Universiteit Leiden
Opleiding Informatica

Multi-surrogate Assisted Efficient Global Optimization:

A Positional Study on Discrete Problems

Name: Qi Huang

Date: 22/03/2022

1st supervisor: Dr Anna V. Kononova
2nd supervisor: Prof. dr. Thomas Bäck

MASTER'S THESIS

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

Decade-long advances in simulation-based surrogate-assisted optimization and unprecedented growth of computational power have enabled us to optimize complex engineering problems once intractable to solve. This thesis has a special focus on investigating the benefit of a concurrent utilization of multiple simulation-based surrogate models to solve complex discrete optimization problems. To fulfill this, the so-called Self-Adaptive Multi-surrogate Assisted Efficient Global Optimization algorithm (SAMA-DiEGO), which features a two-stage online model management strategy, is proposed and further benchmarked on thirty-three binary-encoded combinatorial problems and fifteen ordinal problems against several robust non-surrogate or single-surrogate assisted optimization algorithms. Our findings in a fixed budget analysis indicate that SAMA-DiEGO can rapidly converge to better solutions on a majority of the test problems which shows the feasibility and advantage of using multiple surrogate models in optimizing discrete problems. However, we also identify that the proposed SAMA-DiEGO shows deficiency in handling problems with weak global structure and plentiful local optima in comparison with one of the single-surrogate assisted optimization algorithm (SMAC).

Acknowledgements

I would like to heartily thank Anna V. Kononova, Thomas Bäck for patiently supervising me during this long period thesis study. My special thanks go to Roy de Winter and Furong Ye for their generous help and constructive suggestions throughout the whole research. Furthermore, I would like to express my sincere gratitude and love to my parents for their precious support.

Contents

1

Introduction

Technology has been advanced unprecedentedly in the last few decades, the focus of engineering has been gradually shifting from a new invention that can work properly to one that will efficiently and effectively function for a long while. These expectations and challenges give space to the development and application of mathematical optimization, which comprises a series of techniques and can be found nearly everywhere in modern-day engineering fields e.g., aircraft/vehicle design [19, 37] and chemical engineering [57]. To formulate and scientifically achieve better design, engineers and researchers have set up a variety of complex models based on their expertise and observations. Nowadays, instead of manually finding the best solutions, a computer with its enormous computational power can tirelessly and smartly go through the search space of these models under certain guidelines called optimization algorithms to therefore locate the optimal design at acceptable costs. To facilitate readers, in the remainder of this chapter, a quick walk-through of some key concepts of mathematical optimization that relate to our study will be given.

The primary target of mathematical optimization can be defined as locating desirable (optimal) solution(s) from a group of candidates solutions in respect to their performance on objective function(s). Assume we are solving a single objective maximization problem $f\colon \mathcal{S} \to \mathbb{R}$, a formal statement of the optimization task can be described as,

$$\mathbf{X} = \{\vec{x} \mid \arg\max_{\vec{x}\in\mathcal{S}} f(\vec{x}),\ f(\vec{x}) \in \mathbb{R}\}, \tag{1.0.1}$$

where $\mathcal{S}$ is the search space that contains all feasible solutions and $\mathbf{X}$ are

the optimal solution(s). Broadly speaking, if the objective function is known and (sub-)differentiable, an efficient group of solvers are derivative-based methods [73], e.g. gradient descent optimization [77]. However, these solvers become incompetent in the case of optimizing non-differentiable functions. Fortunately, there exists a group of query-based algorithms, namely iterative optimization heuristics (IOH) [26]. These algorithms are capable of optimizing functions regardless of accessibility of derivatives as long as the outputs of functions are accessible within reasonable computation time. Existing algorithms including the famous genetic algorithms [65], the evolutionary algorithms [3], the covariance matrix adaptation evolution strategies [45] and particle swarm optimization [71] are all symbolic representatives of IOHs.

Common iterative optimization procedures are normally sufficient to perfectly solve numerous optimization tasks at the cost of considerable amount of fitness evaluations (function calls to the objectives). However, in some real-world applications, the objective functions are costly to be evaluated. Examples of these problems vary from different background, e.g. chemical design and synthesis [57], water resources [74], aerodynamic design [37], aerospace [19], etc. As a consequence, it would be unacceptable for IOHs to directly evaluate thousands of feasible solutions using expensive objective functions since it will naturally require massive execution time and computational resources. Hence, an alternative strategy has been proposed, instead of using the objective functions, one can approximate their landscapes using various mathematical models known as surrogate models. The use of surrogate can help to save calls to original objectives by shifting majority of function evaluations to surrogates while periodically adjusting the surrogates using the real output(s) of objective functions. A definition of a competent surrogate model $\hat{f}$ to the function $f\colon \mathcal{S} \to \mathbb{R}$ of Equation 1.0.1 is as follows,

$$\begin{gathered}
\hat{F} = \{f(\vec{x}) \mid \vec{x} \in \hat{\mathcal{S}},\ \hat{\mathcal{S}} \subseteq \mathcal{S}\}, \\
\hat{f}_{\hat{\mathcal{S}},\hat{F}}(\vec{x})\colon \mathcal{S} \to \mathbb{R}, \\
\exists \mathcal{S}^* \subseteq \mathbf{S}\colon |S^*| > 1, \\
\forall \hat{x} \in \mathcal{S}^*\colon \hat{f}_{\hat{\mathcal{S}},\hat{F}}(\vec{x}) \simeq f(\vec{x}),
\end{gathered}$$

where $\hat{\mathcal{S}}$ is the set of available observable input from the solution space $\mathcal{S}$ and their corresponding real objective values of $\hat{\mathcal{S}}$ are recorded in $\hat{F}$. The surrogate model $\hat{f}_{\hat{\mathcal{S}},\hat{F}}$ learns (e.g. curve fitting) the data ($\hat{\mathcal{S}}$ and $\hat{F}$) and therefore, produces simulated objective values of f on some solutions ($\mathcal{S}^*$) from $\mathcal{S}$. Consequently, if the surrogate model can accurately simulate the original functions, i.e., given the same inputs, the surrogate model can generate similar or even the same outputs to the original functions using less computational resources, it is possible to efficiently locate the global optimum of original functions through exclusively doing optimization using the surrogate as a proxy objective function.

Algorithm 1: A high level summary of the Efficient Global Optimization (EGO). The pipeline is adapted from [84]

1 **Procedure** `EGO`**:**
2 Compute a surrogate $\hat{f}$ on the initial data set $\mathcal{X}, y$
3 **while** *termination criterion not met* **do**
4 Instantiate an infill criterion $\mathcal{I}$ using $\hat{f}$
5 Find global optimum of the infill criterion $\mathcal{I}$:

$$\vec{x}^* \leftarrow \arg\max_{\vec{x} \in \mathcal{S}} I(\vec{x})$$

6 Evaluate $\vec{x}^*$: $y^* \leftarrow f(\vec{x}^*)$ and append $\vec{x}^*$, y^* to $\mathcal{X}, y$.
7 Re-estimate the surrogate $\hat{f}^*$ based on $\mathcal{X}, y$
8 **end**
9 **end**

One of the well-studied surrogate-assisted algorithm is the Efficient Global Optimization (EGO) [56], which is also commonly known as Bayesian Optimization (BO). In Algorithm 1, a general framework of the Efficient Global Optimization (EGO) is given. The EGO and merely all its variants feature an iterative process that is consist of two major steps: building and fitting a surrogate model (line 2 and line 7 in Alg 1), normally a Gaussian process regression model a.k.a Kriging model (see Section 2.2), with available data samples (inputs and outputs) obtained on current task; finding new additional data point based on the surrogate model through an acquisition process on a certain infill criterion (line 4 to line 6 in Alg 1). A comprehensive and self-explainable tutorial on the EGO algorithms was made by Frazier in 2018 [33].

Before the EGO algorithm was proposed, history of applying surrogate models (meta-models) for optimization can be traced back to 1980s [43] and even longer ago as the prototype of the surrogate model in EGO (the Kriging model) can be dated back to 1950s [58]. Decades of development have resulted in a wide variety of surrogate assisted/based algorithms each with distinct characteristics, (dis-)advantages and suitable application scenarios. Bhosekar and Ierapetritou [10] made an introductory review in 2018, which summarized frequently used surrogate models; famous derivative-free optimization algorithms (solvers) that are suitable to combine with surrogates; as well as accompanied sampling strategies and validation metrics for surrogate models. Similar constructive reviews have been made time to time for different application scenarios in [1, 2, 32, 74, 90].

Most of these reviews and works on surrogate assisted/based optimization are primarily focused on handling (expensive) optimization problems defined

on continuous search spaces. This type of search space is exclusively comprised of numerical variables each has an infinite number of values/choices between any two values/choices. However, there exists modern real-world optimization problems, e.g., vehicle routing and scheduling [79], supply chain network design [29] and global routing in electronic design [78], that are entirely built upon discrete search space, where each variable has a exact number of values/choices.

In contrast to the flourished development of EGO as well as online surrogate management for real-valued optimization, Jin [53] points out the scarcity of studies on surrogate (model) based discrete/combinatorial optimization in 2011. Later in 2017, a survey by Bartz-Beielstein and Zaefferer [6] identifies the key challenge as finding suitable surrogate models for discrete problems. This challenge becomes critical if EGO algorithms are applied, as these solvers normally rely on single surrogate model that is solely determined beforehand. Therefore, it is naturally to consider the possibility of concurrently using multiple surrogate models that exactly support discrete cases, to tackle complex discrete problems. Hence, this thesis takes steps to study the feasibility and benefit (necessity) of utilizing multiple surrogate models in an EGO-styled algorithm to solve purely discrete problems.

1.1 Research Questions

Given with the background and the motivation, three research questions are firstly raised and further investigated in this thesis:

1. In comparison to non-surrogate optimization algorithm, can the use of surrogate model help achieve better performance in solving discrete problem?

   An important motivation of using surrogate model is to save calls to the objectives. It may take days, weeks or even months to directly evaluate/simulate solutions on an expensive optimization problem [19, 37, 57, 74]. Although given the success of EGO, it is still important to assure the effectiveness of using surrogate models in discrete optimization.

2. Given with a fixed evaluation budget, can multiple surrogate model based EGO achieve better performance if compared with robust single surrogate model based EGO?

   To the best of our knowledge, there is a lack of studies on utilizing multiple surrogate models for discrete problems. Therefore, considering the fact that using multiple surrogates will consume more computational resources, it is reasonable to consider whether it is beneficial to use multiple surrogate models over a single surrogate.

3. When optimizing discrete problems, can we trust the prediction value of surrogate models in EGO if multiple surrogates are used?

   The use of acquisition functions is almost a default setting in EGO-styled algorithms (line 4 in Algorithm 1). However, Rehbach et al. [75] argued that exclusively use prediction values of a promising surrogate can outperform using acquisition functions in handling lower dimensional continuous problems. Meanwhile, as far as we know, the necessity of using acquisition functions for optimizing discrete problems has not been investigated. Thus, in this thesis, a special attention is given on this topic.

1.2 Outline

The rest of the thesis is organized as follows: in Chapter 2, some notable related works on adapting surrogate models to discrete search spaces and on online management strategies for using multiple surrogates will be introduced; in Chapter 3, a newly proposed multi-surrogate assisted EGO algorithm for discrete optimization called SAMA-DiEGO[1] is introduced where the algorithm is decomposed to several building blocks to assist readers understanding our work; this newly proposed algorithm is later experimentally benchmarked in Chapter 4 against existing robust algorithms on two problem sets to answer the research questions raised in Section 1.1; lastly, the results are further concluded and extended in Chapter 5.

[1]The Python implementation of the proposed SAMA-DiEGO algorithm can be found in `https://github.com/BaronH07/SAMA-DiEGO`

2

Related Works

2.1 A Survey of Forerunners

Before directly going into the three research questions raised in Section 1.1, it is beneficial and worthwhile to investigate first what are the existing methods of adapting surrogate modelling to discrete problems and integrating multiple such models in surrogate assisted/based optimization? To answer the question, in this section, an overview of forerunners is given.

Online management of multiple surrogate models denotes the process of maintaining and determining the most suitable proxy (surrogate model) to an objective function while doing optimization. In contrast to offline model management, the online management is capable of utilizing incrementally obtained new data samples [54]. Gorissen et al. [40] proposed an Evolutionary Model Selection (EMS) algorithm to concurrently determine the best surrogate model type and its hyperparameter through minimizing cross-validation error or Akaike Information Criterion using a modified genetic algorithm. Couckuyt et al. [18] further extended the work of [40] by combining the expected improvement function with the EMS algorithm in searching for promising points (to be evaluated on original objective function). Bagheri et al. [4] selected the best-performed type of radial basis function interpolation on the basis of their median absolute errors obtained on newly acquired data samples (unseen to surrogate models). The model selection phase in [81] directly applies cross-validation technique on all candidate models to identify the one with minimum mean squared errors. Jia et al. [51] combined offline model selection with online model selection. In the offline stage, assuming

availability of some pre-selected representative optimization problems and a pool of candidate models, a selector is trained, which maps pre-defined four characteristics of pre-selected problems to the best surrogate model types. Later, in online selection, the trained selector is used to pre-screen the candidate models regarding the four characteristics of the new problem to reduce the number of candidate models. Lastly, the online model selection adapts a similar strategy to [40] to find the best surrogate for the new problem.

Instead of only using the best surrogate out of all candidates, an alternative choice is to ensemble (aggregate) multiple surrogate models through computing a weighted sum of predictions produced by all candidate models [6, 35, 61, 67, 93]. Further, it is feasible to adaptively merge a group of surrogate models with regard to fidelity level, i.e., reliability or accuracy of models in approximating the original problem. An notable example is the Co-Kriging [31, 94]. Co-Kriging builds a new Kriging-styled model through adaptively combining approximations of a coarse (low-fidelity) and a fine (high-fidelity) surrogate model, while exploiting correlations between the two models. Le Gratiet and Cannamela [42] extended the Co-Kriging by combining leave-one-out cross-validation errors with the original error measurement of Co-Kriging. By all means, the previously described methods aim to select/create a surrogate model out of all candidate models. An alternative strategy is proposed in [86], namely the Multiple Surrogate Efficient Global Optimization (MSEGO) algorithm, which maintains all candidate models and samples new infill points by independently maximizing the EI criterion on each candidate model. A similar idea is re-visited in [88] and further developed in [7]. In comparison to exclusively using EI in MSEGO, Beaucair et al. [7] suggests employing different infill criteria for different type of surrogate models and therefore concurrently sampling new points from these independent infill criteria. Another advancement in multi-fidelity EGO is generalizing existed or developing novel infill criteria to suit for multiple surrogate models. As an pioneer example, Huang et al. [49] augmented the original Expected Improvement (EI) [56] by mutually considering three aspects: the correlation between, the errors and the cost of the low-and high-fidelity models. More concrete reviews regarding recent development of EI for multi-fidelity optimization can be found in [95].

With respect to adapting surrogate models to discrete problems, the survey of Bartz-Beielstein and Zaefferer [6] indicates that existing methods for discrete surrogate modelling can be split up in six strategies:

1. The naive approach: directly applying methods used for continuous spaces if the search space can be represented/encoded using a vector.

2. Customized models: models mostly designed by experts.

3. Inherently discrete models: models that naturally support discrete structures.

4. Mapping: discrete variables or structures are mapped to a more easily treatable representations.

5. Feature extraction: extract numeric features from the objective (e.g. depths of trees). The obtained numerical features can be modeled and processed with standard optimization techniques.

6. Kernel-based modeling: where available, discrete kernels/measures of (dis)-similarity may be used to replace continuous kernels.

The need of adapting EGO for discrete problems is recently and constantly emphasized in solving expensive mixed-integer optimization problems. A notable example is hyper-parameter tuning and automatic model selection for machine learning tasks, where, in that field of study, the EGO-styled solvers are more commonly referred as Sequential Model Based Optimization (SMBO) algorithms [50]. Studying the surrogate models used in SMBO algorithms can be a solid entry point to understand the state-of-the-art methodology of applying surrogate models on expensive discrete problems. Existing well-established representatives of SMBO are Gaussian Process optimizer (GP optimizer) [8], adaptive Tree-structured Parzen Estimator (TPE) [8], Sequential Model-based Algorithm Configuration (SMAC) [50, 62], Spearmint [80], Bayesian Optimization and Hyperband (BOHB) [30] and Mixed-Integer Parallel Efficient Global Optimization (MIP-EGO) [85]. Their application scenario is: suppose given a set of mixed-integer (co-existence of discrete and continuous variables) configurations as λ and the corresponding performance of the expensive machine learning model as y, surrogate models in SMBO are commonly required to model the conditional probability $p(y|\lambda)$. With respect to the back-end surrogate models of SMBO: SMAC and MIP-EGO use random forest regression; Spearmint and GP optimizer utilize Gaussian process regression; TPE, instead of modeling the $p(y|\lambda)$, it applies a tree-structure Parzen estimator to concurrently model $p(y|\lambda \geq a)$ and $p(y|\lambda < a)$, where a is a moving threshold; and lastly, BOHB resembles multiple TPE models as its back-end surrogate. Other eye-catching examples besides SMBO algorithms are: Li et al. [59] uses a RBF networks to analyse ultrasound images; Support Vector Regression (SVR) and gradient boosting regression tree [34, 72] are applied in [28] to model machine learning tasks. To come to the point, all the mentioned surrogate models are naturally supported or modified to support discrete variables (e.g. binary, ordinal and categorical) or structures (tree, graph, etc) which primarily fall into the first and the sixth category of the six strategies defined in [6]. A detailed walk-through of four notable surrogate models that suit discrete cases are presented in the next section.

2.2 Types of Surrogate Models Studied in the Thesis

Four types of surrogate models that naturally supported or can be patched to support discrete variables/structures are used in our thesis, namely Radial Basis Function interpolation (RBF), Support Vector Machine (SVM) regression, Kriging models (a.k.a Gaussian process regression) and Random Forest regression (RF) as a representative for tree-based models.

RBF Interpolation

Radial basis function interpolation is firstly proposed by Hardy in [47]. The idea is originated on modeling topography with high accuracy. Researchers later extended the usage of RBF to global optimization tasks [22, 44, 68]. The core idea behind RBF interpolation is to assign a distance-dependant transformation (function) $\varphi(\|\cdot\|)$ to all data points (vectors) $\vec{x}$ w.r.t predefined centers points (vectors) $\vec{c}$ and then linearly combined these functions to simulate the original objective(s). A formal definition of this interpolation process for Equation 1.0.1 can be defined as follows:

$$\hat{f}(\vec{x}) = \sum_{i=1}^{n} w_i \varphi\left(\|\vec{x} - \vec{c}_i\|\right). \tag{2.2.1}$$

where $\{\vec{c}\}^n$ are the observable and trainable data points from $\mathcal{S}$, $\{w\}^n$ are the weights need to be derived and $\hat{f}$ is the approximation function of f. Micchelli introduced in a polynomial tail for $\hat{f}$, to relax the limitations imposed on radial basis functions, so that it can guarantee an unique solution for weights [64]. Afterwards, the interpolation system is shown as follows:

$$\begin{gathered}
\vec{x} \in \mathcal{S}^d,\ \vec{c} \in \mathcal{S}^d \\
p(\vec{x}) = \mu_0 + \overrightarrow{\mu_1}\vec{x} + \overrightarrow{\mu_2}\vec{x}^2 \cdots + \overrightarrow{\mu_k}\vec{x}^k \\
\hat{f}(\vec{x}) = \sum_{i=1}^{n} w_i \varphi\left(\|\vec{x} - \vec{c}_i\|\right). + p(\vec{x}),
\end{gathered}$$

where the problem dimensionality is d, $\vec{p}(x)$ is a k-th order polynomial functions of $\vec{x}$, μ and w are a series of weights that need to be computed. These weights (w and μ) can be obtained by solving the following linear system (in practice using e.g. SVD inversion):

$$\begin{bmatrix} \Phi & \mathrm{P} \\ \mathbf{P}^T & \mathbf{0}^{(kd+1)\times(kd+1)} \end{bmatrix} \begin{bmatrix} \vec{w} \\ \vec{\mu}' \end{bmatrix} = \begin{bmatrix} \vec{f} \\ 0^{(kd+1)} \end{bmatrix}, \tag{2.2.2}$$

where $\Phi \in \mathbb{R}^{n\times n}$ is a matrix of radial basis functions with $\Phi_{ij} = \varphi\left(\|\vec{c}_i - \vec{c}_j\|\right)$, $\mathbf{P} \in \mathbb{R}^{n\times(kd+1)}$ is a matrix with $\left(1, \vec{x}_{(i)}, \ldots, \vec{x}^k_{(i)}\right)$ in its ith row and $\mathbf{0}^{(kd+1)}$

and $\mathbf{0}^{(kd+1)\times(kd+1)}$ are two zero vector/matrix. Illustration and implementations of RBF interpolation in this thesis are largely based on the work of [5] and [22]. Additionally, Bagheri et al [5] introduced in an uncertainty quantification for RBF interpolation on a data point $\vec{x_i} \in \mathcal{S}^d$ as follows,

$$\Sigma_{rbf} = \varphi\left(\|\vec{x}_i - \vec{x}_i\|\right) - \vec{\Phi}_i^T \Phi^{-1} \vec{\Phi}_i, \tag{2.2.3}$$

where $\varphi\left(\|\vec{x}_i - \vec{x}_i\|\right)$ is a RBF-dependant constant value, $\vec{\Phi}_i$ is the RBF matrix for point $\vec{x}_i$. A break-through of their innovation is, it enables us to compute a probabilistic-based infill criterion (e.g. expected improvements) on the RBF interpolation model.

