

Approximating Hypervolume Contributions using Grammatical Evolution

Amín V. Bernabé Rodríguez¹ and Carlos A. Coello Coello²

Abstract—The hypervolume (HV) indicator is widely used in multi-objective evolutionary algorithms (MOEAs) due to its Pareto compliance property. This property makes it very effective for assessing the quality of approximation sets from different MOEAs and for ranking solutions among a population of solutions using the individual hypervolume contribution (HVC). However, the computational cost of computing the HV increases exponentially with the number of objectives. Furthermore, this cost increase is worse when the HVC is adopted as a density estimator, making it prohibitive in many-objective optimization problems (MaOPs). In this work, we propose a novel approach to create HVC approximation functions using Grammatical Evolution (GE). We describe the grammar and fitness functions designed to identify the worst-contributing individual given a population of non-dominated solutions. Then, we use a GE implementation with training data generated from the DTLZ and WFG test problems. The resulting approximation functions, tailored for dimensionalities ranging from 2 to 10, are evaluated against two state-of-the-art methods: HVC-Net and the R2-based HVC approximation. Experimental results on validation data also derived from benchmark problems show that our GE-generated functions consistently outperform both alternative approaches regarding worst-contributing individual identification for dimensions greater than two, while maintaining competitive execution times. These results indicate that GE is a viable and effective tool for generating high-quality HVC approximations, particularly suitable for solving MaOPs.

I. INTRODUCTION

Multi-objective optimization problems (MOPs) arise in several domains such as engineering, economics, logistics, artificial intelligence, and healthcare [1]. The solution of an MOP is a set of the so-called Pareto-optimal solutions, rather than a single global optimum. Pareto-optimal solutions represent the best possible trade-offs among the objectives (i.e., no single objective can be improved without degrading at least another one). One of the most popular approaches to solve MOPs is the use of multi-objective evolutionary algorithms (MOEAs) since evolutionary algorithms have features that work inherently well with these problems.

Currently, there are three main types of MOEAs: Pareto-based, decomposition-based and indicator-based. In this paper, we will focus on indicator-based MOEAs, which rank solutions using a predefined performance indicator [2]. Then, using this ranking, the selection mechanism determines

which solutions should be preserved in the next generation [3]. Consequently, the performance of indicator-based MOEAs is highly dependent on the indicator adopted.

One of the most popular performance indicators used in evolutionary algorithms is the hypervolume (HV), since it has the main advantage of being Pareto compliant (i.e., it strictly preserves the relation established by Pareto dominance). However, it has one crucial drawback: its execution time increases exponentially with the number of objectives. Hence, the hypervolume allows for an easy and effective comparison between two approximation sets, requiring only two hypervolume computations per comparison. Consequently, it is widely used to assess the performance of different MOEAs. However, when it is used in indicator-based MOEAs, the number of hypervolume computations significantly increases since it is required to obtain the contribution of each individual to the population's hypervolume. This process is repeated iteratively until the MOEA ends its execution. Then, hypervolume-based MOEAs typically become prohibitively expensive in MOPs with more than five objective functions.

To deal with the high computational cost of the hypervolume, there have been two main strategies. The first one is to adopt a different performance indicator. However, the best alternatives currently available are weakly Pareto compliant, causing them to select, from time to time, solutions that are not Pareto-optimal. The second strategy is to approximate the hypervolume values with one of the different approximation methods that have been proposed [4], [5]. These approximations are less time-consuming at the unavoidable expense of reducing their accuracy. This work focuses on this second strategy and presents the methodology we followed to create new hypervolume contribution (HVC) approximation functions using grammatical evolution (GE). We will show that our proposed approach outperforms two state-of-the-art methods (HVC_{NET} and HVC_{R2}) to approximate the hypervolume in most of the validation scenarios adopted in our study, which considered MOPs having two or more objectives. Compared to HVC_{NET} , our approach improves the quality of the HVC approximation while maintaining competitive execution times. Moreover, compared to HVC_{R2} , our approach improves both its quality and execution times in 6 out of 7 benchmark instances. Finally, a comparison using real-world MOPs shows that our approach is also competitive in such cases.

The remainder of this paper is organized as follows. Section II introduces some fundamental definitions and related concepts. Section III presents the methodology followed

*This research is supported by the Basque Government through the BERC 2022-2025 program and by the Ministry of Science and Innovation: BCAM Severo Ochoa accreditation CEX2021-001142-S / MICIN / AEI / 10.13039/501100011033.

