Abstract—Scalarizing functions have been widely used to convert a multiobjective optimization problem into a single objective optimization problem. However, their use in solving computationally expensive multi- and many-objective optimization problems using Bayesian multiobjective optimization is scarce. Scalarizing functions can play a crucial role on the quality and number of evaluations required when doing the optimization. In this article, we compare 15 different scalarizing functions in the framework of Bayesian multiobjective optimization and build Gaussian process models on them. We use the expected improvement as infill criterion (or acquisition function) to update the models. In particular, we analyze the performance of different scalarizing functions on several benchmark problems with different number of objectives to be optimized. The review and experiments on different functions provide useful insights in using a scalarizing function, especially for problems with a large number of objectives.

Index Terms—Pareto optimality, evolutionary multiobjective optimization, surrogate, metamodel

I. INTRODUCTION

Many real-world optimization problems have multiple conflicting objectives to be achieved and use black-box computationally expensive simulators. Surrogate-assisted optimization methods have been widely used in the literature to solve such problems. Many of these methods use Bayesian optimization involving Gaussian processes (or Kriging) as surrogate models. The Gaussian processes provide an advantage over other types of surrogate models because of their ability to provide uncertainty in predictions. For more details about surrogate-assisted methods, see e.g. surveys [1], [2], [3], [4], [5].

A generic framework of a Bayesian optimization method is shown in Figure 1. In the first step, a set of samples is generated e.g. using a design of experiment technique like Latin hypercube sampling [6]. In the second step, these samples are then evaluated with expensive objectives. In the third step, a termination criterion is checked. If the termination criterion is met, nondominated solutions from all expensive evaluated solutions are used as the final solutions. Otherwise, surrogate models are built using evaluated solutions in the fourth step. An optimizer e.g. an evolutionary algorithm is then used with the models in the fifth step to find promising samples by using an appropriate infilling criterion (or updating criterion or acquisition function). In the sixth step, a fixed number of samples generated with the optimizer is selected which are then evaluated with expensive objective functions.

All six steps mentioned above are important in the performance of a Bayesian optimization method [7]. In this article, our focus is on the fourth step i.e. building or training surrogates. Once, the samples are evaluated with expensive objective functions in the second step, there can be different ways to build surrogates. The most common way is to build surrogate for each objective function [8], [9], [10]. Another way is to build a surrogate for a scalarizing function after converting multiobjective optimization problem into a single objective optimization problem [11], [12], which is also the focus of this article. There are other approaches for building surrogates like the classification of solutions into different ranks or classes [13], [14], [15]. For details on other ways of building surrogates, see [1], [3], [16]. Note that a scalarizing function can also be used after building models on each objective function e.g. as in [10] to do a local search. In this article, we focus on building surrogate after converting a multiobjective optimization problem into a single objective by using a scalarizing function.

There are two main advantages in using a single surrogate when solving an expensive MOP. The first one is that only one surrogate is used in the solution process instead of multiple surrogates, which reduces the computational burden e.g. the training time especially in many-objective (usually more than three objectives) optimization problems. The second advantage is that one can use an infill criterion proposed for single-objective optimization problems which also reduces the computational complexity. In the literature, a little attention has been paid in using scalarizing functions for building surrogates on them and only a few studies exist in the literature. For instance, in [11], an augmented achievement scalarizing function (AASF) was used and in [12], hypervolume improvement, dominance rank and minimum signed distance were used (more details are provided in the Section III).

It is worth important to be pointed out that several studies exist [17], [18], [19], [20], [21] on using scalarizing functions without using them in a Bayesian optimization method. In this article, we study 15 different scalarizing functions and build surrogates on them in solving (computationally) expensive MOPs and particularly, focus on answering the following research questions:

1) are different scalarizing functions perform different to each other under the same framework?
2) are surrogate models built on a given scalarizing function sensitive to the number of objectives?

