

This is a self-archived version of an original article. This version may differ from the original in pagination and typographic details.

Author(s): Aghaei pour, Pouya; Hakanen, Jussi; Miettinen, Kaisa

Title: A surrogate-assisted a priori multiobjective evolutionary algorithm for constrained multiobjective optimization problems

Year: 2024

Version: Published version

Copyright: © The Author(s) 2024

Rights: CC BY 4.0

Rights url: <https://creativecommons.org/licenses/by/4.0/>

Please cite the original version:

Aghaei pour, P., Hakanen, J., & Miettinen, K. (2024). A surrogate-assisted a priori multiobjective evolutionary algorithm for constrained multiobjective optimization problems. *Journal of Global Optimization*, Early online. <https://doi.org/10.1007/s10898-024-01387-z>



A surrogate-assisted a priori multiobjective evolutionary algorithm for constrained multiobjective optimization problems

Pouya Aghaei pour¹ · Jussi Hakanen¹ · Kaisa Miettinen¹

Received: 24 February 2022 / Accepted: 18 February 2024
© The Author(s) 2024

Abstract

We consider multiobjective optimization problems with at least one computationally expensive constraint function and propose a novel surrogate-assisted evolutionary algorithm that can incorporate preference information given a priori. We employ Kriging models to approximate expensive objective and constraint functions, enabling us to introduce a new selection strategy that emphasizes the generation of feasible solutions throughout the optimization process. In our innovative model management, we perform expensive function evaluations to identify feasible solutions that best reflect the decision maker's preferences provided before the process. To assess the performance of our proposed algorithm, we utilize two distinct parameterless performance indicators and compare them against existing algorithms from the literature using various real-world engineering and benchmark problems. Furthermore, we assemble new algorithms to analyze the effects of the selection strategy and the model management on the performance of the proposed algorithm. The results show that in most cases, our algorithm has a better performance than the assembled algorithms, especially when there is a restricted budget for expensive function evaluations.

Keywords Multiple objectives · Model management · A priori algorithms · Constraint handling · Surrogate-assisted optimization · Constrained problems · Computationally expensive problems

1 Introduction

Many real-world problems involve multiple conflicting objective functions to be optimized simultaneously, subject to constraints, see, e.g., [1–3]. Such problems are known as con-

✉ Pouya Aghaei pour
pouya.ghaeipour@gmail.com

Jussi Hakanen
jussi.hakanen@jyu.fi

Kaisa Miettinen
kaisa.miettinen@jyu.fi

¹ Faculty of Information Technology, University of Jyväskylä, P.O. Box 35, 40014 Agora, Finland

strained multiobjective optimization problems (CMOPs). Because of the conflict between the objective functions, many so-called *Pareto optimal solutions* usually exist, representing different trade-offs. In the absence of additional information, all Pareto optimal solutions are mathematically incomparable. Typically, a domain expert called a decision maker (DM) provides preference information to find the most preferred Pareto optimal solution for practical implementation.

There are three significant challenges in solving real-world CMOPs. First, the evaluation of one (or more) objective or constraint functions can be time-consuming. For instance, many engineering problems rely on simulations, where function evaluation can take minutes or hours [4–10]. Such problems are referred to as *computationally expensive* ones. We can find computationally expensive problems in various fields such as ergonomic well-being [1], manufacturing design [2], shape and design optimization [4–7] and the configuration of energy sources [11].

In this paper, we focus on CMOPs that have at least one computationally expensive constraint function. For simplicity, we refer to them as computationally expensive CMOPs.

The second challenge is to handle the constraints so that we can generate feasible Pareto optimal solutions. In computationally expensive CMOPs, we typically have a limited function evaluation budget that should not be exceeded. Moreover, sometimes, function evaluations for infeasible solutions may take longer than feasible ones [9]. Therefore, evaluating too many infeasible solutions should be avoided. Moreover, generating solutions that the DM is not interested in is a waste of resources.

In addition, comparing many solutions to find the most preferred one can increase the cognitive load set on the DM. Therefore, it is important to generate solutions that reflect the DM's preferences. This is the third challenge. In this way, she/he only needs to compare a handful of solutions that are of interest.

There have been separate studies in the literature that address each of the three challenges mentioned in the previous paragraphs [12, 13]. However, there is currently no algorithm that addresses all three challenges simultaneously. This paper aims to fill this gap by introducing a novel algorithm for CMOPs that can handle all three challenges effectively.

Multiobjective evolutionary algorithms (MOEAs) are well-known algorithms to solve CMOPs [14]. They are population-based algorithms that find an approximation of the set of Pareto optimal solutions (also known as a Pareto front) and have certain advantages that have made them popular over the years. For example, they can handle different kinds of decision variables and deal with objective or constraint functions that are discontinuous or nondifferentiable [14, 15]. In the literature, many different MOEAs have been proposed, see, e.g. [16] and references therein. Among them, decomposition-based MOEAs have become prevalent [17] in recent years.

Decomposition-based MOEAs can scale better than other MOEAs as the number of objective functions increases [17, 18]. Moreover, decomposition-based algorithms can be modified to easily take the DM's preferences into account by focusing on a specific subspace of the objective space that represents the preferences [18]. On the other hand, all MOEAs need many function evaluations to converge toward Pareto optimality [19], and this makes them impractical for computationally expensive CMOPs. For example, according to [20], MOEAs require quite many function evaluations for some test problems.

To speed up computations, one can train computationally inexpensive surrogate models to mimic the behavior of the computationally expensive functions as closely as possible [19]. One then evaluates the expensive functions at new samples iteratively to update the accuracy of the surrogate models and we refer to this as *model management* (also known as active learning [21]). Kriging models (also known as Gaussian process regression) [22] are widely

used surrogate models [12, 19], because they provide uncertainty information of a predicted solution along with the predicted value. This information can be helpful to MOEAs and can be utilized in different ways for model management (see, e.g., [11, 23, 24]). In this paper, we focus on using Kriging in a decomposition-based MOEA.

There have been many studies on solving CMOPs when the constraints are computationally inexpensive, see, e.g., the survey [25] and recent studies [26–28]. Furthermore, as mentioned, e.g., in [29], most constraint handling techniques developed for single-objective optimization can be utilized in multiobjective optimization. However, in a recent survey [30], it is mentioned that this is not always straightforward. Moreover, some studies assume that the objective functions are computationally expensive, but the constraints are still considered to be computationally inexpensive [24, 31].

Although many engineering CMOPs can have computationally expensive objective and constraint functions (see e.g. [1, 2, 32]), there are only a handful of surrogate-assisted MOEAs that can handle them [32–38]. Many take a Bayesian approach for model management (we refer to it as Bayesian model management) based on an acquisition function that is used to select a solution for updating the surrogate models [36]. For example, in [39], an acquisition function is created based on expected hypervolume improvement [40] and the probability of feasibility (POF) of the solutions.

So far, we have discussed the first two challenges in solving CMOPs. The third challenge is to incorporate a DM's preferences. Few studies have incorporated a DM's preferences in the acquisition function [41–45]. In most surrogate-assisted MOEAs for computationally expensive CMOPs, the DM is assumed to select the most preferred solution after a representative set of Pareto optimal solutions has been generated (these are called a posteriori algorithms [46]). However, as mentioned earlier, we typically can only afford a limited number of function evaluations when solving computationally expensive CMOPs, and a posteriori algorithms tend to waste resources and also find solutions that the DM is not interested in. One can address this by asking for the DM's preferences before the optimization process (they are called a priori algorithms [46]). Here, by considering the DM's preferences, we need to find feasible and Pareto optimal solutions that satisfy the DM.

In this paper, we develop a novel Kriging-assisted a priori MOEA for computationally expensive CMOPs, called KAEA-C. To the best of our knowledge, KAEA-C is the first algorithm to incorporate the DM's preferences in problems with at least one computationally expensive constraint. We assume that the preference information is provided as a reference point consisting of desirable values for each objective function. We call the components of the reference point as aspiration levels.

The contributions of this paper are as follows: First, we propose a novel selection strategy for choosing a new population in each generation of the proposed algorithm, employing two distinct criteria for feasible and infeasible solutions. This strategy utilizes either surrogate models or original functions (if they are computationally inexpensive), enabling the generation of more feasible solutions that align with the DM's preferences while converging toward the Pareto optimal solutions.

Second, we devise a unique model management strategy that seeks two types of solutions: those that significantly improve the accuracy of surrogate models, and feasible solutions that closely adhere to the DM's preferences. Our selection strategy increases the likelihood of obtaining feasible solutions, making it more likely to find desired solutions during the model management phase, where we must evaluate expensive functions within a limited budget.

The remainder of the paper is organized as follows. Section 2 presents background material, concepts, and notations. In Sect. 3, we introduce the proposed algorithm KAEA-C. Section 4 is dedicated to numerical experiments, evaluating KAEA-C's performance against

state-of-the-art algorithms and analyzing the effects of KAEA-C's selection strategy and model management on its performance. Finally, Sect. 5 offers concluding remarks and discusses future research directions.

2 Background: basic concepts and notation

In this section, we cover some basic concepts, notation, and relevant terminology in multiobjective optimization that we need. Then, we provide background information about different ways to incorporate DM's preferences and Kriging-assisted MOEAs for CMOPs.

2.1 Multiobjective optimization

We consider problems of this form:

$$\begin{aligned} & \text{minimize } f(x) = (f_1(x), \dots, f_k(x)) \\ & \text{subject to } g_i(x) \geq 0, & i = 1, \dots, m \\ & x_i^l \leq x_i \leq x_i^u & i = 1, \dots, n, \end{aligned} \quad (1)$$

where $f(x)$ denotes an objective vector which consists of the values of k conflicting objective functions at $x = (x_1, \dots, x_n)^T$, an n -dimensional decision variable vector (for short, decision vector). In this paper, we refer to objective vectors as solutions. We call a decision vector x , and the corresponding solution $f(x)$ feasible, if x satisfies all the constraints. The set of all feasible decision vectors is called a feasible region \mathbb{F} . On the other hand, a decision vector and the corresponding solution are infeasible if x violates at least one of the constraints.

For the m inequality constraint functions, the individual constraint violation value $cv_i(x)$, with $i = 1, \dots, m$ of the decision vector x can be defined as follows:

$$cv_i(x) = \begin{cases} 0, & \text{if } g_i(x) \geq 0 \\ |g_i(x)| & \text{otherwise.} \end{cases} \quad (2)$$

We perform a min-max normalization [47] within the box constraints of problem (1), because the constraint functions can have different scales, and the normalization scales the magnitude of violations. In real-world problems, where we do not know the upper and lower bounds for the constraint violations, we can use the current population violations and update it iteratively. Then, we can calculate the sum of all individual constraint violations so that all constraints have an equal effect on the overall constraint violation value. The sum of all individual constraint violations provides the overall constraint violation for a given decision vector:

$$CV(x) = \sum_{i=1}^m cv_i(x). \quad (3)$$

We will use both Eqs. (2) and (3) in Sect. 3.

