Applied Soft Computing 125 (2022) 109103

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

Towards fast approximations for the hypervolume indicator for multi-objective optimization problems by Genetic Programming

Cristian Sandoval [a], Oliver Cuate [b,c], Luis C. González [d,*], Leonardo Trujillo [a,*], Oliver Schütze [c]

[a] Tecnológico Nacional de México/IT de Tijuana, Tijuana, BC, Mexico
[b] Escuela Superior de Física y Matemáticas del Instituto Politécnico Nacional, Mexico City, Mexico
[c] Computer Science Department, Cinvestav-IPN, Mexico City, Mexico
[d] Autonomous University of Chihuahua, School of Engineering, Chihuahua City, Chihuahua, Mexico

ARTICLE INFO

Article history:
Received 15 December 2020
Received in revised form 19 January 2022
Accepted 26 May 2022
Available online 6 June 2022

Keywords:
Multi-objective optimization
Genetic programming
Hypervolume
Automatic algorithm design
Regression

ABSTRACT

Hypervolume (HV) has become one of the most popular indicators to assess the quality of Pareto front approximations. However, the best algorithm for computing these values has a computational complexity of $O(N^{k/3}\text{polylog}(N))$ for N candidate solutions and k objectives. In this study, we propose a regression-based approach to learn new mathematical expressions to approximate the HV value and improve at the same time their computational efficiency. In particular, Genetic Programming is used as the modeling technique, because it can produce compact and efficient symbolic models. To evaluate this approach, we exhaustively measure the deviation of the new models against the real HV values using the DTLZ and WFG benchmark suites. We also test the new models using them as a guiding mechanism within the indicator-based algorithm SMS-EMOA. The results are very consistent and promising since the new models report very low errors and a high correlation for problems with 3, 4, and 5 objectives. What is more striking is the execution time achieved by these models, which in a direct comparison against standard HV calculation achieved extremely high speedups of close to 100X for a single front and over 1000X for all the HV contributions in a population, speedups reach over 10X in full runs of SMS-EMOA compared with the standard Monte Carlo approximations of the HV, particularly for large population sizes. Finally, the evolved models generalize across multiple complex problems, using only two problems to train the problems from the DTLZ benchmark and performing efficiently and effectively on all remaining DTLZ and WFG benchmark problems.

1. Introduction

Real-world problems often require the simultaneous optimization of several competing objectives, leading to multi-objective optimization problems (MOPs). One important characteristic of MOPs is that their solution sets, the so-called Pareto sets, as well as their images, the Pareto fronts, typically form objects of dimension $(k-1)$, where k is the number of objectives considered in the given problem. For the numerical treatment of such problems, multi-objective evolutionary algorithms (MOEAs) have caught many researchers' and practitioners' interest during the last two decades. Reasons for this include that MOEAs are of global nature, very robust, require minimal assumptions on the model, and are capable of computing finite-size approximations of the entire Pareto set/front of the given MOP in a single run of the algorithm. Since the outcome of every MOEA is an entire set of candidate solutions (population) that ideally resembles the solution set (mainly the Pareto front), one question that naturally arises is how to measure the obtained approximation quality. This is needed to compare different solution sets and guide the MOEA towards the "best" Pareto front approximation.

One performance indicator that is widely used is the Hypervolume indicator (HV, [1,2]). Although this indicator has several valuable properties [2–4], it has one critical weakness: the cost for evaluating the HV value of given candidate sets grows exponentially with the number of objectives. That is, while this cost is relatively low for bi-objective problems (compared to the overall cost of a MOEA), the computation of the HV values become the bottleneck for MOPs with more objectives, which represents a severe drawback for the applicability of the HV in modern applications. Since decision-making processes are getting more sophisticated, it is a natural consequence that also the related

* Corresponding authors.
E-mail addresses: sr_cristian@outlook.es (C. Sandoval), ocuateg@ipn.mx (O. Cuate), lcgonzalez@uach.mx (L.C. González), leonardo.trujillo@tectijuana.edu.mx (L. Trujillo), schuetze@cs.cinvestav.mx (O. Schütze).

https://doi.org/10.1016/j.asoc.2022.109103
1568-4946/

MOPs increase their number of optimization objectives. And this is not only valid for the quality assessment of a given solution set, but even for the correct functioning of those MOEAs that are based on computing thousands of HV values and HV contributions (i.e., the contribution of an individual of a given population to the HV value) within one run of the algorithm. A straightforward implementation of the HV value of a given set S with a magnitude N leads to a complexity of $\mathcal{O}(N^{k+1})$, while the best algorithm has a complexity of $O(N^{k/3}\text{polylog}(N))$. Literature reports several methods that aim for a reduction of the computational cost for the HV. For instance, some methods proposed algorithms that reduce the complexity of the computation for specific cases [3,5]; others employ techniques to approximate the values of the Hypervolume [6]; and finally, algorithms specialized on the Hypervolume contributions have also been proposed [7–9].

In this work, we propose using a machine learning regression technique that can produce relatively simple and efficient models that approximate the Hypervolume indicator's behavior. The goal is to approximate the real indicator value, with minimal deviation, for any given problem. The modeling strategy considered is Genetic Programming (GP), which can produce models expressed as symbolic mathematical expressions. The GP system is set up to obtain efficient and straightforward models, avoiding unnecessarily large or complex structures. Thus, the resulting expressions' main advantage is their computational complexity, which significantly speeds up the computational times (mostly runs in linear time) while keeping the quality in the obtained approximation.

Accordingly, we can summarize the main contributions of this study as follows.

- We pose the problem of deriving approximate models of the HV indicator as a supervised learning problem that we approach through GP regression.
- We show that the learned models are highly efficient, particularly when combined with an adequate updating process, achieving large speedups relative to the state-of-the-art.
- We present results that show that the evolved models effectively approximate the HV indicator, allowing them to be used in two common scenarios: (1) quality indicators and (2) guiding the search of an indicator-based MOEA, both tasks tested for 3-objectives, 4-objective, and 5-objective MOPs.
- The evolved models are quite general, since models trained on two benchmarks can be used to guide an indicator based MOEA on a variety of different MOPs.

The remainder of this paper is organized as follows. In Section 2, we briefly present some definitions and related work on multi-objective optimization and GP, respectively. In Section 3, we present the problem formulation, posing it as a supervised learning problem on which to apply GP. Afterwards, we outline our proposed approach in Section 4 and provide details of the main algorithms used. In Section 5 we present the main results of our study. Finally, Section 6 contains the conclusions and future work.

2. Background on multi-objective optimization and performance indicators

A continuous MOP can be mathematically defined as follows:

$$\begin{aligned} \min_{x\in D} \quad & F(x), \\ \text{s.t.} \quad & G(x) \leq 0 \\ & H(x) = 0. \end{aligned} \tag{1}$$

Hereby, $F : D \subset \Re^n \to \Re^k$, $F(x) = (f_1(x), \ldots, f_k(x))$ is the objective function that is defined by the individual objectives $f_i : D \subset \Re^n \to \Re$. The domain D of F is defined by the subset of the $\Re^n$ that satisfies all inequality and equality constraints,

$$D := \{x \in \Re^n \ : \ G(x) \leq 0 \text{ and } H(x) = 0\}. \tag{2}$$

The optimality of a MOP is defined by the concept of *dominance*. Let $v, w \in \Re^k$, then we say that the vector v is *less than* w ($v \prec_p w$), if $v_i < w_i$ for all $i \in \{1, \ldots, k\}$; the relation $\leq_p$ is defined analogously. A vector $y \in D$ is called *dominated* by a vector $x \in D$ ($x \prec y$) with respect to (1) if $F(x) \leq_p F(y)$ and $F(x) \neq F(y)$, else y is called non-dominated by x. A point $x^* \in D$ is Pareto optimal wrt (1) if there is no $y \in D$ which dominates x^*. The set P_D of all the Pareto optimal points is called the Pareto set and its image $F(P_D)$ is called the Pareto front. Typically, i.e., under mild conditions in the MOP, Pareto set and front form objects of dimension $(k-1)$, see [10]. Hence, the numerical solution of a MOP refers to finding an adequate finite-size approximation of the solution set, where most works focus on the Pareto front since for every vector x the image $F(x)$ determines the qualities of x (in terms of the given individual objectives). In order to assess the approximation quality of a given set wrt the Pareto front several performance indicators have been proposed in recent years including the Generational Distance (GD, [11]), the Inverted Generational Distance (IGD, [12]), the averaged Hausdorff distance (Δ_p, [13–15]), R2 [16–18], DOA [19], IGD^+ [20], and the Hypervolume (HV, [1–3]). Though the indicators differ from each other, they all have in common that they aim in their way for convergence, spread and a uniform distribution along the solution set. In this study we will focus on the Hypervolume indicator, since it is one of the most adopted indicators for performing the comparison of MOEAs, and it is also widely used for indicator-based MOEAs such as SMS-EMOA [21] and HypE [6].

The HV indicator, also called S-metric, is defined as follows. Let $y^{(1)}, \ldots, y^{(N)} \in \Re^k$ be a set of non-dominated vectors and $r \in \Re^k$ (a reference point) be such that $y^{(i)} \prec r$ for all $i = 1, \ldots, \mu$. Then, the value

$$\mathcal{H}\left(y^{(1)}, \ldots, y^{(N)}; r\right) = \mathcal{L}\left(\cup_{i=1}^{N}\left[y^{(i)}, r\right]\right), \tag{3}$$

is termed the dominated hypervolume with respect to the reference point r, where $\mathcal{L}(\cdot)$ denotes the Lebesgue measure in $\Re^k$. This indicator hence measures the size of the space that is covered or dominated by the given set, that is, the union of hypercubes defined by a non-dominated point $y^{(i)}$ and a reference point r.

One major drawback of this indicator is that it is computationally expensive, as its estimated complexity is $\mathcal{O}(N^{k+1})$. However, some specialized algorithms can achieve a better complexity for some specific cases. For instance, the Walking Fish Group algorithm (WFG [22]) is a state-of-the-art algorithm that has a worst-case complexity of $O(k \cdot 2^N)$ though experimentally it has a better performance on average. While the best known algorithm for $k \geq 3$ is derived from a classic problem in computational geometry, i.e., Klee's measure problem, which runs in $O(N^{k/3} \text{ polylog } N)$ [23]. On the other hand, when interested in hypervolume contributions, for $k = 2, 3$ the problem has complexity of $O(N \log N)$ [9]; and, for the general case it has complexity of $O(N^{k/2} \log N)$ for $k > 2$ [8].

For this reason, the use of approximations for reducing computational efforts is not uncommon [6,24–26]. Algorithms that aim to approximate the HV are very varied, some of them can guarantee an (ϵ, δ)-approximation [27] or they use polar coordinates [28]. However, for approximating the Hypervolume contributions the most common approaches are instance scalarizing function method [29], modified scalarizing function methods [30,31], and Monte-Carlo methods [32–34]. To add some context to this analysis, consider that in this work, we approximate the Hypervolume contributions with a complexity of $O(kN \log N)$ and

the main advantage of the proposed method over the above mentioned ones is that it can in principle be applied to any other performance indicator.

In addition, using approximations, instead of the exact HV, is supported by some works [35,36]. Also, this search is theoretically supported by the fact that the use of greedy strategies for computing the hypervolume contributions is not necessarily optimal [37].

2.1. Synthesizing new heuristics using supervised learning

One of the most promising application domains of machine learning in general, and GP in particular, is to derive new heuristics or modify existing heuristics to improve their performance or efficiency. Notable examples include TPOT [38], a tool for Automatic Machine Learning (AutoML), that can evolve partial machine learning pipelines that are optimized to solve specific problem instances. For Supervised Learning, GP has shown to be able to dramatically improve classification performance when coupled with Machine Learning algorithms, even outperforming competitive methods such as Support Vector Machines (SVM) and Artificial Neural Networks in a variety of datasets [39,40]. For optimization problems, particularly in scheduling, GP has been used to find heuristic dispatching rules that have been shown to outperform standard techniques in some real-world scenarios [41]. Similarly, in computer vision, this approach has been used to approximate a previously known feature detection method [42] or find new ones that exhibit practical characteristics [43]. Automated Design of Algorithms (ADA) is focused on improving meta-heuristic algorithms using evolutionary techniques [44], where the focus is on evolving high-level instructions to control or manage a search process. ADA attempts to either replace specific algorithmic design features or reorder a meta-heuristic algorithm. Lately, recent work has begun to explore in-situ algorithmic improvements of existing heuristic tools, with a technique called Genetic Improvement that operates directly at the level of source code [45]. One study has applied Genetic Improvement on a GP library directly [46]. Through this revision, we observe that GP can improve different kinds of heuristic mechanisms, which is the main path that this study will follow to generate fast approximations to the HV indicator for MO problems.