Kriging Interpolation

Kriging interpolation, a.k.a Gaussian process regression, is named after D.G. Krige in recognition of proposing the prototype of the method [58]. Originated in geostatistics, Kriging interpolation has now become widely used in global optimization. The original EGO algorithm [56] is in nature based on Kriging interpolation, examples of other applications are [20, 23].

Kriging interpolation features two key components. The first one is a linear combination of known functions, which are exclusively defined on individual data points. The another one is the appending stochastic process. A definition of universal Kriging [76] over function $f(\vec{x})$ can be given as follows:

$$\hat{f}(\vec{x}) = \sum_{i=1}^{k} w_i \xi_i(\vec{x}) + P(\vec{x}),\ \vec{x} \in \mathcal{S} \tag{2.2.4}$$

where ξ is a basic function of $\vec{x}$ (e.g. constants, linear and polynomial), w's are the weights and $P(\vec{x})$ is a predefined random process over x. More importantly, P is with zero mean and its spatial covariance function is defined as,

$$\operatorname{cov}\left[P\left(\vec{x}^{(i)}\right), P\left(\vec{x}^{(j)}\right)\right] = \sigma^2 C\left(\vec{x}^{(i)}, \vec{x}^{(j)}\right),$$

where C is a correlation function (normally defined over distances between data samples) and σ is the variance of P. In terms of solving the equation 2.2.4, Bouhlel et al reviewed several methods for estimating the parameters of Kriging interpolation [12].

To better incorporate discrete variables (ordinal and categorical variables) with Kriging interpolation, Garrido-Merchán and Hernández-Lobato have recently proposed a mechanism to systematically do continuous relaxation for each discrete variable, hence it is reported to improve the performance of Kriging in discrete cases [36]. As the covariance function C is intrinsically defined on continuous space, their method transforms the variables (rounding ordinal types and one-hot-encoding categorical types) before they are fed to covariance function C. Consequently, it guarantees a constant correlation

between any two evaluation points that are different in continuous space but the same in discrete space, e.g. if considering a one-dimensional case, $C(1.4, 0.6)$ will be transformed into $C(1, 1)$. In this thesis, the applied discrete-supported universal Kriging models are largely based on the works of [13], which can be considered as an extended implementation of [36].

Support Vector Machines

Support vector machines (SVMs) is a group of well-known statistical learning algorithms firstly proposed by Cortes and Vapnik in 1995 [17]. It has been widely used for classification and regression analysis and it naturally support discrete variables. The core idea is, given data that are not linearly separable, the SVMs will firstly map original data to higher-dimensions and then determine a set of hyperplanes which can maximize the linear separability of data in the new space. A formal procedure of $\varepsilon-$SVM in approximating a function $f(\vec{x})$ is more or less equivalent to solve the following problem:

$$\min_{w,b,p^+,p^-} \frac{1}{2} w^T w + C \sum_{i=1}^{n} (p_i^+ + p_i^-), \tag{2.2.5}$$

where w and b are the parameters of linear function, C is the coefficient for penalties and p^+, p^- are two positive punishments that measure how far a data sample is above or below its nearest hyperplane ε, respectively. Moreover, formula 2.2.5 is subjected to two constraints,

$$f(\vec{x}_i) - w^T \phi(\vec{x}_i) - b \leq \varepsilon + p_i^+,$$

$$w^T \phi(\vec{x}_i) + b - f(\vec{x}_i) \leq \varepsilon + p_i^-,$$

where $\phi(\vec{x})$ is a kernel function that maps data to higher dimensions. Our implementations of SVMs are exclusively based on the well-known scikit-learn package [69].

Random Forest

Random forest (RF) is a ensemble learning algorithm for solving classification and regression tasks [14]. It features a bagging type of ensemble of decision trees by aggregating results of each tree to make final prediction, thereby RF can overcome the high-variance (over-fitting) phenomenon commonly seen in using a single decision tree. Moreover, during the training (fitting) session, each decision tree is only exposed to a random subset of training samples and the best split in each of its node is determined either from all input features or from a subset of features [15]. Two outstanding aspects of random forest regression are: it is designed to be capable of handling discrete variables; it has a structure that is suitable to parallelize [85]. In this thesis, we mostly followed the implementations of Random Forest regression

provided by scikit-learn [69]. Moreover, in an effort to fit random forest with probabilistic-based infill criteria (see Section 3.4), we applied the uncertainty quantification mechanism proposed in [50] and implemented by [85].

3

The Proposed Algorithm

In this chapter, a new *Self-Adaptive Multi-surrogate Assisted Discrete Efficient Global Optimization* (SAMA-DiEGO) algorithm is proposed, utilizing the power of multiple surrogate models and online model selection technique to efficiently optimize expensive objective functions under limited budgets.

3.1 Overview

The proposed self-adaptive multi-surrogate assisted discrete efficient global optimization (SAMA-DiEGO) algorithm features concurrent management of various kinds of surrogates to approximate and explore the landscape of an objective function from different perspectives. A high-level pseudo-code is presented in Algorithm 2. The algorithm takes all given surrogates into consideration and firstly generate initial samples following the strategy described in Section 3.2 and determine a pool of promising surrogates by screening out incompatible ones based on the initial samples (Section 3.3.1). The main loop starts with selecting the best surrogate to optimize in the current iteration (Section 3.3.2) and then instantiates a pre-defined infill criteria according to the best surrogate. Afterwards, the next solution to be evaluated is chosen by the back-end optimizer based on the infill criteria (Section 3.4). Lastly, all the surrogates within the model pool are updated.

Algorithm 2: SAMA-DiEGO. **Input:** Objective functions $f(\mathbf{x})$, decision parameters' lower and upper bounds $[\mathbf{lb}, \mathbf{ub}]^d \subset \mathbb{R}^d$, number of initial samples N_{init}, maximum evaluation budget N_{max}, model configurations M (φ), number of available parallelisms P, selection criteria C, infill criteria I. **Output:** Evaluated best solution.

Function `SAMA-DiEGO(`f, $[\mathbf{lb}, \mathbf{ub}]$, N_{init}, N_{max}, M, C, P, I`):`
- $\mathbf{X} \leftarrow \{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$ ▷ `Generate initial design,` $\mathbf{X} \in \mathbb{R}^{d \times N}$
- $\mathbf{Y} \leftarrow f(\mathbf{X})$ ▷ `Obtain objective scores,` $\mathbf{Y} \in \mathbb{R}^N$
- ▷ `Validation on feasibility of surrogates`
- $\tilde{SS}, M^* \leftarrow$ ModelVerification $(M, \mathbf{X}, \mathbf{Y}, P, C)$
- ▷ `Get the best model config. after verification`
- $\hat{m} \leftarrow$ GetFirstOf(M^*)
- $n \leftarrow N_{init}$
- **while** $n \leq N_{max}$ **do**
  - $\tilde{SS} \leftarrow []$
  - $\hat{\mathbf{Y}} \leftarrow$ Standardize$(\mathbf{Y})$ ▷ `See Section 3.3.1`
  - **for** $\varphi \in M^*$ **do** ▷ `For each model configuration`
    - $S \leftarrow$ CreateAndFit$(\varphi, \mathbf{X}, \hat{\mathbf{Y}})$ ▷ `Fit models`
    - $\tilde{SS} \leftarrow [\tilde{SS}\ S]$ ▷ `Add the model into sequence`
  - **end**
  - $\bar{\mathbf{y}} \leftarrow$ Mean$(\mathbf{Y})$ ▷ `Get the mean of fitness values`
  - $\sigma \leftarrow$ Std$(\mathbf{Y})$ ▷ `Get the standard deviation`
  - $\mathbf{y}_{best} \leftarrow$ UpdateBest $(\mathbf{Y})$ ▷ `Update the best-so-far`
  - $\hat{\mathbf{y}}^*_{best} \leftarrow (\mathbf{y}_{best} - \bar{\mathbf{y}}) \ / \ \sigma$ ▷ `Standardization`
  - ▷ `Get the best model by configuration`
  - $\hat{S} \leftarrow$ GetModelByConfig$(\tilde{SS}, \hat{m})$
  - $\mathbf{IC} \leftarrow I(\hat{S}, \hat{\mathbf{y}}^*_{best})$ ▷ `Instantiate infill criteria`
  - ▷ `Find best new solution w.r.t` **IC**
  - $\mathbf{x}_n \leftarrow$ MultiStartOptimize(**IC**, $\hat{S}$, P)
  - $\mathbf{y}_n \leftarrow f(\mathbf{x}_n)$ ▷ `Evaluate the new solution`
  - $\hat{\mathbf{y}}^*_n \leftarrow (\mathbf{y}_n - \bar{\mathbf{y}}) \ / \ \sigma$ ▷ `Standardize the new sample`
  - ▷ `Rank and select the best model`
  - $\hat{S} \leftarrow$ ModelRank$(\tilde{SS}, \mathbf{x}_n, \hat{\mathbf{y}}^*_n, P, C)$
  - $\hat{m} \leftarrow$ GetConfigOfModel$(\hat{S})$
  - $\mathbf{X} \leftarrow [\mathbf{X}\ \mathbf{x}_n]$ ▷ `Add new solution,` $\mathbf{X} \in \mathbb{R}^{d \times (n+1)}$
  - $\mathbf{Y} \leftarrow [\mathbf{Y}\ \mathbf{y}_n]$ ▷ `Add the new fitness value,` $\mathbf{Y} \in \mathbb{R}^{n+1}$
  - $n \leftarrow n + 1$
- **end**
- $\mathbf{x}_{best}, \mathbf{y}_{best} \leftarrow$ FindBest $(\mathbf{X}, \mathbf{Y})$ ▷ `The best w.r.t.` $\mathbf{Y}$

return $(\mathbf{x}_{best}, \mathbf{y}_{best})$

3.2 Design of Experiments

A surrogate-assisted algorithm normally requires data samples (with objective values) to initialize and train its surrogate. In [48], three initialization methods (design of experiments methods) for combinatorial optimization are compared. It suggested that D-optimal [21] is the strongest on low-dimensional problems but its performance are largely problem-dependent and relies on its internal model. Whereas Latin hypercube sampling (LHS) is relatively easy-to-configure and independent from problems. Consequently, our SAMA-DiEGO utilizes LHS to generate initial samples.

LHS was firstly proposed by McKay et al. in 1979 [63]. Assume sampling M items for a discrete problem with N variables. LHS firstly divides each variable into M non-overlapping equal-sized intervals and within each interval, a value is determined under a specified strategy, hence there are M values for each variable. Afterwards, an ordinal one-dimensional array of these values for the ith variable X_i as $\vec{v_i}^{(1\times N)}$. M data samples can then be constructed by sequentially concatenating all the $\vec{v_i}^{(1\times N)}$ arrays. Subsequently, the output of LHS is a $S^{(M\times N)}$ matrix of which each row explicitly represents a data sample.

The LHS in SAMA-DiEGO follows the implementation of [13], where a specifically designed enhanced stochastic evolutionary algorithm [52] is used as the strategy to select value in each interval. An example of performing LHS on a search space with 2 variables ($X1 \in \mathcal{Z}^{[0,6]}, X2 \in \mathcal{Z}^{[0,6]}$) to generate 7 samples are given in Figure 3.2.1. Additionally, an example of performing LHS on binary encoded search space is provided in Table 3.2.1.

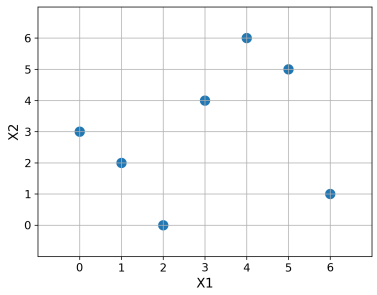


Figure 3.2.1: An example of doing LHS on a discrete search space with N = 2, M = 7. The blue dots are the sampled data points.

	x1	x2	x3	x4	x5	x6	x7
1	0	1	0	0	1	0	0
2	1	1	1	1	0	0	0
3	0	1	0	1	0	1	1
4	0	0	1	0	1	0	1
5	1	1	0	1	1	0	1
6	1	0	1	1	1	1	0
7	1	0	0	0	0	1	0

Table 3.2.1: An example output of doing LHS on a binary encoded search space with N = 7, M = 7 based on the implementation of [13]

With respect to the choice of number of initial samples, [48] suggested that if solving a binary encoding problem, there is no obvious gain in allowing the initial size to be more than 30% of total budgets. Moreover, Bossek et al. [11] similarly concluded that if optimizing the continuous BBOB benchmark,

the performance of EGO worsens with increasing of initial size. In fact, they benchmarked the classical EGO [56] on continuous BBOB with 10 different initial sizes (every 10% from 10% to 100% of the total budget) and experimentally concluded that 10% is the best setting. Here, in SAMA-DiEGO, following their findings, the default initial sampling size is set to 10% of total budget however, it is possible to further decrease the sampling size.

3.3 Online Surrogate Management

The core of SAMA-DiEGO is to maintain a group of promising models and choose the best one to optimize. In the following sections the two-stage model management strategy of SAMA-DiEGO will be presented.

3.3.1 Initial Screening

Considering the case where we are optimizing an expensive black-box function using surrogates, an important step here will be deciding which surrogate to use. To think from the perspective of efficient optimization, this surrogate shall be capable of accurately approximating the original objective function. The concept behind this approximating is regression analysis and hence the famous *no free lunch theorem* [91] applies to all surrogate models. Therefore, it is a challenge for researchers to always choose the right model unless the landscape and mathematical property of the objective are known. Sometimes, if the characteristics of the objective function are known upfront, suggestions can be given based on experience, but this expertise is not always available. Thus, an initial model selection stage is embedded in the SAMA-DiEGO.

The SAMA-DiEGO is initialized with a large pool of surrogate models and a batch of initial data samples (with objective values), it then verifies and selects top-performed models following the procedure described in Algorithm 3. The procedure starts with standardizing the object values (Y) of all initial samples (line 9 in Alg. 2) as follows:

$$\hat{Y} = \frac{Y - \bar{y}}{\sigma},$$

where $\hat{Y}$ represents the standardized objective values, $\bar{y}$ and σ are the mean and the standard deviation of Y, respectively. This standardization is necessary for distance-based surrogates (e.g. Kriging and RBF interpolation). The data samples are further divided into two sets, namely training set and test set. Subsequently, all surrogates are fitted with the samples in training set (same data for all surrogates). Incompetent models are immediately dropped out – here, an incompetent model is the one that incurs arithmetic errors (e.g. division by zero and overflow) or under-fits the training samples according to mean squared error or other metrics e.g. R-squared value. The remaining

competent models are further evaluated using the test samples (again, same data for all surrogates) and are ranked based on a set of criteria E. The evaluation and ranking can focus on exact performance (e.g. mean squared errors) on one hand and on execution time on the other hand. Consequently, the top-T performed surrogates in respect to the initial data samples (X and Y) can be obtained, here, T is a hyperparameter determined beforehand. Notably, surrogates are handled in parallel to save the execution time in SAMA-DiEGO. A practical recommendation from our experiments is to set the size of final surrogate pool (T) equal to the number of available CPU threads (namely, P in Algorithm 2). In this way, the average time needed to handle surrogates in SAMA-DiEGO is theoretically bounded by the model with worst average time complexity.

3.3.2 Model Maintenance in Main Loop

Managing model in the main loop is consist of two steps, training and ranking. As described in the previous section, outcomes ($\tilde{SS}$ and M^*) of model verification, are arranged with respect to scores of models and the best model configuration $\hat{m}$ in verification is the first element of M^* (line 5 in Algorithm 2).

The main loop (line 7 to line 27 in Algorithm 2) starts with training all surrogates on all available standardized data samples. Afterwards, the next point to be evaluated is found by searching on the landscape of currently best surrogate. Notably, the best surrogate is always the model with the best configuration $\hat{m}$. Moreover, the algorithm updates the choice of best configuration $\hat{m}$ in respect to models' performances on the newly obtained sample ($[x_n, y_n]$) in each iteration. Broadly speaking, the SAMA-DiEGO largely adapts the online model selection technique that is introduced in [4], but leaves out the sliding window mechanism since the authors of [4] experimentally discovered that smaller window size produces better results. The sliding window mechanism smooths the performance of surrogates across few iterations and consequently leads to consistency in determining the best surrogate. Our intention of using multiple surrogates in SAMA-DiEGO is to allow the optimizer explore different approximations of search space provided by distinct models without using additional calls to the objective function. Thus, the consistency in model selection is not appreciated in this case.

Training All surrogate models are trained on exactly all the data samples obtained before current iteration (line 8 to 13 in Algorithm 2).

Ranking Similar to the procedure described in Alg. 3, the model selection technique in the main loop is in nature a ranking procedure which is briefly given in Alg. 4. The data sample used in ranking is the new sample obtained in current iteration.

Algorithm 3: Model Verification. **Input:** Model configurations M (φ), solutions X, Fitness values Y, maximum number of candidate models T, evaluation criteria E **Output:** Promising models $\tilde{SS}$ and their corresponding configurations M^*.

Function `ModelVerification`(M, $\mathbf{X}$, $\mathbf{Y}$, T, E)**:**
  $\hat{\mathbf{Y}} \leftarrow$ STANDARDIZE($\mathbf{Y}$) ▷ `Standardize objective space`
  $SS \leftarrow \{\}$
  $L \leftarrow \{\}$
  ▷ `Verify models`
  for $\varphi \in M$ **do** ▷ `For each model configuration`
    try:
      $S^* \leftarrow$ CREATE(φ) ▷ `Create models`
      ▷ `Fit and evaluate models, where` l `is the score of model under criteria` E
      $S, l \leftarrow$ FITANDEVALUATE (S^*, $\mathbf{X}$, $\hat{\mathbf{Y}}$, E)
    catch *Errors***:**
      continue
    end
    $SS \leftarrow [SS\ S]$ ▷ `Add the` ***survived*** `model into pool`
    $L \leftarrow [L\ l]$ ▷ `Add models' corresponding scores`
  end
  ▷ `From best to worst, rank the` ***survived*** `models`
  $Ind \leftarrow$ RANK(L)
  $\tilde{SS} \leftarrow \{\}$
  $M^* \leftarrow \{\}$
  for $i \in Ind$ **do**
    $\tilde{S} \leftarrow$ GET(SS, i) ▷ `Get the good-performed model`
    $m \leftarrow$ GET(M, i) ▷ `Get the model configuration`
    $M^* \leftarrow [M^*\ m]$
    $\tilde{SS} \leftarrow [\tilde{SS}\ \tilde{S}]$
    if LENGTH($\tilde{SS}$) $= T$ **then**
      break
    end
  end
return ($\tilde{SS}$, M^*)

Algorithm 4: Model Rank. **Input:** Candidate models SS, input solutions X, objective values Y, evaluation criteria E **Output:** The most promising model $\tilde{S}$.

```
1  Function ModelRank(SS, X, Y, E):
2      L ← {}
3      for S ∈ SS do                    ▷ For each model configuration
4          try:
                                        ▷ Evaluate each candidate model
5              l ← EVALUATE (S, X, Y, E)
6          catch Error:
7              continue
8          end
9          L ← [L l]                    ▷ Add models' corresponding scores
10     end
                  ▷ Decide the best model w.r.t models' scores
       S̃ ← DECIDEBEST(SS, L)
11 return S̃
```

The training and ranking procedures are designed to be carried out in parallel in SAMA-DIEGO. However, since the surrogates are exclusively making predicting on one sample in each iteration, the ranking procedure can be done sequentially. By all means, the decision to do parallelism or not is subjected to the trade-off between space and time complexity of surrogates.

3.4 Optimization on Surrogates

Locating promising solutions using a surrogate model as the objective function is a key contributor to the success of surrogate-assisted optimization and it is rather critical in SAMA-DiEGO since an additional requirement for allowing rapid switching of surrogate models in the model selection stage is to perform a one-shot reliable optimization on the surrogate. This process is consist of two steps, setting up an acquisition function and optimizing the acquisition function using adequate algorithms.

Acquisition Function The idea of using acquisition functions (a.k.a infill criteria) in EGO-styled algorithm (line 4 in Alg. 1) is ascribed to maximize the chance of selecting a feasible data point that can latter yield considerable improvements in comparison to current best-so-far point. These functions balance the exploration and exploitation of algorithms. Here, exploration means searching samples at positions where the surrogate predicts high uncertainty and exploitation means sampling at locations where the surrogate predicts better objective val-

ues. Some notable examples of these acquisition functions are expected improvement (EI) [56], probability of improvement (PI) [55, 96] and moment-generating function of improvement (MGFI) [87].