A feasible decision vector $x^* \in \mathbb{F}$ and the corresponding $f(x^*)$ are called Pareto optimal, if there does not exist another decision vector $x \in \mathbb{F}$ such that $f_i(x) \leq f_i(x^*)$ for all $i = 1, \dots, k$, and $f_j(x) < f_j(x^*)$ for at least one index j . A feasible decision vector $x^* \in \mathbb{F}$ and the corresponding $f(x^*)$ are called weakly Pareto optimal if there does not exist another feasible decision vector $x \in \mathbb{F}$ such that $f_i(x) < f_i(x^*)$ for all $i = 1, \dots, k$.

Table 1 The concepts (and their descriptions) used in this paper

Concept	Description
Iteration	A fixed number of generations. The model management takes place between iterations
Original functions	All the objective and constraint functions in the original CMOP
Expensive functions	Computationally expensive (objective or constraint) functions that we need to train surrogate models for
Expensive evaluation	Evaluating expensive functions at a decision vector
Expensive solution	A solution whose corresponding decision vector has been used for evaluating an expensive function
Surrogate evaluation	Evaluating surrogate functions at a decision vector
Surrogate solution	A solution whose corresponding decision vector has been used only for evaluating surrogate functions
Unconstraint ideal point $z^* = (z_1^*, \dots, z_k^*)$	This point is obtained by minimizing each of the objectives individually subject to the box constraints only
Utopian point $z^u = (z_1^u, \dots, z_k^u)$	The components of this point are formed by $z_i^* - \epsilon$ for $\epsilon > 1$ [48]
Nadir point $z^{nad} = (z_1^{nad}, \dots, z_k^{nad})$	This point is constructed by all the worst objective values on the Pareto front
Dystopian point $z^{dy} = (z_1^{dy}, \dots, z_k^{dy})$	The components of this point are formed by $z_i^{nad} + \epsilon$ for $\epsilon > 1$

Assume that the set $X = \{x^1, \dots, x^p\}$ is an arbitrary subset of feasible decision vectors, and $F(X) = \{f(x^1), \dots, f(x^p)\}$ is the set of corresponding solutions in the objective space. A solution $f(x^i)$, with $i = 1, \dots, p$, that satisfies the definition of Pareto optimality within the set $F(X)$, is called a nondominated solution [14]. Sometimes, nondominated solutions and Pareto optimal solutions are regarded as synonyms in the literature, but we distinguish the terms since MOEAs can only guarantee nondominance within a set considered but not Pareto optimality. A Pareto optimal solution is always nondominated, but the reverse situation is not necessarily true.

In this paper, we utilize several concepts and notations. We summarize them in Tables 1 and 2, respectively.

Moreover, we use an achievement scalarizing function (ASF) [49] to sort nondominated solutions based on a given reference point $\hat{z} \in \mathbb{R}^k$. There are different ways to formulate an ASF. Here, we use the following formulation to be minimized:

$$\max_{i=1, \dots, k} [w_i(f_i(x) - \hat{z}_i)] + \rho \sum_{i=1}^k w_i(f_i(x) - \hat{z}_i), \tag{4}$$

where $w_i = \frac{1}{z_i^u - z_i^{nad}}$, and $\rho \sum_{i=1}^k w_i(f_i(x) - \hat{z}_i)$ with $\rho > 0$ is an augmentation term to avoid finding weakly Pareto optimal solutions [46, 49]. We use an ASF to order solutions in a set. The lower the ASF value for a given x , the "closer" it is to the DM's reference point [46, 49]. We refer to this as how well the solution reflects the DM's preferences.

2.2 Preference incorporation

Multiobjective optimization algorithms can be classified based on the timing a DM provides preferences [46, 50]: after optimization (a posteriori algorithms), iteratively during the opti-

Table 2 Notation (and their descriptions) used in this paper

Notation	Description
$cv_i(x)$	Constraint violation of the i -th constraint at the decision vector x
$CV(x)$	Sum of constraint violations at the decision vector x
\hat{z}	Reference point provided by the DM
V	Set of reference vectors used to decompose the objective space
θ_i	Angle between the i -th solution vector and the reference point
$\gamma_{i,j}$	Angle between the i -th solution and the j -th reference vector
P_r	Randomly generated population
P_c	Population generated by optimizing individual constraint violations
P_f	Feasible population in P_c
P_0	Initial population
t_{max}	Maximum number of generations for each iteration
FE_{max}	Maximum number of expensive evaluations
FE	Expensive evaluation counter
N_u	Maximum number of expensive function evaluations per update
N_S	Maximum number of expensive solutions that the DM wants to see
A	Archive for storing expensive solutions and corresponding decision vectors
A_θ	Archive for storing the angles between feasible surrogate solutions and reference point
A_d	Archive for storing the distance between feasible surrogate solutions and unconstrained ideal point
t	Generation counter
Q_t	Offspring at generation t
P_t	Population at generation t
Y	Set of subpopulations created by V and P
N_V	Number of violated constraints for a given surrogate solution
N_T	$N_V + N_{CV}$
\hat{U}_j	Individual uncertainty information for a solution
A_{new}	Archive for storing unique surrogate solutions to be used for expensive evaluations

mization (interactive algorithms), or before optimization (a priori algorithms). In a posteriori algorithms, the DM selects the most preferred solution after seeing a set of solutions representing the Pareto front. The DM actively interacts with the algorithm in the second class of algorithms and provides preferences during an iterative solution process. In a priori algorithms, the DM expresses one's preferences before the solution process. Then, at the end of the optimization process, the DM gets a set of solutions that best reflect the preferences.

A posteriori algorithms need a lot of computing resources since they try to represent the whole range of different Pareto optimal solutions. Therefore, the computation resources can go to waste by generating solutions that do not interest the DM. These algorithms are suitable when the DM wants to see a wide range of trade-offs. On the other hand, the DM is actively involved in the optimization process in interactive algorithms, which needs time and involvement. The advantages of interactive algorithms are that the DM can learn about the reachable solutions and adjust her/his preferences iteratively. In a priori algorithms, the DM gets to express her/his preferences before the optimization begins and, thereby, limits the region of interest and chooses the most preferred solution at the end. The drawback of

a priori algorithms is that the DM may provide unrealistic preferences and be disappointed in the solutions obtained. A priori algorithms are suitable when the DM cannot interactively provide preferences but has knowledge of what kind of solutions are desirable.

In a priori algorithms, computation resources are saved compared to a posteriori algorithms. Besides, the cognitive load set on the DM is not as extensive since she/he needs to consider only a set of Pareto optimal solutions in her/his region of interest. To the best of our knowledge, there are no a priori or interactive surrogate-assisted MOEAs designed for CMOPs with computationally expensive constraints. In this paper, we focus on a priori algorithms. In this way, we avoid shortcomings of a posteriori algorithms and avoid making assumptions on the DM having much time to participate in an interactive solution process.

2.2.1 A priori decomposition-based MOEAs

The current decomposition-based MOEAs can easily be adapted to handle a priori preferences. With minor adjustments, most of them can utilize a DM's preferences to decompose the objective space into several subspaces [13]. For example, the weights in MOEA/D [51], the reference vectors in RVEA [52], the reference points in NSGA-III [53], and many more MOEAs (see [54] for more details) can be adjusted to incorporate the DM's preferences.

We can modify any of the above-mentioned algorithms to incorporate a DM's reference point. In this paper, we use the reference vectors (used by RVEA) as an example. Besides, in two recent studies [13, 18], it has been explicitly mentioned that among decomposition-based MOEAs, RVEA has a straightforward way for incorporating a DM's preferences. DMs can express preferences in different ways [55]. We use reference points as preference information because they have been proven to be something that is understandable to the DM [56, 57] (they are in the objective space like objective vectors that a priori MOEAs generate and show to the DM).

2.2.2 Reference vectors

Next, we describe how reference vectors are used in RVEA since we use them in the proposed algorithm KAEA-C. RVEA uses uniformly distributed reference vectors (using the simplex-lattice design algorithm [58]) to divide the objective space into subspaces. Then, solutions are assigned to the closest reference vectors in each generation. Each set of solutions assigned to a reference vector is called a subpopulation. Next, a scalarization function is used to form a single-objective optimization problem. Here, we refer to the scalarization function as a fitness function. Finally, the best solution is selected for the next generation by solving the single-objective optimization problems in each subpopulation.

Moreover, RVEA has an a priori extension (we refer to it as AP-RVEA) that incorporates a reference point \hat{z} [52]. AP-RVEA positions the reference vectors v^i ($i = 1, \dots, p$) based on a normalized reference vector $v^c = \{v_1^c, \dots, v_j^c\}$ (for $j = \{1, \dots, k\}$) according to the following equation:

$$v^i = \frac{r \cdot v^i + (1 - r) \cdot v^c}{\|r \cdot v^i + (1 - r) \cdot v^c\|}, \quad (5)$$

where $v_j^c = \frac{\hat{z}_j}{\|\hat{z}\|}$, and $\|\hat{z}\| \geq 0$ is the Euclidean norm of the reference point. If $\|\hat{z}\| = 0$, then we set v^c to be the unit vector. The parameter $r \in (0, 1)$ controls how the reference vectors are adjusted towards the reference point. If r is close to 1, then the reference point has less effect on the reference vectors, and if it is close to 0, they will get closer to the reference point.

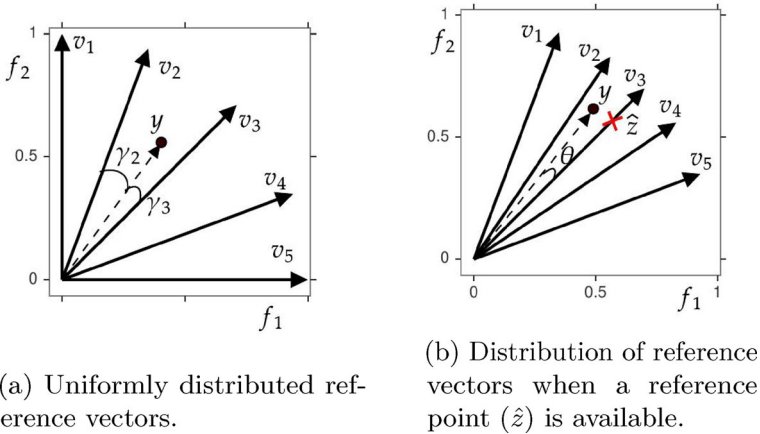


Fig. 1 Example of how reference vectors are distributed **a** uniformly and **b** when we incorporate a reference point \hat{z} (red cross)

Moreover, the cosine value of the angle γ of a solution y and a reference vector v can be used to measure the angle-based distance between them. We calculate this by the following equation:

$$\cos \gamma = \frac{y \cdot v}{\|y\| \cdot \|v\|}. \tag{6}$$

Assume $V = \{v_1, \dots, v_5\}$ is a set of reference vectors. Figure 1a demonstrates how V is uniformly distributed in the objective space. Here, we can observe that y should be assigned to either v_2 or v_3 . First, we calculate the angles (γ_2 and γ_3) between y and these two vectors. Then, we assign y to the reference vector with the smallest angle, which is v_3 .