3. Computational problem statement

In this work, the goal of deriving a MOEA performance indicator is posed as a synthesis or learning problem instead of a traditional analytical or formal derivation. While this could be modeled in different ways, we propose defining a supervised machine learning problem to build a new model to compute a performance indicator. To do so, it is necessary to define a target functionality that a learning algorithm will attempt to match, contained in a set of training instances. From this, it is then essential to define a suitable cost or objective function and search procedure that allows us to synthesize an appropriate computational model for the desired indicator. Thus, we pose the learning task in the following manner.

Target functionality. The target functionality of the proposed solutions is based on their deviation with respect to the expected *ground truth*, which in this case corresponds to the existing HV indicator. The proposed method will attempt to replicate the HV's functionality while biasing the search towards solutions that improve the HV's non-functional properties, i.e., computational complexity. One of this study's contributions is the improvement of the computational cost of the standard HV computation methods. Therefore, we aim to produce a compact and efficient symbolic model.

Training data. Supervised learning problems are defined by the training set $\mathcal{T} = \{I_i\}$ with $i = 1, \ldots, N$, where each training instance is defined as a tuple $I_i = (\mathbf{x}_i, y_i)$, where $\mathbf{x}_i$ represents the input data and y_i is the target output. Regularly, each training instance $\mathbf{x}_i$ is represented in a l dimensional space, where $|l|$ corresponds to the number of features or attributes of learning objects. The target, as defined above, is defined as the HV in this work. On the other hand, defining the input variables is not as straightforward. For the HV, the input is a set of non-dominated solutions A_i, which in principle could be given as input to a learning algorithm. However, in practice, the cardinality of A_i is not fixed; this limits the options of possible learning strategies that could learn on such data. Another approach is to transform set A_i into a fixed-length and real-valued feature vector $\mathbf{d}_i \in \Re^l$, where each feature dimension d_i^j, with $j = 1, \ldots, l$, provides a partial description of A_i. This is similar to how many supervised regression and classification tasks are solved when analyzing complex signals, such as brain signals [47], complex chemical processes [48], or mimicking human judgment on highly subjective tasks [49]. In this case, to build a training set, it will be necessary to extract a set of real Pareto fronts and their corresponding HV values along with a set of descriptive features.

Fitness or objective function. Given our definition of the training data, we will use as an error-based measure the Root Mean Squared Error (RMSE) between the estimation given by a learning model for the HV and the ground truth HV in the training set. This, however, only accounts for the desired functional behavior; it does not take into account the non-functional property of HV estimation's computational cost. There are two options here. One is to add another term to the objective function, such as the candidate model's run time. However, another option is to curtail the learning process not to derive overly complex solutions. The cost of HV computation is basically a by-product of the course of dimensionality; as the number of objectives grows, then the computational cost does so as well. The second approach is used in this work with GP since it simplifies the search process and eliminates costly non-optimal models during training.

4. Methodology

This section outlines our proposed approach to derive models that can efficiently approximate the HV indicator. The proposed methodology includes the following main stages.

1. Generate the learning dataset. From a set of widely used MO benchmarks, we obtain a sample of approximations to the Pareto front. We made this by running a MOEA on these problems, and used a heuristic sampling policy. To compute the HV for each data set (ground truth), we apply the WFG implementation.[1]
2. Feature extraction. A set of suitable descriptive features must be extracted from each Pareto Front to pose the learning task. We use a length-fixed feature vector regardless of the number of points in any given front.
3. Learning method. In this work, GP is used to solve the supervised learning model, aiming to produce compact, efficient, and symbolic models.
4. Performance evaluation. Standard training and testing procedures are used to evaluate the evolved models. In particular, we are interested in evaluating the accuracy of the models relative to: (1) the approximation to the ground truth HV, (2) the efficiency of the models in terms of computation time, and (3) the models' ability to be used in

[1] www.wfg.csse.uwa.edu.au.

conjunction with an indicator-based MOEA (here used to compute HV contributions). In particular, (3) might be the most relevant for practical purposes, since the excessive computational cost of computing the HV is one of the biggest drawbacks of using an indicator-based MOEA.

4.1. Datasets and feature extraction

In this work, we use two sets of multi-objective benchmark problems: the DLTZ functions [50], as well as the WFG [51] functions, considering three, four, and five objectives. We have chosen these benchmarks since they are widely known and used by the EMO community, and since their Pareto fronts pose an interesting range of different characteristics (linear, convex, concave, and mixed fronts, connected and disconnected fronts, as well as degeneration). Please note that in particular, the benchmark of WFG problems is still considered a challenge for state-of-the-art MOEAs. The number of decision variables used for each problem are set as suggested by each benchmark suite. However, to show the generalization of the models produced by way of GP regression, we only use the DTLZ benchmark problems for training, while for testing both the DTLZ and WFG problem are used. In this sense, we use simpler problems (DTLZ) to generate the models, and experimentally evaluate the performance on both simple and complex (WFG) test problems.

To generate the learning dataset, we execute the SMS-EMOA algorithm using the parameters of Table 1. From these executions, we can extract a varied set of non-dominated fronts from each problem. SMS-EMOA uses a steady-state process to replace each individual in the population every generation, where the worst individual in the population (with the lowest HV contribution) is replaced every time a new offspring enters the population. Moreover, the population is always ranked into a series of non-dominated fronts, with the worst individuals are in the front with the lowest rank. Our sampling process focuses on the non-dominated fronts that contain the worst individuals in any given population, because we want to ensure that our models can correctly compute the HV contributions of these individuals, since these are the ones that need to be identified and replaced during the SMS-EMOA search.

To achieve this, we use three sampling rules to obtain our sets of non-dominated solutions. First, every time p new offspring are generated we store the lowest ranked front. Second, every generation we store all of the fronts in the population. Finally, it is important to consider that as the search progresses the population converges to a single front, and the general approximation of the optimal Pareto front is basically set with only minimal improvements with most of the new offspring generated in later generations. Therefore, the Pareto front becomes stagnant, and the sampling process produces many similar fronts, which are not very useful for machine learning purposes. However, sometimes a small subset of individuals does aggregate in a secondary front of dominated individuals with a lower rank, and we want to make sure that all of these fronts are stored since they contain the individuals that will be subjected to replacement in the later generations. Hence, our third rule is to set $p = 1$ when there are only two fronts in the population and the lowest ranked of the two fronts contains less than 1/3rd of the population..

Also, only sets of three or more elements (points) were considered, eliminating all non-dominated sets of one or two elements (since no descriptive features can be obtained). We use $p = 8$ in our experiments described below. Tables 2 and 3 summarize the number of non-dominated sets extracted from each problem using SMS-EMOA, respectively, for the DTLZ and WFG benchmarks.

Table 1
Parameters for SMS-EMOA to generate the non-dominated fronts.

Parameter	Value
Population size	300
Generations	100
Crossover	Simulated Binary Crossover (SBX)
Crossover probability	0.9
Mutation	Polynomial
Mutation Probability	$1/n$

Table 2
Number of non-dominated sets extracted from each problem in the DTLZ benchmarks.

Problem	3 Objectives	4 Objectives	5 Objectives
DTLZ1	8126	8389	12 248
DTLZ2	4081	11 565	12 216
DTLZ3	12 135	10 914	8126
DTLZ4	5796	11 104	10 652
DTLZ5	6212	13 874	12 067
DTLZ6	10 092	13 001	10 973
DTLZ7	7208	11 841	11 982

Table 3
Number of non-dominated sets extracted from each problem in the WFG benchmarks.

Problem	3 Objectives	4 Objectives	5 Objectives
WFG1	4704	14 218	12 875
WFG2	4710	9796	10 952
WFG3	6445	14 407	14 666
WFG4	6352	12 697	13 881
WFG5	6317	11 952	13 221
WFG6	4512	10 254	11 847
WFG7	7947	13 563	14 438
WFG8	3990	10 090	11 631
WFG9	8303	14 063	14 707

Table 4
Statistical features extracted from the non-dominated set of solutions, for the jth objective function.

Feature	Symbol
Mean	$\mu^{(j)}$
Median (Second Quartile)	$Q2^{(j)}$
Standard deviation	$\sigma^{(j)}$
First Quartile	$Q1^{(j)}$
Second Quartile	$Q3^{(j)}$
Kurtosis	$\kappa^{(j)}$
Skewness	$\gamma^{(j)}$

4.1.1. Feature extraction

Feature engineering and feature extraction are well-known as the more complex and critical processes in Machine Learning pipelines [52]. Indeed, the input features from a particular problem often must be transformed into a suitable new space to allow learning to take place effectively and efficiently [53,54]. This is precisely the reason why Deep Learning is pragmatically attractive because the feature engineering and extraction process are off-loaded directly to the learning process. However, such an approach's negative side-effect is the loss of model interpretability, given that the features are in a black-box form [55]. Therefore, in this work, we propose a relatively simple but effective set of statistical features to describe a non-dominated set of solutions. The extracted features are composed of seven descriptive statistics, computed on the non-dominated set relative to each objective function, as summarized in Table 4. An important factor in the HV estimation is the computational cost. A benefit of using relatively simple statistical descriptors for feature extraction is their low complexity and efficient implementation.

It means that for a k objective problem, the number of statistical features is $k \times 7$. This also means that a model learned for three objective problems cannot be used to estimate the HV of a four objective problem. However, having specialized estimation methods that depend on the number of objectives is not uncommon for the HV indicator [9,22,56].

All of the objective values of the solutions in the non-dominated set are normalized using min–max normalization in the range [0, 1]. Normalization of the objective function values is intended to simplify the learned models' generalization to different problem instances. It also allows us to omit the reference point from each training instance's input features, taking for all cases the point $(1.1, \ldots, 1.1)^T \in \Re^k$.

The feature vector's computation must be as efficient as possible, as the HV approximations will also be used in the *Performance Evaluation* stage as part of the SMS-EMOA algorithm. At each generation, the HV values of the entire population and the HV contributions of each individual must be obtained. Fortunately, all the feature vector components are statistical moments (see Table 4), which means that we can improve the HV contributions' computation using special formulae. The derivation of such formulae is detailed in Appendix A. From this, it is possible to reduce the feature vector computation's complexity by one order of magnitude. For instance, consider the computation of standard deviation for a MOP with k objectives and population size of N. Let $\sigma^{(j)}$ denote the standard deviation of the population for the jth objective (as it is stated in Table 4), and let $\sigma_i^{(j)}$ with $i = 1, \ldots, N$, denote the standard deviation for the jth objective of the population without the ith individual. Notice that, in the naive way, obtaining $\sigma^{(j)}$ and $\sigma_i^{(j)}$ has a complexity of $O\left(N^2\right)$ (N operations for $\sigma^{(j)}$ and $N-1$ for each $\sigma_i^{(j)}$, $i = 1, \ldots, N$). However, we only spend N operations to get $\sigma^{(j)}$ and a constant number of computations for each $\sigma_i^{(j)}$, $i = 1, \ldots, N$, that is, the complexity is now $O(N)$. We can perform a similar analysis for each one of the components of the feature vector. In this way, the complexity of computing the whole vector for all the objectives is reduced from $O(kN^2)$ to $O(kN \log N)$, due to the sorting process for computing the quartiles.

4.2. GP-based model induction

As stated in Section 3, the goal is to generate relatively simple, efficient, and symbolic models from the supervised learning problem that is being posed. Therefore, the proposal is to use GP, a well-known evolutionary algorithm for the automatic model or program induction, often used to solve challenging real-world supervised learning problems [57,58]. The standard pseudo-code of a GP system, similar to most EAs, is outlined in Algorithm 1. Some notable aspects of a GP algorithm are the following. Individuals are usually expressed as variable-length syntax trees, where each node represents an element from a Primitive Set. The Primitive Set is composed of both Terminal and Functional elements; the former are input features of the learning problem (e.g. $\mu^{(j)}$, $\sigma^{(j)}$, $\kappa^{(j)}$ etc.), numerical constants, and 0-arity functions. The latter are basic functional building blocks used to construct the syntactic expression, such as arithmetic operations, logical operations, or trigonometric functions. It is possible to use more complex control elements in a GP algorithm, such as conditionals or loops [57]. However, these may increase computational complexity of the evolved models and are therefore not used in this work.

While standard GP is capable of solving complex real-world problems, hybrid approaches tend to achieve the best performance. One promising family of GP-based hybrid methods is Constructive Induction (CI) methods that use GP to build a feature transformation operator and an additional modeling process to derive the final model [53,54]. These are more generally known as wrapper-based techniques for Feature Engineering and have been used successfully in many problem domains. A useful taxonomy of these methods is presented in [53], and one such method is GPTIPS [54], which is used in this work. This algorithm uses a multi-tree representation, where each subtree represents a new feature dimension of the problem, integrates specialized search operators, and builds the final model using regularized multiple linear regression.

