 solution sets form one group. We generate 10 different groups of Type-II for each case of m. Type-II testing solution sets are used to test the generalization ability of HVC-Net since we train HVC-Net using solution sets with 1-100 solutions and test HVC-Net using solution sets with 101-200 solutions.

Parameter Settings For the training of HVC-Net, we use Adam [8], an effective gradient-based optimization method with an adaptive learning rate. The initial learning rate is set to 10^{-4}. For all the other parameters in Adam, we use their default settings in PyTorch [9]. The batch size during training is set to 100. The number of epochs for training is set to 100.

For the number of sampling points in the point-based method, we examine 20 different settings: 5, 10, ..., 100. For the number of lines in the line-based method, we examine 20 different settings: 5, 10, ..., 100. We use the unit normal vector method [5] to generate the direction vector set Λ in the line-based method.

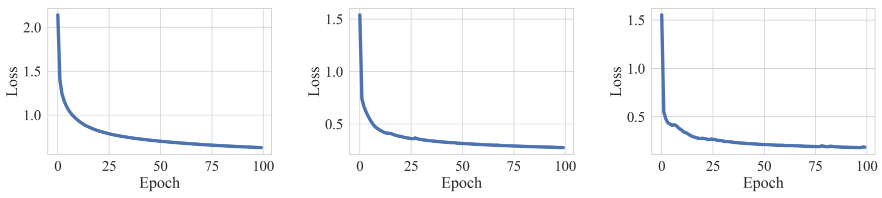
Performance Metrics To compare the performance of different hypervolume contribution approximation methods, we use the correct identification rate (CIR). CIR is a metric which can evaluate the ability of a method to identify the smallest (largest) hypervolume contributor in a solution set. For example, suppose we have P solution sets. If a method can correctly identify the smallest (largest) hypervolume contributor for Q out of P solution sets, then CIR is calculated as Q/P. In our experiments, we use CIR_{min} (i.e., CIR for identifying the smallest hypervolume contributor) and CIR_{max} (i.e., CIR for identifying the largest hypervolume contributor). A larger CIR value means better approximation quality of a method.

We also record the runtime of the three methods to compare their efficiency. Here the runtime of HVC-Net means its evaluation time on the testing solution sets, not the training time.

Platforms All the methods are coded in Python and tested on a server with Intel(R) Xeon(R) Gold 6130 CPU @ 2.10GHz, GeForce RTX 2080 GPU, and Ubuntu 18.04.6 LTS. HVC-Net is implemented based on PyTorch version 1.9.0.

4.2 The Training of HVC-Net

Fig. 5 shows the training curve of HVC-Net in each case of m. We can see that the loss decreases sharply in the first 20 epochs. This is mainly because we use a batch size of 100 for training 1M solutions, which means that the parameters of HVC-Net can be updated 10K times in each epoch. We can also observe that the loss becomes very small at the end of the training process in each figure, which shows the success of the training of HVC-Net.



(a) 5-objective (b) 8-objective (c) 10-objective

Fig. 5. The training curve of HVC-Net in each case.

Table 1 shows the time used for training HVC-Net. Although the training of HVC-Net needs quite a substantial time, the trained HVC-Net models can be saved and are ready to use at any time. That is, once we obtain a well trained HVC-Net model, we can save it and use it directly in the future without spending a lot of time to retrain it.

Table 1. The time (GPU hours) used for training HVC-Net in each case.

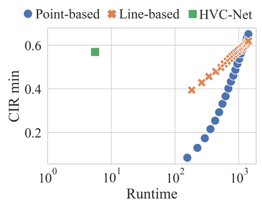
5-objective	8-objective	10-objective
369.13	370.47	383.79

Next, we will examine the performance of our trained HVC-Net models on the testing solution sets.

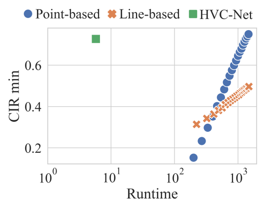
4.3 Testing on Type-I Solution Sets

We apply the three hypervolume contribution approximation methods to Type-I testing solution sets, i.e., solution sets with 1-100 solutions. For fair comparison, all the methods are tested on CPU. That is, we disable GPU when using HVC-Net for evaluation.