In Fig. 1b, we observe how the reference vectors are distributed if a reference point \hat{z} is provided. Here θ is the angle between y and \hat{z} . We will use θ later in Sect. 3 for the selection strategy in KAEA-C. It is worth mentioning that the reference point only provides a search direction \hat{z} and does not matter if it is attainable or unattainable

2.3 Kriging-assisted MOEAs

As mentioned in Sect. 1, we can use surrogate models which are computationally inexpensive to evaluate to replace expensive functions. Naturally, we only fit surrogate models to computationally expensive functions.

Moreover, we use Kriging models as surrogates because of the uncertainty information that they provide [12, 33]. One of the essential functionalities of uncertainty information is that it helps in managing the Kriging models. There are different types of model management in the literature [13]. For example, in [23], the surrogate solutions that have the highest uncertainty are selected to update the Kriging models because the global accuracy of the Kriging models is important in that work. On the other hand, in [11], the surrogate solutions with the lowest uncertainty are chosen because the DM's preferences are involved, and it is important to make sure some expensive solutions follow the preferences.

Another type of model management in Kriging-assisted MOEAs is the Bayesian model management. In Sect. 4, we compare our proposed model management to a Bayesian

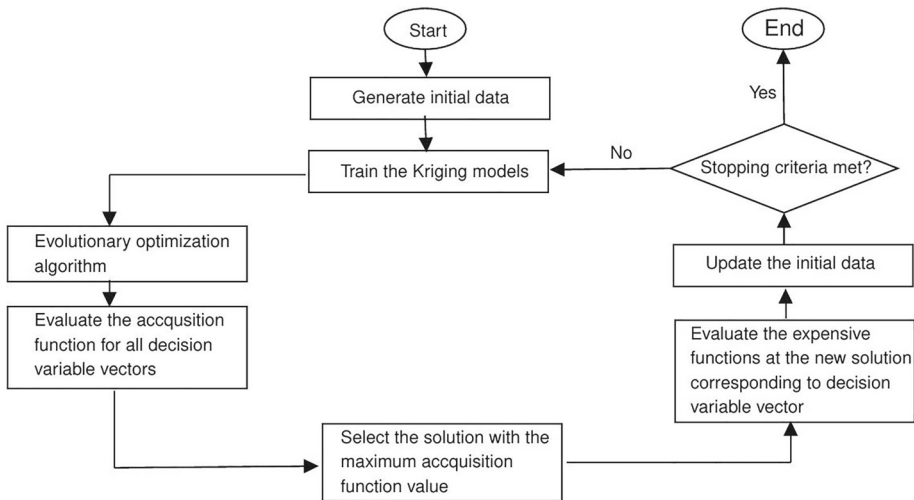


Fig. 2 A diagram of Bayesian evolutionary optimization

Table 3 A list of surrogate-assisted MOEAs and weather they can handle expensive constraints, DM’s preferences and if they have been applied to real-world problems

Paper	Handling expensive constraints	Handling preferences	Real-world problems
[32, 35, 38]	✓	✗	✓
[34, 37, 39, 40]	✓	✗	✗
[41, 43]	✗	✓	✗
[42, 44, 45, 60]	✗	✓	✓

For each row, the sign ✓ indicates that the corresponding header has been considered in the paper, and the sign ✗ indicates it has not

approach. Because of this reason, we outline the basics of Bayesian evolutionary optimization [59] in this subsection. In Bayesian evolutionary optimization, surrogate models are trained for objective and constraint functions and optimized using an evolutionary algorithm. Then, an acquisition function is created [13]. Next, the acquisition function value for each decision vector in the offspring population is calculated. Finally, the decision vectors with the maximum acquisition function values are chosen to update the surrogate models.

An acquisition function can incorporate different criteria. For example, the expected hypervolume improvement for unconstrained problems or the combination of the POF and expected hypervolume improvement for constrained problems can be utilized in an acquisition function. Figure 2 provides a diagram of the main steps of Bayesian evolutionary optimization.

Table 3 summarizes surrogate-assisted MOEAs in the literature that handle constraints or incorporate DM’s preferences. None of them is able to handle both computationally expensive constraints and incorporate preferences.

3 KAEA-C

In this section, we introduce a novel Kriging-assisted a priori multiobjective evolutionary algorithm with the ability to handle computationally expensive objective and constraint func-

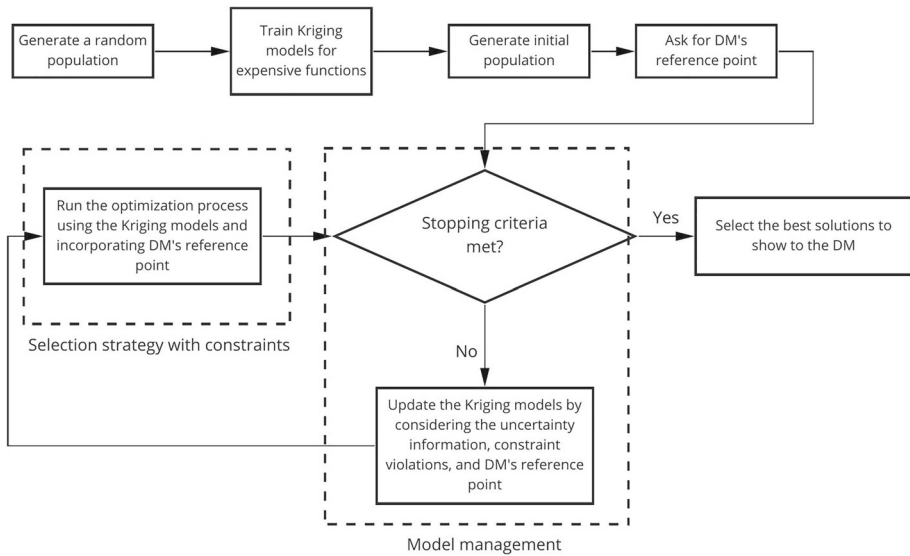


Fig. 3 Flowchart of KAEA-C

tions, called KAEA-C. Moreover, we assume the following: the DM provides the maximum number of expensive solutions, N_S , that she/he wants to see at the end of the optimization process, we have at least one computationally expensive constraint and a limited budget for expensive function evaluations.

The main idea of KAEA-C is to find feasible expensive solutions reflecting the DM's reference point. Here, we have three main phases: initialization, selection strategy, and model management. The novelty of KAEA-C lies in the latter two phases. In the selection strategy phase, we use Kriging models and focus on generating feasible surrogate solutions. On the other hand, in the model management phase, our goal is to generate feasible expensive solutions that improve the Kriging model's accuracy or nondominated feasible expensive solutions that follow the DM's reference point. Figure 3 provides a flowchart of the main steps of KAEA-C.

In the initialization phase, we train surrogate models for the expensive functions. Additionally, it is important to start the optimization process with some feasible surrogate solutions in the initial population [31]. To find some feasible surrogate solutions, we simultaneously minimize the individual constraint violation for each constraint (equation (2)).

We introduce a novel selection strategy by using two fitness functions and prioritizing the feasible surrogate solutions. In the selection strategy, we calculate the angle between surrogate solutions and the reference point (θ in Fig. 1b) as the first fitness function in each subpopulation and the distance between surrogate solutions to the unconstrained ideal point as the second fitness function. Then, we select two types of surrogate solutions to update the Kriging models in model management. Firstly, we select surrogate solutions that can improve the accuracy of the Kriging models the most (they have a high uncertainty). Secondly, we select some feasible surrogate solutions that follow the DM's reference point the best (by using (4)) and have a low uncertainty to update the Kriging models. We repeat these three phases iteratively until the budget of expensive evaluations runs out.

3.1 Description of the KAEA-C algorithm

Algorithm 1 shows the main steps of KAEA-C. The input of Algorithm 1 are as follows: t_{max} is the maximum number of generations in each iteration, FE_{max} is the budget of expensive evaluations, N_u is the maximum number of surrogate solutions that we choose to update the Kriging models, and $|P_r|$ is the size of a randomly generated population. In addition, we ask the DM to provide the reference point \hat{z} and N_S , which is the maximum number of expensive solutions that she/he wants to see. Note that we ask the DM for the upper bound N_S based on her/his cognitive capacity.

As mentioned earlier, in the initialization phase, we minimize individual constraint violations of each constraint that was defined in (2) to find feasible surrogate solutions and add their corresponding decision vectors to a randomly generated population P_r . Moreover, we have an archive A to store all expensive solutions and their corresponding decision vectors. Next, we train Kriging models for each expensive function. We continue optimization with the Kriging models and original functions that are computationally inexpensive. Then, in each generation t , we generate an offspring population Q_t using evolutionary operations crossover and mutation. To select the population P_{t+1} for the next generation, we propose a novel selection strategy which is described in Algorithm 2, where we demonstrate how to push the algorithm to generate more feasible surrogate solutions.

In the model management, we select two types of surrogate solutions to evaluate the expensive functions at them: The surrogate solutions that can improve the accuracy of the Kriging models the most, and feasible expensive solutions that reflect the DM's reference point the best. Then, we store these expensive solutions in the archive A and update the Kriging models.

Lastly, after we have used the whole budget of expensive evaluations (when $FE = FE_{max}$), we must select at most N_S expensive solutions from the archive A to show to the DM. The selected expensive solutions must be feasible, nondominated, and they should follow the DM's reference point. As we mentioned in Sect. 2, we use the ASF (equation (4)) to determine how well a solution reflects the DM's reference point. Therefore, we calculate the ASF values for feasible nondominated expensive solutions in the archive A . Then, we select up to N_S expensive solutions with the lowest ASF values to show to the DM.

3.2 Initialization

In KAEA-C, the optimization process is performed on the Kriging models and the computationally inexpensive functions. It is important to start the optimization process with an initial population with some feasible surrogate solutions [31].

First, we generate the random population P_r (e.g., by using Latin hypercube sampling [61]). The size of P_r is denoted as $|P_r|$. Then, we evaluate the expensive functions at the decision vectors in P_r and store their original function values in the archive A (along with the corresponding decision vectors). Next, we use the archive A to train Kriging models for expensive functions.