Algorithm 1 Standard GP Pseudo-code.

```
for i = 1 to Num_Of_Generations (or until an acceptable solution is found) do
    if 1st generation then
        generate the initial population with the Primitive Set (terminals, constants and math operations)
    end if
    Calculate fitness (minimize and error measure) of population members
    Perform Tournament Selection of parent individuals
    Create offspring using genetic operations according to specified probabilities
end for
Return best individual in last population
```

In particular, GPTIPS uses a representation that performs a mapping $m : \Re^p \rightarrow \Re^d$ where m is the evolved model, p is the number of original features, and d is the size of the new feature space (number of new features produced by the model). The root node of m works as a container and each subtree of the root node ST_i defines a new feature dimension, such that $i = 1, \ldots, d$. Conversely, standard regression optimizes a fixed model structure, Multiple Linear Regression (MLR) fits a linear model to the observed dataset, as given by

$$y_i = \beta_0 + \beta_1 x_{i,1} + \beta_2 x_{i,2} + \cdots + \beta_p x_{i,p} + \epsilon_i \quad \text{for} \quad i = 1, 2, \ldots, n, \quad (4)$$

where the β_j are the model parameters and ϵ_i represents the residual error for the ith training instance.

However, MLR provides poor quality solutions against state-of-the-art algorithms in many difficult real-world scenarios [53]. In general, the only way to improve the accuracy of the model is to extract better predictive features. Therefore, the evolved transformation m, found by GPTIPS, defines a new dataset $\{\widehat{\mathbf{x}}_i, y_i\}$ where $\mathbf{y}_i = m(\widehat{\mathbf{x}}_i)$ and $\widehat{\mathbf{x}}_i = (\widehat{x}_{i,1}, \ldots, \widehat{x}_{i,d})$. Notice that the jth element $\widehat{x}_{i,j}$ of $\widehat{\mathbf{x}}_i$ is generated by the jth subtree ST_j of m. This new dataset is used to build an MLR model, where model parameters $(\beta_0, \ldots, \beta_d)$ are computed using singular value decomposition in GPTIPS.

The search (genetic) operators in GPTIPS include two types of crossover and various forms of mutation. High-level crossover promotes the interchange of entire subtrees ST_i of the root node between two models, while low-level crossover implements standard sub-tree crossover in GP between any internal nodes from two individuals. Mutations can specifically act on root node subtrees or internal nodes following standard-subtree mutation in GP.

The fitness function is based on the prediction accuracy of the MLR model for each GPTIPS transformation given by the Root Mean Square Error (RMSE), thus posing a minimization problem.[2] During testing, the transformation tree m and the model

[2] While fitness is usually expected to be maximized, in this case, GPTIPS poses a minimization problem but prefer to use the term *fitness* to match common usage in evolutionary and GP literature, another option would be to refer to it as a cost function.

Table 5
GPTIPS configuration.

Hyper-parameter	Value
Population size	250
Number of generations	250
Crossover probability	0.95
Mutation probability	0.05
Selection	Tournament, $n = 20$.
Number of Genes	10
Elite fraction	0.05
Max depth	7
Terminals	Feature vector and random constants
Functions	$+, -, \times, \div, \sqrt{}, sin, cos, tan, cot, sec, csc, arsin, arcos, x^y, e^x, ln$

parameters ($\beta_0, \ldots, \beta_d$) are used to predict the output on unseen data.

Models linear in parameters, such as those generated by GPTIPS, are often preferred in many application domains. It also integrates a bloat control mechanism to push the search towards compact models. Given its relatively efficient implementation, it is used in this work to solve the posed learning task. In the case of the Terminal elements, the feature vector for the naive formulation of the problem includes the reference point; however, since the fronts are normalized, the reference point can be omitted since it always has a coordinate of 1 in all dimensions. Therefore, the constant 1 is explicitly added as a final element in this case.

GPTIPS was configured based on the standard values shown in Table 5 for all the runs reported in this work. Manual hyper-parameter tuning was performed to achieve the best empirical performance. However, it must be stressed that GP algorithms generally tend to be very robust to these control settings [59]. In fact, expert tuning tends to be very competitive compared to automatic methods [60]. Most of the parameters are standard EA or GP settings, with only the *Number of Genes* parameters requiring further clarification. This parameter defines the size of the evolved models by GPTIPS. It is basically the number of terms in the linear regression model; i.e., the number of subtrees, or new features, generated in the CI process by GPTIPS. This value was chosen to keep the models small, concise, and efficient.

While other regression methods might also be applicable, such as regularized regression techniques or stepwise linear regression, it is not the goal of this work to find the best possible modeling technique for the posed problem but to find accurate and efficient models that can approximate the HV indicator. Nonetheless, a series of tests were carried out (not reported in this work) using a variety of ensemble techniques based on decision trees and other ML methods, including neural networks, but model accuracy was always similar or worse than GP, generalization was also similar, but in most cases efficiency was lower, since most other techniques tend to require larger model structures.

4.3. Performance evaluation

The experimental evaluation of the evolved models is carried out in the following manner. First, two subsets of problems are used to train (evolve) the models, and their generalization is evaluated on the remaining problems from each problem set. Second, model evaluation is carried out from the following three perspectives:

- **Model Accuracy.** Based on the RMSE on the training and testing problem instances, this is a standard ML evaluation procedure, comparing the ground truth HV and the estimated HV by the best models found by GP. To complement this, the Pearson's correlation coefficient between the true HV value and the estimated HV is also reported in some cases.
- **Run Time.** The goal is to evaluate the computational cost approximating the HV indicator with the evolved models and contrast them with one offered by the WFG algorithm. In this case, the run time is evaluated from two perspectives: (i) computing the HV for a non-dominated set of solutions; and (ii) computing the HV contribution for each individual in the non-dominated set (step performed within the SMS-EMOA algorithm).
- **Indicator-based Evolution with SMS-EMOA.** A benefit of having a performance indicator such as HV is that it can be used beyond the function of measuring the convergence of a given Pareto set. That is, the HV computation can also be part of an internal mechanism of a MOEA. This with the specific purpose to guide the search towards sets of solutions with good convergence. This study is also interested in how the developed models would behave as search-guiding mechanisms within the SMS-EMOA algorithm. Thus, in the context of this MOEA, we will use the evolved solutions to compute the HV contribution of each individual in the population. The convergence of the algorithm over 30 runs will be analyzed, along with examples of the actual Pareto fronts found by SMS-EMOA implementation and the GP-based version.

5. Experiments and results

The experiments were carried out on a Dell R730 Power Edge Server with 2X Intel Xeon E5-2650 processors and 512 GB RAM running KVM virtual machines over Ubuntu Linux. The GPTIPS 2.0 software was downloaded from https://sites.google.com/site/gptips4matlab, running on MATLAB Version: 9.3.0.713579 (R2017b). To compute the HV indicator,we used the WFG implementation,[3] the SMS-EMOA code was obtained from PlatEMO,[4] with the standard approach using Monte Carlo approximations of the HV. The experimental results are presented for each set of benchmark problems, considering estimation accuracy, run time and indicator-based evolution, as previously stated. An important aspect of this study is that we are paying special attention to the generalization properties of the models. Thus, to generate a model, we train the method on fronts extracted from two problems. For training, we only use pairs of problems from the DTLZ benchmark set. Afterwards, we evaluate the performance of the evolved models on unseen fronts of the same pair of problems, but more importantly, on fronts extracted from problems different from those used for training. In particular, we evaluate the models on the remaining DTLZ problems and on all of the WFG problems. This experimental setting makes the evaluation of the models unbiased but also more challenging since Pareto fronts from different problem sets do not necessarily share the same characteristics, particularly when comparing the DTLZ with the more challenging WFG problems.

For simplicity most of the performance metrics and results are summarized in Appendix B. In what follows we highlight the performance of the best evolved models, presenting their symbolic expression, summarizing their speedup relative to standard computations, and focusing on their generalization when evaluated on the WFG problems and their performance when used to guide an SMS-EMOA search.

[3] www.wfg.csse.uwa.edu.au.

[4] https://github.com/BIMK/PlatEMO/tree/master/PlatEMO.

Table 6
Estimation accuracy of the best evolved models on the DTLZ problems, showing RMSE and Pearson on the training and test problems with the last row showing average performance on the respective training and testing problems for each model. Results show the performance of model $M^3_{4,6}$ for 3-objective problems, $M^4_{5,6}$ for the 4-objective problems and model $M^5_{1,5}$ for the five objective case.

	3 Objectives		4 Objectives		5 Objectives	
	RMSE	Pearson	RMSE	Pearson	RMSE	Pearson
DTLZ1	0.17	0.84	0.19	0.86	0.13	0.94
DTLZ2	0.08	0.93	0.13	0.84	0.23	0.91
DTLZ3	0.15	0.87	0.19	0.88	0.18	0.92
DTLZ4	0.11	0.93	0.12	0.80	0.20	0.71
DTLZ5	0.08	0.90	0.04	0.93	0.05	0.93
DTLZ6	0.07	0.96	0.06	0.96	0.07	0.94
DTLZ7	0.08	0.92	0.07	0.94	0.17	0.84
Training	0.09	0.94	0.05	0.94	0.09	0.93
Testing	0.11	0.89	0.14	0.86	0.17	0.86

Table 7
Ratio $\frac{HV_{GP}}{HV}$ of the true HV values of the final fronts produced on the DTLZ problems. HV for the SMS-EMOA front and HV_{GP} for the front found by SMS-EMOA with the GP model as an indicator.

Problem	3 Objectives	4 Objectives	5 Objectives
DTLZ1	0.9999	1.0000	1.0000
DTLZ2	0.9920	0.9950	0.9839
DTLZ3	0.9999	1.0000	1.0000
DTLZ4	1.0019	0.9637	0.9755
DTLZ5	1.0012	0.9772	0.9941
DTLZ6	0.9989	0.9076	0.8675
DTLZ7	0.9974	0.9954	0.9816
Average	0.9988	0.9770	0.9718

5.1. 3-Objective problems

The detailed (and extended) results analyzing the performance on 21 different models, each evolved using a different pair of DTLZ problems for training, are summarized in Appendix B Table 11. It is noteworthy that most models present a very high correlation coefficient, achieving correlation values. In particular, in all problems there are several models that obtain correlation values above 0.9, which are a good indicator that the models could be used to guide SMS-EMOA. There are, however, some models that perform poorly, so choosing a good model is key to achieve the best possible performance.

For further analysis we choose one high-performing model, model $M^3_{4,6}$, which was obtained by training with DTLZ4, and DTLZ6 and achieved correlation values above 0.90 in most problems, except DTLZ1 and DTLZ3 with correlation values of 0.848 and 0.877 respectively. Performance of the chosen model is summarized in the first row of Table 6, with very high correlation values, in general above 0.9 and very low errors below 0.1.

This high performance model is expressed as

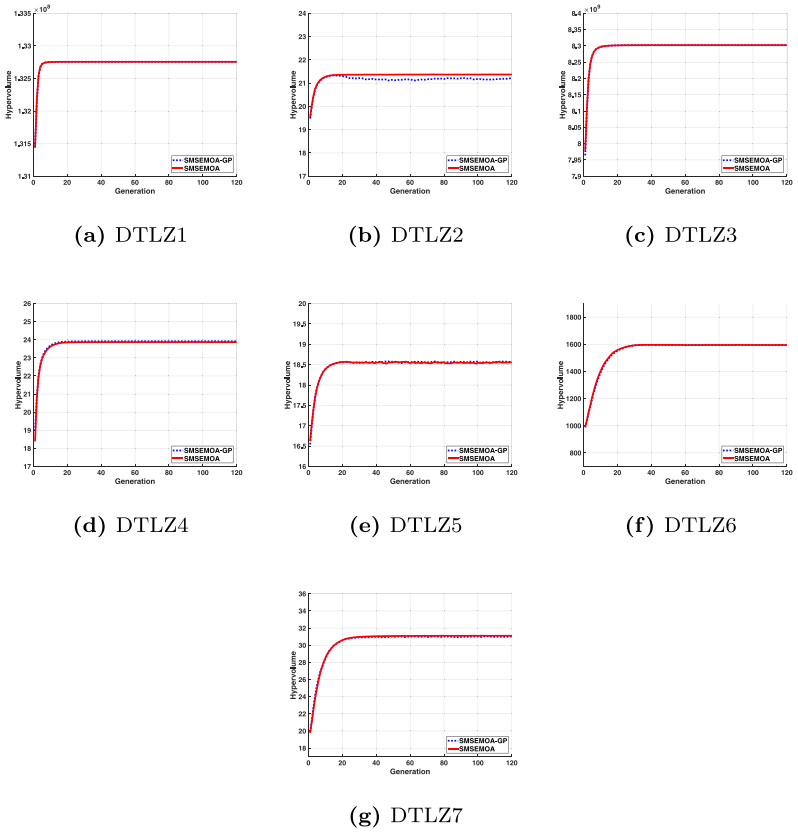
$$\begin{aligned} M^3_{4,6} \quad = \quad & 1.38 \cdot \mu^{(1)} \cdot \mu^{(2)} \cdot Q1^{(1)} - 1.40 \cdot \mu^{(2)} - 0.78 \cdot \mu^{(3)} \\ & -1.17 \cdot \sigma^{(1)} \\ & - 0.24 \cdot Q1^{(1)} - 0.08 \cdot |\gamma^{(3)}| - 0.16 \cdot \log\left(\kappa^{(1)}\right) \\ & -0.32 \cdot \log\left(\sigma^{(3)}\right) \\ & - 0.05 \cdot \log\left(\kappa^{(2)} + \gamma^{(2)} + Q2^{(2)} \cdot \kappa^{(3)}\right) \\ & -0.08 \cdot Q2^{(3)} \cdot \gamma^{(1)} \\ & - 0.18 \cdot Q2^{(3)} \cdot \gamma^{(2)} - 1.34 \cdot \mu^{(1)} \\ & +0.26 \cdot Q2^{(3)} \cdot Q1^{(1)} \cdot \gamma^{(2)} + 2.50 \ . \end{aligned} \tag{5}$$

There are several interesting aspects of model $M^3_{4,6}$. First, that all three objectives are represented by several statistical descriptors. Second, most descriptors are parametric measures, i.e., kurtosis, mean and standard deviation, although non-parametric descriptors are also used. Finally, the model includes several non-linear terms, but most terms are in fact linear.