One of the research questions of this thesis (see Section 1.1) is to experimentally identify whether it is still necessary to use probabilistic-based infill criteria in discrete optimization if multiple surrogate models are used. This argument will be discussed in Chapter 4 where expected improvement (EI) and predicted values of surrogates (PV) are concurrently tested.

Back-end Optimizer Any existing optimizer that can effectively and efficiently optimize (non-linear) discrete functions is feasible to be a back-end optimizer. A function defined on discrete space is non-differentiable, thus it is infeasible to optimize it with gradient-based methods. Although there exists various traditional numerical optimizers that do not take derivatives, our SAMA-DiEGO specifically consults on evolutionary computation algorithms for their proven records in solving discrete problems and consequently selects the following optimizers:

Mixed-Integer Evolutionary Strategy ((μ, λ) MI-ES) In SAMA-DiEGO, the MI-ES is specifically chosen as one of the optimizers. Its implementation is exclusively based on the descriptions by Li et al [60]. The hyper-parameters and the stopping criteria are largely set according to those in MIP-EGO [85]. Default values of hyper-parameters are listed in Table 6.2.2 in the Appendix 6.2. Additionally, the maximum number of function evaluations allowed for each MI-ES run is relatively problem-dependent (see the experimental setup sections in Chapter 4).

Evolutionary Algorithms The $(1 + \lambda)$ evolutionary algorithm with self-adjusting mutation rate [25], shortly $(1 + \lambda)EA_{r/2,2r}$ (the $r/2$ and $2r$ are the two mutation rates controlled by a self-adjusting parameter r), is implemented for optimizing acquisition function in handling pseudo-boolean (binary-encoding) problems. The hyper-parameters are determined in reference to the settings described in [27].

In Section 3.3, parallelism is assumed in managing multiple surrogate models. Similarly, SAMA-DiEGO can be equipped with a parallel multi-start optimization technique (MultiStartOptimize in line 20 of Algorithm 2). The idea is straightforward: concurrently running multiple back-end optimizers from different starting points. These optimizers find various local optimal solutions of surrogate model and the acquisition function. After the back-end optimizers have converged or stopped, the solution with the highest infill criteria score is selected.

This becomes certainly profitable if the surrogate model or the acquisition function features numerous local optimum. The starting points in SAMA-DiEGO are previous optimal solutions and random samples. It might be beneficial to introduce in other heuristically generated samples as starting points but this topic is not within the consideration of our experiments and further investigation is highly recommended in the future. As a matter of fact, existing studies on paralleling the acquisition process of EGO primarily focused on generalizing old or developing novel infill criteria. Famous examples are Ginsbourger et al. [38, 39] invents a multi-point Expected Improvement function (q-EI), which uses either a so-called *Constant Liar* or a *Kriging Believer* strategy to concurrently selects multiple infill points during the acquisition process and evaluates the points in parallel on the original problem. Wang et al. [89] further develops a novel stochastic gradient estimator based on infinitesimal perturbation analysis to speed up the process of maximizing q-EI concurrently on up to 128 infill points. Similar strategies are applied in [50, 85]. These methods have a prerequisite that it is feasible to compute uncertainty on the predictions of surrogate models. However, to the best of our knowledge, there is an absence of studies on uncertainty quantification mechanism for support vector regression. Thus, in compliance with using all the implemented surrogates, we consider our method described in this section as more appropriate one for the uniformity of algorithm.

3.5 Implemented Surrogate Models

As described in Section 2.2, four categories of surrogate models (RBF, Kriging, SVM and RF) are considered in SAMA-DiEGO. The surrogate pool contains 31 models in total, i.e., nine RBF interpolation models, fifteen (5×3) Kriging models, six SVM regression models and one RF model:

RBF A total of nine Euclidean-distance-based radial basis functions ($\varphi(\|\cdot\|)$ in Section 2.2) are considered in this thesis as an extended study of [22, 23]. Suppose using $d_{ij} = \|\vec{x_i} - \vec{x_j}\|_2$ to shortly describe the Euclidean distance between two data samples in the search space $\mathcal{S}$, i.e., $\vec{x_i}, \vec{x_j} \in \mathcal{S}$. The definitions of nine radial basis functions can be therefore written in Table 3.5.1. Moreover, only linear tail $p(\vec{x}) = \sum_{z_t \in \vec{x}} \mu_t \cdot z_t + 1$ is considered in the experiments.

Kriging Five correlation functions together with three types of basic functions are adopted in SAMA-DiEGO. A summary of these functions are given in Table 3.5.2. More details regarding the implementations are introduced in [13].

Table 3.5.1: Specifications of the nine radial basis functions in SAMA-DiEGO (see Section 2.2). Specifically, $\log(d_{ij})$ will programmatically return 1 if $d_{ij} = 0$.

Name	Definition (φ)
Linear	d_{ij}
Cubic	$d_{ij} \cdot d_{ij} \cdot d_{ij}$
Thin plate spline	$d_{ij} \cdot d_{ij} \cdot \log(d_{ij})$
Polyharmonic spline 4	$d_{ij} \cdot d_{ij} \cdot d_{ij} \cdot d_{ij} \cdot \log(d_{ij})$
Polyharmonic spline 5	$d_{ij} \cdot d_{ij} \cdot d_{ij} \cdot d_{ij} \cdot d_{ij}$
Multiquadric	$\sqrt{1 + (d_{ij} \cdot d_{ij})}$
Gaussian function	$exp(-d_{ij} \cdot d_{ij})$
Invmultiquadric	$1/\sqrt{1 + d_{ij} \cdot d_{ij}}$
Invquadric	$1/(1 + d_{ij} \cdot d_{ij})$

Table 3.5.2: Three basic functions and five correlation functions used for Kriging interpolation. Additionally, the mathematical definitions of correlation functions are given in Appendix 6.1.1

Correlation Functions $C\left(\vec{x}^{(i)}, \vec{x}^{(j)}\right)$
Ornstein–Uhlenbeck Process (OUP)
Squared Gaussian Correlation (SGC)
Matérn Correlation 3/2 (Matern32)
Matérn Correlation 5/2 (Matern52)
Gower Distance (Gower)

Basic Functions $\xi_i(\vec{x})$	Specification
Constant	1
Linear	x_i
Quadratic	$x_i \cdot x_i$

SVM and RF As it is introduced in Section 2.2, the implemented SVM and RF in SAMA-DiEGO are mostly based on the scikit-learn [69] package. The six kernels defined on two example data samples x_i, x_j for $\epsilon-$SVM regression are shown in Table 3.5.3, where $\langle\ ,\ \rangle$ is the inner product of two samples, γ and r are two hyper-parameters and d is the degree of polynomial kernel. Three degrees, 2, 3 and 5 are considered for the polynomial kernels of SVM regression in SAMA-DiEGO. The γ and r along with other hyperparameters are set using their default values in scikit-learn.

With respect to random forest, the default implementation and hyper-parameters provided by scikit-learn are used. Note that starting from 0.22, scikit-learn ensembles 100 decision trees in one random forest regression instead of 10 in previous versions and our implementation

Table 3.5.3: The four types of SVM kernels in SAMA-DiEGO (see Section 2.2).

Kernel Name (API)	Definition
linear	$\langle \vec{x_i}, \vec{x_j} \rangle$
poly	$(\gamma \langle \vec{x_i}, \vec{x_j} \rangle + r)^d$
rbf	$\exp(-\gamma \|\vec{x_i} - \vec{x_j}\|^2)$
sigmoid	$\tanh(\gamma \langle \vec{x_i}, \vec{x_j} \rangle + r)$

is based on the 0.24.2 version.

4

Experiments and Results

In this chapter, three evaluations of our SAMA-DiEGO on two types of problem sets with different search spaces are presented. General setups of the three experiments are introduced in Section 4.1. Meanwhile Sections 4.2, 4.3 and 4.4 consists of two parts, experimental setup and results. In experimental setup (Sections 4.2.1, 4.3.1 and 4.4.2), overview of the chosen test problems together with the problem-dependent settings of benchmark algorithms are discussed, whereas SAMA-DiEGO is compared with several existing optimization methods in the results section.

4.1 General Setup

Experiments are carried out on three problem sets:

Single Pseudo-boolean Optimization (PBO) problems Five binary-encoded combinatorial problems introduced in [27];

Composite Pseudo-boolean Optimization (PBO) problems Nine composite binary-encoded combinatorial problems. Each problem is a linear combination of two PBO single problem proposed in [27];

Discretized Black-box Optimization Benchmark (BBOB) problems Fifteen discretized BBOB problems [46] defined on ordinal variables.

Each benchmark algorithm is experimented with 11 independent runs per test function as advised in [27]. All the designed experiments are conducted on Latinum, Octiron and Uridium servers in the LIACS Data Science Lab.

4.2 A Study on Single PBO Problems

4.2.1 Experimental Setup

Test Problems To identify the superiority of utilizing a surrogate model when applying categorical evolutionary strategy [60] to solve combinatorial problems, Horesh et al. [48] chose five famous combinatorial maximization problems:

LABS (F18): low auto-correlation binary sequences problem

Ising1D ring (F19) Ising problem defined on one-dimensional ring

Ising2D torus (F20) Ising problem defined on two-dimensional torus

MIVS (F22): maximum independent vertex set problem

NQP (F23): N-queens problem

These problems are further nicely implemented, tested and integrated into the PBO benchmark [27] for evaluating and comparing combinatorial optimization algorithms. Consequently, the five famous problems are chosen as the test problems for our first experiment. Mathematical formulations of these five tasks in binary-encoding are summarized in Appendix 6.1.2 based on the description of [27]. The global optima of the five functions are known regardless of dimensionality. Moreover, the dimensionalities of problems are determined in compliance with the settings of [48], which are {25, 64, 100}.

Benchmarked Algorithms Eleven algorithms that consist of four types of SAMA-DiEGO and three types of other algorithms are experimented (see Table 4.2.1) on PBO problems. The $(1 + \lambda)$ EAs are mutation only evolutionary algorithms each with a distinct strategy to control its mutation rate [16, 25, 92]. The SA $(1+(\lambda, \lambda))$ GA is a $(1+(\lambda, \lambda))$ genetic algorithm with self-adjusting parameters [24]. Except for the traditional evolutionary algorithms, the Mixed-Integer Parallel Efficient Global Optimization (MIP-EGO) [85] which uses a random forest regressor as surrogate and mixed-integer evolutionary strategy (MI-ES) as solver is also taken into consideration.

The hyper-parameters of evolutionary (genetic) algorithms are set according to their C++ implementations provided by [27]. Experiments on MIP-EGO are carried on its official open-source implementation[1]. The maximum number of fitness evaluations (calls to the real objective function) is fixed to 500 for all benchmarked algorithms across three dimensionalities.

[1] https://github.com/wangronin/MIP-EGO

Table 4.2.1: A table of algorithms implemented for benchmarking on PBO problems. The hyperparameters of all benchmark algorithms except SAMA-DiEGO are taken from their papers and public-available implementations, respectively.

Algorithm	Specification		
$(1+\lambda)$ EA	Type	Strategy to decide mutation strength	
	static	Use one static mutation rate	
	$r/2, 2r$	Use two self-adaptive mutation rates [25]	
	norm	Sample from a normal distribution [92]	
	var	The self-adaptive version of *norm* [92]	
	logN	Use a log-Normal self-adaptive mutation rate (starting with 0.2) [16]	
	Population size λ is set to 10 for all variants of $(1+\lambda)$ EA		
	A family of non-surrogate assisted/based solvers.		
SAMA-DiEGO	Type	Acquisition Function	Back-end Optimizer
	A	Prediction Value	MI-ES
	B	Expected Improvement	MI-ES
	C	Prediction Value	$(1+10)EA_{r/2,2r}$
	D	Expected Improvement	$(1+10)EA_{r/2,2r}$
SA $(1+(\lambda,\lambda))$ GA	A self-adjusting genetic algorithm [24]		
	Population size λ is initialized to 25 and self-updated after each iteration.		
	A non-surrogate assisted/based solver.		
MIP-EGO [85]	Mixed-Integer Parallel Efficient Global Optimization		
	Use random forest regression as surrogate model.		
	Use moment-generating function of improvement.		
	Use MI-ES as back-end optimizer		

In addition to the benchmarking performance of algorithms mentioned in Table 4.2.1, results obtained by two more algorithms reported in [48] are included in our discussion. As described in [48], both algorithms were benchmarked with tuned hyper-parameters and were allowed to use up to 2100 calls to the real objective function:

CatES : Evolutionary Strategy with MIES-based categorical self-adaptive mutation operator [60].

SVM-CatES : Support vector machines regression assisted CatES [48].

Configurations of SAMA-DiEGO Upon running, in our experiments, SAMA-DiEGO starts with the entire surrogate model pool mentioned in Section 3.5, among all these available surrogates, top *seven* models will survive the model verification stage (see Section 3.3.1), this number is a hyper-parameter (T in Algorithm 3) and here, seven is selected in consideration of our available computational resources for parallelism. Moreover, two criteria (namely E in Algorithm 3) are applied simultaneously in the verification stage:

Validation Error : The initial randomly generated data samples are randomly divided into a training part with 70% of total samples and a validation part with the other 30% samples, this ratio is

flexible and it can be changed if needed. All surrogates are fitted with training samples and tested with the validation samples to acquire model's mean squared errors which are later used for ranking model.

Time : The maximum running time for each model to fit the training data is set to 30 seconds to limit the overall execution time of the algorithm. This means our algorithm will disqualify a surrogate if it takes more than 30 seconds to converge, regardless of their performance. An alternative strategy, in contrast to our choice, is to halt the fitting process of a surrogate model as long as it uses up the time limit.

The initial sampling size of all SAMA-DiEGOs by default is set to the dimensionalities of problem plus 1. In an effort to tidily manage computing resources, the number of paralleled multi-start back-end solvers (see Section 3.4) is designed to the number of qualified models after model verification, which is 7 in our experiments. This number is determined on the basis of our available computation resources. The maximum allowed number of function calls to the surrogate for the back-end solver is a hyper-parameter and it is unintentionally set to 500 times of the dimensionality in this experiment.

4.2.2 Analysis of Results

The results of the experiments are shown in Table 4.2.2. For each test case, instead of showing all results obtained by five $(1 + 10)$ EA algorithms (see Table 4.2.1), the result of the best-performed $(1 + 10)$ EA according to the Wilcoxon rank sum significance test is exclusively shown in the result table. SAMA-DiEGO yields best performance on fourteen of the fifteen test cases except for the 64-dimensional MIVS problems where the (1+10) EA obtained better result. Moreover, as it is indicated in **Budgets Mean** columns of the result table, all types of SAMA-DiEGO can find the global optimum of Ising problems across three dimensionalities and are also capable of solving NQP and MIVS on 25 dimensions within 500 fitness evaluations. Unexpectedly, the single surrogate assisted algorithms MIP-EGO performed worse than the (1+10) evolutionary algorithms in fourteen of the fifteen cases. Since SAMA-DiEGO (C and D), SVM-CatES and MIP-EGO all use MI-ES as back-end solvers, it can be inferred that the gaps in performance of the three algorithms shall be ascribed to the performance of their individual surrogate models.

Furthermore, it can be observed that SAMA-DiEGO-B with EI for acquistion function and MI-ES for back-end optimizer achieve better performance on 64 and 100 dimensions in comparison to other three types. Therefore,

Table 4.2.2: The experimental results obtained on PBO defined on 25, 64 and 100 dimensions. The results are highlighted in blue if they are significantly better according to Wilcoxon rank sum test (Mann-Whitney U test) with a confidence level of 0.95. Additionally, the *Budgets Mean* columns show the average number of fitness evaluations over 11 runs used by algorithms to locate the global optimum (if found otherwise 500). Such results for ES and SVM-CatES are not available in case they are not provided in [48].

Function	Algorithm Group	Dimension = 25				Dimension = 64				Dimension = 100			
		Fitness Value			Budgets	Fitness Value			Budgets	Fitness Value			Budgets
		Mean	Std	Best	Mean*	Mean	Std	Best	Mean*	Mean	Std	Best	Mean*
Ising1D	ES	43.36	N/A	50.0	N/A	96.48	N/A	108.0	N/A	142.32	N/A	152.0	N/A
	SVM-CatES	46.8	N/A	50.0	N/A	107.12	N/A	112.0	N/A	154.88	N/A	168.0	N/A
	MIP-EGO	43.0	1.78	46	500	90.18	2.62	96	500	130.91	1.78	132.00	500
	$(1+10)$ EA	43.5	3.09	50	485	105.45	3.09	112	500	162.18	7.88	172.00	500
	SA (1+(25,25)) GA	43.5	1.92	46	500	103.27	2.86	108	500	153.45	5.47	164.00	500
	SAMA-DiEGO-A	50.0	0.0	50.0	259	128	0	128	99	200.00	0.00	200.00	125
	SAMA-DiEGO-B	50.0	0.0	50.0	291	128	0	128	66	200.00	0.00	200.00	106
	SAMA-DiEGO-C	50.0	0.0	50.0	74	128	0	128	101	200.00	0.00	200.00	115
	SAMA-DiEGO-D	50.0	0.0	50.0	27	128	0	128	66	200.00	0.00	200.00	105
Ising2D	ES	84.08	N/A	100.0	N/A	181	N/A	196.0	N/A	266.88	N/A	304.0	N/A
	SVM-CatES	99.84	N/A	100.0	N/A	203.28	N/A	232.0	N/A	288.88	N/A	308.0	N/A
	MIP-EGO	78.54	2.14	88.0	500	166.55	3.53	172	500	245.82	5.75	252.00	500
	$(1+10)$ EA	90.45	9.38	100.0	407	198.55	9.99	212	500	306.91	9.96	328.00	500
	SA (1+(25,25)) GA	88.72	8.14	100.0	485	192.73	7.78	204	500	284.73	5.86	292.00	500
	SAMA-DiEGO-A	100.0	0.0	100.0	150	256	0	256	85	400.00	0.00	400.00	123
	SAMA-DiEGO-B	100.0	0.0	100.0	177	256	0	256	68	400.00	0.00	400.00	110
	SAMA-DiEGO-C	100.0	0.0	100.0	58	256	0	256	85	400.00	0.00	400.00	130
	SAMA-DiEGO-D	100.0	0.0	100.0	27	256	0	256	66	400.00	0.00	400.00	105
NQP	ES	4.16	N/A	5.0	N/A	-36.96	N/A	-3.00	N/A	-264.3	N/A	24.0	N/A
	SVM-CatES	4.56	N/A	5.0	N/A	2.64	N/A	6.00	N/A	-15.2	N/A	26.0	N/A
	MIP-EGO	-0.73	2.67	3.0	500	-247.55	29.94	-193.00	500	-753.3	53.8	-641.0	500
	$(1+10)$ EA	4.36	0.64	5.0	392	3.55	5.25	8.00	500	-46.6	32.8	7.0	500
	SA (1+(25,25)) GA	3.81	0.93	5.0	451	-59.55	22.92	-25.00	500	-288.6	73.7	-186.0	500
	SAMA-DiEGO-A	5.0	0.0	5.0	195	3.36	0.48	4.00	500	6.0	1.0	7.0	500
	SAMA-DiEGO-B	5.0	0.0	5.0	260	6.00	0.43	7.00	500	6.3	0.6	7.0	500
	SAMA-DiEGO-C	5.0	0.0	5.0	260	4.18	0.57	5.00	500	6.3	1.2	8.0	500
	SAMA-DiEGO-D	5.0	0.0	5.0	255	4.55	0.50	5.00	500	-4.4	8.4	4.0	500
LABS	ES	3.93	N/A	6.01	N/A	2.58	N/A	3.58	N/A	2.11	N/A	2.73	N/A
	SVM-CatES	4.31	N/A	6.51	N/A	2.76	N/A	4.03	N/A	2.24	N/A	2.94	N/A
	MIP-EGO	3.26	0.45	4.34	500	2.17	0.2	2.69	500	1.77	0.08	1.91	500
	$(1+10)$ EA	4.16	0.75	4.88	500	3.31	0.32	3.97	500	3.10	0.27	3.55	500
	SA (1+(25,25)) GA	4.01	0.94	6.51	500	2.53	0.21	2.83	500	2.19	0.17	2.48	500
	SAMA-DiEGO-A	4.80	0.78	6.51	500	2.71	0.21	3.03	500	2.77	0.31	3.32	500
	SAMA-DiEGO-B	4.10	0.62	5.58	500	3.69	0.21	4.23	500	3.17	0.34	3.74	500
	SAMA-DiEGO-C	4.08	0.59	5.21	500	3.29	0.38	3.94	500	2.48	0.26	2.88	500
	SAMA-DiEGO-D	4.57	0.54	5.21	500	3.72	0.34	4.53	500	2.48	0.37	3.06	500
MIVS	ES	10.84	N/A	12.0	N/A	-1.0	N/A	24.0	N/A	-308.6	N/A	-99.0	N/A
	SVM-CatES	11.66	N/A	12.0	N/A	17.44	N/A	23.0	N/A	-63.18	N/A	-3.0	N/A
	MIP-EGO	8.82	0.83	10.0	500	-370.36	127.11	-116.00	500	-1657.18	263.19	-1069.00	500
	$(1+10)$ EA	11.45	0.89	12.0	271	26.00	1.54	29.00	500	-21.73	77.64	36.00	500
	SA (1+(25,25)) GA	10.63	1.30	12.0	431	-17.18	49.56	21.00	500	-353.36	251.72	23.00	500
	SAMA-DiEGO-A	12.0	0.0	12.0	276	11.45	0.99	13.00	500	23.45	6.43	36.00	500
	SAMA-DiEGO-B	12.0	0.0	12.0	299	22.73	2.18	26.00	500	28.73	4.05	34.00	500
	SAMA-DiEGO-C	12.0	0.0	12.0	282	11.73	1.60	14.00	500	19.73	3.49	24.00	500
	SAMA-DiEGO-D	12.0	0.0	12.0	348	10.91	0.90	13.00	500	-182.18	140.53	16.00	500

it is reasonable to consider that SAMA-DiEGO-B is more suitable to solve higher dimensional problems in comparison with its three sibling algorithms.