To increase the likelihood that we have some feasible decision vectors in the initial population P_0 , we first formulate the following multiobjective optimization problem:

$$\begin{aligned} & \text{minimize } \{cv_1(x), \dots, cv_m(x)\} \\ & \text{subject to } x_i^l \leq x_i \leq x_i^u, \quad i = 1, \dots, n, \end{aligned} \quad (7)$$

where we simultaneously minimize the individual constraint violations in problem (1), and x is a decision vector with the same box constraints as in problem (1). Note that the individual constraint violations in problem (7) are calculated with regard to the Kriging models or original functions that are computationally inexpensive. In step 5 of Algorithm 1, we solve problem (7) by an MOEA, which is appropriate for this problem. After solving problem (7), we have the final population P_c and we select feasible decision vectors P_f satisfying Kriging models of the constraints from P_c and combine the decision vectors in P_f and P_r to create P_0 . Finally, we remove the duplicate decision vectors from P_0 .

Moreover, to save computation resources, we do not evaluate expensive functions at P_f . Therefore, we have two types of decision vectors in P_0 , those used for expensive evaluations (they could be feasible or infeasible) and those that satisfy the surrogate constraints. In addition, we assume that we can solve problem (7) and generate some feasible surrogate solutions. Discussing the case where we cannot find any feasible surrogate solutions is beyond the scope of this paper.

Algorithm 1: KAEA-C

Input: t_{max} = maximum number of generations for each iteration, FE_{max} = maximum number of expensive evaluations, N_u = maximum number of expensive function evaluations per update, $|P_r|$ = size of random population, N_S = maximum number of expensive solutions that the DM wants to see, \hat{z} = reference point provided by the DM **Result:** Up to N_S feasible expensive solutions reflecting a given reference point

/ Initialization */*

- 2 Initialize expensive evaluation counter $FE = 0$, generation counter $t = 0$, an archive $A = \emptyset$, P_r = a randomly generated population of size $|P_r|$;
- 3 Evaluate expensive functions at P_r ;
- 4 Store P_r and corresponding expensive solutions in the archive A ;
- 5 Train Kriging models for expensive functions using archive A ;
- 6 Solve problem (7) by using an MOEA and store the final decision vectors in P_c ;
- 7 Select the feasible decision vectors (w.r.t. surrogate values) in P_c and store them in P_f ;
- 8 Combine the decision vectors in P_f and P_r and store them in P_0 ;
- 9 Remove duplicate decision vectors from P_0 ;

10 **while** $FE < FE_{max}$ **do**

- 11 **while** $t \leq t_{max}$ **do**
- 12 $Q_t = \text{offspring-creation}(P_t)$;
- 13 $P_t = P_t \cup Q_t$;
- /* Selection strategy */*
- 14 Use Algorithm 2 to generate P_{t+1} ;
- 15 $t = t + 1$
- 16 **end**
- /* Model management */*
- 17 Use Algorithm 3 to update the Kriging models, archive A , and FE ;
- 18 $t = 0$;

19 **end**

- 20 Select maximum N_S feasible expensive solutions from the archive A that follow \hat{z} the best based on (4), and show them to the DM.

3.3 Selection strategy

In the selection strategy of KAEA-C, we generate the next population while using the Kriging models instead of the expensive functions. Typically, in decomposition-based MOEAs, we select one decision vector per subproblem based on a fitness function [17, 18]. This selection strategy can bring up some issues when incorporating DM's reference point in computationally expensive problems. For instance, if very few subproblems generate the majority of feasible surrogate solutions, we do not have enough feasible surrogate solutions to select in the model management for updating the Kriging models. In addition, this can lead to not having enough feasible expensive solutions to show to the DM. In the proposed KAEA-C, we increase the number of feasible surrogate solutions generated by appointing two fitness functions and selecting a set of nondominated solutions based on the two fitness functions in each subproblem.

We take two steps to maximize our selection strategy's likelihood of generating enough feasible surrogate solutions to show to the DM. First, we separate infeasible and feasible surrogate solutions. Second, we provide a unique selection algorithm for generating P_{t+1} .

In each subproblem, our priority is to select feasible surrogate solutions. Here, we create a bi-objective subproblem with two fitness functions for feasible surrogate solutions, and we select a set of nondominated surrogate solutions based on the two fitness functions. However, if all the surrogate solutions are infeasible, we use a ranking system that considers the overall constraint violation and the number of violated constraints. Algorithm 2 shows the main steps of our selection strategy.

3.3.1 Dealing with feasible surrogate solutions

If feasible surrogate solutions exist for any subpopulation, we only consider them in the selection strategy. We are interested in generating feasible surrogate solutions, and one way to focus on that is to increase the number of feasible surrogate solutions when the next population P_{t+1} is being generated. Then, crossover and mutation have a higher chance of generating feasible surrogate solutions. Therefore, we aim to select only feasible surrogate solutions if possible.

We use two fitness functions. The first is the distance d of the surrogate solutions to the unconstrained ideal point z^* , and the second fitness function is the angle θ between feasible surrogate solutions and the reference point \hat{z} (see Fig. 1b). We can calculate θ by replacing v with \hat{z} in (6). Therefore, d and θ are the two fitness functions we use in our novel selection strategy, where we select nondominated surrogate solutions based on them.

Figure 4 illustrates how our selection strategy works for feasible surrogate solutions in a bi-objective case. In Fig. 4a, $\{y_1, \dots, y_5\}$ are feasible surrogate solutions of a subpopulation (black dots), \hat{z} is the reference point (red cross), and z^* is the unconstrained ideal point (blue square). After calculating the fitness function values d and θ for all feasible surrogate solutions, we can identify the nondominated solutions based on these two fitness functions (see Fig. 4b). In this example, based on Fig. 4b, it is clear that y_2 , y_3 , and y_4 are the nondominated surrogate solutions for the next generation.

Since we select a set of solutions for each subproblem, the computation time for optimizing the surrogate models would increase compared to typical decomposition-based MOEAs. However, thanks to the low computation time of surrogate evaluations, we can afford to generate more surrogate solutions during the selection strategy. In return, we will have more candidate solutions to choose from in the model management, and therefore, more likely find feasible expensive solutions that reflect the DM's preferences.

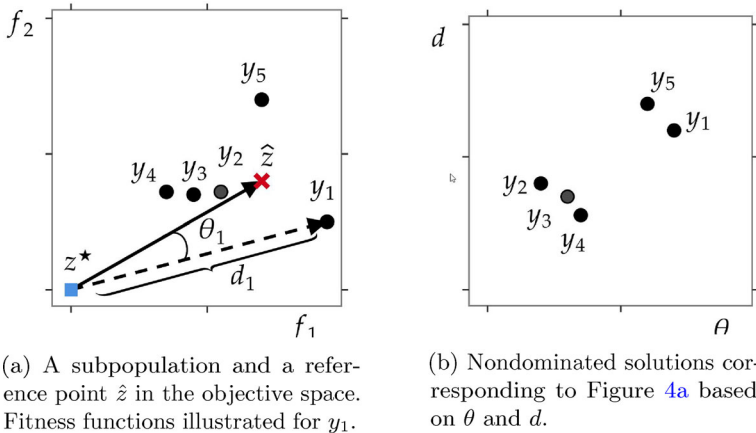


Fig. 4 An example of the selection strategy for feasible surrogate solutions described in Algorithm 2

Table 4 An example of how an infeasible surrogate solution is selected

	g_1	g_2	g_3	CV	N_V	N_{CV}	N_T	R
y_1	0	0.9	0	0.9	0	2	2	1
y_2	0.5	0.2	0	0.7	1	0	1	0
y_3	0.25	0	0.8	1.05	1	3	4	2
y_4	0.2	0.4	0.4	1	3	1	4	2

3.3.2 Dealing with infeasible surrogate solutions

In case all the surrogate solutions in a subpopulation are infeasible, we focus on moving toward the feasible region to generate feasible surrogate solutions in the next generation. Here, we create a ranking system by considering two factors, the overall constraint violation, which is calculated by equation (3), and the number of constraints that are violated. In each subpopulation, we pick the infeasible surrogate solution with the lowest rank for the next generation. If two infeasible surrogate solutions have the same rank, we select one of them randomly.

Assume the surrogate solutions in $\{y_1, \dots, y_q\}$ are infeasible. To select the best surrogate solution for the next generation, we define:

$$N_T(y_j) = N_V(y_j) + N_{CV}(y_j), \tag{8}$$

where $j = 1, \dots, q$, $N_V(y_j)$ is the number of surrogate solutions that violate fewer constraints than y_j , $N_{CV}(y_j)$ is the number of surrogate solutions that have a lower overall constraint violation value than y_j , and their sum is denoted by N_T . We rank all infeasible surrogate solutions based on $N_T(y_j)$ and refer to their rank as $R(y_j)$. Note that the rank 0 represents the best infeasible solution.

Table 4 shows an example of how the ranking system works for four infeasible solutions $\{y_1, \dots, y_4\}$. Here, y_2 is the best surrogate solution because there is only one surrogate solution that violates fewer constraints ($N_V(y_2) = 1$), and none of the infeasible surrogate solutions have a lower overall constraint violation ($N_{CV}(y_2) = 0$). Therefore, y_2 has the lowest rank ($R(y_2) = 0$), and it is selected for the next generation.

Algorithm 2: Selection strategy

Input: The current population P_t , h = number of reference vectors, unconstrained ideal point, and reference point \hat{z}

Output: The next population P_{t+1} ;

- 1 Initialize $P_{t+1} = \emptyset$, $A_\theta = \emptyset$ to store the angles between feasible surrogate solutions and \hat{z} , and $A_d = \emptyset$ to store the distance between feasible surrogate solutions and the unconstrained ideal point;
- 2 Generate reference vectors $V = \{v_1, \dots, v_h\}$ using (5);
- 3 Generate subpopulations $Y = \{Y_1, \dots, Y_h\}$, using V and P_t ;
- 4 **for** $i \leftarrow 1$ **to** h **do**
- 5 **if** all surrogate solutions in Y_i are infeasible **then**
- 6 Use (8) to calculate the rank of all infeasible surrogate solutions in Y_i ;
- 7 select the surrogate solution with the lowest rank and add it to P_{t+1} ;
- 8 **else**
- 9 **if** number of feasible surrogate solutions == 1 **then**
- 10 Select the only feasible surrogate solution for v_i and add it to P_{t+1} ;
- 11 **else**
- 12 Calculate the angle between feasible surrogate solutions and \hat{z} and store the values in A_θ ;
- 13 Calculate the distance between feasible surrogate solutions and the unconstrained ideal point and store the values in A_d ;
- 14 Select the nondominated surrogate solutions based on A_d and A_θ for Y_i and add them to P_{t+1} ;
- 15 **end**
- 16 **end**
- 17 **end**
- 18 Return P_{t+1} ;

In Algorithm 2 we use the current population (P_t), the number of reference vectors (h), the unconstrained ideal point, and \hat{z} as the input. If an unconstrained ideal point is not available, it can be estimated at every generation from the current population. Next, we generate the reference vectors by using (5). In step 3 of Algorithm 2, we generate the set of subpopulations Y by assigning each surrogate solution to the closest reference vector. Then, we look for feasible surrogate solutions in each subpopulation. For a subpopulation Y_i , if there is no feasible surrogate solution, we use (8) to select the best infeasible surrogate solution for this subpopulation.