¹ Amín V. Bernabé Rodríguez is with the Basque Center for Applied Mathematics, Bilbao, Biscay, Spain. abernabe@bcamath.org

² Carlos A. Coello Coello is with the Centro de Investigación y de Estudios Avanzados-IPN, Mexico City, Mexico. carlos.coellocoello@cinvestav.mx

to generate the hypervolume contribution approximations. Then, Section IV describes our experimental setup. Our results are evaluated and compared with respect to two other hypervolume contribution approximations in Section V. Finally, Section VI provides our conclusions and some paths for future research.

II. BACKGROUND

In order to determine the notion of optimality in MOPs, the concept of Pareto optimality is often adopted. Given a set of solutions $X = \{\vec{x}_1, \dots, \vec{x}_n\}$, solution $\vec{x}^* \in X$ is called Pareto optimal, or non-dominated, if there is no other $\vec{x} \in X$ such that: $f_i(\vec{x}) \leq f_i(\vec{x}^*) \forall i \in \{1, \dots, d\}$, with at least one strict inequality $f_j(\vec{x}) < f_j(\vec{x}^*)$.

Given a set of Pareto-optimal vectors $X^* = \{\vec{x}_1^*, \dots, \vec{x}_n^*\}$, with $\vec{x}_i^* \in \mathbb{R}^d$ and a reference point $\vec{r} \in \mathbb{R}^d$ which is dominated by every vector in X^d . The hypervolume of this set is defined as:

$$HV(X^*, \vec{r}) = \lambda \left(\bigcup_{\vec{x}^* \in X^*} V(\vec{x}^*, \vec{r}) \right), \quad (1)$$

where $V(\vec{x}^*, \vec{r})$ is the hyper-region formed between each vector \vec{x}^* and the reference point \vec{r} , and $\lambda(\cdot)$ is the Lebesgue measure in d -dimensional space. When comparing two sets, the hypervolume of both sets must be computed using the same reference point \vec{r} . The larger the hypervolume value, the better the quality of the approximation set [6]. The hypervolume contribution of a single solution \vec{x}_i^* is given by:

$$HVC(\vec{x}_i^*) = HV(X^*) - HV(X^* \setminus \{\vec{x}_i^*\}), \quad (2)$$

where $HV(X^* \setminus \{\vec{x}_i^*\})$ is the hypervolume of the set X^* after removing solution \vec{x}_i^* . This value represents the hypervolume lost if the solution \vec{x}_i^* was removed. Hence, a higher HVC means that the corresponding solution contributes more to the population's global HV.

Indicator-based MOEAs employ the HVC of each individual in the population to rank them. Consequently, the population's global HV and one additional hypervolume value per individual must be obtained, significantly increasing the computational cost of the MOEA. Some HVC approximations have been proposed to reduce this cost, two of which are the R2-based HVC approximation [7] and the HVC-Net [8]. The former is an approximation of the HVC formulated as an R2 indicator, and it is based on the lengths of line segments between a reference point and the attainment surface of the non-dominated set. The latter is a deep learning-based approximation method. These approximations can handle solution sets of different sizes and reduce the execution time required to obtain HVC compared to the actual value. However, it is important to remember that approximations of the HVC will always present an error percentage compared to the actual HVC.

Grammatical Evolution (GE) is an evolutionary computation technique that allows the automatic generation of computer programs or functions to solve a given problem. Similarly to other evolutionary techniques, GE operates on a

population of individuals, which are iteratively altered via genetic operators until some termination criterion is met. However, one of the most distinctive features of GE is that its individuals encode executable code in binary lists [9], which provides versatility and ease in applying it to different domains. An implementation of GE requires two elements: (1) a grammar to define the syntax of potential solutions and (2) a fitness function to evaluate such solutions[10].

III. PROPOSAL

Here, we propose generating new HVC approximation functions using GE. To achieve this, we first generated training and validation files. Each of these files contains a set of non-dominated points, the actual HVC value of each point, and their ranked position, considering all contributions in the same file. Then, using the training sets, as well as the grammar and fitness function described next, different HVC approximations were obtained.

A. Grammar

For each data file X with a set of n coordinated points in m -dimensional space, $X = \{\vec{x}_1, \vec{x}_2, \dots, \vec{x}_n\}$, three variables were obtained: the cumulative sum of each component of all points $S = \{\sum_{i=0}^n \vec{x}_{i,1}, \sum_{i=0}^n \vec{x}_{i,2}, \dots, \sum_{i=0}^n \vec{x}_{i,n}\}$, the maximum component value $X_\Omega = \max_{i,j=1}^n x_{i,j}$ and the minimum component value $X_\omega = \min_{i,j=1}^n x_{i,j}$.