To answer the questions above, we embed different scalarizing functions into the framework of efficient global optimization (EGO) [22]. The algorithm uses the expected improvement criterion for updating the surrogates build on the scalarizing function. We then test the method with different scalarizing functions on several benchmark problems with different numbers of objectives. We analyze the accuracy and uncertainty provided by Gaussian process models on different scalarizing functions with a different number of objectives.

The rest of the article is organized as follows. In the next section, we provide a literature survey on using scalarizing functions with and without building surrogates on them. In Section 3, we give a brief introduction to different functions with their mathematical formulations and merits and demerits. In Section 4, numerical experiments are conducted to answer the research questions mentioned above. We finally conclude and mention the future research directions in Section 5.

II. RELATED WORK

Scalarizing functions have been used since decades in the Multiple Criteria Decision Making (MCDM) community [20], [21], [23], [24]. In evolutionary multiobjective optimization (EMO) algorithms especially decomposition based, utilization of scalarizing functions became popular in the last few years. Many studies and algorithms exist in the literature utilizing different functions for solving multi- and many-objectives optimization problems. Some of the well-known decomposition based EMO algorithms utilizing different scalarizing functions are nondominated-sorting genetic algorithm III (NSGA-III) [25], multiobjective optimization based on decomposition (MOEA/D) [26] and its numerous versions [27] and reference vector guided evolutionary algorithm (RVEA) [28]. For more details on the working principle of decomposition based algorithms, see surveys on many-objective optimization evolutionary algorithms [29], [30].

In [31], two different scalarizing functions weighted sum and augmented Chebyshev were used adaptively in the solution process in the framework of MOEA/D by using a multi-grid scheme. The proposed idea was tested on a knapsack problem with four and six objectives and performed better than the original version of MOEA/D. The authors extended their work in [32], and modified the penalty boundary intersection (PBI) function in MOEA/D to handle different kinds of Pareto fronts. Two modified PBI functions called two-level PBI and quadratic PBI were tested in the MOEA/D framework and the algorithm with two new scalarizing functions performed better than the original version.

A detailed study on different scalarizing functions and their corresponding parameters was conducted in [33]. Instead of proposing a new algorithm, the authors showed the performance of the different scalarizing functions in a simple (1+λ)-evolutionary algorithm [33] on bi-objective optimization problems. In [17], the performance of 15 different scalarizing functions was tested in MOEA/D [26] and MOMBI-II [34] algorithms. The authors used a tool called EVOCA [35] to tune the parameter values in different functions. Two algorithms MOEA/D and MOMBI-II [34] with different scalarizing functions were tested on Lame Supersphere test problems [36]. The authors found out that the performance of two algorithms depends on the choice of scalarizing functions.

In [19], a hyper-heuristic was used to rank different scalarizing functions with a measure called s-energy [37]. The proposed algorithm was tested on ZDT, DTLZ and WFG benchmark problems with 2-10 objectives and compared with MOMBI-II, MOEA/D and NSGA-III and found better results in most of the instances. In [38], two new scalarizing functions called the multiplicative scalarizing function (MSF) and penalty-based scalarizing function (PSF) were used in MOEA/D-DE [39]. The proposed scalarizing functions performed better than penalty boundary intersection, Chebyshev and weighted sum scalarizing functions.

The first algorithm in Bayesian multiobjective optimization method using a scalarizing function and building a surrogate on it was proposed in [11] and known as ParEGO (for Pareto based efficient global optimization). A Gaussian process model [40] was used as a surrogate of the scalarizing function. In the algorithm, a set of reference vectors was uniformly generated in the objective space using simplex lattice-design method [41]. In each iteration, the algorithm randomly selected a reference vector among a set of vectors, which was then used in the Chebyshev function to build the surrogate. The algorithm was compared with NSGA-II [42] and performed significantly better.