On the other hand, when feasible surrogate solutions exist in Y_i , then two cases can happen. If there is only one feasible surrogate solution, it is selected as the best one for the current subpopulation. The second case is when more than one feasible surrogate solutions exist in a subpopulation. Here, we calculate the distance A_d of feasible surrogate solutions to the unconstrained ideal point and the angle A_θ between feasible surrogate solutions and \hat{z} . Then, based on A_θ and \hat{z} we select nondominated feasible surrogate solutions and store them in P_{t+1} .

3.4 Model management

After each iteration, we update the Kriging models. Typically, in Kriging-assisted optimization algorithms, the surrogate solutions with the highest uncertainty (standard deviation of the

predicted value) are chosen to update the Kriging models since they can potentially improve the most with these surrogate solutions [23]. It is essential to evaluate expensive functions at some decision vectors with a high uncertainty (regardless of whether they are feasible or infeasible). As mentioned in [13], assume a decision vector x' is infeasible regarding the Kriging models of the constraints. If x' has a high uncertainty, then its corresponding expensive solution may be feasible. Therefore, if we do not evaluate the expensive functions at such decision vectors, the algorithm may not identify the feasible region correctly, and the search direction may be misleading.

In addition, we are interested in finding feasible expensive solutions reflecting the DM's preferences. As mentioned earlier, the most common approach is to use a Bayesian evolutionary optimization algorithm that uses an acquisition function that considers some criteria to select some surrogate solutions to update the surrogate models. However, in our case, creating an acquisition function that can consider selecting surrogate solutions with a high uncertainty and feasible surrogate solutions with a low uncertainty that follow the DM's preferences could be complicated. Therefore, we select surrogate solutions that have a high uncertainty and ignore whether they violate constraints or not. We select some of the feasible surrogate solutions reflecting the DM's reference point and having a low uncertainty to increase the likelihood of finding such expensive solutions separately. The main steps of the model management are shown in Algorithm 3.

In case some of the functions are inexpensive, we do not train any Kriging models for them. Therefore, the uncertainty for these objectives is zero. Here, for a surrogate solution y , we define the uncertainty information U_y as:

$$U_y = \prod_j^{\bar{m}} \frac{u_j - u_{jmin}}{u_{jmax} - u_{jmin}} \leq 1, \quad (9)$$

where, \bar{m} is the number of expensive functions, u_j is the uncertainty information from the j -th Kriging model for y , and u_{jmin} and u_{jmax} are the minimum and maximum uncertainties of the j^{th} corresponding Kriging function, respectively. Note that, the values of u_{jmin} and u_{jmax} are updated after each iteration.

As input, Algorithm 3 needs FE and FE_{max} from Algorithm 1, the current population, N_u , and the archive A from Algorithm 1. In steps 1–3, we check how many expensive evaluations are left. In this paper, improving the Kriging models' accuracy is as important as finding feasible expensive solutions that follow the DM's preferences. Therefore, we spend half of N_u on expensive evaluations that can improve the Kriging models, and the other half on expensive evaluations that can lead to finding feasible expensive solutions reflecting the DM's reference point.

To update the Kriging models, we select $N_u/2$ surrogate solutions with the highest uncertainty to improve the Kriging models in areas with the most potential of improving their accuracy and store them in A_u . Then, we select N_u feasible surrogate solutions with the lowest ASF values and store them in A_{asf} . Next, we select $N_u/2$ surrogate solutions with the lowest uncertainty values from A_{asf} and store them in A_u . After that, we evaluate the expensive functions at the corresponding decision vectors of A_u that are not in the archive A . Finally, we add all new expensive solutions to the archive A , update FE , and train the Kriging models again with the archive A . Note that, in case N_U is an odd number, use $N_U - 1$ as the maximum number of expensive evaluations per update.

As we mentioned earlier, in the selection strategy, we select a set of nondominated surrogate solutions based on their distance to the unconstrained ideal point and their angles with the reference point. Moreover, the proposed selection strategy complements the proposed

Algorithm 3: Model management

Input: FE , FE_{max} , current population P_{tmax} , maximum number of expensive function evaluations per update N_u , and updated archive A from Algorithm 1

Output: updated Kriging models for expensive functions, status of stopping criterion u , and the archive A ;

- 1 Initialize archives $A_u = \emptyset$ to store candidate surrogate solutions for updating the Kriging models, $A_{asf} = \emptyset$ to store surrogate solutions with lowest ASF values, and $A_{new} = \emptyset$ to store new expensive solutions;
- 2 **if** $N_u + FE \geq FE_{max}$ **then**
- 3 | $N_u = FE_{max} - FE$;
- 4 **end**
- 5 Select $N_u/2$ surrogate solutions with the highest uncertainty from P_{tmax} , and store them in A_u ;
- 6 Select N_u feasible surrogate solutions from $\{P_{tmax} - A_u\}$ with the lowest ASF values and store them in A_{asf} ;
- 7 Select $N_u/2$ surrogate solutions from A_{asf} with the lowest uncertainty value, and store them in A_u ;
- 8 Use the surrogate solutions in A_u that do not belong to A to evaluate expensive functions and store them in A_{new} . ;
- 9 Add the expensive solutions in A_{new} to the archive A ;
- 10 Use the archive A to train all Kriging models for all expensive functions;
- 11 Update $FE = FE + |A_{new}|$;
- 12 Return Kriging models, FE , and the archive A

model management by enabling us to spend the available limited budget for expensive evaluations wisely in two main ways. First, sometimes one subproblem can lead to a set of feasible surrogate solutions that follow the DM's reference point better than other subproblems. In a typical decomposition-based MOEA's selection strategy, only one of these solutions makes it to the next generation, and this can lead to a lack of surrogate solutions for expensive evaluations in the model management phase. On the other hand, in our proposed selection strategy, we keep all the surrogate solutions that follow the DM's preferences better than other surrogate solutions, and this helps the model management select the best surrogate solution available for expensive evaluations. The second way that the selection strategy complements the model management is when most of the surrogate solutions are assigned to one reference vector. Here, if we choose only one solution to pass on to the next generation, it is possible not to have enough surrogate solutions for the model management phase to use for expensive evaluations. However, by selecting a set of nondominated solutions, we increase the number of solutions we can choose from in the model management phase.

4 Numerical experiments

In this section, we demonstrate the performance of the proposed KAEA-C algorithm. We first compare it with some existing algorithms. Moreover, we analyze the effect of Algorithms 2 and 3 on the performance of KAEA-C. In the evaluation, we use 12 problems and two performance indicators.

4.1 Algorithms compared with

As mentioned in Sect. 1, KAEA-C is the first algorithm for the purpose considered. However, we can adjust some existing algorithms in the literature for comparison. First, we use a random search approach as our baseline algorithm. Next, we chose two algorithms from the RVEA

family, which had competitive results with many MOEAs on different benchmark problems [52] as examples. Here, we selected AP-RVEA and interactive K-RVEA (IK-RVEA) [11] for comparison. They both use (5) to decompose the objective space and employ a fitness function called *angle penalized distance* [52] to select the best solution for each sub-space. In the following, we adapt these algorithms to handle computationally expensive constrained problems.

To modify AP-RVEA (which was not originally proposed for expensive problems), we train Kriging models for expensive functions and, in the end, randomly select some of the surrogate solutions generated by AP-RVEA for expensive evaluations. For IK-RVEA, we use the same reference point for every iteration to treat it as an a priori algorithm. Additionally, we need to implement a constraint handling technique for IK-RVEA. In this case, we apply AP-RVEA's constraint handling technique. Thus, the key difference between IK-RVEA and AP-RVEA lies in IK-RVEA's model management. To highlight the importance of proper model management, we do not modify these two algorithms in this aspect. AP-RVEA does not have model management, and we do not create one for it. Conversely, IK-RVEA possesses model management that considers the DM's preferences, but it has not been designed for computationally expensive constraints. We follow the suggestions in the original papers to set their parameters for the problems.

4.2 Using different components

The majority of KAEA-C's novelty lies within the selection strategy (Algorithm 2) and the model management (Algorithm 3). These two components have a modular structure, which means that we can combine different model management and selection strategies to assemble new algorithms.

For the selection strategies, we choose the RVEA's selection strategy (referred to as RVEA^{SS}) since it has a competitive performance compared to other algorithms according to [52], and it is straightforward to reflect the DM's preferences [13]. Additionally, we also choose the R-NSGA-III [62] selection strategy (R-NSGA-III^{SS}) to be able to assemble more algorithms.

For model management, we need an algorithm that is suitable for problems with expensive objectives and constraints. Model managements used in Bayesian optimization usually have this property [13]. In this paper, we use the model management used in an algorithm called BMOO [63], which uses the POF and expected hypervolume improvement to create an acquisition function. Then, in each iteration, the solution with the highest acquisition function value is selected to update the surrogate models. We refer to this model management as BMOO^{MM}. We choose BMOO^{MM} because it efficiently performed on multiple benchmark problems in [33, 63]. Finally, we combine the model management used in [41] with POF to create a new model management that can handle DM's preferences along with computationally expensive constraints.

Surrogate-assisted evolutionary algorithms typically feature a modular design, dividing functionalities between model management and selection strategy. This modularity facilitates the theoretical assembly of novel algorithms through the integration of diverse selection strategies and model management modules from various evolutionary algorithms. In this work, we leverage this flexibility to assemble eight distinct algorithms, succinctly denoted as assembled algorithms (AAs), AA1 through AA8. They are formed by combining the selection strategy from KAEA-C (Algorithm 2) and the model management from Algorithm 3 with external modules such as the AP-RVEA selection strategy and BMOO model management.

Table 5 Assembled algorithms by using different selection and model management strategies

	KAEA-C	AA1	AA2	AA3	AA4	AA5	AA6	AA7	AA8
Selection strategy									
Algorithm 2	✓	✓	✓	✗	✗	✗	✗	✗	✗
RVEA ^{SS} [52]	✗	✗	✗	✓	✓	✓	✗	✗	✗
R-NSGA-III ^{SS} [62]	✗	✗	✗	✗	✗	✗	✓	✓	✓
Model management									
Algorithm 3	✓	✗	✗	✓	✗	✗	✓	✗	✗
BMOO ^{MM} [63]	✗	✓	✗	✗	✓	✗	✗	✓	✗
TEHVI [41]+POF	✗	✗	✓	✗	✗	✓	✗	✗	✓

The sign ✓ indicates the selection and model management strategy that is used for the corresponding assembled algorithm

The specific configurations of these assembled algorithms are detailed in Table 5, providing a clear depiction of the module combinations in use. It is imperative to emphasize that the primary objective of this endeavor is not the creation of new algorithmic entities, but rather a comprehensive assessment of the performance characteristics of Algorithms 2 and 3, ensuring a robust evaluation of their capabilities in diverse situations.