These variables were used to generate the individuals in GE using the following grammar:

$$\begin{aligned} \langle e \rangle &::= \langle e \rangle + \langle e \rangle \mid \langle e \rangle - \langle e \rangle \mid \langle e \rangle * \langle e \rangle \mid \langle e \rangle / \langle e \rangle \\ &\mid \text{sqrt}(\langle e \rangle) \mid \sin(\langle e \rangle) \mid \cos(\langle e \rangle) \\ &\mid \exp(\langle e \rangle) \mid \log(\langle e \rangle) \mid \vec{x}_{i,\langle d \rangle} \\ &\mid X_\omega \mid X_\Omega \mid S_{\langle d \rangle} \mid n \\ &\mid \langle c \rangle \langle c \rangle . \langle c \rangle \langle c \rangle \end{aligned}$$

$$\langle c \rangle ::= 0 \mid 1 \mid 2 \mid 3 \mid 4 \mid 5 \mid 6 \mid 7 \mid 8 \mid 9$$

$$\langle d \rangle ::= 1 \mid 2 \mid \dots \mid m$$

The previous grammar was selected experimentally after some trial executions, considering only arithmetic operators, as well as combined with trigonometric operators and exponential, logarithmic, and square root functions. We observed that the quality of the approximation functions was better with the combination of all these functions.

B. Fitness function

Once GE generates a candidate HVC approximation function using the previously described grammar, we evaluated it to assess how well the new function approximated the actual HVC values. We focused specifically on identifying the worst-contributing individual in the population rather than approximating the real HVC of every individual. We decided to use this approach since this is a mechanism commonly adopted by indicator-based MOEAs.

Given an HVC approximation function f , its fitness was obtained by averaging how well it identified the worst-contributing individual in each file X of the training set. Using function f , an approximated value of HVC was obtained for each point in the file. Afterwards, these values

were used to rank each individual, and the worst-performing one was identified. Then, we used the following expression to quantify the approximation error:

$$fitness(f, X) = \frac{|r_{HVC}(f, X, \vec{x}_w) - r_{HVC}(f, X, \vec{x}_{aw})| \cdot 100}{n} \quad (3)$$

where $r_{HVC}(f, X, \vec{x})$ is the ranking of the point \vec{x} considering the actual HVC values, \vec{x}_w is the point with the worst real HVC and \vec{x}_{aw} is the point with the worst approximated HVC. It is essential to note that two or more individuals can have the same HVC value, which should be reflected in the ranking to prevent favoring any particular solution. Hence, we assign a rank to each individual that corresponds to its position in the sorted HVCs list. However, if two or more individuals have the same HVC, all of them share the same rank, which results from averaging the ranks they would have if they were considered to have different HVC values. Finally, (3) is used to obtain the fitness of the given approximation function for each file in the training set, and these values are averaged to assign a final fitness to it.

C. Training and validation sets

The data used to train and validate the HVC approximation functions was generated using independent MOEA executions solving MOPs from the test suites DTLZ [11] and WFG [12] with 2, 3, 4, 5, 7, 8 and 10 objectives. The MOEA adopted to solve these problems was NSGA-III, as it can obtain well-distributed Pareto fronts in these problems [13].

The first step was to determine an appropriate number of function evaluations for each instance, as some converge much faster than others. This means that the maximum number of function evaluations for each problem should be different, since using the same maximum number would cause some of the easier instances to have well-converged populations (very close to the 100% of the real Pareto front) for most of the populations stored from said instances. Hence, using NSGA-III, we performed eight independent executions, considering 100,000 function evaluations for problems with 2-8 objectives and 150,000 function evaluations for 10-objective problems. Then, the average hypervolume values were obtained every 5,000 function evaluations to gather information on the levels of convergence at different points of the execution. Using this information, we identified the point at which 99% of the final hypervolume was achieved for each problem. Then, we selected the closest 5k interval as the stopping point for the next step. Afterwards, twenty executions were performed for every MOP instance, using the custom maximum number of function evaluations previously found. During these executions, the MOEA's population was stored after a certain number of generations in such a way that between 35 and 40 populations were stored per execution. Each resulting population was filtered to store only the non-dominated points, which were then normalized to the interval $[0, 1]$ per population. This normalization removes the requirement of a reference point in the approximation. Then, the actual HVC of each point

and the corresponding ranking were obtained. These values were stored in each file, along with the coordinates of each non-dominated, normalized point. Finally, the files were randomly assigned to the training or validation set with a probability of 50% each. In Table I, we present the maximum population size before the filtering of dominated solutions, as well as the size of the resulting training and validation sets for each dimension considered.

TABLE I
TRAINING AND VALIDATION SET SIZES.

d	Maximum population size	Training set size	Validation set size
2	100	5662	5662
3	106	5895	5895
4	120	5805	5805
5	212	5778	5778
7	210	5751	5751
8	120	5751	5751
10	220	5535	5535

IV. APPROXIMATION FUNCTIONS GENERATION

A. Experimental setup

The GE implementation adopted to generate the HVC approximation functions is `PonyGE2`, a GE implementation coded in Python [14], which handles the creation of individuals based on a given grammar and the reproduction of these individuals based on a pre-defined fitness function. These two elements were implemented as described in the previous section.