In a recent study in [12], three scalarizing functions, hy-
pervolume improvement, dominance ranking, and minimum signed distance were proposed in solving expensive MOP. These scalarizing functions do not use reference vectors and preserve the dominance relationship. The authors also used the framework of EGO and compared with SMS-EGO [9] and ParEGO (i.e. with Chebyshev scalarizing function). The results outperformed the ParEGO algorithm and performed similarly to SMS-EGO in some cases.

III. SCALARIZING FUNCTIONS

In this section, we summarize 15 different scalarizing functions. All these functions have already been explained in details in the literature [43], [21]. Therefore, we provide a brief summary of these functions.

We define a multiobjective optimization problem (MOP) as:

\[
\text{minimize } \{ f_1(x), \ldots, f_k(x) \} \\
\text{subject to } x \in S
\]

with \( k \geq 2 \) objective functions \( f_i(x) : S \rightarrow \mathbb{R}^n \). The vector of objective function values is denoted by \( f(x) = (f_1(x), \ldots, f_k(x))^T \). The (nonempty) feasible space \( S \) is a subset of the decision space \( \mathbb{R}^n \) and consists of decision vectors \( x = (x_1, \ldots, x_n)^T \) that satisfy all the constraints. The scalarizing functions are as follows:

1) **Weighted sum (WS)**: The weighted sum combines different objectives linearly and has been widely used [21]. It converts the a MOP into a single-objective optimization problems as:

\[
g = \sum_{i=1}^{k} w_i f_i.
\]

One major limitation in using WS is that it cannot find solutions in non-convex parts of the Pareto front.

2) **Exponential weighted criterion (EWC)**: EWC was first used in [44] to overcome the limitations of WS:

\[
g = \sum_{i=1}^{k} \exp(p w_i) \exp(p f_i).
\]

As mentioned in [43], the performance of the function depends on the value of \( p \) and usually a large value of \( p \) is needed.

3) **Weighted power (WPO)**: The weighted power defined below can find solutions when the Pareto front is not convex. However, it also depends on the parameter \( p \) as in EWC.

\[
g = \sum_{i=1}^{k} w_i (f_i)^p.
\]

4) **Weighted norm (WN)**: The weighted norm (or \( L_p \) metric) is a generalized form of the weighted sum:

\[
g = \left( \sum_{i=1}^{k} w_i |f_i|^p \right)^{1/p}.
\]

The weighted norm has been used in MOEA/D [26]. Likewise EWC and WPO, its performance also depends on the choice of the \( p \) parameter.

5) **Weighted product (WPR)**: The weighted power is defined as:

\[
g = \prod_{i=1}^{k} (f_i)^{w_i}.
\]

It is also called as product of powers [43] and can find solutions in non-convex parts of the Pareto front. However, like EWC, WPO and WN, its performance depends on the value of \( p \).

6) **Chebyshev function (TCH)**: The Chebyshev function can be derived from the WN function with \( p = \infty \). This function has been used in many EMO algorithms such as MOEA/D [26] and its versions [27]:

\[
g = \max_i |w_i (f_i - z_i^*)|,
\]

where \( z_i^* \) is the ideal or utopian objective vector. In this work, we normalize the expensive objective function values in the range [0,1] before building the model. Therefore, \( z_i^* \) is a vector of zeros.

7) **Augmented Chebyshev (ATCH)**: In [24], it was suggested that weakly Pareto optimal solutions can be avoided by adding an augmented term to TCH:

\[
g = \max_i |w_i (f_i - z_i^*)| + \alpha \sum_{i=1}^{k} |f_i - z_i^*|, \quad \alpha \geq 1
\]

Moreover, this function was the first to be used as a surrogate in [11].

8) **Modified Chebyshev (MTCH)**: A slightly modified form of MTCH was used in [45]:

\[
g = \max_i \left[ w_i (f_i - z_i^*) + \alpha \sum_{i=1}^{k} |f_i - z_i^*| \right], \quad \alpha > 0
\]

As mentioned in [21] that main different between ATCH and MTCH is the slope to avoid weakly Pareto optimal solutions.