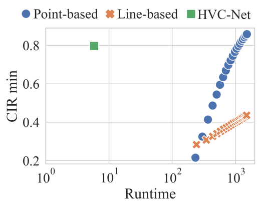
Fig. 6 shows the results of CIR_{min} and CIR_{max} for each case. We can see that HVC-Net clearly dominates the other two methods in most cases in terms of both the correct identification rate and the runtime, which shows the advantage of using HVC-Net for identifying the smallest (largest) hypervolume contributor in a solution set. It is worth noting that the point-based and line-based methods are very time-consuming compared with HVC-Net. HVC-Net is able to process the testing solution sets in less than 10 seconds and achieves a good CIR value. The



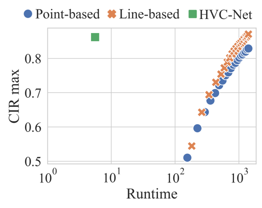
(a) 5-objective



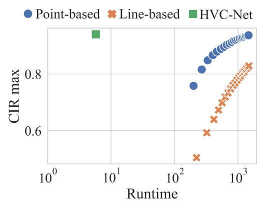
(b) 8-objective



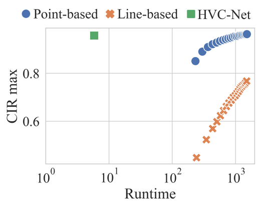
(c) 10-objective



(d) 5-objective



(e) 8-objective



(f) 10-objective

Fig. 6. Experimental results of the three hypervolume contribution approximation methods on Type-I testing solution sets in each case. The runtime (in seconds) means the total time for evaluating 10,000 testing solution sets. The CIR (CIR_{min} in (a)-(c) and CIR_{max} in (d)-(f)) means the correct identification rate over 10,000 testing solution sets. All the results are the average over 10 groups of Type-I testing solution sets.

other two methods even perform worse (i.e., a smaller CIR value) than HVC-Net by consuming 1000 seconds. Of course, the other two methods can achieve better CIR values than HVC-Net using more points or lines in some cases (e.g., in Fig. 6 (a) the other two methods achieve better CIR values than HVC-Net by consuming more than 1000 seconds). However, the runtime cost is too high to realize this goal for the point-based and line-based methods.

4.4 Testing on Type-II Solution Sets

We also apply the three hypervolume contribution approximation methods to Type-II testing solution sets, i.e., solution sets with 101-200 solutions. We use Type-II testing solution sets to examine the generalization ability of HVC-Net.

Fig. 7 shows the results of CIR_{min} and CIR_{max} for each case. We can observe that HVC-Net performs well in general. It can still dominate the other two methods in terms of the correct identification rate and the runtime in most cases. There is almost no runtime increase for HVC-Net compared with the results on Type-I solution sets. However, there is a significant runtime increase for the point-based and line-based methods. These results show the strong generalization ability and the efficiency of HVC-Net. Although HVC-Net is trained using solution sets with 1-100 solutions, it is able to handle solution sets with 101-200 solutions effectively and efficiently.

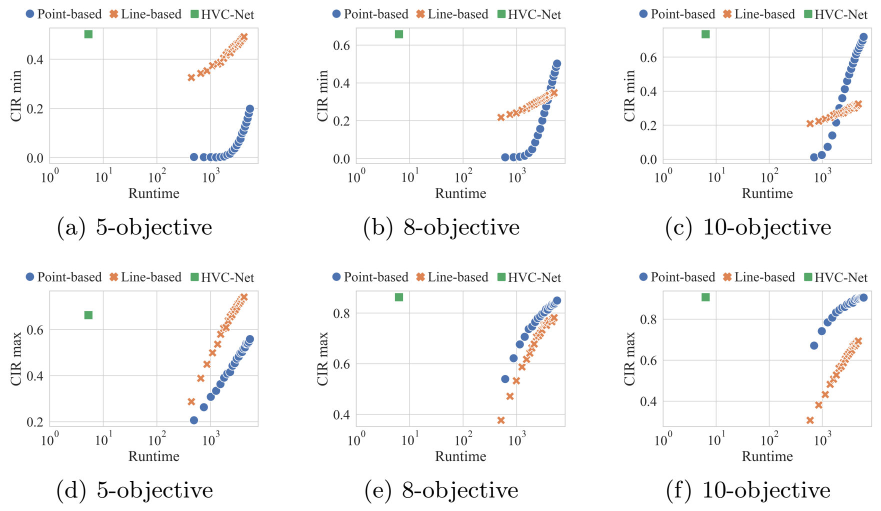


Fig. 7. Experimental results of the three hypervolume contribution approximation methods on Type-II testing solution sets in each case. The runtime (in seconds) means the total time for evaluating 10,000 testing solution sets. The CIR ($\mathrm{CIR}_{\min}$ in (a)-(c) and $\mathrm{CIR}_{\max}$ in (d)-(f)) means the correct identification rate over 10,000 testing solution sets. All the results are the average over 10 groups of Type-II testing solution sets.