Note that there are many different selection strategies and model managements in the literature that we can use to assemble new algorithms (e.g., see [13]). However, in this research, our goal is to show a gap in the literature for algorithms like KAEA-C. Analyzing all the algorithms that can be assembled is outside of the scope of this research.

4.3 Problems considered

There are not many constrained benchmark problems with more than two objective functions in the literature [64]. Constrained versions of well-known benchmark problems DTLZ [65] (referred to as CDTLZ) are introduced in [66]. We begin with a relatively simple problem, C2DTLZ2, to examine whether all the algorithms can successfully solve it. Furthermore, we choose C3DTLZ4, which according to [67], is one of the few problems in the CDTLZ family that genuinely requires a constraint-handling technique. For this reason, we select C3DTLZ4 from the CDTLZ family. In addition to the CDTLZ family, we also utilize some test problems introduced in [68]. Here, the problems named “MW4”, “MW8”, and “MW14” are CMOPs (single constraint) with a scalable number of objectives [68].

We use a total of 14 problems to evaluate the performance of KAEA-C. In Table 6, we display their dimensions and references. The superscript in the problem name refers to the number of objectives. In the C3DTLZ4 problem, the objective functions are the same as in DTLZ4, but the constraints make DTLZ4’s Pareto front infeasible. We use C3DTLZ4 with three objectives (C_3^3DTLZ4) and seven objectives (C_3^7DTLZ4) and assume that all functions are computationally expensive. In addition to the benchmark problems, we also employ four real-world problems, which include conceptual marine design (CMD) [69], the car-side impact (CSI) problem [66], the water resource (WR) problem [70], and the multiple-disk clutch brake design (MDCBD) problem [71]. In the first two problems, we assume that all functions are computationally expensive. However, for the MDCBD problem, we consider the last three objectives inexpensive since they are individual decision variables. The formulations of all these problems can be found in the supplementary materials.

Table 6 The number of objectives (k), constraints (m), and decision variables (n) in the problems considered with references

	k	m	n
$C_2^3 DTLZ2$ [66]	3	3	7
$C_2^7 DTLZ2$ [66]	7	7	11
$C_3^3 DTLZ4$ [66]	3	3	7
$C_3^7 DTLZ4$ [66]	7	7	11
M^3W4 [68]	3	1	7
M^7W4 [68]	7	1	11
M^3W8 [68]	3	1	7
M^7W8 [68]	7	1	11
M^3W14 [68]	3	1	7
M^7W14 [68]	7	1	11
CMD [72]	4	9	6
CSI [66]	3	10	5
WR [70]	5	7	3
$MDCBD$ [71]	5	8	5

4.4 Parameter settings

For the experiments, we need to set the values of five parameters. Based on [73, 74], we set the size of the randomly generated population P_r as

$$|P_r| = 11n - 1, \quad (10)$$

where n is the number of decision variables. The DM must provide the maximum number of expensive solutions that she/he wants to see (N_S), and it can be different for each problem. However, we use here the same number for all problems for simplicity, and we set $N_S = 5$. The next parameter is the maximum number of expensive evaluations at each iteration (N_u). In Algorithm 3 we use $N_u/2$ expensive evaluations for finding feasible expensive solutions that follow the DM's reference point. Therefore, in a worst-case scenario, we should have $N_u = 2N_S$ so that if we have only one iteration, we attempt to find N_S new expensive solutions to show to the DM at least once. Hence, we set $N_u = 10$.

Moreover, the maximum number of expensive evaluations (FE_{max}) determines the number of iterations that we can have. There is no exact way of allocating the correct value for FE_{max} [75]. In practice, we can set FE_{max} according to the amount of time that the DM has. However, in this paper, we assume that we have the function evaluation budget of 100 after we have generated P_0 , and therefore, $FE_{max} = P_r + 100$. The last parameter is the maximum number of generations (t_{max}) at each iteration, where we use the surrogate evaluations. Since this part of the optimization process is computationally inexpensive, we use 40000 surrogate evaluations. We set $N_u = 10$, and $N_S = 5$. In the ASF function (equation (4)), we set $\rho = 0.0001$.

4.5 Performance indicators

Typically, in a posteriori MOEAs, we aim at the convergence of solutions toward the Pareto front and the diversity of solutions to evaluate the performance of an algorithm [76]. However,

when dealing with a priori algorithms, where the DM's preferences are incorporated in the solution process, the performance should be evaluated regarding the region in which the DM is most interested in. This region is also known as a region of interest. Different performance indicators in the literature can incorporate DM's preferences (expressed as a reference point) (see, e.g., [11, 77–81]). In most of the performance indicators that we mentioned, the DM must provide additional information such as the size of the region of interest to evaluate the performance of an algorithm. To reduce the cognitive load and simplicity, we choose two parameterless indicators. For the first performance indicator, we use the ASF, which does not need any new information from the DM, and has been used to evaluate the performance of different algorithms [11, 82]. The second parameterless indicator we use is the expanding hypercube-metric (EH-metric) in [81]. In [81], the EH-metric is compared to the well-known performance indicator R-metric [77], and there are some examples where the EH-metric and the R-metric disagree on which algorithm is better, and after visualizing the solutions, it is evident that the EH-metric was able to identify the best performance. Based on [81], the EH-metric is more consistent to find the better algorithm than the R-metric.

4.6 Experimental results

As previously mentioned, we compared 12 algorithms, focusing on their performance for feasible, computationally expensive solutions. We generated 15 random reference points for each problem as preference information. Each algorithm was independently run 31 times for every reference point and problem, after which we calculated the median ASF and EH-metric values of the final five ($N_S = 5$) expensive solutions. All 12 algorithms were able to find at least five feasible expensive solutions.

We employed a pairwise Wilcoxon significance test [83] to compare the 12 algorithms, using a significance level of $\alpha = 0.05$. Subsequently, we implemented a scoring system to rank the algorithms. For each test, a score of 0 was assigned to a pair if the performance difference between the algorithms was insignificant, +1 to an algorithm if it significantly outperformed the other algorithm and -1 to the algorithm that performed significantly worse. We then ranked all 12 algorithms in an descending order based on the sum of these scores, with rank 1 indicating the best performance and rank 12 representing the worst performance.

Figure 5 presents the heatmaps for the rankings of ASF and EH-metric scores across the 12 problems and their 15 reference points. A paired colormap (employing Viridis color code) was utilized to generate the heatmaps, with darker colors indicating better ranks (dark blue for rank 1 and yellow for rank 12). As illustrated in Fig. 5, KAEA-C outperformed the other five algorithms for the majority of the problems based on both performance indicators.

Upon analyzing the results with a budget of 100 function evaluations, IK-RVEA, AP-RVEA, and RS demonstrated inferior performance. Furthermore, AA1, AA6, AA7, and AA8 consistently exhibited darker colors compared to AA3, AA4, and AA5. KAEA-C's performance displayed darker colors than all other algorithms for both ASF and EH results, signifying its superior performance for most problems.

These observations indicate that other algorithms struggle to effectively reflect the DM's reference point in computationally expensive constrained problems. The poor performance stems from their reliance on constraints solely within their selection strategies and the fact that their model management is not specifically designed to handle constraints. It is also noteworthy that IK-RVEA, which incorporates model management, generally outperformed AP-RVEA, which lacks model management, further underscoring the significance of model management.

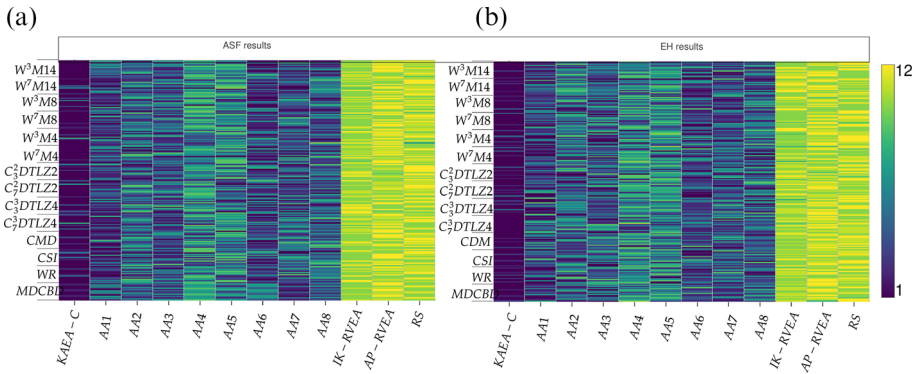


Fig. 5 Heatmaps of the rankings of **a** ASF and **b** EH-metric scores for 12 problems and their 15 reference points for KAEA-C, AA1-AA8, IK-RVEA, AP-RVEA and RS, respectively

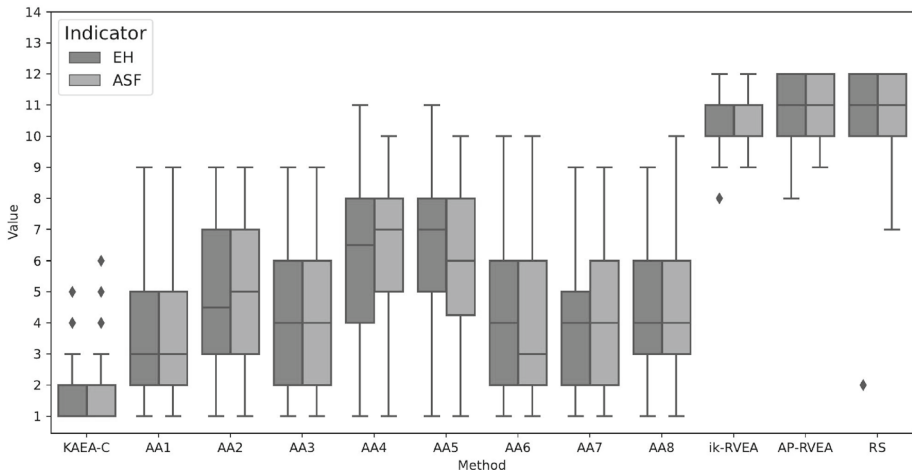


Fig. 6 Boxplot of frequency for each rank for KAEA-C, AA1-AA8, IK-RVEA, AP-RVEA and RS, respectively

Building on our previous analysis, we further examine the results of the two performance indicators by presenting the frequency of appearance for each rank across all problems as boxplots in Fig. 6. As demonstrated, both indicators exhibit similar outcomes in identifying KAEA-C as the superior performing algorithm compared to the other algorithms. Additionally, the other assembled algorithms display somewhat comparable performances. However, it is worth noting that AA1 and AA6 appear to have better median values than the other algorithms, indicating that they may be more consistent in their performance across the problems.