Using this implementation¹, we ran one execution using each of the seven training sets generated with different dimensions (2-5, 7, 8, and 10). The population size was set at 100 individuals in all cases, and two termination criteria were initially adopted: a maximum of 2,000 generations and a maximum execution time of 10 days, whichever occurred first. Each execution used four cores of an Intel Xeon Gold 6248R processor and 40 GB of RAM.

B. Approximation functions

In practice, all executions reached the 10-day timeout before the maximum number of generations, exhibiting variation in the maximum generation reached since the computation time of each approximation function generated throughout the evolutionary search differs. The best individual for each training set is shown in (7) to (13).

$$HVC_{GE2}(\vec{x}_i, X) = e^{X_\Omega} - X_\omega \cdot \frac{\frac{S_2}{e^{X_\omega \cdot S_2}}}{e^{X_\omega \cdot S_2} - \cos \sqrt{e^{\frac{X_\Omega}{S_2} - \frac{X_\omega}{e^{\sqrt{X_\Omega}}}} + \vec{x}_{i,1} + X_\omega \cdot X_\omega - X_\omega + e^{\sin X_\omega}} \cdot S_2} \quad (4)$$

¹The source code of our implementation is available at https://github.com/amin-vanya/PonyGE2_EMO

$$\begin{aligned}
HVC_{GE3}(\vec{x}_i, X) = & \vec{x}_{i,2} \cdot S_2 - \cos \sin \log X_\omega + 95.91 \\
& - n + S_2 + \vec{x}_{i,3} + \log X_\omega \\
& + X_\Omega \cdot e^{X_\Omega \cdot e^{\vec{x}_{i,1}}} \cdot e^{\vec{x}_{i,1}} + \vec{x}_{i,2} \cdot S_3 \\
& - n - \frac{54.56}{S_2} + n - S_3 + \log \vec{x}_{i,3}
\end{aligned} \quad (5)$$

$$\begin{aligned}
HVC_{GE4}(\vec{x}_i, X) = & \sqrt{n} \cdot \log X_\Omega + \sin \vec{x}_{i,4} + n \\
& + e^{\sin X_\Omega} + \vec{x}_{i,2} + \vec{x}_{i,1} + n + \vec{x}_{i,3}
\end{aligned} \quad (6)$$

$$\begin{aligned}
HVC_{GE5}(\vec{x}_i, X) = & X_\Omega \cdot X_\Omega - X_\omega + \cos X_\Omega \\
& + 81.70 + e^{\cos 0.95 - \vec{x}_{i,4}} + X_\Omega + X_\omega \\
& + \frac{X_\Omega}{\cos X_\Omega} + \sin \vec{x}_{i,3} + \vec{x}_{i,1} \cdot \frac{\cos n}{\vec{x}_{i,1}} - \cos \vec{x}_{i,2} \\
& + X_\omega + 10.00 + \cos S_1 - \log S_2 \cdot \vec{x}_{i,5} \\
& - \cos n \cdot e^{\cos 05.69 - X_\Omega} + \sin \sin \sqrt{\cos 25.09} \\
& + X_\omega + \vec{x}_{i,2} + n + \vec{x}_{i,1} \cdot X_\Omega - \cos \sqrt{X_\Omega \cdot \vec{x}_{i,2}} \\
& + \frac{X_\omega}{\vec{x}_{i,4}} \cdot X_\omega - \cos \vec{x}_{i,5} + \cos \vec{x}_{i,5} \cdot \cos \vec{x}_{i,2} + X_\omega \\
& + X_\Omega - X_\omega + \cos X_\omega - \sin \cos S_2 \cdot \vec{x}_{i,5} \\
& - \cos \sin 87.07 + S_5 + X_\omega + 70.72 + \sqrt{X_\omega} \\
& + 81.71 + \cos \sin 89.00 \\
& + \cos (S_1 - S_2 + 42.06 - X_\omega) \cdot \cos X_\omega \\
& + \frac{X_\Omega}{88.17} + e^{\cos 15.51 - X_\omega + \cos X_\Omega} + 81.70 \\
& + \cos \cos 00.95 - \vec{x}_{i,4} + X_\Omega + X_\omega + \frac{X_\Omega}{\cos X_\Omega} \\
& + \sin \vec{x}_{i,3} + \vec{x}_{i,1} \cdot \frac{\cos n}{\vec{x}_{i,1}} - \cos \vec{x}_{i,2} + X_\omega + X_\Omega \\
& - X_\omega + \cos S_1 - \log S_2 \cdot \vec{x}_{i,5} - \cos n + X_\omega \\
& + \vec{x}_{i,1} + \vec{x}_{i,2} + n + \vec{x}_{i,1} + X_\Omega \\
& - \cos \sqrt{X_\Omega \cdot \cos X_\Omega} \cdot \frac{X_\omega}{\vec{x}_{i,4}} \cdot X_\omega - \cos \vec{x}_{i,5} \\
& + 12.78 \cdot X_\Omega + 60.03 - X_\Omega + X_\omega \cdot \log \sqrt{X_\omega} \\
& + X_\Omega + \vec{x}_{i,4} + \log 60.61
\end{aligned} \quad (7)$$