9) **Penalty boundary intersection (PBI)**: The PBI function was first used in MOEA/D and used as the selection criterion for balancing convergence and diversity:

\[
g = d_1 + \theta d_2, \quad \theta \geq 0
\]

where \( d_1 = |\mathbf{f} \cdot \mathbf{w}| \) and \( d_2 = |\mathbf{f} - d_1 \mathbf{w}| \). The PBI function has been widely used in EMO algorithms [27]. However, as shown in [46] its performance is effected by the \( \theta \) parameter.

10) **Inverted penalty boundary intersection (IPBI)**: To enhance the diversity of solutions, IPBI was proposed in [47]:

\[
g = \theta d_2 - d_1, \quad \theta > 0
\]

where \( d_1 = |\mathbf{f} \cdot \mathbf{w}| \) and \( d_2 = |\mathbf{f} - d_1 \mathbf{w}| \) and \( \mathbf{f}' = \mathbf{z}_{nadir} - \mathbf{f} \). In this work, we considered the vector of worst expensive objective function values as \( \mathbf{z}_{nadir} \).

11) **Quadratic PBI (QPBI)**: Recently, an enhanced version of PBI function was proposed in [32]:

\[
g = d_1 + \theta d_2 \frac{d_2}{d_1^2}, \quad \theta > 0
\]
where \(d_1\) and \(d_2\) are same as in PBI and \(d^*\) is an adaptive parameter and defined as:

\[
d^* = \alpha \frac{1}{k} \sum \left( z^{\text{nadir}} - z^{\text{ideal}} \right)
\]

where \(\alpha\) is a pre-defined parameter and \(H\) is a parameter used in generating the reference (or weight vectors) in decomposition based EMO algorithms. For more details about these parameters, see [26], [28].

12) **Angle penalized distance (APD):** The APD function is a recently proposed scalarizing function and used as the selection criterion in reference vector guided many-objective evolutionary algorithm (RVEA) [28]:

\[
g = 1 + P(\theta) \cdot ||f||,
\]

where \(P(\theta) = k(\frac{FE}{FE_{\text{max}}})^{\alpha \theta}\), \(FE\) is the number of expensive function evaluations at the current iteration and \(FE_{\text{max}}\) is the maximum number of expensive function evaluations. The angle between an objective vector \(f\) and the reference vector to which it is assigned is represented by \(\theta\) and the minimum of all angles between a reference vector selected and other reference vectors is represented by \(\gamma\). This function adaptively balances the convergence and diversity based on the maximum number of function evaluations. The performance of the function depends on the value of \(\alpha\) and \(FE_{\text{max}}\).

13) **Hypervolume improvement (HypI):** The HypI is a recently proposed scalarizing function in a surrogate-based algorithm [12]. Given a set of solutions \(X\), a nondominated sorting is performed to find fronts of different ranks as in [42]. Let the different fronts of ranks 1, 2, \ldots, \(k\) are denoted by \(P_1, P_2, \ldots\) and the hypervolume of a front \(P_k\) given a reference point \(r\) is denoted by \(H(P_k)\). Then hypervolume improvement a solution \(x \in X\) belongs to the front \(P_k\) is then given by:

\[
g = H(x \cup P_{k+1}).
\]

In this way, the Pareto dominance is preserved when calculating the hypervolume contributions of solutions. The performance of the function can be sensitive to the reference point in calculating the hypervolume.