4.3 A Study on Composite PBO Problems

4.3.1 Experimental Setup

Test Problems An initiative of organizing this experiment is to extend the previous study on five single problem to more PBO single problems at an acceptable computation cost. Similar to the five single problems introduced in previous section, the PBO problem set [27] also hold

other 18 functions which, as illustrated in the paper, are primarily organised by transforming two single problems:

OneMax : count the number of ones in the function.

LeadingOnes : count the number of initial ones in the functions.

Our proposed evaluation method is, despite of experimenting on every single PBO problems, the benchmark algorithms are evaluated on nine composite PBO problems ($g(\vec{x})$ in Eq. 4.3.1).

These problems are created by linearly combining two single PBO problems as follows:

$$g(\vec{x}) = 0.5 \cdot f_1(\vec{x}) + 0.5 \cdot f_2(\vec{x}) \quad \vec{x} \in [0,1]^n, \tag{4.3.1}$$

where n is the dimensionality of the problem, f_1, f_2 are two different single PBO problems. A specification of the nine pair of single problems used to create composite problems are given in Table 4.3.1. The first five cases (Case 1 to Case 5) are determined by us based on conclusions of [27] and our previous study on single PBO problems (Table 4.2.2):

Case 1 : F10 is the hardest OneMax problems and F17 is the second-hardest LeadingOnes problem as concluded in [27].

Case 2 : The composite function is a combination of two single PBO problems (F18 LABS and F20 Ising2D on Torus) tested in Section 4.2.

Case 3 : Combining a previously tested PBO problem (MIVS) with a new Ising2D problem defined on triangular lattice graph.

Case 4 : *Concatenated Trap* (F24) and *NK landscapes* (F25) are two problems added into the official repository[2] of the PBO problem set [27].

Case 5 : A combination of F19 Ising1D on Ring and F23 N-queen problems.

On the contrary, the choices of single problems in the rest four cases (Case 6 to Case 9) are uniformly randomly sampled from F1 to F25. All nine composite problems are instantiated on 25 and 36 dimensions. Moreover, the *instance* mechanism proposed in [27], which camouflages all input bit strings by methodical permutations, is applied to introduce in more difficulties. Doerr et al. [27] mentioned that the *instance id* which controls the permutation can be any arbitrary value. Therefore, this value is unintentionally set to 7 in this study.

[2]https://iohprofiler.github.io/IOHproblem/

Table 4.3.1: The nine pairs of single PBO problems used to create composite PBO problems (Eq. 4.3.1).

	Function 1	Function 2
Case 1	OneMax with fitness perturbation (F10)	LeadingOnes with fitness perturbation (F17)
Case 2	LABS (F18)	Ising2D on Torus (F20)
Case 3	Ising2D on Triangular Lattice (F21)	MIVS (F22)
Case 4	Concatenated Trap (F24)	NK Landscapes (F25)
Case 5	Ising1D on Ring (F19)	N-queen Problems (F23)
Case 6	Ising2D on Torus (F20)	LeadingOnes with Neutrality (F13) Input bit string is downsampled.
Case 7	MIVS (F22)	OneMax with Epistasis (F7) Input bit string is locally perturbated
Case 8	LeadingOnes with Dummy Variables (F12)	NK Landscapes (F25)
Case 9	OneMax with Dummy Variables (F5)	LeadingOnes with fitness perturbation (F16)

Configurations of Benchmarked Algorithms The eleven algorithms (Table 4.2.1) tested on PBO single problems are also benchmarked on composite PBO problems. Moreover, another efficient non-surrogate solver called Univariate Marginal Distribution Algorithm [66], namely UMDA, is employed as well. Implementations of UMDA and its hyperparameters are also determined in accordance to [27]. The maximum number of fitness evaluations (calls to real objective function) is specified to 200 for 25 dimensions and 300 for 36 dimensions, which are approximately eight times of the dimensionality. Only SAMA-DiEGO-C which uses prediction value as infill criteria and (1+10) EA with two self-adjusting mutation rates ($r/2, 2r$ in Table 4.2.1) as back-end solver was benchmarked due to the availability of computation resources. The maximum number of calls to the surrogate for the back-end solver is still 500 times the dimensionality, which remains the same as that in previous single PBO test cases.

4.3.2 Analysis of Results

It can be seen from the results in Table 4.3.2, SAMA-DiEGO clearly outperforms other benchmark algorithms in all cases on 25 dimensions and eight of the nine cases on 36 dimensions at the significance level of 0.05 of a Mann-Whitney U test. The $(1 + 10)$ EA family achieved a comparable result on 36 dimensions in Cases 4, 7 and 8 if compared to SAMA-DiEGO.

Moreover, the global optimum of each test problem on 25 dimensions is found using a brute-force search over all 2^{25} solutions. The number of runs that a benchmark algorithm found the global optimum per test problem is recorded in the **Hit** column of the table. It can be observed that SAMA-DiEGO-C can locate the optimum in Cases 3, 5 and 9 in all the eleven independent runs, and in at least nine out of eleven runs in Cases 2 and 6. Additionally, SAMA-DiEGO is the only algorithm that can reach the global optimum in Cases 1 and 7. It can be concluded that among all

the benchmarked algorithms in this study, SAMA-DiEGO-C shows higher capability of finding promising results in limited budgets.

The performance of MIP-EGO can be improved if hyper-parameter optimization is applied beforehand as it is suggested by its authors [85]. However, we argue that the influence of hyper-parameters on performance should not be critical for surrogate assisted algorithm as the essential purpose of using surrogate is to save calls to original problems.

4.4 A Study on Discretized BBOB

Having discussed how our algorithms performed on the pseudo-boolean *maximization* problems, the last experiment addresses benchmarking on integer-valued *minimization* problems. The benchmark is constructed by discretizing the well-known Black-Box Optimization Benchmark noiseless (BBOB-noiseless) [46] as the original BBOB-noiseless focuses on continuous *minimization* problems. The original BBOB-noiseless problem set contains 24 specifically designed noise-free-problems which can be classified into five categories with respect to their properties e.g. degree of conditioning and structural characteristics [46]. Each of these problem challenges a search algorithm from unique perspectives and is capable of suggesting the effectiveness of an algorithm in handling similar problems. More details regarding definition and property of these problems can be found in [46]. The utilized discretization method in our study is proposed in and implemented by [70]. More concrete explanations can be found in [83].

Similar to the categorical space defined in previous two PBO experiments, the ordinal search space for BBOB shall hold a sizeable amount of solutions, on which it is generally infeasible to conduct brute-force search. Bossek et al. [11] considered 10 as the highest dimension to benchmark a Bayesian optimization algorithm on the original BBOB-noiseless function set. In reference to their settings along with our expectation of challenging benchmark algorithms with an enormous search space, a solution in our experiment contains 15 variables and each variable has 101 values (0 to 100). Therefore, the search space comprises approximately 10^{30} distinct solutions in total.

4.4.1 A Preliminary Study on Discretized BBOB Problems

An independent preliminary study is firstly carried out to identify the capability of all the implemented surrogate models (see Section 3.5) in fitting the created discretized BBOB problems. The first five problems (P1 to P5) are not taken into considerations for the reason that they are separable functions, which means the search process on these problems can be simply reduced to multiple one-dimensional search procedures [46]. Thus, our interest is on the other 19 non-separable functions (P6 to P24) that are more complex and more difficult to solve. For each function between F9 to F24, 500 data

Table 4.3.2: Performance of algorithms benchmarked on nine PBO composite maximization problems defined on 25 and 36 dimensions. The result(s) are highlighted in blue if they are significantly better according to Wilcoxon rank sum test (Mann-Whitney U test) with $\alpha = 0.05$. On 25 dimensions, the number of runs that each algorithm found global optimum per test case is recorded in the *Hit* column.

Case	Algorithm Group	Dimension = 25				Dimension = 36		
		Fitness Value			Hit	Fitness Value		
		Mean	Std	Best		Mean	Std	Best
Case 1	$(1+10)$ *EA*	-742.34	2.64	-737.35	0	-738.46	1.84	-734.59
	SA (1+(25,25)) GA	-742.55	1.93	-737.35	0	-738.28	1.33	-736.56
	UMDA	-741.01	2.12	-737.35	0	-737.21	2.60	-731.82
	MIP-EGO	-740.40	1.94	-737.75	0	-737.82	1.40	-735.38
	SAMA-DiEGO-C	-737.78	3.19	-734.59	2	-728.52	3.07	-725.90
Case 2	$(1+10)$ *EA*	-738.04	0.96	-736.13	0	-731.34	1.27	-729.01
	SA (1+(25,25)) GA	-738.50	1.04	-737.65	0	-731.98	1.23	-729.80
	UMDA	-739.85	0.72	-738.45	0	-734.20	0.85	-732.84
	MIP-EGO	-739.21	0.66	-737.66	0	-733.98	0.87	-732.05
	SAMA-DiEGO-C	-734.84	0.89	-734.56	10	-728.22	2.09	-725.88
Case 3	$(1+10)$ *EA*	-726.15	1.70	-724.72	6	-712.87	1.41	-711.69
	SA (1+(25,25)) GA	-727.98	1.81	-724.72	1	-720.05	6.21	-713.66
	UMDA	-728.34	1.46	-724.72	1	-715.53	1.39	-713.66
	MIP-EGO	-731.72	1.46	-728.66	0	-733.80	9.35	-721.16
	SAMA-DiEGO-C	-724.72	0.00	-724.72	11	-711.87	0.57	-711.69
Case 4	$(1+10)$ *EA*	-752.94	0.09	-752.77	0	-751.92	0.12	-751.74
	SA (1+(25,25)) GA	-752.99	0.06	-752.91	0	-752.20	0.12	-751.96
	UMDA	-753.06	0.07	-752.98	0	-752.32	0.10	-752.12
	MIP-EGO	-753.12	0.04	-753.08	0	-752.42	0.09	-752.28
	SAMA-DiEGO-C	-752.89	0.04	-752.78	0	-751.92	0.05	-751.81
Case 5	$(1+10)$ *EA*	-744.64	0.31	-744.46	8	-740.69	0.90	-740.11
	SA (1+(25,25)) GA	-745.82	1.64	-744.46	3	-746.29	4.33	-740.51
	UMDA	-745.32	0.67	-744.46	1	-740.83	0.33	-740.51
	MIP-EGO	-748.69	1.64	-744.85	0	-758.99	2.70	-754.33
	SAMA-DiEGO-C	-744.46	0.00	-744.46	11	-740.11	0.00	-740.11
Case 6	$(1+10)$ *EA*	-734.87	1.95	-731.43	1	-728.20	3.47	-722.74
	SA (1+(25,25)) GA	-735.88	2.53	-731.43	1	-729.42	2.38	-725.11
	UMDA	-737.75	1.85	-733.01	0	-732.58	1.46	-730.24
	MIP-EGO	-737.31	1.39	-734.59	0	-731.75	1.09	-729.85
	SAMA-DiEGO-C	-732.43	2.14	-731.43	9	-723.14	3.32	-721.16
Case 7	$(1+10)$ *EA*	-743.77	0.74	-742.48	0	-739.22	0.92	-737.75
	SA (1+(25,25)) GA	-744.21	0.83	-742.88	0	-742.77	5.91	-738.53
	UMDA	-745.46	0.93	-743.67	0	-744.06	4.44	-741.30
	MIP-EGO	-744.82	0.85	-743.27	0	-756.48	8.46	-742.48
	SAMA-DiEGO-C	-743.42	0.74	-742.09	1	-739.68	1.26	-737.35
Case 8	$(1+10)$ *EA*	-748.88	0.62	-747.82	0	-747.12	2.82	-741.86
	SA (1+(25,25)) GA	-750.05	0.70	-748.58	0	-748.21	1.58	-744.24
	UMDA	-750.86	0.65	-749.79	0	-749.88	0.98	-747.80
	MIP-EGO	-748.83	0.94	-746.24	0	-749.06	0.50	-747.79
	SAMA-DiEGO-C	-747.15	1.35	-745.84	4	-746.12	2.46	-741.85
Case 9	$(1+10)$ *EA*	-738.86	3.00	-735.77	4	-733.73	4.07	-727.48
	SA (1+(25,25)) GA	-740.11	2.14	-737.75	0	-736.60	1.33	-735.38
	UMDA	-740.94	1.64	-738.53	0	-736.60	1.42	-734.59
	MIP-EGO	-742.02	0.82	-740.51	0	-739.00	0.99	-737.35
	SAMA-DiEGO-C	-735.77	0.00	-735.77	11	-727.48	0.00	-727.48

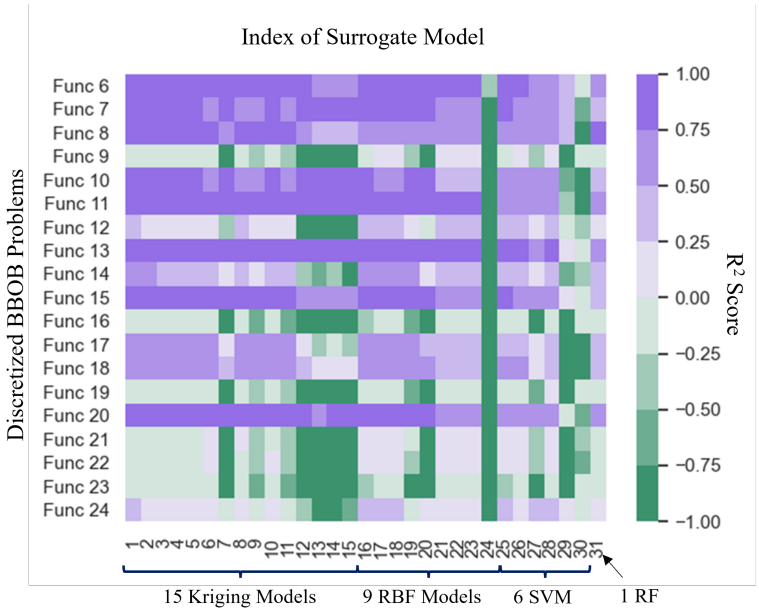


Figure 4.4.1: The heat map of R^2 scores of 31 surrogate models obtained on 19 discretized BBOB functions using 500 data samples. Notably, all scores that are less than -1 are raised to -1 for nicer visualization.

samples (input solutions and their output, i.e., values of objective function for these solutions) are generated by performing Latin hyper cube sampling (see Section 3.2). The data samples are later utilized to test the performance (using R^2 score) of surrogate models through a 5-fold cross-validation. The whole procedure is then repeated twice with different data samples to obtain more reliable outcomes. R^2 score is chosen since it is independent from the scale of problems.

Through the proportion of explained variance, R^2 score can suggest goodness of fit of a surrogate model and, as a result, a measure of how well the model is likely to predict unseen landscape of a problem. Generally speaking, the closer the score is to one, the better the surrogate model performs. A heat map is shown in Fig. 4.4.1 describing the results (mean R^2 score over three repetitions of 5-fold cross-validation) obtained by each surrogate model on 19 discretized BBOB functions. The purple cells are cases where surrogate models obtained a mean R^2 score larger than 0. The raw scores are given in Appendix Tab. 6.2.1. It is possible to roughly categorize the discretized BBOB problems into three groups based on the wellness of fitting of the implemented surrogates:

Good (G) Most of the surrogates obtained R^2 scores greater than 0.5 on **problems 6, 7, 8, 10, 11, 13, 15 and 20**, which suggests that these models can extrapolate unseen landscapes of problems from available

data samples.

Moderate (M) Most of the surrogates obtained R^2 scores that are greater than 0 but less than 0.5 on **problems 12, 14, 17, 18 and 24**, whereas the other models only got negative scores. It means that surrogate are capable of explaining some landscapes of the problems.

Bad (B) It seems to be very hard for surrogate models to interpolate **problems 9, 16, 19, 21, 22, 23** as nearly all models acquired negative R^2 scores. Considering the modality of problems, except for the problem 9 which is uni-modal, the other 5 are all multi-modal problems.

4.4.2 Experimental Setup

Test Problems Given the results of preliminary study, all the six *bad* problems are chosen since it is important to study the performance of a surrogate assisted algorithm (SAMA-DiEGO) if its back-end surrogate(s) struggle with interpolating problems. Moreover, nine out of thirteen *simple* and *moderate* problems are randomly sampled into the test problems. The full list of fifteen test problems are given below, the capital letter G, M, B is the wellness of fitting as described in previous section.

Problem 6 (G) : Attractive Sector Function

Problem 7 (G) : Step Ellipsoidal Function

Problem 8 (G) : Rosenbrock Function

Problem 9 (B) : Rosenbrock Function, rotated

Problem 11 (G) : Discus Function

Problem 13 (G) : Sharp Ridge Function

Problem 15 (G) : Rastrigin Function

Problem 16 (B) : Weierstrass Function

Problem 18 (M) : Schaffers F7 Function, moderately ill-conditioned

Problem 19 (B) : Composite Griewank-Rosenbrock Function F8F2

Problem 20 (G) : Schwefel Function

Problem 21 (B) : Gallagher's Gaussian 101-me Peaks Function

Problem 22 (B) : Gallagher's Gaussian 21-hi Peaks Function

Problem 23 (B) : Katsuura Function

Problem 24 (M) : Lunacek bi-Rastrigin Function

More descriptions on the definitions and properties of these functions can be found in [46].

Benchmarked Algorithms Three famous surrogate assisted algorithms that can exactly accommodate to discrete variables, namely Sequential Model Algorithm Configuration (SMAC) [50, 62], adaptive Tree-structured Parzen Estimator approach (TPE) [8] and MIP-EGO [85] are selected. Moreover, two non-surrogate assisted optimization algorithms are also experimented with, namely MI-ES [60] and P3-GOMEA [28]. MI-ES is the back-end solver of SAMA-DiEGO and MIP-EGO, therefore it can help to identify whether it is beneficial to use surrogate model to solve ordinal problems. P3-GOMEA is a state-of-the-art version of the Gene-Pool Optimal Mixing Algorithm (GOMEA) [82] that can efficiently handle discrete optimization problems. As for SAMA-DiEGO, only the A, B types are considered (see Table 4.2.1) since the alternative back-end optimizer $(1 + 10)$ $EA_{r/2,2r}$ does not directly fit for ordinal variables. Comparing the result of A type and B type of SAMA-DiEGO can also provide guidance to the selection of infill criteria (i.e. Prediction Value versus Expected Improvement).

All algorithms are allowed to use 500 fitness evaluations (calls to real objective function), which is more than 30 times of the dimensionality of problems. With this relatively abundant budget, it is feasible to analyse the convergence speed. Additionally, the initial sampling sizes for SAMA-DiEGO and MIP-EGO are both set to 10% of the total budgets in reference to the suggestions of [11].

With respect to realization of algorithms, the benchmarking is executed upon the widely used and stable implementations of SMAC[3] (v1.1.1) [62], TPE[4] (v0.2.5) [9], MIP-EGO[5] (v2.0.0) and P3-GOMEA[6].

4.4.3 Analysis of Results

Discussions on the experimental results are divided into two parts with regard to the wellness of surrogate models in fitting problems (Section 4.4.1).