$$\begin{aligned}
HVC_{GE7}(\vec{x}_i, X) = & X_\Omega + \sqrt{X_\omega} / \log(11.36 - \sin((n \\
& - \sin(11.54 - 15.64 - 39.03 + 11.36 \\
& - \sin((n - \sin(11.13 - 09.39 - X_\omega \\
& + \log(11.36 - \sin(n - \sin(11.84 - \frac{\cos \vec{x}_{i,3}}{32.19} \\
& \cdot \vec{x}_{i,4} \cdot \vec{x}_{i,4} - \vec{x}_{i,3} - \log 47.44 + X_\omega - X_\Omega) \\
& - 43.95)^{\frac{1}{2}} + \log(11.36 - \sin((n - \sin(11.13 \\
& - 09.36 - \sqrt{X_\omega} + X_\Omega) - e^{\sin \vec{x}_{i,3}})^{\frac{1}{2}}) \\
& + \log 47.44))) - X_\omega - 11.15 + 41.56 \cdot 43.95 \\
& \cdot \log(11.36 - \sin((n - \sin(11.13 - 09.36 \\
& - \sqrt{X_\omega} + e^{\sin \vec{x}_{i,3}})^{\frac{1}{2}}) - \log 43.44)^{\frac{1}{2}}) \\
& + \sqrt{\log 21.75 - 05.41} + 41.56 + 62.95)^{\frac{1}{2}} \\
& + \log(11.35 \cdot 11.36 - \sin(11.15 - 41.56 \\
& - 43.95) + \log(11.56 - \sin((n - \sin(11.13 \\
& - 09.36 - \sqrt{X_\omega} + e^{\sin \vec{x}_{i,3}})^{\frac{1}{2}}) - \log 47.24)^{\frac{1}{2}} \\
& + \sin X_\omega))) - \vec{x}_{i,1})^{1/2}) + S_1 \cdot X_\omega)
\end{aligned} \quad (8)$$

$$\begin{aligned}
HVC_{GE8}(\vec{x}_i, X) = & \cos n \cdot X_\Omega \cdot \log S_5 \cdot X_\omega \\
& + \vec{x}_{i,2} \cdot \vec{x}_{i,4} \cdot \vec{x}_{i,6} + \log(X_\Omega \cdot \log S_5 \cdot X_\omega \\
& + \vec{x}_{i,2} \cdot \vec{x}_{i,4} \cdot \vec{x}_{i,6} + X_\Omega + \log n \cdot S_3 \\
& + S_3 \cdot e^{X_\Omega} + X_\Omega \cdot \log S_5 + S_5) \cdot X_\Omega \\
& \cdot \log((X_\Omega \cdot e^{X_\Omega} \cdot X_\Omega \cdot \log S_5 + \vec{x}_{i,5} \\
& + X_\Omega \cdot \log n + X_\Omega \cdot \cos(X_\Omega \cdot \vec{x}_{i,2} + \vec{x}_{i,6}) \\
& + \log S_5 + \vec{x}_{i,1} + X_\Omega \cdot 78.44 + \log(\cos S_1 \\
& + e^{X_\Omega}) + X_\Omega \cdot \log 07.17 \cdot S_7 + X_\Omega + \vec{x}_{i,2} \\
& \cdot \log S_5 + \vec{x}_{i,1} + X_\omega \cdot X_\Omega \cdot 78.44 - \vec{x}_{i,2} \\
& + \vec{x}_{i,4} \cdot \vec{x}_{i,6} + \log n \cdot S_1 + \cos S_6 \\
& \cdot \log e^{X_\Omega} + 56.88 + 55.76)^{\frac{1}{2}}) + \log S_5 \\
& + \vec{x}_{i,1} + \vec{x}_{i,2} \cdot X_\Omega + \vec{x}_{i,4} + n \\
& + (X_\Omega \cdot \log n \cdot S_1 + S_3 \cdot e^{X_\Omega} + X_\Omega \\
& \cdot \log S_5 + \vec{x}_{i,5})^{\frac{1}{2}} + X_\Omega \cdot \log S_7 + e^{X_\Omega} \\
& \cdot X_\omega \cdot \log \log S_7 + \vec{x}_{i,1} + X_\Omega \cdot e^{X_\Omega} \cdot X_\Omega \\
& + \log(57.88 \cdot 55.76) \cdot \log S_5 + \vec{x}_{i,1} \cdot \vec{x}_{i,2} \\
& + X_\Omega + \vec{x}_{i,3} + \log S_5 \cdot X_\omega + X_\Omega \cdot X_\Omega \\
& + \log S_5 + \vec{x}_{i,4} \cdot X_\Omega + \vec{x}_{i,3} + \frac{\log S_5}{18.46} + \vec{x}_{i,2}
\end{aligned} \quad (9)$$