We report additional experiments with increased numbers of function evaluations (200 and 300) in the supplementary materials. As the number of function evaluations increases, the performance of all algorithms improves, reaching a point where, with 300 function evaluations, most performances become competitive. However, the results obtained with 100 expensive evaluations indicate that KAEA-C’s performance outshines that of AP-RVEA, IK-RVEA, and the assembled algorithm. This observation has practical relevance since real-world problems often involve working with a limited number of function evaluations.

The EH-metric results reveal that solutions generated by KAEA-C exhibit better diversity and convergence around the reference point for the majority of the tests conducted. Furthermore, the ASF results suggest that our proposed algorithm can more effectively generate solutions that align with the DM's preferences compared to the other algorithms examined in this paper.

In real-world problems, we cannot assume to have access to such information. We can utilize the current population at the end of each iteration to update the estimates, ensuring that our algorithm remains applicable in practice.

5 Conclusions

In this paper, we introduced KAEA-C, a novel Kriging-assisted a priori multiobjective evolutionary algorithm designed for computationally expensive problems with at least one expensive constraint. To the best of our knowledge, no other algorithm in the literature simultaneously incorporates DM's preferences and handles computationally expensive constraints. As the name suggests, KAEA-C employs Kriging models as surrogates to replace computationally expensive functions.

KAEA-C decomposes the objective space into subspaces and employs a novel selection strategy, wherein each subspace solves a bi-objective subproblem, and the nondominated solutions form the offspring population. In this selection strategy, surrogate evaluations generate the offspring population, allowing for numerous function evaluations. KAEA-C leverages this opportunity by selecting a set of nondominated solutions based on two fitness functions rather than selecting one solution per subproblem. Subsequently, we designed a model management strategy that capitalizes on our selection approach. Here, we evaluate the expensive functions for some decision vectors generated during the optimization process. Our selection strategy provides numerous options to choose from, increasing our probability of finding feasible expensive solutions aligned with the DM's preferences.

We compared KAEA-C with 11 other algorithms and observed that its performance consistently outperformed the others, even under a limited function evaluation budget of 100 function evaluations. When we increased the budget, the performance of other methods improved, but KAEA-C still remained competitive. This result highlights the ability of KAEA-C to perform well under restricted function evaluation budgets, which is often the case in real-world applications.

We examined various combinations of selection and model management strategies, with results indicating that the proposed ones perform best when used together. Furthermore, we observed that altering any of KAEA-C's two components led to a worse performance appearing more frequently, signifying that the assembled algorithms' performance was not as strong as KAEA-C's.

For future research, we plan to develop an interactive algorithm based on KAEA-C that can adapt to potentially significant changes in the DM's preference information. Additionally, we have assumed that the latencies of the expensive functions are relatively uniform. In future work, we aim to address problems with nonuniform latencies and explore strategies for effectively utilizing computing time.

Supplementary Information The online version contains supplementary material available at <https://doi.org/10.1007/s10898-024-01387-z>.

Acknowledgements This research was partly supported by the Academy of Finland (grant no 311877) and is related to the thematic research area DEMO (Decision Analytics utilizing Causal Models and Multiobjective Optimization. jyu.fi/demo) of the University of Jyväskylä. KAEA-C will be implemented in the *open-source* software DESDEO (<https://desdeo.it.jyu.fi/>) funded by the Academy of Finland. The authors would like to thank Bhupinder Saini and Henrik Smedberg for providing the visualization of the heatmap plot and the EH-metric code, respectively, and Dr. Tinkle Chugh for providing feedback during the writing process.

Funding Open Access funding provided by University of Jyväskylä (JYU).

Data Availability The datasets generated during and/or analyzed during the current study are available from the corresponding author upon reasonable request.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

1. Iriondo Pascual, A., Högberg, D., Syberfeldt, A., Brodin, E., Perez Luque, E., Hanson, L., Lämkuill, D.: Multi-objective optimization of ergonomics and productivity by using an optimization framework. In: Black, N.L., Neumann, W.P., Noy, I. (eds.) Proceedings of the 21st Congress of the International Ergonomics Association (IEA 2021), pp. 374–378. Springer, Cham (2022)
2. Diaz, C., Aslam, T., Ng, A.H., Flores-García, E., Wiktorsson, M.: Simulation-based multi-objective optimization for reconfigurable manufacturing system configurations analysis. In: 2020 Winter Simulation Conference (WSC), pp. 1527–1538 (2020)
3. Wang, H., Jin, Y.: A random forest-assisted evolutionary algorithm for data-driven constrained multiobjective combinatorial optimization of trauma systems. *IEEE Trans. Cybern.* **50**(2), 536–549 (2018)
4. Li, M., Li, G., Azarm, S.: A Kriging metamodel assisted multi-objective genetic algorithm for design optimization. *J. Mech. Des.* **130**(3), 031401 (2008)
5. Sun, C., Song, B., Wang, P.: Parametric geometric model and shape optimization of an underwater glider with blended-wing-body. *Int. J. Naval Archit. Ocean Eng.* **7**(6), 995–1006 (2015)
6. Gu, J., Zhang, H., Zhong, X.: Hybrid meta-model-based global optimum pursuing method for expensive problems. *Struct. Multidiscip. Optim.* **61**(2), 543–554 (2020)
7. Chugh, T., Kratyk, T., Miettinen, K., Jin, Y., Makkonen, P.: Multiobjective shape design in a ventilation system with a preference-driven surrogate-assisted evolutionary algorithm. In: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO'19, pp. 1147–1155. ACM, New York (2019)
8. Amrit, A., Leifsson, L.: Applications of surrogate-assisted and multi-fidelity multi-objective optimization algorithms to simulation-based aerodynamic design. *Eng. Comput.* **37**(2), 430–457 (2019)
9. Müller, J., Day, M.: Surrogate optimization of computationally expensive black-box problems with hidden constraints. *INFORMS J. Comput.* **31**(4), 689–702 (2019)
10. He, C., Zhang, Y., Gong, D., Ji, X.: A review of surrogate-assisted evolutionary algorithms for expensive optimization problems. *Expert Syst. Appl.* **217**(1), 119495 (2023)
11. Aghaei Pour, P., Rodemann, T., Hakanen, J., Miettinen, K.: Surrogate assisted interactive multiobjective optimization in energy system design of buildings. *Optim. Eng.* **23**, 303–327 (2022)
12. Chugh, T., Sindhya, K., Hakanen, J., Miettinen, K.: A survey on handling computationally expensive multiobjective optimization problems with evolutionary algorithms. *Soft. Comput.* **23**(9), 3137–3166 (2019)
13. Jin, Y., Wang, H., Sun, C.: *Data-Driven Evolutionary Optimization*. Springer, Berlin (2021)
14. Deb, K.: *Multi-Objective Optimization Using Evolutionary Algorithms*. Wiley, New York (2001)
15. Branke, J., Deb, K., Miettinen, K., Slowiński, R. (eds.): *Multiobjective Optimization: Interactive and Evolutionary Approaches*. Springer, Berlin (2008)
16. Coello Coello, C.A., Lamont, G.B., van Veldhuizen, D.A.: *Evolutionary Algorithms for Solving Multi-objective Problems*. Springer, Berlin (2007)

17. Trivedi, A., Srinivasan, D., Sanyal, K., Ghosh, A.: A survey of multiobjective evolutionary algorithms based on decomposition. *IEEE Trans. Evol. Comput.* **21**(3), 440–462 (2016)
18. Xu, Q., Xu, Z., Ma, T.: A survey of multiobjective evolutionary algorithms based on decomposition: variants, challenges and future directions. *IEEE Access* **8**, 41588–41614 (2020)
19. Jin, Y.: Surrogate-assisted evolutionary computation: recent advances and future challenges. *Swarm Evol. Comput.* **1**(2), 61–70 (2011)
20. Müller, J.: Socemo: surrogate optimization of computationally expensive multiobjective problems. *INFORMS J. Comput.* **29**(4), 581–596 (2017)
21. Zuluaga, M., Sergent, G., Krause, A., Püschel, M.: Active learning for multi-objective optimization. In: *International Conference on Machine Learning*, pp. 462–470. PMLR (2013)
22. Sacks, J., Schiller, S.B., Welch, W.J.: Designs for computer experiments. *Technometrics* **31**(1), 41–47 (1989)
23. Chugh, T., Jin, Y., Miettinen, K., Hakanen, J., Sindhya, K.: A surrogate-assisted reference vector guided evolutionary algorithm for computationally expensive many-objective optimization. *IEEE Trans. Evol. Comput.* **22**(1), 129–142 (2018)
24. Habib, A., Singh, H.K., Chugh, T., Ray, T., Miettinen, K.: A multiple surrogate assisted decomposition-based evolutionary algorithm for expensive multi/many-objective optimization. *IEEE Trans. Evol. Comput.* **23**(6), 1000–1014 (2019)
25. Coello Coello, C.A.: Theoretical and numerical constraint-handling techniques used with evolutionary algorithms: a survey of the state of the art. *Comput. Methods Appl. Mech. Eng.* **191**(11), 1245–1287 (2002)
26. Fan, Z., Fang, Y., Li, W., Lu, J., Cai, X., Wei, C.: A comparative study of constrained multi-objective evolutionary algorithms on constrained multi-objective optimization problems. In: *2017 IEEE Congress on Evolutionary Computation, Proceedings*, pp. 209–216 (2017)
27. Liu, Z.-Z., Wang, B.-C., Tang, K.: Handling constrained multiobjective optimization problems via bidirectional coevolution. *IEEE Trans. Cybern.* **52**(10), 10163–10176 (2022)
28. Coello Coello, C.A.: Constraint-handling techniques used with evolutionary algorithms. In: *Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO'21*, pp. 692–714. ACM, New York (2021)
29. Mezura-Montes, E., Coello Coello, C.A.: Constraint-handling in nature-inspired numerical optimization: past, present and future. *Swarm Evol. Comput.* **1**(4), 173–194 (2011)
30. Fukumoto, H., Oyama, A.: A generic framework for incorporating constraint handling techniques into multi-objective evolutionary algorithms. In: *International Conference on the Applications of Evolutionary Computation, Proceedings*, pp. 634–649. Springer (2018)
31. Chugh, T., Sindhya, K., Miettinen, K., Hakanen, J., Jin, Y.: On constraint handling in surrogate-assisted evolutionary many-objective optimization. In: *International Conference on Parallel Problem Solving from Nature, Proceedings*, pp. 214–224. Springer (2016)
32. Belakaria, S., Deshwal, A., Doppa, J.R.: Max-value entropy search for multi-objective bayesian optimization with constraints, arXiv preprint [arXiv:2009.01721](https://arxiv.org/abs/2009.01721) (2020)
33. Rojas-Gonzalez, S., Van Nieuwenhuijse, I.: A survey on Kriging-based infill algorithms for multiobjective simulation optimization. *Comput. Oper. Res.* **116**, 104869 (2020)
34. de Winter, R., van Stein, B., Thomas, B.: SAMO-COBRA: a fast surrogate assisted constrained multi-objective optimization algorithm. In: Hisao, I., Qingfu, Z., Ran, C., Ke, L., Hui, L., Handing, W., Aimin, Z. (eds.) *Evolutionary Multi-criterion Optimization, Proceedings*, pp. 270–282. Springer, Cham (2021)
35. Datta, R., Regis, R.G.: A surrogate-assisted evolution strategy for constrained multi-objective optimization. *Expert Syst. Appl.* **57**, 270–284 (2016)
36. Balandat, M., Karrer, B., Jiang, D., Daulton, S., Letham, B., Wilson, A.G., Bakshy, E.: BoTorch: a framework for efficient Monte-Carlo Bayesian optimization. *Adv. Neural. Inf. Process. Syst.* **33**, 21524–21538 (2020)
37. Daulton, S., Balandat, M., Bakshy, E.: Differentiable expected hypervolume improvement for parallel multi-objective Bayesian optimization. *Adv. Neural. Inf. Process. Syst.* **33**, 9851–9864 (2020)
38. Eriksson, D., Poloczek, M.: Scalable constrained Bayesian optimization. In: *International conference on artificial intelligence and statistics*, pp. 730–738. PMLR (2021)
39. Martínez-Frutos, J., Herrero-Pérez, D.: Kriging-based infill sampling criterion for constraint handling in multi-objective optimization. *J. Global Optim.* **64**(1), 97–115 (2016)
40. Emmerich, M.: Single- and multi-objective evolutionary design optimization assisted by Gaussian random field metamodels. Computer Science Department, TU Dortmund, Germany, Ph.D. dissertation (2005)
41. Yang, K., Li, L., Deutz, A., Back, T., Emmerich, M.: Preference-based multiobjective optimization using truncated expected hypervolume improvement. In: *2016 12th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD)*, pp. 276–281. IEEE (2016)