***Good* and *Moderate* problems** The results showed in Table 4.4.1 are diverse with respect to the best-performed algorithm for different test functions. SAMA-DiEGO-A with prediction value as infill criteria and MI-ES as back-end solvers outperformed in four out of eight test cases (problems 6, 7, 8, 11 and 20), but slightly lost to adaptive TPE on P18 and SMAC on P13 and P15. What's more, SAMA-DiEGO-B leads the performance with respect to minimum average and median objective values on problem 24 to a small degree. Lastly, adaptive TPE per-

[3]https://automl.github.io/SMAC3/master
[4]https://github.com/hyperopt/hyperopt
[5]https://github.com/wangronin/MIP-EGO
[6]https://github.com/ArkadiyD/SAGOMEA

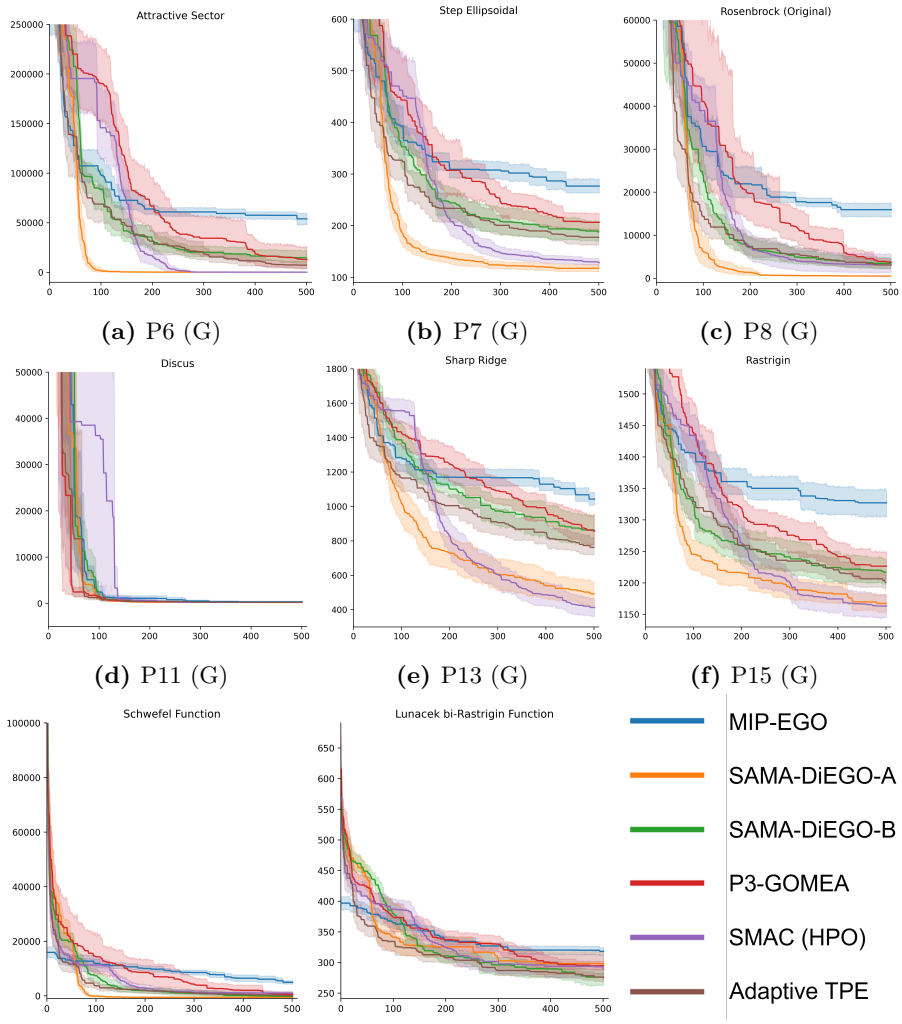
forms the best on the moderately ill-conditioned Schaffers F7 problem (P18 in Table 4.4.2). In addition to the tabular result, the convergence plots of the benchmarked algorithms are displayed in Figs. 4.4.2, 4.4.4 and 4.4.5. From the figures, it can be observed that SAMA-DiEGO-A shows advantage in solving four unimodal problems (P6, P7, P8, P11), followed by SMAC which overtakes SAMA-DiEGO-A in latter phase on P13. Comparing the plots of SAMA-DiEGO-A and SMAC, the former converges faster, whereas the latter shows strong capability of escaping from local optima (e.g. P6, P13 and P24). It is also noticeable that adaptive TPE and MIP-EGO always converge rapidly in the beginning phase (up to 50 function calls) on all problems among all the benchmarked algorithms. Another finding is that both MIP-EGO and SAMA-DiEGO-B are suffering from slow convergence speed in comparison to the other algorithms.

***Bad* problems** The results are presented in Table 4.4.2. SAMA-DiEGO-A obtains the minimum (best-performed) mean and median performance on problems 9 and 19. SAMA-DiEGO-B slightly beats other algorithms on Katsuura Function (P23) and marginally lost to SMAC on the two Gallagher's Gaussian problems (P21 and P22). Besides problems 21 and 22, SMAC also secures a lead on Weierstrass problem (P16) and is closely followed by SAMA-DiEGO-A. In contrast to the comparatively poor performance observed on G and M problems, MI-ES Trajectory of convergence on the problems of benchmarked algorithms are given in Figs. 4.4.3, 4.4.6 and 4.4.7. Similar to the plots obtained on *good* and *moderate* problems, Adaptive TPE and MIP-EGO still converges faster in the beginning in all cases but are surpassed by others in later phase. SMAC dominates the performance on the two Gallagher's Gaussian problems (P21 and P22) which is in agreement with the tabular results. Moreover, it can be observed from Fig. 4.4.6 that SAMA-DiEGO-A is also a competent solver for multi-modal problems with adequate global structure as defined by [46] (P16 and P19). The trajectories of SAMA-DiEGO-A and SAMA-DiEGO-B acquired on problems 21, 22 and 23 indicate that the former is suffering from multi-modal problems with numerous global optima, whereas the latter with expected improvements as infill criteria is a more suitable option here.

Generally speaking, if discussing the usage of SAMA-DiEGO in respect of structural property of ordinal problems as defined in [46]: SAMA-DiEGO-A is a competent solvers for uni-modal problems and multi-modal problems with adequate global structure, whereas SAMA-DiEGO-B appears to be more capable of handling multi-modal problems with weak global structure.

Table 4.4.1: Benchmark results on the discretized G and M (P24) BBOB problems (Section 4.4.1). The best performance for each test problem in terms of median and mean values are highlighted in blue.

Function	Algorithm	Algorithm Specification	Dimension = 15			
			Fitness Value			
			Best	Median	Mean	Std
P6 (G) Attractive Sector	MI-ES	N/A	172869.15	407604.29	428103.05	209158.25
	MIP-EGO	Random Forest	21227.12	61558.02	53880.90	13787.24
	SAMA-DiEGO-A	PV-ES	76.35	93.44	94.86	17.81
	SAMA-DiEGO-B	EI-ES	233.04	11822.81	14705.90	10733.29
	P3-GOMEA	N/A	148.02	3884.69	12891.50	18872.99
	SMAC4	HPO	72.05	99.15	99.16	16.20
	Adaptive TPE	N/A	387.80	6884.71	7191.29	5945.59
P7 (G) Step Ellipsoidal	MI-ES	N/A	361.10	682.51	765.20	314.34
	MIP-EGO	Random Forest	223.94	269.91	276.73	32.11
	SAMA-DiEGO-A	PV-ES	104.07	114.74	117.48	11.39
	SAMA-DiEGO-B	EI-ES	146.07	180.11	188.11	31.78
	P3-GOMEA	N/A	156.84	209.53	206.48	28.50
	SMAC4	HPO	111.35	126.16	128.40	11.47
	Adaptive TPE	N/A	139.31	164.78	177.11	25.68
P8 (G) Rosenbrock Original	MI-ES	N/A	43955.62	104023.45	111845.50	46027.14
	MIP-EGO	Random Forest	14296.34	17477.50	17779.66	2520.72
	SAMA-DiEGO-A	PV-ES	283.74	606.00	609.30	191.79
	SAMA-DiEGO-B	EI-ES	1073.39	5357.91	4928.82	2720.72
	P3-GOMEA	N/A	42008.57	59620.57	69886.26	25249.70
	SMAC4	HPO	565.94	2225.38	4038.31	4345.35
	Adaptive TPE	N/A	1302.25	4796.49	5614.59	2990.26
P11 (G) Discus	MI-ES	N/A	206.25	279.45	207713.44	452104.85
	MIP-EGO	Random Forest	166.11	233.11	337.58	276.67
	SAMA-DiEGO-A	PV-ES	152.51	179.92	185.90	19.26
	SAMA-DiEGO-B	EI-ES	168.88	215.24	217.75	30.85
	P3-GOMEA	N/A	160.71	205.91	207.61	27.70
	SMAC4	HPO	135.17	187.86	199.90	41.14
	Adaptive TPE	N/A	125.13	217.01	229.29	78.24
P13 (G) Sharp Ridge	MI-ES	N/A	1805.62	2177.03	2186.14	225.16
	MIP-EGO	Random Forest	943.29	1009.77	1042.65	87.45
	SAMA-DiEGO-A	PV-ES	268.56	514.22	492.19	122.41
	SAMA-DiEGO-B	EI-ES	625.11	827.61	863.90	146.71
	P3-GOMEA	SVR	656.29	856.28	855.72	145.99
	SMAC4	HPO	258.09	417.76	412.22	88.60
	Adaptive TPE	N/A	668.11	758.38	762.45	84.39
P15 (G) Rastrigin	MI-ES	N/A	1337.55	1544.87	1572.05	223.00
	MIP-EGO	Random Forest	1231.29	1341.62	1327.01	51.03
	SAMA-DiEGO-A	PV-ES	1123.21	1172.55	1167.24	23.65
	SAMA-DiEGO-B	EI-ES	1172.35	1207.25	1217.00	34.37
	P3-GOMEA	N/A	1177.40	1217.94	1226.36	36.73
	SMAC4	HPO	1098.96	1170.15	1163.20	30.73
	Adaptive TPE	N/A	1173.00	1198.76	1201.05	20.33
P20 (G) Schwefel	MI-ES	N/A	25393.04	72963.97	76338.17	33981.88
	MIP-EGO	Random Forest	1580.41	5264.21	5734.52	2624.57
	SAMA-DiEGO-A	PV-ES	-544.35	-543.55	-543.59	0.40
	SAMA-DiEGO-B	EI-ES	-535.31	-366.33	-193.45	416.77
	P3-GOMEA	N/A	-542.92	-201.39	564.72	1157.05
	SMAC4	HPO	-543.55	298.68	1000.94	1143.17
	Adaptive TPE	N/A	-514.11	-128.67	66.48	474.78
P24 (M) Lunacek bi-Rastrigin	MI-ES	N/A	358.86	437.38	456.27	71.83
	MIP-EGO	Random Forest	282.88	320.01	319.88	15.99
	SAMA-DiEGO-A	PV-ES	245.91	303.27	297.76	31.59
	SAMA-DiEGO-B	EI-ES	229.96	276.68	276.03	21.78
	P3-GOMEA	N/A	276.85	295.09	294.50	11.66
	SMAC4	HPO	246.68	279.30	288.88	33.66
	Adaptive TPE	N/A	253.99	280.45	276.14	10.94



(a) P6 (G) **(b)** P7 (G) **(c)** P8 (G)

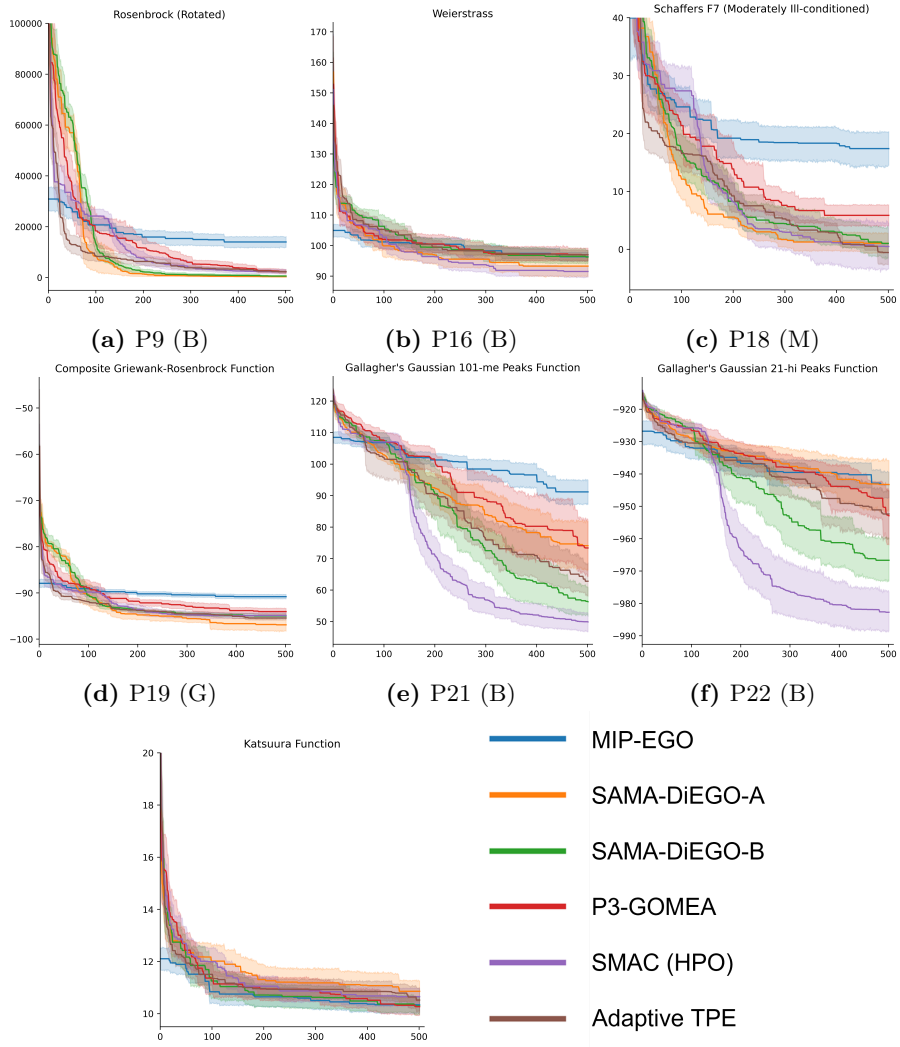
(d) P11 (G) **(e)** P13 (G) **(f)** P15 (G)

(g) P20 (G) **(h)** P24 (M) **(i)** Legend

Figure 4.4.2: Convergence plots of benchmarked algorithms on discretized G & M BBOB problems (see Section 4.4.1 and 4.4.2). The relatively poorly performed MI-ES is removed from the figures for nicer visualization. For each plot, the **horizontal axis** is the **number of function calls** to real objective function (fitness evaluation) to the objective problem, whereas the **vertical axis** records the iterative **fitness values**. The solid lines are the statistical means of **best-so-far** fitness across 11 independent runs in regard to number of function calls and the hues around the lines of means are the corresponding 95% confidence interval.

Table 4.4.2: Benchmark results on the discretized M (P18) and B BBOB problems (Section 4.4.1). The best performance for each test problem in terms of median and mean values are highlighted in blue.

Function	Algorithm	Algorithm Specification	Dimension = 15			
			Fitness Value			
			Best	Median	Mean	Std
P9 (B) Rosenbrock Rotated	MI-ES	N/A	44756.84	73232.62	79370.41	31158.36
	MIP-EGO	Random Forest	5123.26	14679.66	13981.47	5139.36
	SAMA-DiEGO-A	PV-ES	214.83	256.11	316.88	159.97
	SAMA-DiEGO-B	EI-ES	214.83	426.35	581.77	328.22
	P3-GOMEA	SVR	579.97	1547.67	2241.56	1597.13
	SMAC4	HPO	500.69	2478.33	2293.00	1502.34
	Adaptive TPE	N/A	1401.03	2181.28	2284.56	753.30
P16 (B) Weierstrass	MI-ES	N/A	84.24	100.79	96.79	7.81
	MIP-EGO	Random Forest	89.66	96.05	96.33	3.50
	SAMA-DiEGO-A	PV-ES	85.10	93.22	93.25	5.70
	SAMA-DiEGO-B	EI-ES	90.34	96.64	96.23	3.63
	P3-GOMEA	N/A	91.99	96.09	96.96	3.46
	SMAC4	HPO	85.14	92.29	91.49	3.10
	Adaptive TPE	N/A	87.63	98.08	96.99	3.23
P18 (M) Schaffers F7 (moderately ill-conditioned)	MI-ES	N/A	15.54	29.97	31.97	11.10
	MIP-EGO	Random Forest	0.80	17.16	17.39	7.09
	SAMA-DiEGO-A	PV-ES	-3.17	1.19	1.04	2.93
	SAMA-DiEGO-B	EI-ES	-4.82	0.44	1.02	4.57
	P3-GOMEA	N/A	0.65	6.62	5.86	3.06
	SMAC4	HPO	-9.13	0.31	0.48	6.99
	Adaptive TPE	N/A	-9.57	0.11	-0.56	3.12
P19 (B) Composite Griewank-Rosenbrock	MI-ES	N/A	-82.41	-70.90	-70.03	11.00
	MIP-EGO	Random Forest	-92.40	-90.98	-90.82	1.12
	SAMA-DiEGO-A	PV-ES	-102.30	-96.38	-96.91	2.11
	SAMA-DiEGO-B	EI-ES	-96.90	-94.76	-95.00	0.87
	P3-GOMEA	N/A	-96.13	-94.03	-94.05	1.20
	SMAC4	HPO	-97.05	-94.95	-94.90	1.03
	Adaptive TPE	N/A	-96.42	-95.68	-95.44	0.79
P21 (B) Gallagher's Gaussian 101-me Peaks	MI-ES	N/A	97.56	112.49	111.21	6.81
	MIP-EGO	Random Forest	75.47	91.13	91.16	9.26
	SAMA-DiEGO-A	PV-ES	49.80	71.38	74.09	12.93
	SAMA-DiEGO-B	EI-ES	46.81	51.16	56.28	7.76
	P3-GOMEA	N/A	51.45	72.57	73.36	15.09
	SMAC4	HPO	42.45	51.12	49.88	4.96
	Adaptive TPE	N/A	53.32	60.82	62.80	8.61
P22 (B) Gallagher's Gaussian 21-hi Peaks	MI-ES	N/A	-925.42	-917.58	-918.79	3.07
	MIP-EGO	Random Forest	-959.34	-937.17	-942.87	10.24
	SAMA-DiEGO-A	PV-ES	-963.68	-939.61	-943.27	13.19
	SAMA-DiEGO-B	EI-ES	-978.12	-971.07	-966.68	11.30
	P3-GOMEA	N/A	-980.78	-943.83	-952.91	14.21
	SMAC4	HPO	-997.29	-979.23	-982.75	10.64
	Adaptive TPE	N/A	-977.57	-948.56	-952.38	12.52
P23 (B) Katsuura	MI-ES	N/A	9.92	11.13	11.21	0.69
	MIP-EGO	Random Forest	8.99	10.65	10.34	0.75
	SAMA-DiEGO-A	PV-ES	9.85	10.81	10.86	0.68
	SAMA-DiEGO-B	EI-ES	9.64	10.16	10.30	0.60
	P3-GOMEA	N/A	9.03	10.30	10.27	0.53
	SMAC4	HPO	9.20	10.55	10.65	0.58
	Adaptive TPE	N/A	9.50	10.63	10.52	0.76

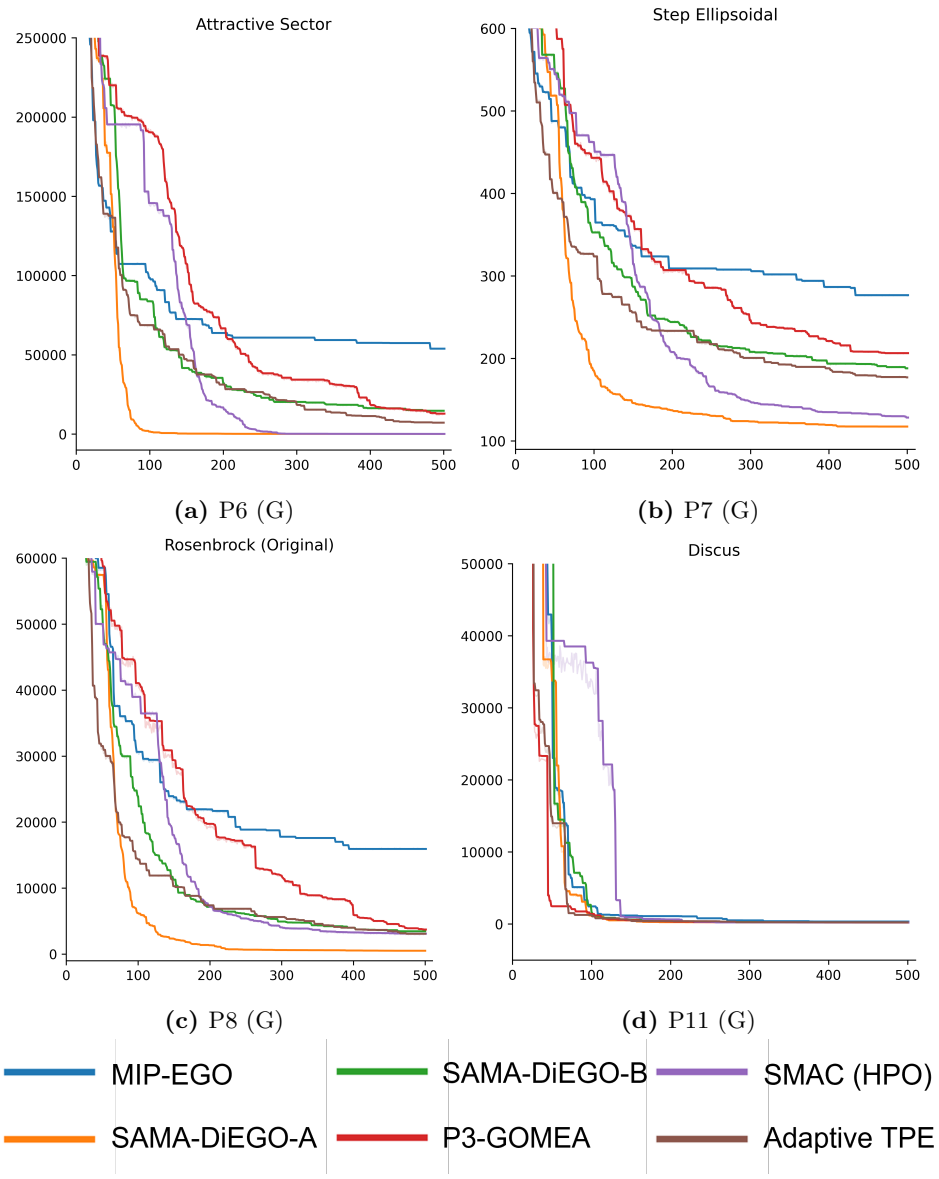


(a) P9 (B) **(b)** P16 (B) **(c)** P18 (M)

(d) P19 (G) **(e)** P21 (B) **(f)** P22 (B)

(g) P23 (B) **(h)** Legend

Figure 4.4.3: Convergence plots of benchmarked algorithms on discretized M (P18) and B BBOB problems (see Section 4.4.1 and 4.4.2). The relatively poorly performed MI-ES is removed from the figures for nicer visualization. For each plot, the **horizontal axis** is the **number of function calls** to real objective function (fitness evaluation) to the objective problem, whereas the **vertical axis** records the iterative **fitness values**. The solid lines are the **means** of **best-so-far** fitness across 11 independent runs in regard to number of function calls and the hues around the lines of means are the corresponding 95% confidence interval.