$$\begin{aligned}
HVC_{GE10}(\vec{x}_i, X) = & \log X_\Omega + e^{\vec{x}_{i,2}} + \vec{x}_{i,7} + n \\
& + S_2 + \vec{x}_{i,6} + \sin x_{10} \\
& + e^{\log \vec{x}_{i,5} + X_\omega + X_\Omega} + \vec{x}_{i,8} \cdot \vec{x}_{i,2} \\
& + X_\Omega + \cos \vec{x}_{i,7} + n + S_2 \\
& + 55.49 \\
& + e^{\log \vec{x}_{i,4} + 12.17 + \sin \vec{x}_{i,9} + \vec{x}_{i,1} + e^{X_\Omega} + \log e^{\vec{x}_{i,1}}}
\end{aligned} \quad (10)$$

V. VALIDATION

To assess the performance of the new approximation functions found with our GE implementation, we compared them against the R2-based HVC and the HVC-Net approaches. For each of them, we used two metrics related to the identification of the worst-contributing individual in each file of the validation set. First, we measured how many of these individuals were correctly identified by each approximation method. Second, since none of the approximations can always correctly identify the worst-contributing individual, we measured the error in this identification using the same fitness function defined by (3). The values of these two metrics and the comparison of execution times are shown next. These results were obtained using the validation sets described in Section III and averaging the values for all files in the set. In all three cases, our proposed approach (HVC_{GE}) is shown in the last column. In the case of HVC_{NET} , the available models were used for 3, 5, 8, and 10 dimensions. In each row, the best value is shown in **boldface**.

The comparison of results using the first metric are shown in Table II. The higher the percentage of correctly identified worst individuals, the better the approximation. Since multiple points can have the worst-contributing HVC value, we adopted an average ranking as in the training phase. Hence, as long as one of the solutions with the worst HVC value was selected, it was counted as a correctly identified individual. For two dimensions, HVC_{R2} obtained

a slightly better value than the approximation found using our approach. However, our approximations obtained a significantly higher percentage than the other two approaches for the other dimensionalities. Remarkably, they correctly identified the worst individuals between 4.26% and 7.86% more frequently than HVC_{NET} , and between 7.36% and 40.31% more frequently than HVC_{R2} .

TABLE II
COMPARISON OF CORRECTLY IDENTIFIED WORST INDIVIDUALS
BETWEEN HYPERVOLUME CONTRIBUTION APPROXIMATIONS.

Dimensions	Correct identification of worst individual (%)		
	HVC_{NET}	HVC_{R2}	HVC_{GE}
2	-	23.93	20.03
3	15.11	12.30	19.66
4	-	11.21	31.08
5	23.55	9.03	27.81
7	-	5.86	29.21
8	38.01	5.56	45.87
10	39.84	5.64	44.46

The comparison results using the second metric are shown in Table III. In this case, the smaller the error, the better the approximation. A similar behavior to the previous comparison can be observed: HVC_{R2} obtained the best value in the two-dimensional comparison, while our approximations obtained the best value in all the other dimensions. In this case, our approach obtained between 4.31% and 18.28% less error than HVC_{NET} and between 21.04% and 62.03% less error than HVC_{R2} .

TABLE III
ERROR COMPARISON BETWEEN HYPERVOLUME CONTRIBUTION
APPROXIMATIONS.

Dimensions	Error (%)		
	HVC_{NET}	HVC_{R2}	HVC_{GE}
2	-	18.612	26.048
3	30.803	33.563	12.5225
4	-	35.864	10.101
5	15.245	48.987	10.939
7	-	65.143	8.969
8	12.988	56.218	5.251
10	18.115	66.766	4.738

It is worth noting that although HVC_{R2} obtains a better value in both metrics for two-dimensional files, adopting an HVC approximation is particularly useful in many-objective problems. Furthermore, increasing the dimensionality seems to deteriorate the performance of HVC_{R2} , as it exhibits a smaller percentage of correctly identified worst individuals and a higher error rate. In contrast, both HVC_{NET} AND HVC_{GE} improved their performance as the dimensionality increased, with HVC_{GE} obtaining better overall results. This behaviour may occur because HVC_{R2} relies on a set of weight vectors that should ideally increase as the number of dimensions increases. However, this would also increase its computational cost, which is already the highest of the three compared approaches.