42. Astudillo, R., Frazier, P.I.: Bayesian optimization with uncertain preferences over attributes. In: Proceedings of the 23rd International Conference on Artificial Intelligence and Statistics. AISTATS, (2020)
43. Abdolshah, M., Shilton, A., Rana, S., Gupta, S., Venkatesh, S.: Multi-objective Bayesian optimisation with preferences over objectives. *Adv. Neural Inf. Process. Syst.* **32** (2019)
44. Palar, P.S., Yang, K., Shimoyama, K., Emmerich, M., Bäck, T.: Multi-objective aerodynamic design with user preference using truncated expected hypervolume improvement. In: Proceedings of the Genetic and Evolutionary Computation Conference, pp. 1333–1340 (2018)
45. Gaudrie, D., Le Riche, R., Picheny, V., Enaux, B., Herbert, V.: Targeting solutions in Bayesian multi-objective optimization: sequential and batch versions. *Ann. Math. Artif. Intell.* **88**(1), 187–212 (2020)
46. Miettinen, K.: *Nonlinear Multiobjective Optimization*. Kluwer Academic Publishers, Boston (1999)
47. Han, J., Pei, J., Kamber, M.: *Data Mining: Concepts and Techniques*. Elsevier, Amsterdam (2011)
48. Steuer, R.E.: *Multiple Criteria Optimization: Theory, Computation, and Application*. Wiley, New York (1986)
49. Wierzbicki, A.P.: A mathematical basis for satisficing decision making. *Math. Model.* **3**(5), 391–405 (1982)
50. Hwang, C.-L., Masud, A.S.M.: *Multiple Objective Decision Making Methods and Applications: A State-of-the-Art Survey*. Springer, Berlin (1979)
51. Zhang, Q., Li, H.: MOEA/D: a multiobjective evolutionary algorithm based on decomposition. *IEEE Trans. Evol. Comput.* **11**(6), 712–731 (2007)
52. Cheng, R., Jin, Y., Olhofer, M., Sendhoff, B.: A reference vector guided evolutionary algorithm for many-objective optimization. *IEEE Trans. Evol. Comput.* **20**(5), 773–791 (2016)
53. Deb, K., Jain, H.: An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part I: Solving problems with box constraints. *IEEE Trans. Evol. Comput.* **18**(4), 577–601 (2014)
54. Wang, H., Olhofer, M., Jin, Y.: A mini-review on preference modeling and articulation in multi-objective optimization: current status and challenges. *Complex Intell. Syst.* **3**(4), 233–245 (2017)
55. Miettinen, K., Ruiz, F., Wierzbicki, A.: Introduction to multiobjective optimization: interactive approaches. In: Branke, J., Deb, K., Miettinen, K., Slowinski, R. (eds.) *Multiobjective Optimization: Interactive and Evolutionary Approaches*, pp. 27–57. Springer, Berlin (2008)
56. Wierzbicki, A.P.: The use of reference objectives in multiobjective optimization. In: Fandel, G., Gal, T. (eds.) *Multiple Criteria Decision Making Theory and Application*, pp. 468–486. Springer, Berlin (1980)
57. Bechikh, S., Kessentini, M., Ben Said, L., Ghédira, K.: Chapter four—preference incorporation in evolutionary multiobjective optimization: a survey of the state-of-the-art. In: Hurson, A.R. (ed.) *Advances in Computers*, pp. 141–207. Elsevier, Amsterdam (2015)
58. Cornell, J.A.: *Experiments with Mixtures: Designs, Models, and the Analysis of Mixture Data*. Wiley, New York (2011)
59. Lan, G., Tomczak, J.M., Roijers, D.M., Eiben, A.: Time efficiency in optimization with a Bayesian-evolutionary algorithm. *Swarm Evol. Comput.* **69**, 100970 (2021)
60. Lepird, J.R., Owen, M.P., Kochenderfer, M.J.: Bayesian preference elicitation for multiobjective engineering design optimization. *J. Aerosp. Inf. Syst.* **12**(10), 634–645 (2015)
61. McKay, M.D., Beckman, R.J., Conover, W.J.: A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics* **42**(1), 55–61 (2000)
62. Vesikar, Y., Deb, K., Blank, J.: Reference point based nsga-iii for preferred solutions. In: 2018 IEEE Symposium Series on Computational Intelligence (SSCI), pp. 1587–1594 (2018)
63. Feliot, P., Bect, J., Vazquez, E.: A Bayesian approach to constrained single-and multi-objective optimization. *J. Global Optim.* **67**(1–2), 97–133 (2017)
64. de Winter, R., van Stein, B., Bäck, T.: SAMO-COBRA: a fast surrogate assisted constrained multi-objective optimization algorithm. In: *Evolutionary Multi-Criterion Optimization: 11th International Conference, EMO 2021, Proceedings*, pp. 270–282. Springer, (2021)
65. Deb, K., Thiele, L., Laumanns, M., Zitzler, E.: Scalable multi-objective optimization test problems. In: *Proceedings of the 2002 Congress on Evolutionary Computation, CEC'02, vol. 1*, pp. 825–830. IEEE (2002)
66. Jain, H., Deb, K.: An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part II: Handling constraints and extending to an adaptive approach. *IEEE Trans. Evol. Comput.* **18**(4), 602–622 (2014)
67. Tanabe, R., Oyama, A.: A note on constrained multi-objective optimization benchmark problems. In: *2017 IEEE Congress on Evolutionary Computation, Proceedings*, pp. 1127–1134 (2017)
68. Ma, Z., Wang, Y.: Evolutionary constrained multiobjective optimization: test suite construction and performance comparisons. *IEEE Trans. Evol. Comput.* **23**(6), 972–986 (2019)

69. Parsons, M.G., Scott, R.L.: Formulation of multicriterion design optimization problems for solution with scalar numerical optimization methods. *J. Ship Res.* **48**(01), 61–76 (2004)
70. Ray, T., Tai, K., Seow, K.C.: Multiobjective design optimization by an evolutionary algorithm. *Eng. Optim.* **33**(4), 399–424 (2001)
71. Osyczka, A.: *Computer Aided Multicriterion Optimization System (CAMOS): Software Package in Fortran*. International Software Publisher, Hampshire (1992)
72. Tanabe, R., Ishibuchi, H.: An easy-to-use real-world multi-objective optimization problem suite. *Appl. Soft Comput.* **89**, 106078 (2020)
73. Knowles, J.: ParEGO: a hybrid algorithm with on-line landscape approximation for expensive multiobjective optimization problems. *IEEE Trans. Evol. Comput.* **10**(1), 50–66 (2006)
74. Zhang, Q., Liu, W., Tsang, E., Virginas, B.: Expensive multiobjective optimization by MOEA/D with Gaussian process model. *IEEE Trans. Evol. Comput.* **14**(3), 456–474 (2010)
75. Kazikova, A., Pluhacek, M., Senkerik, R.: How does the number of objective function evaluations impact our understanding of metaheuristics behavior? *IEEE Access* **9**, 44032–44048 (2021)
76. Li, M., Yao, X.: Quality evaluation of solution sets in multiobjective optimisation: a survey. *ACM Comput. Surv.* **52**(2), 1–38 (2019)
77. Li, K., Deb, K., Yao, X.: R-metric: evaluating the performance of preference-based evolutionary multi-objective optimization using reference points. *IEEE Trans. Evol. Comput.* **22**(6), 821–835 (2017)
78. Hou, Z., Yang, S., Zou, J., Zheng, J., Yu, G., Ruan, G.: A performance indicator for reference-point-based multiobjective evolutionary optimization. In: *Proceedings of the 2018 IEEE Symposium Series on Computational Intelligence*, pp. 1571–1578 (2018)
79. Mohammadi, A., Omidvar, M.N., Li, X.: A new performance metric for user-preference based multi-objective evolutionary algorithms. In: *2013 IEEE Congress on Evolutionary Computation, Proceedings*, pp. 2825–2832 (2013)
80. Szczepański, M., Wierzbicki, A.P.: Application of multiple criteria evolutionary algorithms to vector optimisation, decision support and reference point approaches. *Journal of Telecommunications and Information Technology*, pp. 16–33 (2003)
81. Bandaru, S., Smedberg, H.: A parameterless performance metric for reference-point based multi-objective evolutionary algorithms. In: *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO'19*, pp. 499–506. ACM (2019)
82. Saini, B.S., Hakanen, J., Miettinen, K.: A new paradigm in interactive evolutionary multiobjective optimization. In: *International Conference on Parallel Problem Solving from Nature, Proceedings*, pp. 243–256. Springer (2020)
83. Frank, W.: Individual comparisons by ranking methods. *Biometrics* **1**, 80–83 (1945)

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.