To evaluate generalization, we further compare the performance of the above model on each of the DTLZ and WFG problems, relative to the ground truth HV values HV_{GT}. The experiments where repeated with different population sizes, $N = \{100, 300, 600, 1000, 2000\}$, using 30 runs on each problem and 120 generations of the SMS-EMOA. The goal here is to evaluate how the models behave using different population sizes, and thus different sizes of the final Pareto front. This comparison is done based absolute error given as $AE = |HV_{approx} - HV_{GT}|$ between the exact or ground truth HV HV_{GT} and the approximation by the GP model HV_{approx}, from which we compute the relative error given by $RE = \frac{AE}{HV_{GT}}$ which is in the range [0, 1]. Detailed results are presented in Table 12 of Appendix B, using the average over all runs for each problem and showing the average over all problems with different population sizes. In all cases the relative error is always very small, below 0.1 in all cases. Moreover, performance seems to be very stable, across different population sizes and different benchmark problems, including all of the testing DTLZ and WFG problems.

The next set of results evaluates model $M^3_{4,6}$ as a search guiding mechanism within the algorithm SMS-EMOA. To do this experiment, we run the algorithm SMS-EMOA in two versions: (1) the standard PlatEMO version and (2) where we use $M^3_{4,6}$ as a search guiding mechanism to compute HV contributions. Results are presented as the average of 30 runs. There are some implementation details to consider. First, given that, in some cases, the GP models produced undefined floating-point results, the algorithm checks for such a condition and updates the value of the HV estimation to 0 in such cases. Similarly, when computing the HV contributions of each individual in the Pareto front, when the contribution is below 0.0015 according to the evolved model, it was also rounded to zero.

Figs. 1 and 2 compare the convergence over each generation of each version of SMS-EMOA, showing the true HV at each generation of both versions, for the DTLZ and WFG problems respectively. It is clear that the convergence on all problems is basically equivalent, with only slight differences in some problems. Figs. 3 and 4 show the Pareto front found in all problems by each version of the SMS-EMOA, taking a single representative run in each case. The similarity in the shape and spread of solutions of the fronts is high. If we compare the true HV values of the final fronts found by each algorithm, HV for the SMS-EMOA front and HV_{GP} for the front found by SMS-EMOA with the GP model, we can compute the ratio $\frac{HV_{GP}}{HV}$ to evaluate the relative performance of each, with values below 1 indicating that there was a performance dropoff using the GP model or vice versa. These values are summarized in Table 7 for the DTLZ problems and Table 8 for the WFG problems; in particular results for the 3-objective problems are presented in the first column of these tables. The last row of these columns show the average ratio on all problems. We can see that on average the ratio is very close to 1 on both sets of problems, which is a very good result considering that training was carried out solely on a pair of DTLZ problems while most of the remaining problems are only used for testing.



(a) DTLZ1 (b) DTLZ2 (c) DTLZ3

(d) DTLZ4 (e) DTLZ5 (f) DTLZ6

(g) DTLZ7

Fig. 1. Convergence comparison of the standard SMS-EMOA and the GP model $M^3_{4,6}$ on the 3-objective WFG problems; problems DTLZ4 and DTLZ6 were used to train the model.

Table 8
Ratio $\frac{HV_{GP}}{HV}$ of the true HV values of the final fronts on the WFG problems. HV for the SMS-EMOA front and HV_{GP} for the front found by SMS-EMOA with the GP model as an indicator.

Problem	3 Objectives	4 Objectives	5 Objectives
WFG1	0.9556	0.9654	0.8261
WFG2	1.0027	0.9951	0.9845
WFG3	0.9984	0.9698	0.8623
WFG4	0.9897	0.9037	0.8152
WFG5	0.9746	0.9251	0.8113
WFG6	0.9846	0.9129	0.7772
WFG7	0.9808	0.8629	0.7552
WFG8	0.9695	0.8445	0.7597
WFG9	0.9743	0.9244	0.8081
Average	0.9812	0.9226	0.8222

A run time evaluation of the chosen model was also performed. First, we perform run time analysis of our evolved models on the DTLZ problems compared to the exact computation of the HV. The comparison is carried out at two levels: when computing a single HV from a Pareto front and also when calculating the HV contribution of each individual in the non-dominated set (the latter is an important step in the selection mechanism of SMS-EMOA). Note that for this experiment we use the WFG algorithm[5] to compute the HV and contrast with the evolved model. These results are averaged over 30 runs. Table 13 in Appendix B presents a detailed run time evaluation of $M^3_{4,6}$ on 3 objective problems using different population sizes.

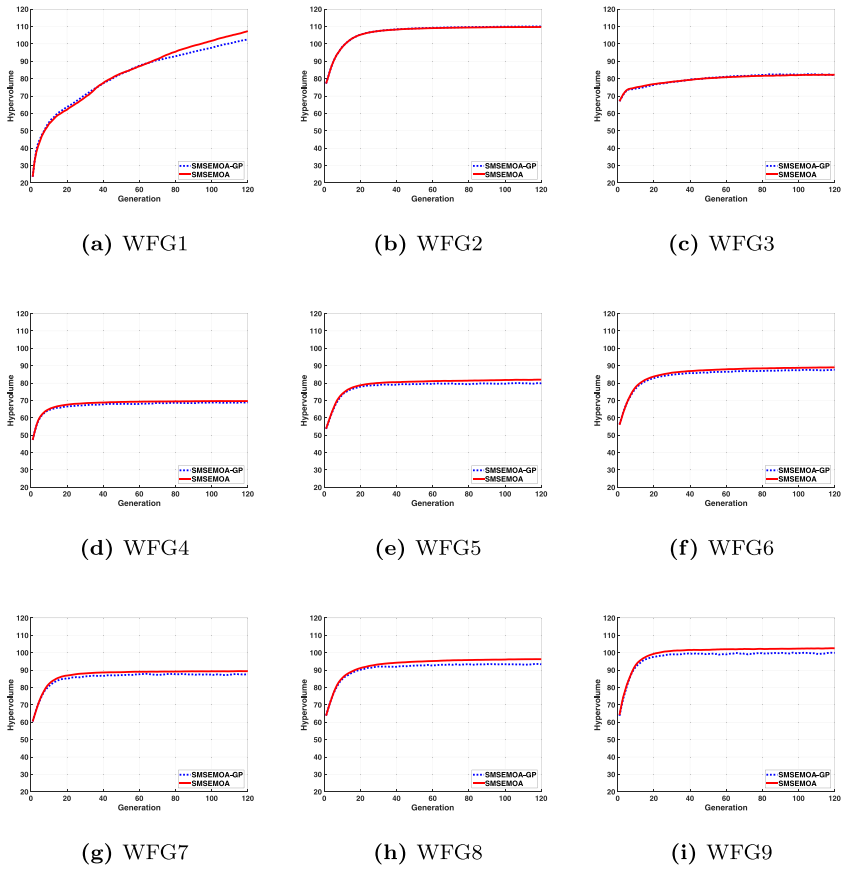
5 www.wfg.csse.uwa.edu.au.

In these tests, it is clear that the evolved models vastly outperform the exact method, reaching speedups from one order of magnitude up to 1000X when computing all of the HV contributions, due to the relative simplicity of the feature extraction process and model composition. A summary of the speedups for HV computation for each problem are shown in Table 9 considering a population of $N = 300$, contrasting the time (seconds) required by the WFG algorithm and the evolved model on a single front. The first column in Table 9 shows results on the 3-objective case. We can see that as the number of objectives increases, the speedup also improves, reaching two orders of magnitude improvement on these tests.

Additionally, a run time comparison between SMS-EMOA guided by the GP model and the standard PlatEMO implementation that uses a Monte Carlo approximation of the HV is presented in Table 10 for both problem sets (DTLZ and WFG). The first column of Table 10 shows performance on the 3-objective case. Averages are given in the final row of each table. Speedups are high on all test problems, from both DTLZ and WFG suites, providing basically on order of magnitude improvement.

5.2. 4-objective and 5-objective problems

For the 4-objective and 5-objective problems, a similar strategy is used to evolve, evaluate and compare the evolved models. First, after comparing the 21 different models generated by using different training pairs of DTLZ problems, the following models were chosen. For the 4-objective case it is model $M^4_{5,6}$ which is expressed as



(a) WFG1 (b) WFG2 (c) WFG3

(d) WFG4 (e) WFG5 (f) WFG6

(g) WFG7 (h) WFG8 (i) WFG9

Fig. 2. Convergence comparison of the standard SMS-EMOA and the GP model $M^3_{4,6}$ on the 3-objective WFG problems; problems DTLZ4 and DTLZ6 were used to train the model.

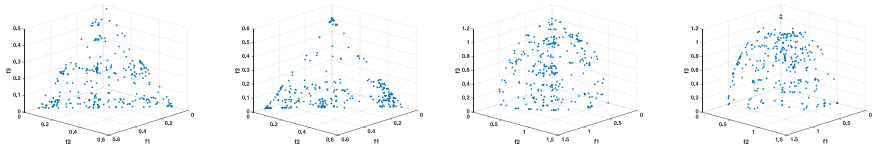
Table 9
Run time of the HV computation for a single front achieved by the evolved GP models relative to the WFG algorithm, applied on the DTLZ problems.

	3 Objectives			4 Objectives			5 Objectives		
	WFG	GP	Speedup	WFG	GP	Speedup	WFG	GP	Speedup
DTLZ1	0.0009	7.4E−05	12.68	0.0018	9.1E−05	20.47	0.006	9.7E−05	68.10
DTLZ2	0.0010	7.8E−05	13.49	0.0027	1.2E−04	21.99	0.009	9.6E−05	97.96
DTLZ3	0.0009	5.8E−05	16.54	0.0008	7.8E−05	10.19	0.001	1.0E−04	16.13
DTLZ4	0.0011	8.5E−05	13.02	0.0020	9.0E−05	22.65	0.004	9.6E−05	47.32
DTLZ5	0.0010	7.7E−05	13.30	0.0016	9.4E−05	17.13	0.002	9.0E−05	31.68
DTLZ6	0.0009	8.1E−05	11.76	0.0021	1.3E−04	15.68	0.007	8.1E−05	89.96
DTLZ7	0.0010	8.8E−05	11.61	0.0019	9.4E−05	20.76	0.004	9.1E−05	52.73
Average	0.0010	7.8E−05	13.20	0.0180	1.0E−04	17.47	0.005	9.4E−05	57.70