The execution time comparison is shown in Table IV. In the available instances, HVC_{NET} is the fastest approximation except for three-dimensional files. However, HVC_{GE} is the second fastest approximation, being one or even two orders of magnitude faster than HVC_{R2} and of the same order of magnitude than HVC_{NET} for 3, 5 and 10 dimensions.

TABLE IV
EXECUTION TIME COMPARISON BETWEEN HYPERVOLUME
CONTRIBUTION APPROXIMATIONS.

Dimensions	Execution time (s)		
	HVC_{NET}	HVC_{R2}	HVC_{GE}
2		9.93E-02	4.67E-03
3	7.58E-03	1.46E-01	7.52E-03
4		1.74E-01	4.11E-03
5	1.04E-02	4.45E-01	6.12E-02
7		4.66E-01	4.30E-02
8	6.17E-03	2.45E-01	3.28E-02
10	1.00E-02	1.55E+00	1.80E-02

It should be emphasized that these execution times only consider the calculation of the HVC approximation values of the validation set. Hence, the time required for the training steps of HVC_{NET} and HVC_{GE} , as well as the time required to generate the vector files used by HVC_{R2} , are left out of these comparisons. We consider this to be a fair comparison since, realistically, the training steps only need to be done once, in order to find the models/approximation functions. Then, they can be used in any desirable way to obtain the HVC approximations of any given file.

Finally, we evaluated the performance using real-world problems with 2 to 6 objectives. These problems were selected from a benchmark proposed by Tanabe and Hishibuchi [15], and consist of continuous MOPs. The instance with two objectives has a convex Pareto front, but the remaining ones have Pareto fronts with unknown shapes. A brief description of each problem is shown in Table V.

TABLE V
REAL-WORLD PROBLEMS USED TO COMPARE THE PERFORMANCE OF
THE HVC APPROXIMATIONS.

Problem	Problem type	Objectives	Variables	Constraints
RE2-1	Four bar truss design	2	4	0
RE3-1	Two bar truss design	3	3	3
RE4-1	Car side impact design	4	7	10
CRE5-1	Water resource planning	5	3	7
RE6-1	Water resource planning	6	3	7

For this comparison, we used a similar methodology for generating the training/evaluation data. In this case, a total of five independent executions of NSGA-III were performed, solving each of the selected test instances. Then, for each execution, the population was stored at different points measured by the number of generations. In all cases,

a maximum of 1000 generations was used, and a total of 24 populations were stored per execution, resulting in 120 different populations per test instance. Then, the real HVCs were obtained to compare the performance of the HVC approximations. The results obtained using real-world MOPs are shown in Table VI. It is worth noting that, as in the previous results, there are some missing values from HVC_{NET} because this model is only available for 3, 5, 7, and 10 dimensions.

TABLE VI
ERROR COMPARISON BETWEEN HYPERVOLUME CONTRIBUTION
APPROXIMATIONS IN REAL-WORLD PROBLEMS.

Problem	Error (%)		
	HVC_{NET}	HVC_{R2}	HVC_{GE}
RE2-1	-	20.2220	83.0077
RE3-1	43.8017	70.0410	67.9640
RE4-1	-	54.1897	29.9099
CRE5-1	10.3538	17.1416	15.4663
RE6-1	-	67.9101	30.9821

The behavior in the 2-objective real-world MOP is similar to that of the 2-objective benchmark MOP, since it is the only test instance where HVC_{R2} obtained the best value. However, in all the other instances HVC_{R2} obtained the worst values: in the problems where a model of HVC_{NET} was available (problems with 3 and 5 objectives), it had the best performance, followed by HVC_{GE} . In the remaining instances (problems with 4 and 6 objectives), HVC_{GE} obtained a better value than HVC_{R2} . These results show that HVC_{NET} performs the best when available in the real-world MOPs selected, while HVC_{GE} achieves a competitive performance, much better than HVC_{R2} in instances with 4 and 6 objectives.

VI. CONCLUSIONS

In this work, we have introduced a methodology to generate new HVC approximation functions using a GE implementation. Our experimental validation compared the quality of the approximation using two metrics related to identifying the worst-contributing individual in each validation file and an execution time comparison. Using the proposed validation sets, the HVC_{GE} functions obtained HVC approximations with better quality than the other two approaches for three or more dimensions. Regarding execution times, HVC_{GE} obtained significantly faster times than HVC_{R2} and slightly slower times than HVC_{NET} in most cases.