5 Conclusions

In this paper, we proposed HVC-Net, a deep learning based method for hypervolume contribution approximation. We compared HVC-Net with the point-based method and the line-based method. The experimental results showed that HVC-Net outperformed the other two methods in terms of both the correct identification rate and the runtime, which showed the potential of using deep learning technique for hypervolume contribution approximation.

Our future work is the development of a hypervolume-based EMO algorithm and a hypervolume subset selection algorithm based on HVC-Net for many-objective optimization. Of course, we will also try to further improve the performance of HVC-Net by improving its structure, parameter settings, training method, and so on.

All the source codes and the trained HVC-Net models are available at `https://github.com/HisaoLabSUSTC/HVC-Net`.

Acknowledgements This work was supported by National Natural Science Foundation of China (Grant No. 62002152, 61876075), Guangdong Provincial Key Laboratory (Grant No. 2020B121201001), the Program for Guangdong Introducing Innovative and Enterpreneurial Teams (Grant No. 2017ZT07X386), The Stable Support Plan Program of Shenzhen Natural Science Fund (Grant No. 20200925174447003), Shenzhen Science and Technology Program (Grant No. KQTD2016112514355531).

References

1. Bader, J., Deb, K., Zitzler, E.: Faster hypervolume-based search using Monte Carlo sampling. In: Multiple Criteria Decision Making for Sustainable Energy and Transportation Systems, pp. 313–326. Springer (2010)
2. Beume, N., Naujoks, B., Emmerich, M.: SMS-EMOA: Multiobjective selection based on dominated hypervolume. European Journal of Operational Research **181**(3), 1653–1669 (2007)
3. Bringmann, K., Friedrich, T.: Approximating the least hypervolume contributor: NP-hard in general, but fast in practice. In: International Conference on Evolutionary Multi-Criterion Optimization. pp. 6–20. Springer (2009)
4. Chen, W., Ishibuchi, H., Shang, K.: Fast greedy subset selection from large candidate solution sets in evolutionary multi-objective optimization. IEEE Transactions on Evolutionary Computation (Early Access)
5. Deng, J., Zhang, Q.: Approximating hypervolume and hypervolume contributions using polar coordinate. IEEE Transactions on Evolutionary Computation **23**(5), 913–918 (2019)
6. Emmerich, M., Beume, N., Naujoks, B.: An EMO algorithm using the hypervolume measure as selection criterion. In: International Conference on Evolutionary Multi-Criterion Optimization. pp. 62–76 (2005)
7. Guerreiro, A.P., Fonseca, C.M., Paquete, L.: Greedy hypervolume subset selection in low dimensions. Evolutionary Computation **24**(3), 521–544 (2016)
8. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
9. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al.: Pytorch: An imperative style, high-performance deep learning library. Advances in Neural Information Processing Systems **32**, 8026–8037 (2019)
10. Shang, K., Chen, W., Liao, W., Ishibuchi, H.: HV-Net: Hypervolume approximation based on deepsets. IEEE Transactions on Evolutionary Computation (Early Access)
11. Shang, K., Ishibuchi, H., He, L., Pang, L.M.: A survey on the hypervolume indicator in evolutionary multiobjective optimization. IEEE Transactions on Evolutionary Computation **25**(1), 1–20 (2021)
12. Shang, K., Ishibuchi, H., Ni, X.: R2-based hypervolume contribution approximation. IEEE Transactions on Evolutionary Computation **24**(1), 185–192 (2020)
13. Ulrich, T., Thiele, L.: Bounding the effectiveness of hypervolume-based $(\mu + \lambda)$-archiving algorithms. In: International Conference on Learning and Intelligent Optimization. pp. 235–249. Springer (2012)
14. Zaheer, M., Kottur, S., Ravanbhakhsh, S., Póczos, B., Salakhutdinov, R., Smola, A.J.: Deep Sets. Advances in Neural Information Processing Systems pp. 3394–3404 (2017)
15. Zitzler, E., Thiele, L., Laumanns, M., Fonseca, C., da Fonseca, V.: Performance assessment of multiobjective optimizers: an analysis and review. IEEE Transactions on Evolutionary Computation **7**(2), 117–132 (2003)
16. Zitzler, E., Brockhoff, D., Thiele, L.: The hypervolume indicator revisited: On the design of Pareto-compliant indicators via weighted integration. In: International Conference on Evolutionary Multi-Criterion Optimization. pp. 862–876. Springer (2007)