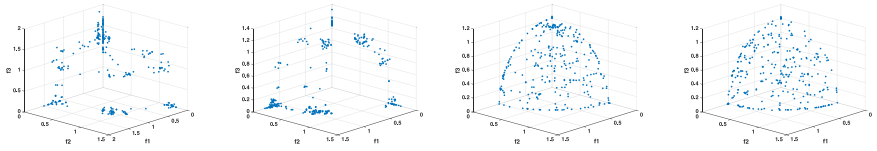
$$
\begin{aligned}
M^4_{5,6} = {} & 217.04 \cdot \sigma^{(3)\frac{9}{2}} - 0.9290 \cdot Q1^{(4)} \\
& + 4.16 \cdot \sin\left(\sin\left(\mu^{(2)} \cdot \sigma^{(4)}\right)\right) \\
& - 1.07 \cdot \sin\left(\frac{\sigma^{(2)^2} \cdot \gamma^{(2)}}{\cos\left(Q1^{(3)}\right)}\right) - 48.14 \cdot \exp\left(\exp\left(\sigma^{(3)^3}\right)\right) \\
& - 0.48 \cdot Q1^{(3)} + \frac{68.83}{\cos\left(Q1^{(3)}\right)} + 0.3350 \cdot \cot\left(\sigma^{(4)\frac{1}{4}}\right) \\
& - 1.45 \cdot \mu^{(1)} \cdot \sigma^{(3)} \\
& - 0.20 \cdot Q1^{(2)} \cdot \cos\left(Q1^{(2)}\right) \cdot \sin\left(\gamma^{(4)}\right) + 62.82 \, . \qquad (6)
\end{aligned}
$$

For the 5-objective case it is model $M^4_{1,5}$, expressed as

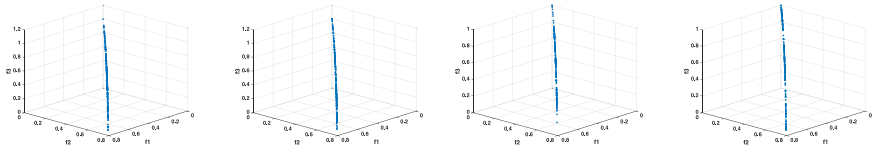
$$
\begin{aligned}
M^5_{1,5} = {} & 0.40 \cdot \mu^{(4)} - 0.34 \cdot \mu^{(1)} + 0.56 \cdot Q2^{(2)} + 0.23 \cdot Q2^{(3)} \\
& + 0.46 \cdot \sigma^{(2)} - 15.70 \cdot \sigma^{(3)} \\
& + 0.05 \cdot Q1^{(1)} + 0.72 \cdot Q1^{(2)} \\
& + 3.67 \cdot \arcsin\left(\arcsin\left(Q2^{(j)} \cdot \sigma^{(3)}\right)\right) \\
& - 14.61 \cdot \arcsin\left(\sigma^{(3)^2}\right) + 10.29 \cdot \arcsin\left(\tan\left(\sigma^{(3)}\right)\right) \\
& + 0.11 \cdot \cos\left(Q2^{(2)} + Q2^{(3)} - Q1^{(5)} + Q3^{(4)}\right. \\
& \left. + Q3^{(5)} + \log\left(\kappa^{(3)}\right)\right) \\
& + 1.14 \cdot \cos\left(\mu^{(3)} \cdot \mu^{(4)} \cdot Q1^{(5)} \cdot Q3^{(1)} + |\sigma^{(2)}|\right) \\
& - 0.17 \cdot \tan\left(Q2^{(j)}\right) \\
& + 10.29 \cdot \tan\left(\sigma^{(3)}\right) - \frac{0.17}{\cos\left(Q3^{(5)}\right)} + 0.23 \cdot Q3^{(1)} \cdot Q3^{(4)} \\
& - 0.17 \cdot Q1^{(2)} \cdot \gamma^{(5)} \\
& - 5.22 \cdot Q2^{(7)} \cdot \sin\left(\sigma^{(5)}\right) + 0.27 \, . \qquad (7)
\end{aligned}
$$



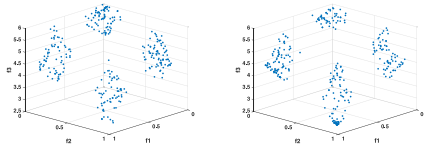
(a) DTLZ1: SMS-EMOA and GP (b) DTLZ2: SMS-EMOA and GP



(c) DTLZ3: SMS-EMOA and GP (d) DTLZ4: SMS-EMOA and GP



(e) DTLZ5: SMS-EMOA and GP (f) DTLZ6: SMS-EMOA and GP



(g) DTLZ7: SMS-EMOA and GP

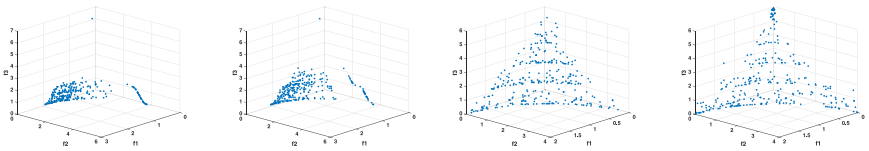
Fig. 3. Sample Pareto Front approximations found by SMS-EMOA (left) and the GP model $M^3_{4,6}$ (right) for the 3-objective DTLZ problems; problems DTLZ4 and DTLZ6 were used to train the model.

Table 10
Run time comparison of the standard PlatEMO SMS-EMOA implementation, with Monte Carlo approximations of the HV, and SMS-EMOA using the GP model to compute HV contributions on both problem sets. Values are given in seconds, and SU is the speedup.

	3 Objectives			4 Objectives			5 Objectives		
	SMS-EMOA	GP	SU	SMS-EMOA	GP	SU	SMS-EMOA	GP	SU
DTLZ1	6.8e+02	5.2e+01	12.9	9.3e+02	8.5e+01	10.9	1.3e+03	1.1e+02	11.59
DTLZ2	1.6e+03	1.1e+02	13.5	1.7e+03	1.7e+02	10.0	2.3e+03	2.6e+02	8.63
DTLZ3	1.6e+02	3.5e+01	4.47	2.4e+02	4.5e+01	5.47	4.2e+02	6.3e+01	6.63
DTLZ4	1.4e+03	1.1e+02	13.0	1.5e+03	1.4e+02	10.6	2.1e+03	2.0e+02	10.24
DTLZ5	7.9e+02	8.3e+01	9.56	1.5e+03	1.9e+02	7.64	1.8e+03	2.7e+02	6.81
DTLZ6	8.0e+02	7.5e+01	10.6	2.5e+03	1.8e+02	13.8	2.9e+03	2.4e+02	12.0
DTLZ7	1.3e+03	1.0e+02	12.7	1.4e+03	1.4e+02	9.56	1.7e+03	2.2e+02	8.03
WFG1	6.6e+02	6.0e+01	10.9	1.4e+03	1.0e+02	14.8	2.2e+03	1.7e+02	12.83
WFG2	1.4e+03	8.0e+01	17.5	2.2e+03	1.2e+02	17.6	3.0e+03	2.0e+02	15.04
WFG3	3.2e+03	1.4e+02	22.4	2.9e+03	2.2e+02	13.0	3.1e+03	3.1e+02	10.12
WFG4	1.6e+03	1.2e+02	13.3	2.3e+03	1.9e+02	12.2	3.0e+03	2.7e+02	11.03
WFG5	1.9e+03	1.2e+02	16.0	2.2e+03	1.8e+02	12.3	2.9e+03	2.6e+02	10.98
WFG6	1.4e+03	9.3e+01	15.5	1.7e+03	1.5e+02	11.3	2.1e+03	2.3e+02	9.22
WFG7	2.4e+03	1.3e+02	17.5	2.0e+03	2.0e+02	10.2	2.7e+03	2.9e+02	9.17
WFG8	1.3e+03	8.5e+01	16.2	1.6e+03	1.6e+02	10.1	2.0e+03	2.5e+02	8.07
WFG9	2.6e+03	1.5e+02	16.6	2.4e+03	2.0e+02	11.6	2.7e+03	3.1e+02	8.67
Average	1.4e+03	9.9e+01	13.95	1.8e+03	1.5e+02	11.35	2.3e+03	2.3e+02	9.94

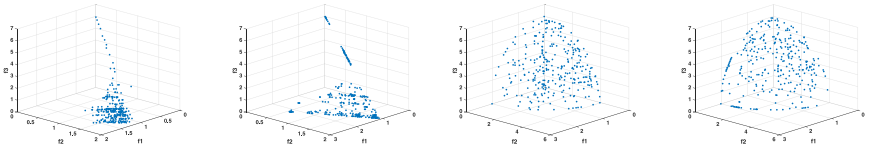
As was the case for the 3-objective case, both models include descriptors from all of the objectives, and while they basically use all of the available descriptors the models do prefer the parametric features. Also, we can see that as the number of objectives increased the use of non-linear terms becomes more prevalent, probably due to the increased complexity in the problem. .

Performance of these models is summarized in the same tables presented before, namely Tables 6–10. First, regarding the RMSE and Pearson metrics, performance of the chosen model are mostly equivalent to the 3-objective case, with relatively high correlation (Pearson above 0.9) and a low RMSE on most problems. Indeed, it is notable that performance of all three models is almost identical, showing good robustness in the learning process. Second,



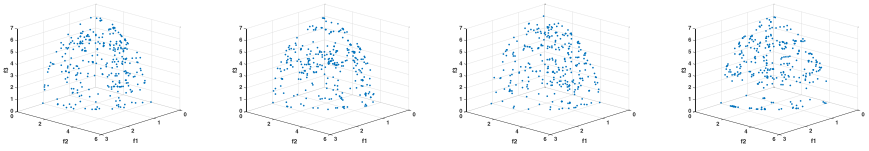
(a) WFG1: SMS-EMOA and GP

(b) WFG2: SMS-EMOA and GP



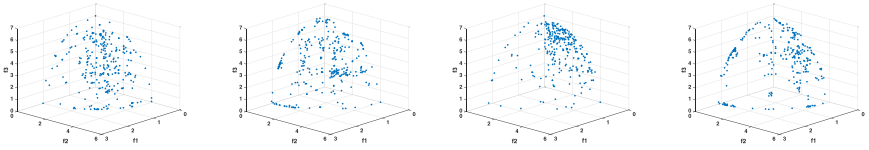
(c) WFG3: SMS-EMOA and GP

(d) WFG4: SMS-EMOA and GP



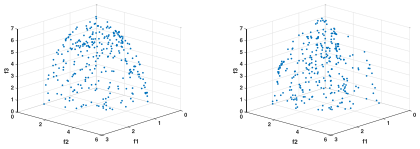
(e) WFG5: SMS-EMOA and GP

(f) WFG6: SMS-EMOA and GP



(g) WFG7: SMS-EMOA and GP

(h) WFG8: SMS-EMOA and GP



(i) WFG9: SMS-EMOA and GP

Fig. 4. Sample Pareto Front approximations found by SMS-EMOA (left) and the GP model $M^3_{4,6}$ (right) for the 3-objective WFG problems; problems DTLZ4 and DTLZ6 were used to train the model.

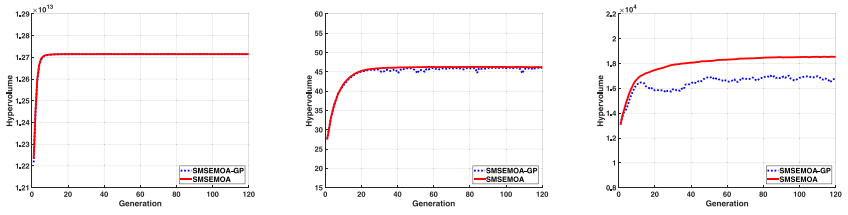
regarding the performance of the models when used to guide SMS-EMOA, Tables 7 and 8 also show good results. On the DTLZ problems, Table 7, performance is close to 1 on all problems except one, DTLZ6, which seems to be the most difficult case. In all other cases, for bot 4 and five objectives, the ratio is between 0.96 and 1. Fig. 5, shows the average convergence plots on some example DTLZ problems for 4 and 5 objectives, namely DTLZ7 and DTLZ3, two test problems for both models where performance is high, and DTLZ6 the most difficult case.

Conversely, on the WFG there is a larger performance variance, both among problems and between both sets of problems (4 or 5 objectives). For 4 objectives, performance is still relatively high, with ratio values close to 1 on several problems and an average of 0.922. However, for 5 objective problems performance is notably lower, with an average of 0.82, a maximum of 0.98 of WFG2 and a minimum of 0.75 on both WFG7 and WFG8. Fig. 6 presents some sample convergence plots, using WFG2, WFG3 and WFG8 as examples. On WFG2, performance is similar for all cases shown in Table 8, independent of the number of objectives. On the other hand, in WFG3 performance is similar for 3 and 4 objective versions of the problem, but drops off for the 5-objective version. Lastly, performance on WFG8 steadily, and almost proportionally, declines as the number of objectives increases.

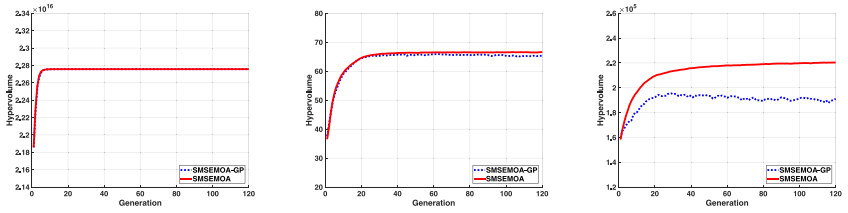
Regarding execution times, Table 9 presents the results for HV computation. It as clear that as the number of objectives increases, so does the relative speedup. Additionally, the total runtime comparison between the standard SMS-EMOA and one that uses the evolved GP models is summarized in Table 10. In this case, we can see that the speedup achieved is on average 10X in all problems, independent of the number of objectives or the benchmark suite.