14) **Dominance ranking (DomRank):** This function is also recently proposed in [12] and assigns fitness values based on the ranks of different solutions as done in the MOGA algorithm [48]. In [48], a solution is assigned a rank as:

\[
\text{rank}(x) = 1 + p, \quad \text{where } p \text{ is number of solutions dominating } x.
\]

Similarly, in [12], given a set of solutions (with expensive evaluations) \(X\) the fitness of a solution \(x\) is:

\[
g = 1 - \frac{\text{rank}(x) - 1}{|X| - 1}.
\]

For instance, the rank of a solution \(x'\) dominated by all other solutions would be \(\text{rank}(x') = 1 + |X| - 1\), and therefore, the fitness of solution would be \(g(x') = 1 - \frac{1 + |X| - 1}{|X| - 1} = 0\). Similarly, the rank of solutions belong the first front will be 1. This function is maximized to find samples for training the surrogates. For problems with many-objectives, where all solutions are non-dominated

### Table I: Parameters used when building the Kriging model, \(lb\) and \(ub\) denote the lower and upper bounds, respectively

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Lower Bound</th>
<th>Upper Bound</th>
</tr>
</thead>
<tbody>
<tr>
<td>Length scale bounds</td>
<td>lower: 1e-6</td>
<td>upper: 100</td>
</tr>
<tr>
<td>Noise standard deviation</td>
<td>lower: 1e-6</td>
<td>upper: 100</td>
</tr>
<tr>
<td>Optimization algorithm</td>
<td>lower: 1e-9</td>
<td>upper: 1</td>
</tr>
</tbody>
</table>

### Table II: Number of variables (\(n\)) for WFG and DTLZ suites

<table>
<thead>
<tr>
<th>Suite</th>
<th>Number of Objectives</th>
<th>Number of Variables</th>
</tr>
</thead>
<tbody>
<tr>
<td>WFG</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>DTLZ</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>WFG</td>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>DTLZ</td>
<td>10</td>
<td>20</td>
</tr>
</tbody>
</table>
TABLE III: Parameters values used in different scalarizing functions

<table>
<thead>
<tr>
<th>Scalarizing function</th>
<th>Parameter value</th>
</tr>
</thead>
<tbody>
<tr>
<td>EWC</td>
<td>( p = 1 )</td>
</tr>
<tr>
<td>WPO</td>
<td>( p = 3 )</td>
</tr>
<tr>
<td>WN</td>
<td>( p = 0.5 )</td>
</tr>
<tr>
<td>ATCH</td>
<td>( \alpha = 0.0001 )</td>
</tr>
<tr>
<td>MTCH</td>
<td>( \alpha = 0.0001 )</td>
</tr>
<tr>
<td>PBI</td>
<td>( \theta = 5 )</td>
</tr>
<tr>
<td>IPBI</td>
<td>( \theta = 5 )</td>
</tr>
<tr>
<td>QPBI</td>
<td>( \theta = 1 )</td>
</tr>
<tr>
<td>APD</td>
<td>( \alpha = 2 )</td>
</tr>
</tbody>
</table>

TABLE IV: Reference point in calculating hypervolumes in HypI and as performance measure

<table>
<thead>
<tr>
<th>Problem</th>
<th>Reference point</th>
</tr>
</thead>
<tbody>
<tr>
<td>DTLZ1</td>
<td>( 400 \times (1,k) )</td>
</tr>
<tr>
<td>DTLZ2</td>
<td>( 1.5 \times (1,k) )</td>
</tr>
<tr>
<td>DTLZ3</td>
<td>( 900 \times (1,k) )</td>
</tr>
<tr>
<td>DTLZ4</td>
<td>( 2 \times (1,k) )</td>
</tr>
<tr>
<td>DTLZ5</td>
<td>( 1.5 \times (1,k) )</td>
</tr>
<tr>
<td>DTLZ6</td>
<td>( 6 \times (1,k) )</td>
</tr>
<tr>
<td>DTLZ7</td>
<td>( 5 \times (1,k) )</td>
</tr>
</tbody>
</table>

III. For WFG problems, the reference point \( r \) in HyPI is used as \( 2 \times \mathbf{g}^{\text{nadir}} \). The vector \( \mathbf{g}^{\text{nadir}} \) contains the maximum objective function values in the Pareto front of the given problem. We ran 21 independent runs for each scalarizing function with 300 maximum number of expensive function evaluations. We show the performance of different functions with inverted generational distance (IGD) and hypervolume. To compare the results of different functions, we used the Wilcoxon rank sum test with Bonferroni correction. The mean IGD values with standard deviation (in parentheses) for DTLZ problems are provided in Table V. The results with hypervolume and other detailed plots are in the supplementary material.