(a) P6 (G)

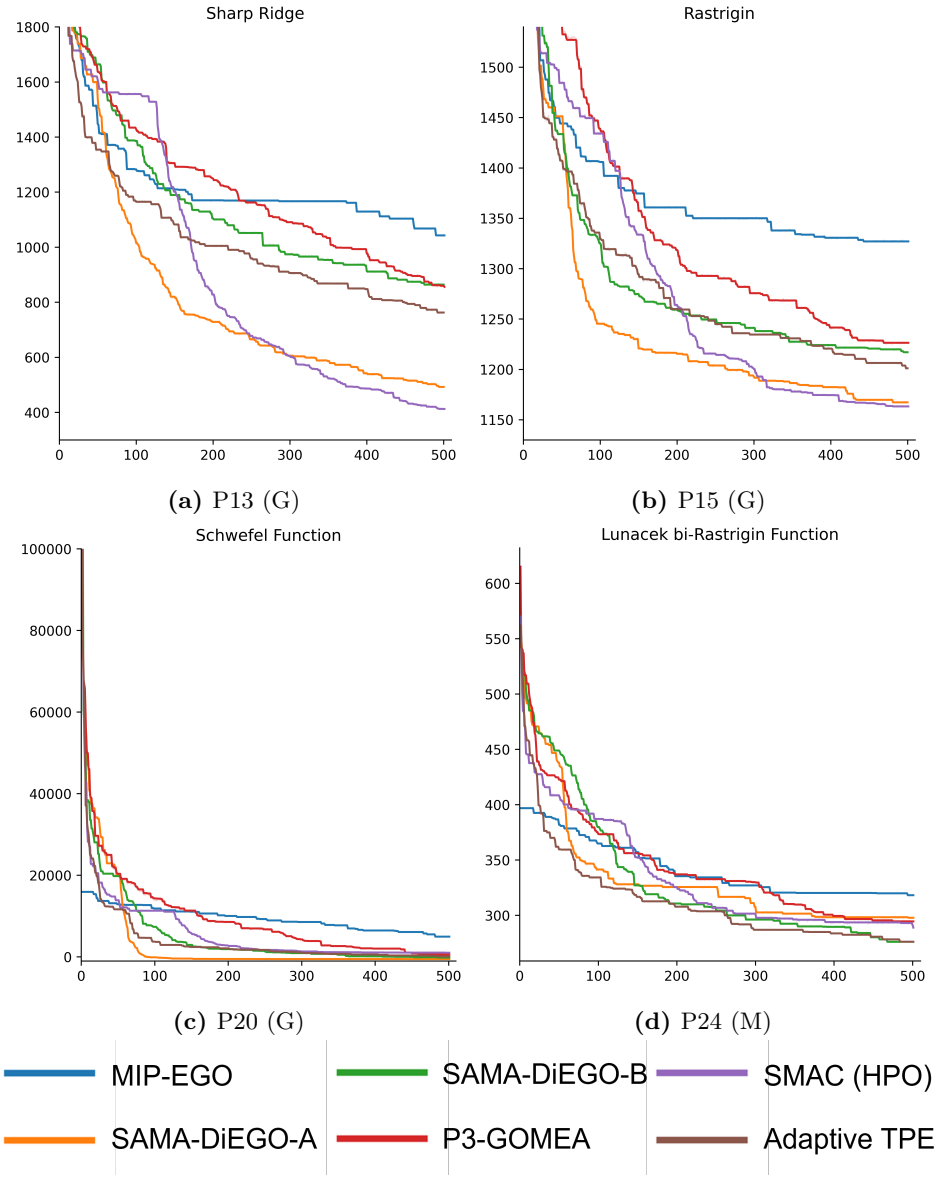
(b) P7 (G)

(c) P8 (G)

(d) P11 (G)

(e) Legend

Figure 4.4.4: Convergence plots of benchmarked algorithms on discretized BBOB problems 6,7,8 and 11 (see Section 4.4.2). The relatively poorly performed MI-ES is removed from the figures for nicer visualization. For each plot, the **horizontal axis** is the **number of function calls** (fitness evaluation) to the problem, whereas the **vertical axis** records the iterative **fitness values**. The solid lines are the **means** of **best-so-far** fitness across 11 independent runs in regard to number of function calls.



(a) P13 (G)

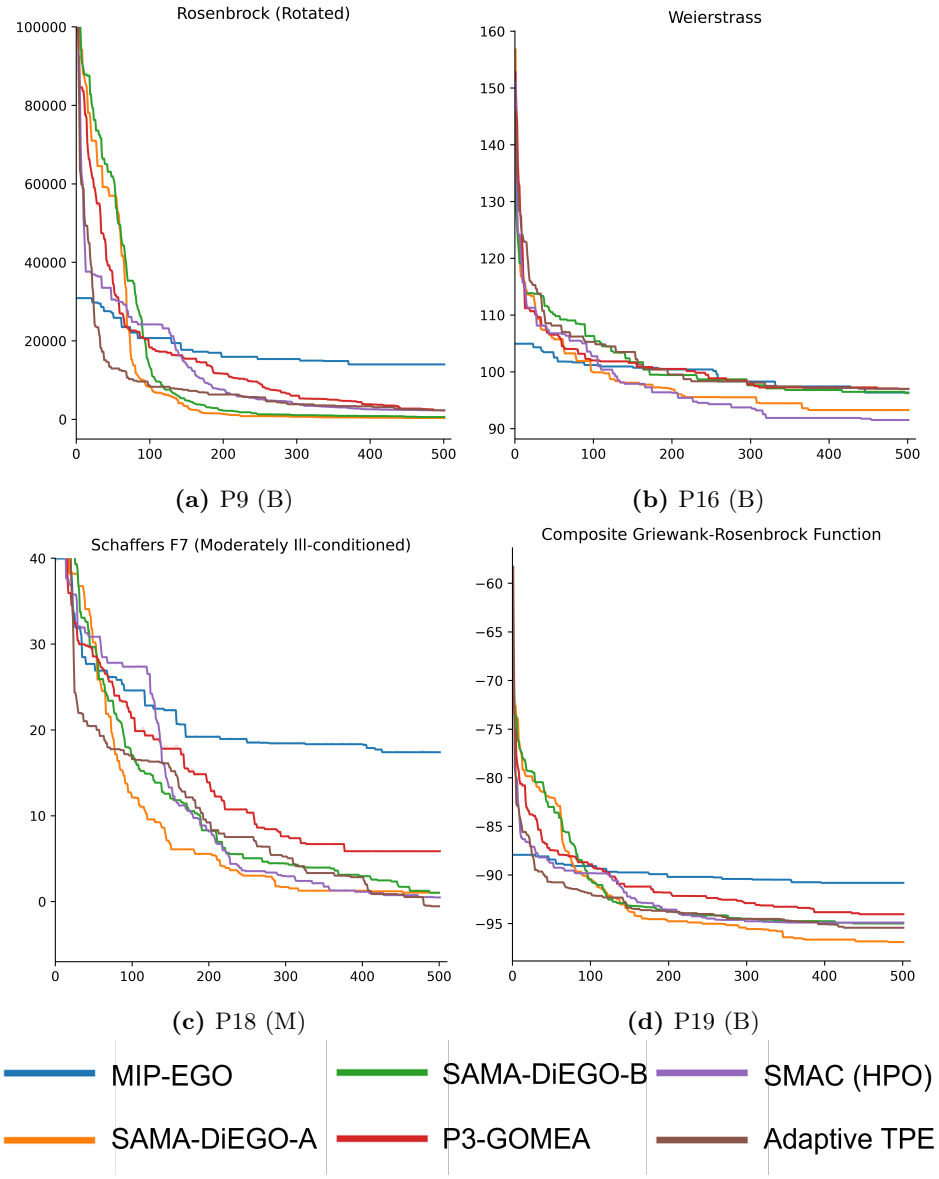
(b) P15 (G)

(c) P20 (G)

(d) P24 (M)

(e) Legend

Figure 4.4.5: Convergence plots of benchmarked algorithms on discretized BBOB problems 13, 15, 20 and 24 (see Section 4.4.2). The relatively poorly performed MI-ES is removed from the figures for nicer visualization. For each plot, the **horizontal axis** is the **number of function calls** to real objective function (fitness evaluation) to the problem, whereas the **vertical axis** records the iterative **fitness values**. The solid lines are the **means** of **best-so-far** fitness across 11 independent runs in regard to number of function calls.



(a) P9 (B)

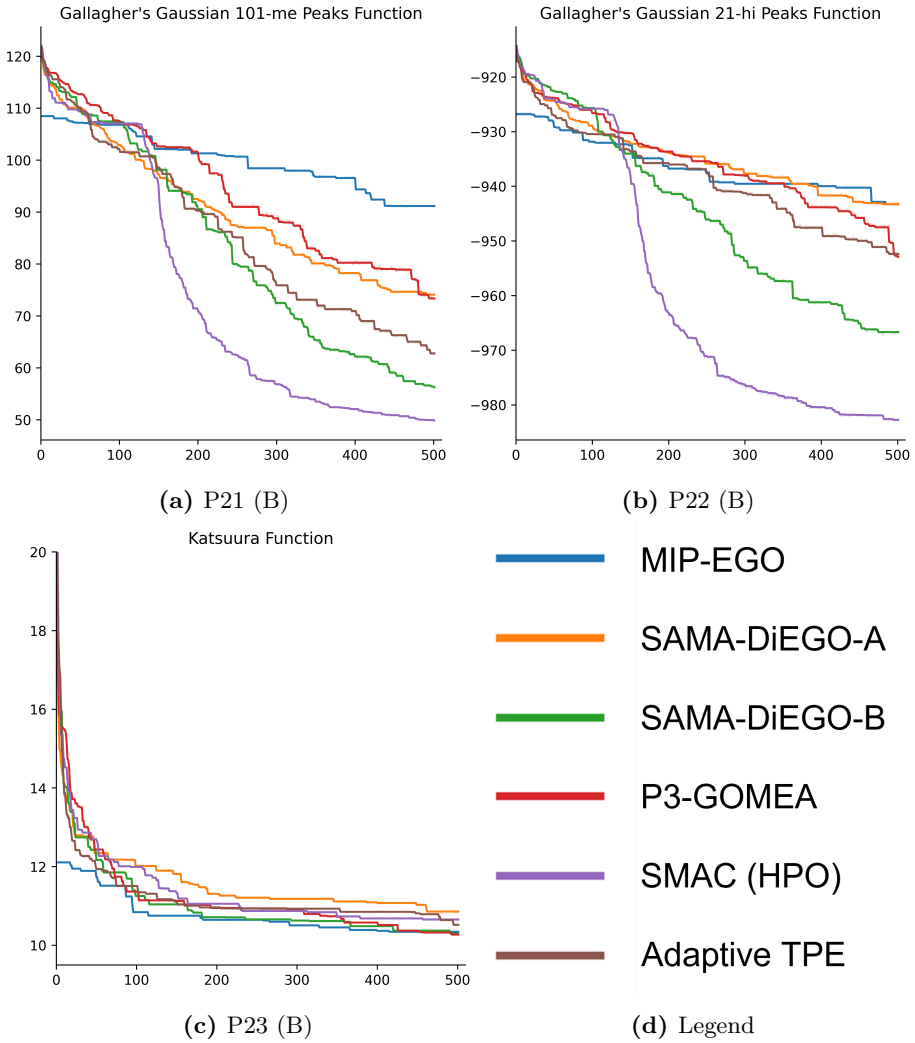
(b) P16 (B)

(c) P18 (M)

(d) P19 (B)

(e) Legend

Figure 4.4.6: Convergence plots of benchmarked algorithms on discretized BBOB problems 9, 16, 18 and 19 (see Section 4.4.2). The relatively poorly performed MI-ES is removed from the figures for nicer visualization. For each plot, the **horizontal axis** is the **number of function calls** (fitness evaluation) to the problem, whereas the **vertical axis** records the iterative **fitness values**. The solid lines are the **means** of **best-so-far** fitness across 11 independent runs in regard to number of function calls.



(a) P21 (B) **(b)** P22 (B)

(c) P23 (B) **(d)** Legend

Figure 4.4.7: Convergence plots of benchmarked algorithms on discretized BBOB problems 21, 22 and 23 (see Section 4.4.2). The relatively poorly performed MI-ES is removed from the figures for nicer visualization. For each plot, the **horizontal axis** is the **number of function calls** to real objective function (fitness evaluation) to the problem, whereas the **vertical axis** records the iterative **fitness values**. The solid lines are the **means** of **best-so-far** fitness across 11 independent runs in regard to number of function calls.

5

Conclusions

5.1 Summary & Discussion

In this thesis, we firstly study the basic concept of surrogate assisted/based optimization and review existing works on strategies for manipulating multiple surrogate models as well as applying surrogate model on discrete problems. Secondly, a new EGO algorithm that utilizes multiple surrogate models, namely Self-Adaptive Multiple-surrogate Assisted Efficient Global Optimization (SAMA-DiEGO) algorithm is proposed. The SAMA-DiEGO strictly follows the iterative optimization process of common EGO algorithms 1. In addition to existing EGO algorithms that can adapt to discrete problems, it features an online model selection mechanism which, in each iteration, chooses the most suitable surrogate from 31 candidates to serve as the proxy to objective function. Lastly, in experimental study, we benchmark SAMA-DiEGO against several robust non-surrogate solvers and single-surrogate assisted solvers on thirty-three binary-encoded combinatorial problems and fifteen ordinal-encoded problems. Meanwhile, we also study the performance of SAMA-DiEGO using probabilistic-based infill criteria (e.g. expected improvement) against that of exclusively using prediction value of the promising surrogate.

With respect to experimental results, SAMA-DiEGO outperforms non-surrogate solvers (evolutionary algorithms) and a single-surrogate assisted solver MIP-EGO in 28 out of 33 binary-encoded cases and achieves equivalent performance in the other five cases. This shows that given a fixed budget (number of calls to the real objective function), in comparison to not use and

only use one surrogate, well-guided usage of multiple surrogate models in EGO yields better solutions to the binary-encoded problem. Further observations obtained on handling another complex test suite, the discretized BBOB problems, show that SAMA-DiEGO performs better than non-and single-surrogate-assisted optimization algorithms on uni-modal problems, but gets slightly beaten by SMAC on 4 of 9 multi-modal problems. This suggests that the proposed model selection mechanism in SAMA-DiEGO is capable of utilizing multiple surrogates to rapidly optimize problems with clear global structures (i.e. with less local optima). Moreover, we also found that, in comparison with directly using prediction values, using expected improvement during acquisition process is beneficial if the target problem is multi-modal (i.e. with more local optima).

Lastly, in reference to the observed experimental results, we can try to answer the three research questions, namely Q1, Q2 and Q3, raised in section 1.1:

Q1 *In comparison to non-surrogate optimization algorithm, can the use of surrogate model help achieve better performance in solving discrete problem?*

Answer Yes. If compared with traditional non-surrogate assisted optimization algorithms under same budget, EGO-styled algorithms can locate better solution for discrete (combinatorial and ordinal) problems.

Q2 Given with a fixed budget, can EGO algorithms equipped with multiple surrogate model achieve better performance on discrete problems if compared with robust single surrogate model based EGO?

Answer Yes, to a large extent. Utilizing multiple surrogate models yield better performance on solving unimodal discrete problems. This advantage becomes obvious if some of the candidate surrogates approximate the target problem with acceptable fidelity levels (e.g. positive r^2 scores). However, if all candidate surrogates tend to interpolate the target with low fidelity, it is not exactly beneficial to prefer multi-surrogates to single-surrogate EGO.

Q3 When optimizing discrete problems, can we trust the prediction value of surrogate models in EGO if multiple surrogates are used?

Answer Yes, to a certain extent. Provided with a fixed budget, directly using prediction value of multi-surrogate is a wise choice in terms of finding better solutions when optimizing problems with clear global structure. However, although multiple surrogates can differently approximate a target problem from various perspectives, probabilistic-based infill criteria is still needed to allow the algorithms exploring problems with multiple local structures.

5.2 Future Work

The online model management strategy applied in SAMA-DiEGO is based on the idea of selecting the best out of a group. In view of the experimental results showed in Chapter 4, this selection-based strategy is indeed a promising way for concurrently managing and utilizing a dozens of surrogate models to optimize discrete problems. However, we can critically say that it has a waste of information since, in each iteration, only the selected model is used whereas the other trained models are set aside. A solution, as briefly mentioned in section 2.1, is doing ensemble of models (e.g. compute a weighted sum of predictions of surrogates [6, 35, 61, 67, 93]). Although simultaneously computing an ensemble of dozens of surrogate models is not wise as it requires tremendous computational resources, it would still be potentially feasible to ensemble promising models after the selection procedure. Besides managing surrogate models, efficiently paralleling the acquisition process is another important but not thoroughly discussed component of SAMA-DiEGO. The reason why SAMA-DiEGO is not equipped with existing robust methods (e.g. q-EI [38, 39, 89]) but uses a compromised strategy is that these methods exclusively focus on probabilistic-based infill criteria. However, before our study, there is a lack of investigations on the necessity of using probabilistic-based infill criteria for discrete problems, especially using with multi-surrogate assisted/based algorithms. Now, as we have evidently find that probabilistic-based infill criteria is still critically needed in handling some multi-modal problems, it is worthwhile to bring the multi-point infill criteria into discussion for better performance of SAMA-DiEGO. A further promising potential study would be an online strategy to determine the timing of using probabilistic-based infill criteria.

Lastly, given the fact that all of the implemented surrogate model in SAMA-DiEGO are essentially support both continuous and discrete variables, we can anticipate that the algorithm will also work for expensive mixed-integer problems. Given the fact that existing robust and state-of-the-art EGO/BO solvers for mixed-integer problems (e.g. hyperparameter tuning) are built on single surrogate, it would be a rewarding attempt to further investigate the benefit of using multi-surrogate assisted/based EGO on mixed-integer cases.

6

Appendix A

In this chapter, additional formulas, figures and pseduo-codes of this thesis will be presented.

6.1 Additional Formulas

6.1.1 Correlation Functions of Kriging

To neatly and conventionally write the definition, it is assumed that x^i and x^j is the ith and jth vectorized data sample, respectively. Moreover, θs are the parameters need to be estimated. More details can be found in [13].

Ornstein–Uhlenbeck process

$$C\left(x^{(i)}, x^{(j)}\right) = \prod_{l=1}^{nx} \exp\left(-\theta_l \left|x_l^{(i)} - x_l^{(j)}\right|\right), \quad \forall \theta_l \in \mathbb{R}^+$$

Squared Gaussian Correlation

$$C\left(x^{(i)}, x^{(j)}\right) = \prod_{l=1}^{nx} \exp\left(-\theta_l \left(x_l^{(i)} - x_l^{(j)}\right)^2\right), \quad \forall \theta_l \in \mathbb{R}^+$$

Gower Distance Gower distance is a strategy specifically proposed to handle categorical variables, its definitions (rules) are explicitly explained in [41].