6. Conclusions and future work

In this work, we have presented a new methodology that allows approximating the Hypervolume (HV) values for MOPs. In particular, we have, for the first time in related literature, a supervised learning problem and used GP to evolve solutions for it. We have confirmed the reliability of our models through

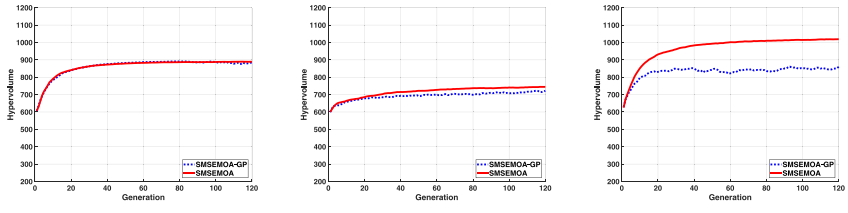


(a) DTLZ3 (b) DTLZ7 (c) DTLZ6

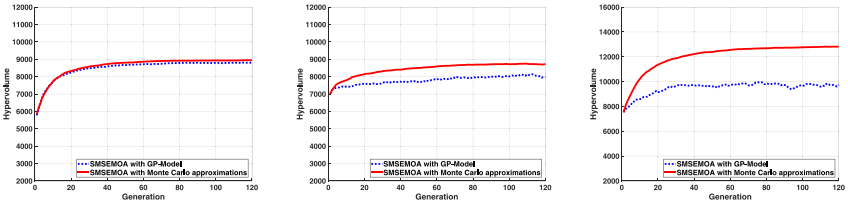


(d) DTLZ3 (e) DTLZ7 (f) DTLZ6

Fig. 5. Convergence comparison of the standard SMS-EMOA and the GP models for 4 (top row) and 5 (bottom row) objective DTLZ problems.



(a) WFG2 (b) WFG3 (c) WFG8



(d) WFG2 (e) WFG3 (f) WFG8

Fig. 6. Convergence comparison of the standard SMS-EMOA and the GP models for 4 (top row) and 5 (bottom row) objective WFG problems.

a comprehensive set of experimental evaluations. Numerical results show that our models approximate the real HV value of the selected benchmark problems, DTLZ and WFG, with great accuracy but require significantly less time than exact methods. These resulting models are hence of great use for HV-based MOEAs. Here, we have provided particular models for standard benchmark suites considering 3, 4 and 5-objective vases. The approach, however, is, in principle, applicable to any set of test problems. We stress that the approach can also be extended to other performance indicators due to GP's flexibility. Another significant contribution of this work is providing an efficient way to compute statistical moments for a particular case, which contributed to achieving a substantial reduction in the execution time of the SMS-EMOA. We expect that the proposed models can be used by the EMO community to reduce the time to tackle similar problems, especially in the development phase, as well as motivate GP's use for the construction of new alternatives to measure the behavior of MOEAs.

The evolved models are quite general, since they have been evolved using only a pair of problems for training, and have been tested on 14 different problems, from the same benchmark (DTLZ) and a more complex one (WFG). Performance generalized quite well, particularly for 3 and 4 objective problems, across all problems, which are of varying degrees of complexity. This is particularly the case when the models are used to guide an indicator based MOEA, in our case SMS-EMOA. We have shown that our evolved models that approximate the HV indicator can be used to accurately and efficiently approximate the HV contributions and guide the SMS-EMOA search, reducing the search time by a one orders of magnitude in most cases. However, performance is not as robust when dealing with 5-objective problems. While performance was comparable to the standard SMS-EMOA on the DTLZ problems, it was notably reduced on the more complex WFG benchmark problems. We hypothesized that this can be resolved by incorporating more complex problem in the training phase of the models, or other alternatives as described below.

There are several possible paths for future work. First, we conjecture that the reported execution time can be reduced even more as the HV contributions' computation using our formulas is highly parallelizable. Further, a more detailed analysis of

Table 11
Estimation accuracy of the evolved models on the DTLZ 3-objective benchmark problems. For each row the training pair of problems is different, while the final row summarizes the training and testing performance.

No.	Training problems	Problems													
		DTLZ1		DTLZ2		DTLZ3		DTLZ4		DTLZ5		DTLZ6		DTLZ7	
		RMSE	Pearson	RMSE	Pearson	RMSE	Pearson	RMSE	Pearson	RMSE	Pearson	RMSE	Pearson	RMSE	Pearson
1	DTLZ1-DTLZ2	0.120	0.924	0.071	0.947	0.135	0.908	0.139	0.902	0.484	0.251	0.329	0.422	0.147	0.854
2	DTLZ1-DTLZ3	0.121	0.922	0.117	0.911	0.125	0.921	0.157	0.875	0.168	0.899	0.172	0.914	0.136	0.919
3	DTLZ1-DTLZ4	0.120	0.924	0.076	0.951	0.134	0.909	0.103	0.945	0.141	0.826	0.190	0.855	0.107	0.926
4	DTLZ1-DTLZ5	0.125	0.919	0.178	0.791	0.139	0.906	0.197	0.787	0.066	0.937	0.099	0.935	0.116	0.910
5	DTLZ1-DTLZ6	0.126	0.916	0.132	0.871	0.144	0.903	0.167	0.848	0.088	0.893	0.081	0.955	0.099	0.939
6	DTLZ1-DTLZ7	0.124	0.918	0.116	0.903	0.131	0.913	0.145	0.888	0.152	0.808	0.143	0.892	0.081	0.953
7	DTLZ2-DTLZ3	0.371	0.651	0.067	0.953	0.121	0.925	0.862	0.489	0.336	0.461	0.177	0.823	0.110	0.916
8	DTLZ2-DTLZ4	0.321	0.608	0.066	0.954	0.173	0.868	0.100	0.948	0.240	0.647	0.190	0.765	0.097	0.932
9	DTLZ2-DTLZ5	0.712	0.146	0.071	0.949	0.287	0.675	0.180	0.846	0.067	0.934	0.127	0.894	0.152	0.892
10	DTLZ2-DTLZ6	0.556	0.408	0.066	0.956	0.328	0.623	0.421	0.762	0.090	0.913	0.070	0.966	0.091	0.952
11	DTLZ2-DTLZ7	0.492	0.372	0.059	0.964	0.212	0.809	0.228	0.748	0.203	0.753	0.135	0.901	0.075	0.960
12	DTLZ3-DTLZ4	0.181	0.850	0.085	0.942	0.127	0.918	0.110	0.937	0.158	0.846	0.199	0.871	0.099	0.936
13	DTLZ3-DTLZ5	0.129	0.913	0.168	0.854	0.126	0.919	0.161	0.868	0.062	0.940	0.082	0.955	0.117	0.926
14	DTLZ3-DTLZ6	0.132	0.912	0.121	0.906	0.127	0.918	0.175	0.860	0.086	0.897	0.076	0.961	0.105	0.934
15	DTLZ3-DTLZ7	0.137	0.903	0.071	0.948	0.126	0.919	0.141	0.898	0.181	0.802	0.140	0.904	0.082	0.952
16	DTLZ4-DTLZ5	0.219	0.776	0.102	0.908	0.208	0.800	0.111	0.937	0.072	0.932	0.094	0.944	0.106	0.931
17	DTLZ4-DTLZ6	0.177	0.848	0.084	0.934	0.156	0.877	0.114	0.933	0.087	0.908	0.074	0.963	0.085	0.927
18	DTLZ4-DTLZ7	0.157	0.876	0.068	0.955	0.152	0.887	0.106	0.942	0.233	0.698	0.151	0.904	0.076	0.959
19	DTLZ5-DTLZ6	0.432	0.687	0.132	0.890	0.253	0.821	0.385	0.756	0.060	0.946	0.071	0.965	0.105	0.936
20	DTLZ5-DTLZ7	0.145	0.846	0.143	0.881	0.150	0.882	0.171	0.847	0.060	0.946	0.075	0.957	0.078	0.957
21	DTLZ6-DTLZ7	0.567	0.314	0.110	0.916	0.285	0.641	0.347	0.473	0.072	0.926	0.074	0.963	0.078	0.957
	Training	0.123	0.920	0.067	0.954	0.125	0.920	0.107	0.940	0.137	0.891	0.075	0.961	0.090	0.947
	Testing	0.315	0.674	0.113	0.904	0.192	0.828	0.211	0.839	0.181	0.769	0.154	0.862	0.111	0.922

Table 12
Performance comparison, given as relative error, of the Pareto front approximation of the evolved GP model $M^3_{4,6}$ compared with the true HV value, using different number of points N in the Pareto front.

	N = 100	N = 300	N = 600	N = 1000	N = 2000
DTLZ1	0.0105	0.0117	0.0187	0.0369	0.0439
DTLZ2	0.0242	0.0497	0.0599	0.0696	0.0789
DTLZ3	0.0093	0.0107	0.0156	0.0336	0.0487
DTLZ4	0.0163	0.0307	0.0441	0.0582	0.0597
DTLZ5	0.0314	0.0374	0.0620	0.0756	0.0922
DTLZ6	0.0102	0.0149	0.0214	0.0380	0.0509
DTLZ7	0.0010	0.0020	0.0049	0.0073	0.0327
WFG1	0.0023	0.0089	0.0204	0.0403	0.0502
WFG2	0.0083	0.0196	0.0214	0.0304	0.0536
WFG3	0.0013	0.0115	0.0263	0.0457	0.0702
WFG4	0.0231	0.0787	0.1026	0.1019	0.1040
WFG5	0.0248	0.0484	0.0707	0.0795	0.0898
WFG6	0.0529	0.0548	0.0791	0.0894	0.1028
WFG7	0.0212	0.0411	0.0780	0.0805	0.0952
WFG8	0.0336	0.0124	0.0536	0.0793	0.0922
WFG9	0.0609	0.0501	0.0634	0.0676	0.0852
Average	0.0207	0.0301	0.0463	0.0583	0.0718

the our approach's performance against different state-of-the-art MOEAs is also left for future work. The proposed approach can of course be used to generate new models based any benchmark suite or even using real-world case studies, and alternative ways can be developed to build these models, such as using different descriptors for the Pareto fronts or using alternative learning paradigms. Another aspect is that our approach produces models that do not scale to different numbers of objectives; i.e., we cannot use a model trained on 3-objective problems on a 5-objective problem. To achieve this a different learning strategy is required, since in our work the number of features increases proportionally with the number of problem objectives. Finally, focusing on approximating the HV might be set aside altogether, and instead use a learning method to generate new Pareto front indicators that can be used to both evaluate Pareto front approximations and guide an indicator-based MOEA. The data for existing models and test problems can be found in https://github.com/NumericalEvolutionaryOptimization/HVApproximations.

CRediT authorship contribution statement

Cristian Sandoval: Methodology, Software, Validation, Investigation, Data curation, Visualization. **Oliver Cuate:** Formal analysis, Investigation, Writing – review & editing, Methodology. **Luis C. González:** Conceptualization, Writing – original draft, Writing – review & editing, Supervision, Project administration, Funding acquisition. **Leonardo Trujillo:** Conceptualization, Methodology, Resources, Writing – original draft, Writing – review & editing, Supervision, Project administration, Funding acquisition. **Oliver Schütze:** Conceptualization, Formal analysis, Writing – original draft, Supervision, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The first author was supported by CONACYT (Mexico) doctoral scholarship with CVU number 789493. Oliver Cuate acknowledges Instituto Politécnico Nacional and funding from project SIP 20221947.

Appendix A

In the following, we describe the update strategies we have used to speed up the computations of the Hypervolume contributions.

Let $A := \{x_1, \ldots, x_N\} \subset \Re^n$ be an archive of size N, and let $F_A := F(A) = \{y_1, \ldots, y_N\} \subset \Re^k$ be the image of A (i.e., $y_i = F(x_i)$). We are interested in efficiently computing the first four statistical moments (mean, variance, kurtosis, and skewness) for all the subsets F_{Ai}, $i = 1, \ldots, N$, where $F_{Ai} \subset F_A$ denotes the images of the sub-populations $A_i \subset A$, $i = 1, \ldots, N$ where the ith individual is removed,

$$F_{Ai} = \{y_1, y_2, \ldots, y_{i-1}, y_{i+1}, \ldots, y_N\}.$$

Table 13
Run time analysis of the evolved model on the 3 objective DTLZ benchmarks, using $M^3_{4,6}$.