Based on our experimental results, we claim that GE is a viable technique for generating HVC approximation functions that can improve the performance of the currently available methods for approximating hypervolume values. However, two aspects should be taken into account when adopting this methodology. First, we should consider the computational time required to generate the training set and to find the actual approximation function via the GE implementation. Both of these steps required several days to be completed, although once an approximation has been

found, this process does not need to be repeated. Second, the performance of these type of approximation depends on the data used for the training. For the generation of training and validation sets, we extracted the population of NSGA-III during the solution of problems provided by the DTLZ and WFG test suites in an effort to consider data converging to Pareto sets with different geometries and at different levels of convergence. However, a further comparison adopting different validation data could help to determine the efficiency of our approximations in different scenarios.

This work could be extended to evaluate the performance of these new approximations as part of a density estimator to solve MOPs. Moreover, a similar methodology can be designed to create new density estimators that do not necessarily approximate any existing performance indicator but are rather a completely new performance measure.

REFERENCES

- [1] C. A. Coello Coello, G. B. Lamont, and D. A. Van Veldhuizen, *Evolutionary Algorithms for Solving Multi-Objective Problems*. New York: Springer, second ed., September 2007. ISBN 978-0-387-33254-3.
- [2] J. G. Falcón-Cardona and C. A. C. Coello, "Indicator-based multi-objective evolutionary algorithms: A comprehensive survey," *ACM Comput. Surv.*, vol. 53, Mar. 2020.
- [3] E. Zitzler and S. Künzli, "Indicator-based Selection in Multiobjective Search," in *Parallel Problem Solving from Nature - PPSN VIII* (X. Y. et al., ed.), (Birmingham, UK), pp. 832–842, Springer-Verlag. Lecture Notes in Computer Science Vol. 3242, September 2004.
- [4] S. Jiang, J. Zhang, Y.-S. Ong, A. N. Zhang, and P. S. Tan, "A Simple and Fast Hypervolume Indicator-Based Multiobjective Evolutionary Algorithm," *IEEE Transactions on Cybernetics*, vol. 45, pp. 2202–2213, October 2015.
- [5] A. Menchaca-Mendez and C. A. Coello Coello, "An alternative hypervolume-based selection mechanism for multi-objective evolutionary algorithms," *Soft Computing*, vol. 21, p. 861–884, Aug. 2015.
- [6] E. Zitzler, D. Brockhoff, and L. Thiele, "The Hypervolume Indicator Revisited: On the Design of Pareto-compliant Indicator Via Weighted Integration," in *Evolutionary Multi-Criterion Optimization, 4th International Conference, EMO 2007* (S. O. et al.
- [7] K. Shang, H. Ishibuchi, and X. Ni, "R2-based hypervolume contribution approximation," *IEEE Transactions on Evolutionary Computation*, vol. 24, p. 185–192, Feb. 2020.
- [8] K. Shang, W. Liao, and H. Ishibuchi, *HVC-Net: Deep Learning Based Hypervolume Contribution Approximation*, p. 414–426. Springer International Publishing, 2022.
- [9] M. O'Neill and C. Ryan, "Grammatical evolution," *IEEE Transactions on Evolutionary Computation*, vol. 5, no. 4, pp. 349–358, 2001.
- [10] M. Nicolau and A. Agapitos, "Understanding grammatical evolution: Grammar design," in *Handbook of Grammatical Evolution*, pp. 23–53, Springer International Publishing, 2018.
- [11] K. Deb, L. Thiele, M. Laumanns, and E. Zitzler, "Scalable test problems for evolutionary multiobjective optimization," in *Evolutionary multiobjective optimization*, pp. 105–145, Springer, 2005.
- [12] S. Huband, L. Barone, L. While, and P. Hingston, "A scalable multi-objective test problem toolkit," in *Evolutionary Multi-Criterion Optimization* (C. Coello Coello, ed.), (Berlin, Heidelberg), pp. 280–295, Springer Berlin Heidelberg, 2005.
- [13] K. Deb and H. Jain, "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*, vol. 18, pp. 577–601, Aug. 2014.
- [14] M. Fenton, J. McDermott, D. Fagan, S. Forstenlechner, M. O'Neill, and E. Hemberg, "PonyGE2: Grammatical Evolution in Python," in *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 1194–1201, July 15–19, 2017.
- [15] R. Tanabe and H. Ishibuchi, "An easy-to-use real-world multi-objective optimization problem suite," *Applied Soft Computing*, vol. 89, p. 106078, Apr. 2020.