Matérn Correlation (3/2)

$$C\left(x^{(i)}, x^{(j)}\right) = \prod_{l=1}^{nx} \left(1 + \sqrt{3}\theta_l \left|x_l^{(i)} - x_l^{(j)}\right|\right) \exp\left(-\sqrt{3}\theta_l \left|x_l^{(i)} - x_l^{(j)}\right|\right), \quad \forall \theta_l \in \mathbb{R}^+$$

Matérn Correlation (5/2)

$$C\left(x^{(i)}, x^{(j)}\right) = \prod_{l=1}^{nx} \left(1 + \sqrt{5}\theta_l \left|x_l^{(i)} - x_l^{(j)}\right| + \frac{5}{3}\theta_l^2 \left(x_l^{(i)} - x_l^{(j)}\right)^2\right) \times \exp\left(-\sqrt{5}\theta_l \left|x_l^{(i)} - x_l^{(j)}\right|\right), \quad \forall \theta_l \in \mathbb{R}^+$$

6.1.2 Five Hard PBO Problems

LABS $F_{\text{LABS}}(\vec{x}) = \frac{n^2}{2\sum_{k=1}^{n-1}\left(\sum_{i=1}^{n-k} x'_i \cdot x'_{i+k}\right)^2}$ where $x'_i = 2x_i - 1$, where x_i is the ith element of a n-sized binary sequence.

Ising1D To formulate the Ising problems, [27] defined an undirected graph $G = (V, E)$, where $V = [n]$. Furthermore, an affine transformation $\{-1, +1\}^n \to \{0, 1\}^n$ is applied on search space to change the n spins to binary decision variable. Hence, the final objective function is formulated as

$$F_{\text{Ising}}(\vec{x}) = \sum_{\{u,v\} \in E} \left[x_u x_v - (1 - x_u)(1 - x_v)\right]$$

Given with the objective function, the Ising1D problem is defined over a one-dimensional lattice graph G_{Ising1D}:

$$e_{ij} = 1 \Leftrightarrow j = i + 1 \quad \forall i \in \{1, \ldots, n-1\} \vee j = n, i = 1$$

Ising2D The Ising2D (the torus) problem shares the same objective function with Ising1D but is defined on a two-dimensional lattice graph G_{Ising2D}:

$$\begin{aligned} e_{(i,j)(k,\ell)} = 1 \Leftrightarrow & [k = (i+1) \bmod N \ \wedge \ \ell = j \quad \forall i, j \in \{0, \ldots, N-1\}] \\ \vee \ & [k = (i-1) \bmod N \wedge \ell = j \quad \forall i, j \in \{0, \ldots, N-1\}] \\ \vee \ & [\ell = (j+1) \bmod N \wedge k = i \quad \forall j, i \in \{0, \ldots, N-1\}] \\ \vee \ & [\ell = (j-1) \bmod N \wedge k = i \quad \forall j, i \in \{0, \ldots, N-1\}] \end{aligned}$$

MIVS The maximum independent vertex set problem is defined on a graph $G = ([n], E)$. The objective is to find the largest independent vertex

set (no adjacent vertices in the set) $V' = \{i \in [n] \mid x_i = 1\}$ in the G. [27] formulates this maximization problem as:

$$F_{\text{MIVS}}(x) = \sum_i x_i - n \cdot \sum_{i,j} x_i x_j e_{i,j},$$

where $e_{i,j} = 1$ if $\{i, j\} \in E$ and $e_{i,j} = 0$ otherwise. The binary-encoded version of $e_{i,j}$ is specifically defined as:

$$\begin{aligned} e_{ij} = 1 \Leftrightarrow j = i + 1 \quad & \forall i \in \{1, \ldots, n-1\} - \{n/2\} \\ \vee\, j = i + n/2 + 1 \quad & \forall i \in \{1, \ldots, n/2 - 1\} \\ \vee\, j = i + n/2 - 1 \quad & \forall i \in \{2, \ldots, n/2\} \end{aligned}$$

NQP The N-queens problem asks to put N queens on a N × N chessboard under the constraint that no queens can attack others. The binary-encoded unconstrained version of this problem is formulated as below:

$$F_{\text{NOP}}(\vec{x}) = \sum_{i=1}^{N} \sum_{j=1}^{N} x_{ij} - N \cdot F_{\text{penalty}},$$

where $x_{ij} = 1$ if a queen is placed in the $\{i, j\}$ cell and $x_{ij} = 0$ otherwise. The penalty for infeasible solution F_{penalty} is defined as,

$$\begin{aligned} F_{\text{penalty}} = & \sum_{i=1}^{N} \max\left\{0, -1 + \sum_{j=1}^{N} x_{ij}\right\} \\ & + \sum_{j=1}^{N} \max\left\{0, -1 + \sum_{i=1}^{N} x_{ij}\right\} \\ & + \sum_{k=-N+2}^{N-2} \max\left\{0, -1 + \sum_{j \in i-k} x_{ij}\right\} \\ & + \sum_{\ell=3}^{2N-1} \max\left\{0, -1 + \sum_{\substack{j+i-\ell \\ i,j \in (1,2,\ldots,N]}} x_{ij}\right\} \end{aligned}$$

6.1.3 Composite PBO Problems

The three basis functions:

OneMax :

$$f : \{0,1\}^n \to [0..n], x \mapsto \sum_{i=1}^{n} x_i$$

LeadingOnes :

$$f : \{0,1\}^n \to [0..n], x \mapsto \max\{i \in [0..n] \mid \forall j \leq i : x_j = 1\} = \sum_{i=1}^{n} \prod_{j=1}^{i} x_i$$

A Linear Function with Harmonic Weights :

$$f : \{0,1\}^n \to \mathbb{R}, x \mapsto \sum_i i x_i$$

6.2 Additional Tables & Figures

Table 6.2.1: R^2 scores obtained by 31 surrogate models on discretized BBOB problems. Each score is the average of 15 scores gathered from three repetitions of a 5-fold cross-validation as described in section 4.4.1. The full names of abbreviations in *Specification* column can be found in section 3.5.

Model Family	ID	Specification	P6	P7	P8	P9	P10	P11	P12	P13	P14
Kriging	1	Constant + SGC	0.866	0.902	0.765	-0.071	0.899	0.922	0.294	0.919	0.532
	2	Linear + SGC	0.840	0.902	0.799	-0.092	0.897	0.925	0.218	0.912	0.520
	3	Constant + Matern52	0.867	0.900	0.790	-0.104	0.896	0.929	0.169	0.915	0.478
	4	Constant + Matern32	0.866	0.887	0.829	-0.100	0.894	0.942	0.165	0.907	0.495
	5	Linear + Matern52	0.815	0.897	0.814	-0.042	0.888	0.932	0.161	0.904	0.403
	6	Linear + OUP	0.812	0.699	0.894	-0.048	0.733	0.941	0.249	0.829	0.421
	7	Quadratic + OUP	0.777	0.868	0.736	-1.560	0.880	0.900	-0.417	0.894	0.007
	8	Constant + OUP	0.826	0.714	0.897	-0.069	0.743	0.942	0.308	0.831	0.420
	9	Linear + Gower	0.801	0.669	0.892	-0.427	0.705	0.937	0.147	0.822	0.247
	10	Linear + Matern32	0.818	0.885	0.817	-0.062	0.891	0.939	0.139	0.891	0.403
	11	Constant + Gower	0.815	0.681	0.895	-0.357	0.712	0.939	0.196	0.826	0.256
	12	Quadratic + Gower	0.752	0.860	0.720	-1.903	0.860	0.882	-1.041	0.867	-0.284
	13	Quadratic + Matern52	0.632	0.789	0.349	-3.092	0.804	0.801	-1.207	0.817	-0.708
	14	Quadratic + Matern32	0.647	0.799	0.473	-2.213	0.838	0.839	-0.958	0.860	-0.361
	15	Quadratic + SGC	0.611	0.782	0.317	-3.114	0.797	0.822	-1.644	0.805	-0.759
RBF Interpolation	1	Thin Plate Spline	0.881	0.863	0.714	-0.050	0.846	0.904	0.389	0.908	0.594
	2	Multiquadric	0.863	0.767	0.699	0.037	0.731	0.857	0.406	0.864	0.544
	3	Linear	0.863	0.767	0.699	0.037	0.731	0.857	0.406	0.864	0.544
	4	Cubic	0.866	0.881	0.624	-0.472	0.895	0.921	0.249	0.867	0.511
	5	Polyharmonic spline 4	0.815	0.893	0.565	-0.916	0.895	0.894	-0.206	0.904	0.246
	6	Invmultiquadric	0.808	0.567	0.669	0.023	0.491	0.758	0.289	0.778	0.358
	7	Invquadric	0.807	0.567	0.669	0.023	0.491	0.758	0.288	0.778	0.358
	8	Gaussian function	0.807	0.567	0.669	0.023	0.491	0.758	0.288	0.778	0.358
	9	Polyharmonic spline 5	-0.391	-51.558	-8.708	-14.586	-5.401	-1367661.901	-187.263	-6.520	-7.458
SVM Regression	1	poly 2	0.811	0.765	0.671	-0.084	0.666	0.720	0.407	0.804	0.495
	2	linear	0.797	0.556	0.666	0.019	0.510	0.745	0.327	0.774	0.396
	3	poly 2	0.739	0.696	0.653	-0.488	0.503	0.526	0.050	0.706	0.177
	4	rbf	0.726	0.642	0.630	0.069	0.574	0.640	0.321	0.755	0.436
	5	poly 5	0.392	0.298	0.499	-0.797	-0.539	-0.305	-0.116	0.234	-0.611
	6	sigmoid	-0.116	-0.665	-1.020	-0.215	-0.877	-0.933	-0.421	-0.247	-0.446
Random Forest Regression		100 decision trees	0.611	0.441	0.764	-0.035	0.354	0.672	0.106	0.549	0.298

Model Family	ID	Specification	P15	P16	P17	P18	P19	P20	P21	P22	P23	P24
Kriging	1	Constant + SGC	0.846	-0.170	0.684	0.672	-0.148	0.900	-0.047	-0.141	-0.168	0.290
	2	Linear + SGC	0.852	-0.169	0.683	0.691	-0.136	0.912	-0.026	-0.105	-0.215	0.141
	3	Constant + Matern52	0.846	-0.120	0.653	0.664	-0.092	0.913	-0.128	-0.225	-0.110	0.080
	4	Constant + Matern32	0.844	-0.099	0.622	0.682	-0.069	0.928	-0.097	-0.205	-0.118	0.161
	5	Linear + Matern52	0.845	-0.179	0.585	0.643	-0.042	0.891	-0.125	-0.021	-0.209	0.045
	6	Linear + OUP	0.844	-0.167	0.639	0.679	-0.103	0.937	0.038	0.022	-0.180	0.036
	7	Quadratic + OUP	0.758	-2.440	0.126	0.393	-1.821	0.845	-1.494	-1.527	-2.619	-0.079
	8	Constant + OUP	0.836	-0.089	0.632	0.684	-0.074	0.939	0.003	-0.035	-0.117	0.106
	9	Linear + Gower	0.822	-0.723	0.527	0.636	-0.414	0.938	-0.377	-0.393	-0.728	-0.143
	10	Linear + Matern32	0.847	-0.169	0.614	0.677	-0.090	0.928	-0.063	0.015	-0.199	0.010
	11	Constant + Gower	0.823	-0.669	0.543	0.648	-0.376	0.939	-0.334	-0.379	-0.666	-0.070
	12	Quadratic + Gower	0.721	-3.351	0.005	0.252	-2.514	0.858	-1.919	-1.985	-3.215	-0.347
	13	Quadratic + Matern52	0.647	-4.310	-0.289	0.062	-3.474	0.747	-3.007	-2.895	-5.388	-1.269
	14	Quadratic + Matern32	0.604	-3.276	-0.220	0.203	-2.698	0.780	-2.131	-2.396	-4.103	-0.782
	15	Quadratic + SGC	0.611	-4.069	-0.258	0.136	-3.263	0.756	-3.243	-3.209	-4.585	-0.748
RBF Interpolation	1	Thin Plate Spline	0.844	-0.324	0.619	0.695	-0.168	0.845	0.012	0.043	-0.393	0.392
	2	Multiquadric	0.799	-0.141	0.587	0.646	-0.047	0.794	0.100	0.144	-0.189	0.314
	3	Linear	0.799	-0.141	0.587	0.646	-0.046	0.794	0.100	0.144	-0.189	0.314
	4	Cubic	0.833	-0.683	0.544	0.685	-0.731	0.833	-0.249	-0.367	-0.810	-0.022
	5	Polyharmonic spline 4	0.801	-1.668	0.363	0.529	-1.367	0.867	-0.838	-0.988	-1.823	0.005
	6	Invmultiquadric	0.702	-0.076	0.441	0.485	-0.035	0.711	0.054	0.120	-0.108	0.089
	7	Invquadric	0.702	-0.076	0.441	0.485	-0.035	0.711	0.053	0.120	-0.108	0.089
	8	Gaussian function	0.702	-0.076	0.441	0.485	-0.035	0.711	0.053	0.120	-0.108	0.089
	9	Polyharmonic spline 5	-1.898	-146231.756	-19.007	-3.038	-169.010	-5.618	-357.175	-1057.016	-26.085	-116.984
SVM Regression	1	poly 2	0.766	-0.242	0.452	0.527	-0.099	0.726	0.069	0.151	-0.317	0.345
	2	linear	0.707	-0.074	0.472	0.501	-0.016	0.699	0.056	0.166	-0.137	0.054
	3	poly 2	0.666	-0.752	0.094	0.211	-0.554	0.606	-0.289	-0.372	-0.823	0.318
	4	rbf	0.652	-0.077	0.416	0.496	0.016	0.683	0.139	0.157	-0.097	0.260
	5	poly 5	0.222	-0.859	-1.182	-0.989	-1.551	-0.051	-1.016	-1.207	-1.236	0.118
	6	sigmoid	-0.159	-0.154	-1.272	-1.198	-0.242	-0.642	-0.303	-0.523	-0.091	-0.075
Random Forest Regression		100 decision trees	0.482	-0.133	0.268	0.336	-0.035	0.724	-0.021	-0.056	-0.092	0.055

Table 6.2.2: Default hyper-parameters of MI-ES used in SAMA-DiEGO. Detailed introduction of these parameters is presented in [60].

Hyper-parameters	Values	Comments
μ	10	Population size
λ	70	Offspring size. The selection pressure is set to 7.
σ_r	$0.05 \times (\text{Rmax} - \text{Rmin})$	Default mutation step for a continuous variable R. $Rmax$ and $Rmin$ are the upper and lower bounds of R, respectively.
η_i	$0.05 \times (\text{Imax} - \text{Imin})$	Default mutation step for an ordinal variable I. $Imax$ and $Imin$ are the upper and lower bounds of I, respectively.
p	$\frac{1}{n_d}$	Default mutation step for all nominal variables in the input. n_d is the number of nominal variables in a solution.
τ_σ	$\frac{1}{\sqrt{2n_r}}$	Global learning rate for σ. n_r is the number of continuous variables in a solution.
$\tau_\sigma^{'}$	$\frac{1}{\sqrt{2\sqrt{n_r}}}$	Local learning rate for σ. n_r is the number of continuous variables in a solution
τ_η	$\frac{1}{\sqrt{2n_i}}$	Global learning rate for η. n_i is the number of ordinal variables in a solution.
$\tau_\eta^{'}$	$\frac{1}{\sqrt{2\sqrt{n_i}}}$	Local learning rate for η. n_i is the number of ordinal variables in a solution.
τ_p	$\frac{1}{\sqrt{2n_d}}$	Global learning rate for p. n_d is the number of nominal variables in a solution.
$\tau_p^{'}$	$\frac{1}{\sqrt{2\sqrt{n_d}}}$	Local learning rate for p. n_d is the number of nominal variables in a solution.

Bibliography

[1] Reza Alizadeh, Janet K Allen, and Farrokh Mistree. Managing computational complexity using surrogate models: a critical review. *Research in Engineering Design*, 31:275–298, 2020. ISSN 1435-6066. doi: 10.1007/s00163-020-00336-7. URL https://doi.org/10.1007/s00163-020-00336-7.

[2] M. J. Asher, B. F. W. Croke, A. J. Jakeman, and L. J. M. Peeters. A review of surrogate models and their application to groundwater modeling. *Water Resources Research*, 51:5957–5973, 8 2015. ISSN 0043-1397. doi: 10.1002/2015WR016967. URL https://doi.org/10.1002/2015WR016967. https://doi.org/10.1002/2015WR016967.

[3] Thomas Bäck. *Evolutionary Algorithms in Theory and Practice: Evolution Strategies, Evolutionary Programming, Genetic Algorithms*. Oxford University Press, Inc., USA, 1996. ISBN 0195099710.

[4] Samineh Bagheri, Wolfgang Konen, and Thomas Bäck. Online selection of surrogate models for constrained black-box optimization. In *2016 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8, 2016. doi: 10.1109/SSCI.2016.7850206. URL https://doi.org/10.1109/SSCI.2016.7850206.

[5] Samineh Bagheri, Wolfgang Konen, Michael Emmerich, and Thomas Bäck. Self-adjusting parameter control for surrogate-assisted constrained optimization under limited budgets. *Applied Soft Computing*, 61:377–393, 12 2017. ISSN 1568-4946. doi: 10.1016/J.ASOC.2017.07.060. URL https://doi.org/10.1016/J.ASOC.2017.07.060.

[6] Thomas Bartz-Beielstein and Martin Zaefferer. Model-based methods for continuous and discrete global optimization. *Applied Soft Computing*, 55:154–167, 6 2017. ISSN 1568-4946. doi: 10.1016/J.ASOC.2017.01.039. URL https://doi.org/10.1016/J.ASOC.2017.01.039.

[7] Paul Beaucaire, Charlotte Beauthier, and Caroline Sainvitu. Multi-point infill sampling strategies exploiting multiple surrogate models. In *Proceedings of the Genetic and Evolutionary Computation Conference*

Companion, pages 1559–1567. Association for Computing Machinery, 2019. ISBN 9781450367486. doi: 10.1145/3319619.3328527. URL https://doi.org/10.1145/3319619.3328527.

[8] James Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. Algorithms for hyper-parameter optimization. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS'11, page 2546–2554, Red Hook, NY, USA, 2011. Curran Associates Inc. ISBN 9781618395993. URL https://proceedings.neurips.cc/paper/2011/file/86e8f7ab32cfd12577bc2619bc635690-Paper.pdf.

[9] James Bergstra, Daniel Yamins, and David Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In Sanjoy Dasgupta and David McAllester, editors, *Proceedings of the 30th International Conference on Machine Learning*, volume 28, pages 115–123. PMLR, 2 2013. URL https://proceedings.mlr.press/v28/bergstra13.html.

[10] Atharv Bhosekar and Marianthi Ierapetritou. Advances in surrogate based modeling, feasibility analysis, and optimization: A review. *Computers & Chemical Engineering*, 108:250–267, 1 2018. ISSN 0098-1354. doi: 10.1016/J.COMPCHEMENG.2017.09.017. URL https://doi.org/10.1016/J.COMPCHEMENG.2017.09.017.

[11] Jakob Bossek, Carola Doerr, and Pascal Kerschke. Initial design strategies and their effects on sequential model-based optimization: An exploratory case study based on bbob. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, GECCO '20, page 778–786, New York, NY, USA, 2020. Association for Computing Machinery. ISBN 9781450371285. doi: 10.1145/3377930.3390155. URL https://doi.org/10.1145/3377930.3390155.

[12] Mohamed Amine Bouhlel, Nathalie Bartoli, Abdelkader Otsmane, and Joseph Morlier. Improving kriging surrogates of high-dimensional design models by partial least squares dimension reduction. *Structural and Multidisciplinary Optimization*, 53:935–952, 2016. ISSN 1615-1488. doi: 10.1007/s00158-015-1395-9. URL https://doi.org/10.1007/s00158-015-1395-9.

[13] Mohamed Amine Bouhlel, John T. Hwang, Nathalie Bartoli, Rémi Lafage, Joseph Morlier, and Joaquim R. R. A. Martins. A python surrogate modeling framework with derivatives. *Advances in Engineering Software*, 135:102662, 2019. ISSN 0965-9978. doi: https://doi.org/10.1016/j.advengsoft.2019.03.005. URL https://www.sciencedirect.com/science/article/pii/S0965997818309360.

[14] Leo Breiman. Random forests. *Machine Learning*, 45:5–32, 2001. ISSN 1573-0565. doi: 10.1023/A:1010933404324. URL https://doi.org/10.1023/A:1010933404324.

[15] Lars Buitinck, Gilles Louppe, Mathieu Blondel, Fabian Pedregosa, Andreas Mueller, Olivier Grisel, Vlad Niculae, Peter Prettenhofer, Alexandre Gramfort, Jaques Grobler, Robert Layton, Jake VanderPlas, Arnaud Joly, Brian Holt, and Gaël Varoquaux. API design for machine learning software: experiences from the scikit-learn project. In *ECML PKDD Workshop: Languages for Data Mining and Machine Learning*, pages 108–122, 2013.

[16] Thomas Bäck and Martin Schütz. Intelligent mutation rate control in canonical genetic algorithms. In Zbigniew W Raś and Maciek Michalewicz, editors, *Foundations of Intelligent Systems*, pages 158–167. Springer Berlin Heidelberg, 1996. ISBN 978-3-540-68440-4.