	N = 100		N = 300		N = 600		N = 1000		N = 2000	
	Time (s)	Speedup	Time (s)	Speedup	Time (s)	Speedup	Time (s)	Speedup	Time (s)	Speedup
DTLZ1										
GP-Model	2.8733e−0.5	–	7.4267e−05	–	0.00015	–	0.00023	–	0.00048	–
WFG	0.00036	**12.819**	0.00094	**12.682**	0.00285	**17.995**	0.00663	**27.892**	0.024732	**51.046**
GP-Model contributions	0.00152	–	0.00049	–	0.00102	–	0.00143	–	0.00329	–
WFG contributions	0.01127	**7.38**	0.21361	**433.59**	1.4918	**1462.21**	6.5862	**4591.95**	51.3467	**15 584.63**
DTLZ2										
GP-Model	3.25E−05	–	7.80E−05	–	0.00014	–	0.00021	–	0.00049	–
WFG	0.00033	**10.33**	0.00105	**13.49**	0.00313	**22.34**	0.00750	**34.78**	0.02768	**56.09**
GP-Model contributions	0.00025	–	0.00053	–	0.00092	–	0.00137	–	0.00376	–
WFG contributions	0.01119	**43.07**	0.2225	**418.90**	1.5502	**1683.48**	6.79606	**4948.09**	53.3537	**14 183.50**
DTLZ3										
GP-Model	2.45E−05	–	5.98E−05	–	0.00010	–	0.00015	–	0.00044	–
WFG	0.00033	**13.74**	0.00099	**16.54**	0.00258	**24.35**	0.00644	**41.72**	0.02587	**57.55**
GP-Model contributions	0.00023	–	0.00055	–	0.00097	–	0.00111	–	0.00321	–
WFG contributions	0.01145	**49.50**	0.20848	**376.75**	1.53	**1562.14**	6.2444	**5615.08**	49.1003	**15 253.26**
DTLZ4										
GP-Model	2.61E−05	–	8.85E−05	–	0.00013	–	0.00022	–	0.00038	–
WFG	0.00042	**16.10**	0.00115	**13.02**	0.00317	**22.91**	0.00840	**36.75**	0.03138	**82.38**
GP-Model contributions	0.00021	–	0.00066	–	0.00094	–	0.00146	–	0.00260	–
WFG contributions	0.01289	**60.38**	0.2228	**336.34**	1.6344	**1728.36**	7.8537	**5366.55**	59.35976	**22 777.82**
DTLZ5										
GP-Model	3.27E−05	–	7.87E−05	–	0.00014	–	0.00024	–	0.00047	–
WFG	0.00035	**10.94**	0.001046	**13.30**	0.002718	**19.025**	0.007113	**29.37**	0.026915	**57.12**
GP-Model contributions	0.00024	–	0.000537	–	0.00093	–	0.00151	–	0.00274	–
WFG contributions	0.01136	**46.59**	0.21763	**404.86**	1.6917	**1816.28**	7.32541	**4841.97**	54.6090	**19 917.26**
DTLZ6										
GP-Model	2.83E−05	–	8.71E−05	–	0.00015	–	0.00023	–	0.00047	–
WFG	0.00043	**13.61**	0.00098	**11.76**	0.00302	**18.58**	0.00689	**30.42**	0.02668	**55.64**
GP-Model contributions	0.00025	–	0.00053	–	0.00097	–	0.00138	–	0.00283	–
WFG contributions	0.01189	**47.43**	0.21529	**404.97**	1.5153	**1560.96**	6.8322	**4944.04**	55.4811	**19 574.58**
DTLZ7										
GP-Model	3.21E−05	–	08.38E−05	–	0.00016	–	0.00022	–	0.00047	–
WFG	0.00041	**14.61**	0.00101	**11.61**	0.00285	**18.77**	0.006892	**28.89**	0.02553	**54.14**
GP-Model contributions	0.00022	–	0.00056	–	0.00097	–	0.00146	–	0.00272	–
WFG contributions	0.01237	**54.97**	0.23472	**413.17**	1.4653	**1501.62**	6.6333	**4524.87**	51.0786	**18 753.62**

In the same way, we desire to obtain the order moments related to the quartiles 10, 25, 50 (the median), and 75. For sake of simplification, denote by $T \in \Re^N$ the set of values of the jth objective in the archive F_A, and analog T_i.

The computation of the mean is an easy task. Since

$$\bar{T} = \frac{1}{N}\sum_{j=1}^{N} t_j \tag{8}$$

it follows that

$$\bar{T}_i = \frac{1}{N-1}\sum_{\substack{j=1\\ j\neq i}}^{N} t_j = \frac{N\bar{T} - t_i}{N-1}. \tag{9}$$

That is, instead of making the sum of $N-1$ elements for *each* T_i (the naive way), we first compute the sum of all the N elements, and then we subtract the corresponding value t_i. This represents a saving in the number of sums from $N+N(N-1) = N+N^2-N = N^2$ to $N+N = 2N$. That is, using the Landau symbol, a reduction of the complexity of the procedure from $O(N^2)$ down to $O(N)$.

For the variance, we have the following for each set T_i:

$$\sigma^2 = \frac{1}{N-2}\sum_{\substack{j=1\\ j\neq i}}^{N-1} \left(t_j - \bar{T}_i\right)^2. \tag{10}$$

Here, every term of the sum depends on a different parameter (i.e., $\bar{T}_i$). Thus, we cannot proceed in the same way as for the mean. In order to avoid unnecessary computations, we will consider the following procedure.

First, notice that by (9):

$$\bar{T} = \bar{T}_i + \frac{t_i - \bar{T}_i}{N}$$

and then, we compute the following sum

$$\begin{aligned} S^2 &= \sum_{j=1}^{N}\left(t_j - \bar{T}\right)^2 = \sum_{j=1}^{N}\left[\left(t_j - \bar{T}_i\right) - \left(\frac{t_i - \bar{T}_i}{N}\right)\right]^2 \\ &= \sum_{j=1}^{N}\left[\left(t_j - \bar{T}_i\right)^2 - \frac{2}{N}\left(t_j - \bar{T}_i\right)\left(t_i - \bar{T}_i\right) + \frac{(t_i - \bar{T}_i)^2}{N^2}\right] \\ &= \sum_{j=1}^{N}\left(t_j - \bar{T}_i\right)^2 - \frac{2(t_i - \bar{T}_i)}{N}\sum_{j=1}^{N}\left(t_j - \bar{T}_i\right) + \frac{(t_i - \bar{T}_i)^2}{N} \\ &= \sum_{j=1}^{N}\left(t_j - \bar{T}_i\right)^2 - 2(t_i - \bar{T}_i)\left(\bar{T} - \bar{T}_i\right) + \frac{(t_i - \bar{T}_i)^2}{N} \\ &= S_i^2 + \left(t_i - \bar{T}_i\right)^2 - 2(t_i - \bar{T}_i)\left(\bar{T} - \bar{T}_i\right) + \frac{(t_i - \bar{T}_i)^2}{N} \end{aligned} \tag{11}$$

Defining $S_i := \sum_{j=1}^{N-1}(t_i - \bar{T}_i)$, we can obtain the desirable expression

$$S_i^2 = S^2 - 2(\bar{T}_i - \bar{T})S_i - (N-1)(\bar{T}_i - \bar{T})^2 - (t_i - T)^2. \tag{12}$$

We can hence proceed in the same way to obtain similar expressions for the following sums.

$$S^3 = \sum_{j=1}^{N} \left(t_j - \bar{T}\right)^3 = \sum_{j=1}^{N} \left[\left(t_j - \bar{T}_i\right) - \left(\frac{t_i - \bar{T}_i}{N}\right) \right]^3 \tag{13}$$

and

$$S^4 = \sum_{j=1}^{N} \left(t_j - \bar{T}\right)^3 = \sum_{j=1}^{N} \left[\left(t_j - \bar{T}_i\right) - \left(\frac{t_i - \bar{T}_i}{N}\right) \right]^3 \tag{14}$$

from which we can obtain, respectively

$$S_i^3 = S^3 - 3(\bar{T}_i - \bar{T})S_i^2 - 3(\bar{T}_i - \bar{T})^2 S_j - (N-1)(\bar{T}_i - \bar{T})^3 - (t_i - \bar{T})^3 \tag{15}$$

and

$$S_i^4 = S^4 - 4(\bar{T}_i - \bar{T})S_i^3 - 6(\bar{T}_i - \bar{T})^2 S_i^2 - 4(\bar{T}_i - \bar{T})^3 \times S_j - (N-1)(\bar{T}_i - \bar{T})^4 - (t_i - \bar{T})^4 \tag{16}$$

Observe that for the computation of every S_i^j, where $i = 1, \ldots, N$ and $j = 1, \ldots, 4$, we only need to know the sum S^j. This reduces the number of operations that are needed by a factor of N. Notice that, for computing S^j we need N sums and powers of j. On the other hand, if we compute every S_i^j in the naive way, we will require $N - 1$ sums and powers of j for each one of the N sums, that is, $N(N-1)$. Now, with our formulas, we only need c multiplications for each S_i^j, that is, cN. In other words, we speed up the computations by a factor of N.

Finally, using the above formulas we can compute the desirable statistical moments. Since in this work we assume that our data represents a sample from a population, we use the following expressions:

$$\sigma_i = \sqrt{\frac{S_i^2}{N-2}} \tag{17}$$

$$\gamma_i = \frac{\frac{1}{N-1} S_i^3}{\left(\sqrt{\frac{1}{N-1} S_i^2}\right)^3} \tag{18}$$

$$\kappa_i = \frac{\frac{1}{N-1} S_i^4}{\left(\frac{1}{N-1} S_i^2\right)^2} \tag{19}$$

For the computation of the quartiles we can also be reduce the complexity via a simple rule. Notice that, after sorting the complete array T, the only thing we have to do is to compare the value t_i when we ask for the quartiles of the set T_i. If t_i is less than the value in the desirable position, then we compute the new value of the quartile. Otherwise, we conserve the value of the quartile, since removing t_i does not alter the order of the first values. The only cost we have to consider is the first sorting which has a complexity of $O(N \log N)$.

Appendix B

Table 11 presents the performance of the 21 different models evolved for different pairs of DTLZ problems and tested on problems from the same benchmark set, considering the 3-objective problems.

Performance of the $M_{4,6}^3$ model for 3-objective problems on each of the DTLZ and WFG problems, relative to the ground truth HV values HV_{GT} are presented in Table 12. The experiments where repeated with different population sizes, $N = \{100, 300, 600, 1000, 2000\}$, using 30 runs on each problem and 120 generations of the SMS-EMOA. The following measures were computed: the absolute error given as $AE = |HV_{approx} - HV_{GT}|$, and the relative error given by $RE = \frac{AE}{HV_{GT}}$, computing the average over all fronts.

Table 13 presents a detailed run time evaluation of $M_{4,6}^3$ on 3 objective problems. The comparison is carried out at two levels: when computing a single HV from a Pareto front and also when calculating the HV contribution of each individual in the non-dominated set (the latter is an important step of SMS-EMOA used as a selection mechanism). Note that for this experiment we use the WFG algorithm to compute the HV and contrast with the evolved model. These results are averaged over 15 runs.

References

[1] E. Zitzler, L. Thiele, Multiobjective evolutionary algorithms: a comparative case study and the strength Pareto approach, IEEE Trans. Evol. Comput. 3 (4) (1999) 257–271.

[2] E. Zitzler, D. Brockhoff, L. Thiele, The hypervolume indicator revisited: On the design of Pareto-compliant indicators via weighted integration, in: International Conference on Evolutionary Multi-Criterion Optimization, Springer, 2007, pp. 862–876.

[3] N. Beume, C.M. Fonseca, M. López-Ibáñez, L. Paquete, J. Vahrenhold, On the complexity of computing the hypervolume indicator, IEEE Trans. Evol. Comput. 13 (5) (2009) 1075–1082.

[4] H. Ishibuchi, R. Imada, N. Masuyama, Y. Nojima, Comparison of hypervolume, IGD and IGD+ from the viewpoint of optimal distributions of solutions, in: K. Deb, E. Goodman, C.A. Coello Coello, K. Klamroth, K. Miettinen, S. Mostaghim, P. Reed (Eds.), Evolutionary Multi-Criterion Optimization, Springer International Publishing, Cham, 2019, pp. 332–345.

[5] A. Jaszkiewicz, Improved quick hypervolume algorithm, Comput. Oper. Res. 90 (2018) 72–83.

[6] J. Bader, E. Zitzler, HypE: An algorithm for fast hypervolume-based many-objective optimization, Evol. Comput. 19 (1) (2011) 45–76.

[7] L. Bradstreet, L. Barone, L. While, Updating exclusive hypervolume contributions cheaply, in: 2009 IEEE Congress on Evolutionary Computation, 2009, pp. 538–544.