In Table V, the values statistically similar to the best value are in bold, the best value is in the circle and the worst value is underlined. As can be seen, the performance of different functions varies with different problems and three functions PBI, Hypl, and APD outperformed other functions. Moreover, WS, ATCH, and DomRank never had the best IGD value in any of 64 instances. Note that, all these three functions have statistically similar values to the best IGD value in many cases. Another interesting observation from the results is that three different scalarizing functions ATCH, DomRank and MSD used in the literature for building surrogates on them were not in the top list. On the other hand, scalarizing functions used as the selection criterion in EMO algorithms e.g. APD and PBI performed significantly better. These results indicate that one needs to be cautious in selecting a scalarizing function in solving an expensive MOP.

B. Sensitivity towards number of objectives

When using scalarizing functions for building surrogates on them, the number of objectives can play a crucial role in their performance. Before using the surrogate built on the scalarizing function, we can analyze and visualize the fitness landscape with respect to uncertainty of predictions.

An example of fitness landscapes of DTLZ2 with two and 10 objectives for different scalarizing functions are shown in Figures 2 and 3, respectively. We used the same training data for all scalarizing functions. Then we generated a testing data set and used the model to get the predicted values (solid lines in figures) and uncertainties of the predicted values (shaded region in figures). For visualization purposes, we could show only one of the variables values.

As can be seen, for the same training data set (and therefore same objective function values), different scalarizing functions gave different values. Therefore, the surrogate model to be built strongly depends on the \( g \) function values. For instance, in many-objective case, most of the solutions were nondominated and functions which used the property of Pareto dominance produced similar \( g \) values, as can be seen in the plots of DomRank, and MSD in 10 objective cases. As the \( g \) function values were the same for most of the solutions, the algorithm (EGO in this work) for finding a sample to update the surrogate by optimizing an infill criterion (EI in this work) was not able to enhance the accuracy of the surrogates. Scalarizing functions like APD and PBI did not use the property of Pareto dominance and suitable for a large number of objectives. In addition to the sensitivity towards many objectives, some functions, EWC and WS did not perform well in most of the problems. In EWC, the \( g \) function landscape is very rugged as can be seen in figures and WS has the problem of finding solutions in the non-convex parts of the Pareto front. These results show that one needs to see or analyze the landscape of scalarizing functions for the given training data set before using them in the optimization. We observed the similar behavior on other problems and landscapes on other problems with different objectives are provided in the supplementary material\(^1\).

V. Conclusions

This work focused on reviewing, analyzing and comparing different scalarizing functions in Bayesian multiobjective optimization for solving (computationally) expensive multi- and many-objective optimization problems. We provided an overview of different functions with their merits and demerits. We built the surrogates on the different functions and compared them with the different number of objectives. The results clearly showed that some functions outperformed others in many cases and some did not work in most of the cases. We then analyzed the fitness landscape of different functions with respect to the number of objectives. We found out that some functions are sensitive to the number of objectives. In this work, we did not compare the scalarizing functions with other methods which do not use scalarizing function. Therefore, comparison with other algorithms is a topic for future research.

\(^1\)https://github.com/tichugh/SCF
Fig. 2: Approximated scalarizing function values (in solid line) with their uncertainty (in gray) on DTLZ2 two objectives. The dots represent the training data set.

Fig. 3: Approximated scalarizing function values (in solid line) with their uncertainty (in gray) on DTLZ2 10 objectives. The dots represent the training data set.
ACKNOWLEDGEMENT

This research was supported by the Natural Environment Research Council, UK [grant number NE/P017436/1] and Youth and Sports of the Czech Republic under the project CZ.02.1.01/0.0/0.0/17_049/0008408: Hydrodynamic design of pumps.

REFERENCES