[8] K. Bringmann, T. Friedrich, An efficient algorithm for computing hypervolume contributions, Evol. Comput. 18 (3) (2010) 383–402.

[9] M.T.M. Emmerich, C.M. Fonseca, Computing hypervolume contributions in low dimensions: Asymptotically optimal algorithm and complexity results, in: R.H.C. Takahashi, K. Deb, E.F. Wanner, S. Greco (Eds.), Evolutionary Multi-Criterion Optimization, Springer Berlin Heidelberg, Berlin, Heidelberg, 2011, pp. 121–135.

[10] C. Hillermeier, Nonlinear Multiobjective Optimization: A Generalized Homotopy Approach, Vol. 135, Springer, 2001.

[11] D.A. Van Veldhuizen, Multiobjective Evolutionary Algorithms: Classifications, Analyses, and New Innovations, Tech. Rep., Air Force Institute of Technology, 1999.

[12] C.A.C. Coello, N.C. Cortés, Solving multiobjective optimization problems using an artificial immune system, Genet. Program. Evol. Mach. 6 (2) (2005) 163–190.

[13] O. Schütze, X. Esquivel, A. Lara, C.A.C. Coello, Using the averaged Hausdorff distance as a performance measure in evolutionary multiobjective optimization, IEEE Trans. Evol. Comput. 16 (4) (2012) 504–522.

[14] J.M. Bogoya, A. Vargas, O. Cuate, O. Schütze, A (p,q)-averaged Hausdorff distance for arbitrary measurable sets, Math. Comput. Appl. 23 (3) (2018).

[15] J.M. Bogoya, A. Vargas, O. Schütze, The averaged Hausdorff distances in multi-objective optimization: A review, Mathematics 7 (10) (2019).

[16] D. Brockhoff, T. Wagner, H. Trautmann, On the properties of the R2 indicator, in: Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation, in: GECCO '12, ACM, New York, NY, USA, 2012, pp. 465–472.

[17] M.P. Hansen, A. Jaszkiewicz, Evaluating the Quality of Approximations to the Non-Dominated Set, IMM, Department of Mathematical Modelling, Technical Universityof Denmark, 1994.

[18] E. Zitzler, J. Knowles, L. Thiele, Quality assessment of Pareto set approximations, in: J. Branke, K. Deb, K. Miettinen, R. Słowiński (Eds.), Multiobjective Optimization: Interactive and Evolutionary Approaches, Springer Berlin Heidelberg, Berlin, Heidelberg, 2008, pp. 373–404.

[19] E. Dilettoso, S.A. Rizzo, N. Salerno, A weakly Pareto compliant quality indicator, Math. Comput. Appl. 22 (1) (2017).

[20] H. Ishibuchi, H. Masuda, Y. Tanigaki, Y. Nojima, Modified distance calculation in generational distance and inverted generational distance, in: A. Gaspar-Cunha, C. Henggeler Antunes, C.C. Coello (Eds.), Evolutionary Multi-Criterion Optimization, Springer International Publishing, Cham, 2015, pp. 110–125.

[21] N. Beume, B. Naujoks, M. Emmerich, SMS-EMOA: MUltiobjective selection based on dominated hypervolume, European J. Oper. Res. 181 (3) (2007) 1653–1669.
[22] L. While, L. Bradstreet, L. Barone, A fast way of calculating exact hypervolumes, IEEE Trans. Evol. Comput. 16 (1) (2012) 86–95.
[23] T.M. Chan, Klee's measure problem made easy, in: 2013 IEEE 54th Annual Symposium on Foundations of Computer Science, 2013, pp. 410–419.
[24] K. Shang, H. Ishibuchi, L. He, L.M. Pang, A survey on the hypervolume indicator in evolutionary multiobjective optimization, IEEE Trans. Evol. Comput. 25 (1) (2020) 1–20.
[25] K. Bringmann, T. Friedrich, Approximating the least hypervolume contributor: NP-hard in general, but fast in practice, Theoret. Comput. Sci. 425 (2012) 104–116, Theoretical Foundations of Evolutionary Computation.
[26] K. Deb, P.C. Roy, R. Hussein, Surrogate modeling approaches for multiobjective optimization: methods, taxonomy, and results, Math. Comput. Appl. (ISSN: 2297-8747) 26 (1) (2021) 5, http://dx.doi.org/10.3390/mca26010005, https://www.mdpi.com/2297-8747/26/1/5.
[27] K. Bringmann, T. Friedrich, Approximating the volume of unions and intersections of high-dimensional geometric objects, Comput. Geom. 43 (6–7) (2010) 601–610.
[28] J. Deng, Q. Zhang, Approximating hypervolume and hypervolume contributions using polar coordinate, IEEE Trans. Evol. Comput. 23 (5) (2019) 913–918, http://dx.doi.org/10.1109/TEVC.2019.2895108.
[29] H. Ishibuchi, N. Tsukamoto, Y. Sakane, Y. Nojima, Hypervolume approximation using achievement scalarizing functions for evolutionary many-objective optimization, in: 2009 IEEE Congress on Evolutionary Computation, 2009, pp. 530–537, http://dx.doi.org/10.1109/CEC.2009.4982991.
[30] K. Shang, H. Ishibuchi, M.-L. Zhang, Y. Liu, A new R2 indicator for better hypervolume approximation, in: Proceedings of the Genetic and Evolutionary Computation Conference, in: GECCO '18, Association for Computing Machinery, New York, NY, USA, 2018, pp. 745–752, http://dx.doi.org/10.1145/3205455.3205543.
[31] K. Shang, H. Ishibuchi, X. Ni, R2-based hypervolume contribution approximation, IEEE Trans. Evol. Comput. 24 (1) (2020) 185–192, http://dx.doi.org/10.1109/TEVC.2019.2909271.
[32] W. Tang, H.-L. Liu, L. Chen, K.C. Tan, Y. ming Cheung, Fast hypervolume approximation scheme based on a segmentation strategy, Inform. Sci. 509 (2020) 320–342, http://dx.doi.org/10.1016/j.ins.2019.02.054, URL http://www.sciencedirect.com/science/article/pii/S0020025519301690.
[33] J.E. Fieldsend, Efficient real-time hypervolume estimation with monotonically reducing error, in: Proceedings of the Genetic and Evolutionary Computation Conference, in: GECCO '19, Association for Computing Machinery, New York, NY, USA, 2019, pp. 532–540, http://dx.doi.org/10.1145/3321707.3321730, URL https://doi.org/10.1145/3321707.3321730.
[34] J. Deng, Q. Zhang, Combining simple and adaptive Monte Carlo methods for approximating hypervolume, IEEE Trans. Evol. Comput. 24 (5) (2020) 896–907, http://dx.doi.org/10.1109/TEVC.2020.2969965.
[35] A. Menchaca-Mendez, C.A.C. Coello, A new selection mechanism based on hypervolume and its locality property, in: 2013 IEEE Congress on Evolutionary Computation, 2013, pp. 924–931.
[36] A. Menchaca-Mendez, E. Montero, M.-C. Riff, C.A.C. Coello, A more efficient selection scheme in iSMS-EMOA, in: A.L. Bazzan, K. Pichara (Eds.), Advances in Artificial Intelligence – IBERAMIA 2014, Springer International Publishing, Cham, 2014, pp. 371–380.
[37] K. Bringmann, T. Friedrich, Don't be greedy when calculating hypervolume contributions, in: Proceedings of the Tenth ACM SIGEVO Workshop on Foundations of Genetic Algorithms, in: FOGA '09, Association for Computing Machinery, New York, NY, USA, 2009, pp. 103–112.
[38] R.S. Olson, N. Bartley, R.J. Urbanowicz, J.H. Moore, Evaluation of a tree-based pipeline optimization tool for automating data science, in: Proceedings of the Genetic and Evolutionary Computation Conference 2016, in: GECCO '16, Association for Computing Machinery, New York, NY, USA, 2016, pp. 485–492.
[39] J. Aguilera, L. González, M. Montes-y Gómez, P. Rosso, A new weighted k-nearest neighbor algorithm based on newton's gravitational force, in: Lect. Notes Comput. Sci. (Including Subser. Lect. Notes Artif. Intell. Lect. Notes Bioinformatics), in: Iberoamerican Congress on Pattern Recognition, 2018, Springer, 2019, pp. 305–313.
[40] J. Aguilera, L.C. González, M. Montes-y Gómez, R. López, H.J. Escalante, From neighbors to strengths - the k-strongest strengths (kSS) classification algorithm, Pattern Recognit. Lett. 136 (2020) 301–308.
[41] J.C. Tay, N.B. Ho, Evolving dispatching rules using genetic programming for solving multi-objective flexible job-shop problems, Comput. Ind. Eng. 54 (3) (2008) 453–473.
[42] M. Ebner, E. karls-universitat Tubingen, A. Rechnerarchitektur, On the evolution of interest operators using genetic programming, in: In Proc. EuroGP'98, 1998, pp. 6–10.
[43] G. Olague, L. Trujillo, Evolutionary-computer-assisted design of image operators that detect interest points using genetic programming, Image Vis. Comput. 29 (7) (2011) 484–498.
[44] S.O. Haraldsson, J.R. Woodward, Automated design of algorithms and genetic improvement: contrast and commonalities, in: J. Woodward, J. Swan, E. Barr (Eds.), GECCO 2014 4th Workshop on Evolutionary Computation for the Automated Design of Algorithms, ACM, Vancouver, BC, Canada, 2014, pp. 1373–1380.
[45] J. Petke, S.O. Haraldsson, M. Harman, W.B. Langdon, D.R. White, J.R. Woodward, Genetic improvement of software: A comprehensive survey, IEEE Trans. Evol. Comput. 22 (3) (2018) 415–432.
[46] V.R. López-López, L. Trujillo, P. Legrand, Applying genetic improvement to a genetic programming library in C++, Soft Comput. 23 (22) (2018) 11593–11609.
[47] E. Z-Flores, L. Trujillo, A. Sotelo, P. Legrand, L.N. Coria, Regularity and matching pursuit feature extraction for the detection of epileptic seizures, J. Neurosci. Methods 266 (2016) 107–125.
[48] E. Z-Flores, M. Abatal, A. Bassam, L. Trujillo, P. Juárez-Smith, Y.E. Hamzaoui, Modeling the adsorption of phenols and nitrophenols by activated carbon using genetic programming, J. Cleaner Prod. 161 (2017) 860–870.
[49] J.R. López, L.C. González, J. Wahlström, M.M. y Gómez, L. Trujillo, G. Ramírez-Alonso, A genetic programming approach for driving score calculation in the context of intelligent transportation systems, IEEE Sens. J. 18 (17) (2018) 7183–7192.
[50] K. Deb, L. Thiele, M. Laumanns, E. Zitzler, Scalable test problems for evolutionary multiobjective optimization, in: Evolutionary Multiobjective Optimization, Springer, 2005, pp. 105–145.
[51] S. Huband, P. Hingston, L. Barone, L. While, A review of multiobjective test problems and a scalable test problem toolkit, IEEE Trans. Evol. Comput. 10 (5) (2006) 477–506.
[52] R. Jenke, A. Peer, M. Buss, Feature extraction and selection for emotion recognition from EEG, IEEE Trans. Affect. Comput. 5 (3) (2014) 327–339.
[53] L.M. noz, L. Trujillo, S. Silva, Transfer learning in constructive induction with genetic programming, Genet. Program. Evol. Mach. (2019).
[54] D.P. Searson, GPTIPS 2: An open-source software platform for symbolic data mining, in: A.H. Gandomi, A.H. Alavi, C. Ryan (Eds.), Handbook of Genetic Programming Applications, Springer International Publishing, Cham, 2015, pp. 551–573.
[55] D. Castelvecchi, Can we open the black box of AI? Nature 538 (7623) (2016) 20–23.
[56] T.M. Chan, Klee's measure problem made easy, in: 2013 IEEE 54th Annual Symposium on Foundations of Computer Science, 2013, pp. 410–419.
[57] W.B. Langdon, R. Poli, Foundations of Genetic Programming, Springer-Verlag New York, Inc., New York, NY, USA, 2002.
[58] A. Sotelo, E. Guijarro, L. Trujillo, L.N. Coria, Y. Martínez, Identification of epilepsy stages from ECoG using genetic programming classifiers, Comput. Biol. Med. 43 (11) (2013) 1713–1723.
[59] M. Sipper, W. Fu, K. Ahuja, J.H. Moore, Investigating the parameter space of evolutionary algorithms, BioData Min. 11 (1) (2018).
[60] L. Trujillo, E.A. González, E. Galván, J.J. Tapia, A. Ponsich, On the analysis of hyper-parameter space for a genetic programming system with iterated F-Race, Soft Comput. (2020).