[17] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20:273–297, 1995. ISSN 1573-0565. doi: 10.1023/A:1022627411411. URL https://doi.org/10.1023/A:1022627411411.

[18] Ivo Couckuyt, Filip De Turck, Tom Dhaene, and Dirk Gorissen. Automatic surrogate model type selection during the optimization of expensive black-box problems. In *Proceedings of the Winter Simulation Conference*, WSC '11, page 4274–4284. Winter Simulation Conference, 2011.

[19] Siva Krishna Dasari, Abbas Cheddad, and Petter Andersson. Random forest surrogate models to support design space exploration in aerospace use-case. In John MacIntyre, Ilias Maglogiannis, Lazaros Iliadis, and Elias Pimenidis, editors, *Artificial Intelligence Applications and Innovations*, pages 532–544. Springer International Publishing, 2019. ISBN 978-3-030-19823-7.

[20] Eddie Davis and Marianthi Ierapetritou. A kriging based method for the solution of mixed-integer nonlinear programs containing black-box functions. *Journal of Global Optimization*, 43:191–205, 2009. ISSN 1573-2916. doi: 10.1007/s10898-007-9217-2. URL https://doi.org/10.1007/s10898-007-9217-2.

[21] P. F. de Aguiar, B. Bourguignon, M. S. Khots, D. L. Massart, and R. Phan-Than-Luu. D-optimal designs. *Chemometrics and Intelligent Laboratory Systems*, 30:199–210, 1995. ISSN 0169-7439. doi: https://doi.org/10.1016/0169-7439(94)00076-X. URL https://www.sciencedirect.com/science/article/pii/016974399400076X.

[22] Roy de Winter, Bas van Stein, Matthys Dijkman, and Thomas Bäck. Designing ships using constrained multi-objective efficient global optimization. In Giuseppe Nicosia, Panos Pardalos, Giovanni Giuffrida, Renato Umeton, and Vincenzo Sciacca, editors, *Machine Learning, Optimization, and Data Science*, pages 191–203. Springer International Publishing, 2019. ISBN 978-3-030-13709-0.

[23] Roy de Winter, Bas van Stein, and Thomas Bäck. Samo-cobra: A fast surrogate assisted constrained multi-objective optimization algorithm. In Hisao Ishibuchi, Qingfu Zhang, Ran Cheng, Ke Li, Hui Li, Handing Wang, and Aimin Zhou, editors, *Evolutionary Multi-Criterion Optimization*, pages 270–282. Springer International Publishing, 2021. ISBN 978-3-030-72062-9.

[24] Benjamin Doerr and Carola Doerr. Optimal static and self-adjusting parameter choices for the $(1 + (\lambda, \lambda))$ genetic algorithm. *Algorithmica*, 80:1658–1709, 2018. ISSN 1432-0541. doi: 10.1007/s00453-017-0354-9. URL `https://doi.org/10.1007/s00453-017-0354-9`.

[25] Benjamin Doerr, Christian Gießen, Carsten Witt, and Jing Yang. The $(1 + \lambda)$ evolutionary algorithm with self-adjusting mutation rate. *Algorithmica*, 81:593–631, 2019. ISSN 1432-0541. doi: 10.1007/s00453-018-0502-x. URL `https://doi.org/10.1007/s00453-018-0502-x`.

[26] Carola Doerr, Hao Wang, Furong Ye, Sander van Rijn, and Thomas Bäck. Iohprofiler: A benchmarking and profiling tool for iterative optimization heuristics, 2018. URL `https://arxiv.org/abs/1810.05281`.

[27] Carola Doerr, Furong Ye, Naama Horesh, Hao Wang, Ofer M. Shir, and Thomas Bäck. Benchmarking discrete optimization heuristics with iohprofiler. *Applied Soft Computing*, 88:106027, 3 2020. ISSN 1568-4946. doi: 10.1016/J.ASOC.2019.106027. URL `https://doi.org/10.1016/j.asoc.2019.106027`.

[28] Arkadiy Dushatskiy, Tanja Alderliesten, and Peter A. N. Bosman. A novel surrogate-assisted evolutionary algorithm applied to partition-based ensemble learning. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 583–591. Association for Computing Machinery, 2021. ISBN 9781450383509. URL `https://doi.org/10.1145/3449639.3459306`.

[29] Majid Eskandarpour, Pierre Dejax, Joe Miemczyk, and Olivier Pëton. Sustainable supply chain network design: An optimization-oriented review. *Omega*, 54:11–32, 2015. ISSN 0305-0483. doi: 10.1016/j.omega.2015.01.006. URL `https://www.sciencedirect.com/science/article/pii/S0305048315000080`.

[30] Stefan Falkner, Aaron Klein, and Frank Hutter. Bohb: Robust and efficient hyperparameter optimization at scale. In Jennifer Dy and Andreas Krause, editors, *Proceedings of the 35th International Conference on Machine Learning*, volume 80, pages 1437–1446. PMLR, 2 2018. URL https://proceedings.mlr.press/v80/falkner18a.html.

[31] Alexander I. J. Forrester, András Sóbester, and Andy J Keane. Multi-fidelity optimization via surrogate modelling. *Proceedings of the Royal Society A: Mathematical, Physical and Engineering Sciences*, 463:3251–3269, 12 2007. doi: 10.1098/rspa.2007.1900. URL https://doi.org/10.1098/rspa.2007.1900. doi: 10.1098/rspa.2007.1900.

[32] Alexander I.J. Forrester and Andy J. Keane. Recent advances in surrogate-based optimization. *Progress in Aerospace Sciences*, 45:50–79, 1 2009. ISSN 0376-0421. doi: 10.1016/J.PAEROSCI.2008.11.001. URL https://doi.org/10.1016/j.paerosci.2008.11.001.

[33] Peter I. Frazier. A tutorial on Bayesian optimization, 2018.

[34] Jerome H Friedman. Greedy function approximation: A gradient boosting machine. *The Annals of Statistics*, 29:1189–1232, 10 2001. doi: 10.1214/aos/1013203451. URL https://doi.org/10.1214/aos/1013203451.

[35] Martina Friese, Thomas Bartz-Beielstein, Thomas Bäck, Boris Naujoks, and Michael Emmerich. Weighted ensembles in model-based global optimization. *AIP Conference Proceedings*, page 020003, 2019. doi: 10.1063/1.5089970.

[36] Eduardo C. Garrido-Merchán and Daniel Hernández-Lobato. Dealing with categorical and integer-valued variables in Bayesian optimization with gaussian processes. *Neurocomputing*, 380:20–35, Mar 2020. ISSN 0925-2312. doi: 10.1016/j.neucom.2019.11.004. URL http://dx.doi.org/10.1016/j.neucom.2019.11.004.

[37] K. C. Giannakoglou, D. I. Papadimitriou, and I. C. Kampolis. Aerodynamic shape design using evolutionary algorithms and new gradient-assisted metamodels. *Computer Methods in Applied Mechanics and Engineering*, 195:6312–6329, 2006. ISSN 0045-7825. doi: 10.1016/j.cma.2005.12.008. URL https://www.sciencedirect.com/science/article/pii/S0045782506000338.

[38] David Ginsbourger, Rodolphe Le Riche, and Laurent Carraro. A Multi-points Criterion for Deterministic Parallel Global Optimization based on Gaussian Processes, March 2008. URL https://hal.archives-ouvertes.fr/hal-00260579.

[39] David Ginsbourger, Rodolphe Le Riche, and Laurent Carraro. Kriging is well-suited to parallelize optimization. In Yoel Tenne and Chi-Keong Goh, editors, *Computational Intelligence in Expensive Optimization Problems*, pages 131–162. Springer Berlin Heidelberg, 2010. ISBN 978-3-642-10701-6. doi: 10.1007/978-3-642-10701-6_6. URL https://doi.org/10.1007/978-3-642-10701-6_6.

[40] Dirk Gorissen, Tom Dhaene, and Filip De Turck. Evolutionary model type selection for global surrogate modeling. *Journal of Machine Learning Research*, 10(71):2039–2078, 2009. URL http://jmlr.org/papers/v10/gorissen09a.html.

[41] J. C. Gower. A general coefficient of similarity and some of its properties. *Biometrics*, 27:857–871, 1971. ISSN 0006341X, 15410420. doi: 10.2307/2528823. URL http://www.jstor.org/stable/2528823.

[42] Loic Le Gratiet and Claire Cannamela. Cokriging-based sequential design strategies using fast cross-validation techniques for multi-fidelity computer codes. *Technometrics*, 57:418–427, 7 2015. ISSN 0040-1706. doi: 10.1080/00401706.2014.928233. URL https://doi.org/10.1080/00401706.2014.928233. doi: 10.1080/00401706.2014.928233.

[43] John J. Grefenstette and J. Michael Fitzpatrick. Genetic search with approximate function evaluation. In *International Conference on Genetic Algorithms and Their Applications*, pages 112–120. Psychology Press, 1985.

[44] Hans-Martin Gutmann. A radial basis function method for global optimization. *Journal of Global Optimization*, 19:201–227, 2001. ISSN 1573-2916. doi: 10.1023/A:1011255519438. URL https://doi.org/10.1023/A:1011255519438.

[45] Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, 9:159–195, 2001. doi: 10.1162/106365601750190398. URL https://doi.org/10.1162/106365601750190398.

[46] Nikolaus Hansen, Steffen Finck, Raymond Ros, and Anne Auger. Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions. Research Report RR-6829, INRIA, 2009. URL https://hal.inria.fr/inria-00362633.

[47] Rolland L Hardy. Multiquadric equations of topography and other irregular surfaces. *Journal of Geophysical Research (1896-1977)*, 76:1905–1915, 3 1971. ISSN 0148-0227. doi: 10.1029/JB076i008p01905. URL https://doi.org/10.1029/JB076i008p01905. https://doi.org/10.1029/JB076i008p01905.

[48] Naama Horesh, Thomas Bäck, and Ofer M. Shir. Predict or screen your expensive assay: DoE vs. surrogates in experimental combinatorial optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '19, page 274–284, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450361118. doi: 10.1145/3321707.3321801. URL https://doi.org/10.1145/3321707.3321801.

[49] D. Huang, T. T. Allen, W. I. Notz, and R. A. Miller. Sequential kriging optimization using multiple-fidelity evaluations. *Structural and Multidisciplinary Optimization*, 32:369–382, 2006. ISSN 1615-1488. doi: 10.1007/s00158-005-0587-0. URL https://doi.org/10.1007/s00158-005-0587-0.

[50] Frank Hutter, Holger H. Hoos, and Kevin Leyton-Brown. Sequential model-based optimization for general algorithm configuration. In Carlos A Coello Coello, editor, *Learning and Intelligent Optimization*, pages 507–523. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-25566-3. URL https://link.springer.com/chapter/10.1007/978-3-642-25566-3_40.

[51] Liangyue Jia, Reza Alizadeh, Jia Hao, Guoxin Wang, Janet K. Allen, and Farrokh Mistree. A rule-based method for automated surrogate model selection. *Advanced Engineering Informatics*, 45:101123, 8 2020. ISSN 14740346. doi: 10.1016/j.aei.2020.101123. URL https://doi.org/10.1016/j.aei.2020.101123.

[52] Ruichen Jin, Wei Chen, and Agus Sudjianto. An efficient algorithm for constructing optimal design of computer experiments. *Journal of Statistical Planning and Inference*, 134(1):268–287, 2005. ISSN 0378-3758. doi: 10.1016/j.jspi.2004.02.014. URL https://www.sciencedirect.com/science/article/pii/S0378375804001922.

[53] Yaochu Jin. Surrogate-assisted evolutionary computation: Recent advances and future challenges. *Swarm and Evolutionary Computation*, 1: 61–70, 6 2011. ISSN 22106502. doi: 10.1016/j.swevo.2011.05.001. URL https://doi.org/10.1016/j.swevo.2011.05.001.

[54] Yaochu Jin, Handing Wang, Tinkle Chugh, Dan Guo, and Kaisa Miettinen. Data-driven evolutionary optimization: An overview and case studies. *IEEE Transactions on Evolutionary Computation*, 23:442–458, 6 2019. ISSN 1089-778X. doi: 10.1109/TEVC.2018.2869001.

[55] Donald R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21:345–383, 2001. ISSN 1573-2916. doi: 10.1023/A:1012771025575. URL https://doi.org/10.1023/A:1012771025575.

[56] Donald R. Jones, Matthias Schonlau, and William J Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:455–492, 1998. ISSN 1573-2916. doi: 10.1023/A:1008306431147. URL https://doi.org/10.1023/A:1008306431147.

[57] Ksenia Korovina, Sailun Xu, Kirthevasan Kandasamy, Willie Neiswanger, Barnabas Poczos, Jeff Schneider, and Eric P. Xing. Chembo: Bayesian optimization of small organic molecules with synthesizable recommendations, 2019.

[58] Danie G. Krige. A statistical approach to some basic mine valuation problems on the witwatersrand. *Journal of the Southern African Institute of Mining and Metallurgy*, 52:119–139, 12 1951. doi: 10.10520/AJA0038223X_4792.

[59] Rui Li, Michael T.M. Emmerich, Jeroen Eggermont, Ernst G.P. Bovenkamp, Thomas Bäck, Jouke Dijkstra, and Johan H.C. Reiber. Metamodel-assisted mixed integer evolution strategies and their application to intravascular ultrasound image analysis. In *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pages 2764–2771. IEEE, 6 2008. ISBN 978-1-4244-1822-0. doi: 10.1109/CEC.2008.4631169.

[60] Rui Li, Michael T.M. Emmerich, Jeroen Eggermont, Thomas Bäck, M. Schütz, J. Dijkstra, and J.H.C. Reiber. Mixed Integer Evolution Strategies for Parameter Optimization. *Evolutionary Computation*, 21 (1):29–64, 03 2013. ISSN 1063-6560. doi: 10.1162/EVCO_a_00059. URL https://doi.org/10.1162/EVCO_a_00059.

[61] Dudy Lim, Yew-Soon Ong, Yaochu Jin, and Bernhard Sendhoff. A study on metamodeling techniques, ensembles, and multi-surrogates in evolutionary computation. In *Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation*, GECCO '07, page 1288–1295, New York, NY, USA, 2007. Association for Computing Machinery. ISBN 9781595936974. doi: 10.1145/1276958.1277203. URL https://doi.org/10.1145/1276958.1277203.

[62] Marius Lindauer, Katharina Eggensperger, Matthias Feurer, André Biedenkapp, Difan Deng, Carolin Benjamins, Tim Ruhopf, René Sass, and Frank Hutter. Smac3: A versatile Bayesian optimization package for hyperparameter optimization, 2022.

[63] Michael D. McKay, Richard J. Beckman, and William J. Conover. Comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21:239–245, 5 1979. ISSN 0040-1706. doi: 10.1080/

00401706.1979.10489755. URL https://doi.org/10.1080/00401706.1979.10489755. doi: 10.1080/00401706.1979.10489755.

[64] Charles A Micchelli. Interpolation of scattered data: Distance matrices and conditionally positive definite functions. *Constructive Approximation*, 2:11–22, 1986. ISSN 1432-0940. doi: 10.1007/BF01893414. URL https://doi.org/10.1007/BF01893414.

[65] Melanie Mitchell. *An Introduction to Genetic Algorithms*. MIT Press, 1998. ISBN 0262631857.

[66] Heinz Mühlenbein. The Equation for Response to Selection and Its Use for Prediction. *Evolutionary Computation*, 5(3):303–346, 09 1997. ISSN 1063-6560. doi: 10.1162/evco.1997.5.3.303. URL https://doi.org/10.1162/evco.1997.5.3.303.

[67] Juliane Müller and Christine A. Shoemaker. Influence of ensemble surrogate models and sampling strategy on the solution quality of algorithms for computationally expensive black-box global optimization problems. *Journal of Global Optimization*, 60:123–144, 10 2014. ISSN 0925-5001. doi: 10.1007/s10898-014-0184-0. URL https://doi.org/10.1007/s10898-014-0184-0.

[68] Juliane Müller, Christine A Shoemaker, and Robert Piché. So-i: a surrogate model algorithm for expensive nonlinear integer programming problems including global optimization applications. *Journal of Global Optimization*, 59:865–889, 2014. ISSN 1573-2916. doi: 10.1007/s10898-013-0101-y. URL https://doi.org/10.1007/s10898-013-0101-y.

[69] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, Jake Vanderplas, Alexandre Passos, David Cournapeau, Matthieu Brucher, Matthieu Perrot, and Édouard Duchesnay. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12(85):2825–2830, 2011. URL http://jmlr.org/papers/v12/pedregosa11a.html.

[70] Bartosz Piaskowski. Benchmarking for efficient global optimization algorithms on mixed-integer problems. Master's thesis, Leiden University, Rapenburg 70, 2311 EZ Leiden, Netherlands, 2021.

[71] Riccardo Poli, James Kennedy, and Tim Blackwell. Particle swarm optimization. *Swarm Intelligence*, 1:33–57, 2007. ISSN 1935-3820. doi: 10.1007/s11721-007-0002-0. URL https://doi.org/10.1007/s11721-007-0002-0.

[72] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: Unbiased boosting with categorical features. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, NIPS'18, page 6639–6649, Red Hook, NY, USA, 2018. Curran Associates Inc.

[73] Singiresu S. Rao. *Classical Optimization Techniques*, chapter 2, pages 57–108. John Wiley & Sons, Ltd, 2019. ISBN 9781119454816. doi: 10.1002/9781119454816.ch2. URL https://onlinelibrary.wiley.com/doi/abs/10.1002/9781119454816.ch2.

[74] Saman Razavi, Bryan A Tolson, and Donald H Burn. Review of surrogate modeling in water resources. *Water Resources Research*, 48, 7 2012. ISSN 0043-1397. doi: 10.1029/2011WR011527. URL https://doi.org/10.1029/2011WR011527. https://doi.org/10.1029/2011WR011527.

[75] Frederik Rehbach, Martin Zaefferer, Boris Naujoks, and Thomas Bartz-Beielstein. Expected improvement versus predicted value in surrogate-based optimization. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, pages 868–876. Association for Computing Machinery, 2020. ISBN 9781450371285. doi: 10.1145/3377930.3389816. URL https://doi.org/10.1145/3377930.3389816.

[76] Olivier Roustant, David Ginsbourger, and Yves Deville. Dicekriging, diceoptim: Two r packages for the analysis of computer experiments by kriging-based metamodeling and optimization. *Journal of Statistical Software*, 51:1 – 55, 10 2012. doi: 10.18637/jss.v051.i01. URL https://www.jstatsoft.org/index.php/jss/article/view/v051i01.

[77] Sebastian Ruder. An overview of gradient descent optimization algorithms, 2017.

[78] Sadiq M. Sait and Umair F. Siddiqi. A stochastic evolution algorithm based 2d vlsi global router. *Integration*, 53:115–125, 2016. ISSN 0167-9260. doi: 10.1016/j.vlsi.2015.12.007. URL https://www.sciencedirect.com/science/article/pii/S0167926015001625.

[79] Abdelkader Sbihi and Richard W. Eglese. Combinatorial optimization and green logistics. *4OR*, 5:99–116, 2007. ISSN 1614-2411. doi: 10.1007/s10288-007-0047-3. URL https://doi.org/10.1007/s10288-007-0047-3.

[80] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical Bayesian optimization of machine learning algorithms. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, volume 25. Curran Associates, Inc.,

2012. URL https://proceedings.neurips.cc/paper/2012/file/05311655a15b75fab86956663e1819cd-Paper.pdf.

[81] Yoel Tenne. An optimization algorithm employing multiple metamodels and optimizers. *International Journal of Automation and Computing*, 10:227–241, 2013. ISSN 1751-8520. doi: 10.1007/s11633-013-0716-y. URL https://doi.org/10.1007/s11633-013-0716-y.

[82] Dirk Thierens and Peter A.N. Bosman. Optimal mixing evolutionary algorithms. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, GECCO '11, page 617–624, New York, NY, USA, 2011. Association for Computing Machinery. ISBN 9781450305570. doi: 10.1145/2001576.2001661. URL https://doi.org/10.1145/2001576.2001661.

[83] Tea Tušar, Dimo Brockhoff, and Nikolaus Hansen. Mixed-integer benchmark problems for single- and bi-objective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '19, page 718–726, New York, NY, USA, 2019. Association for Computing Machinery. ISBN 9781450361118. doi: 10.1145/3321707.3321868. URL https://doi.org/10.1145/3321707.3321868.

[84] Bas van Stein, Hao Wang, Wojtek Kowalczyk, and Thomas Bäck. A novel uncertainty quantification method for efficient global optimization. In Jesús Medina, Manuel Ojeda-Aciego, José Luis Verdegay, Irina Perfilieva, Bernadette Bouchon-Meunier, and Ronald R Yager, editors, *Information Processing and Management of Uncertainty in Knowledge-Based Systems. Applications*, pages 480–491. Springer International Publishing, 2018. ISBN 978-3-319-91479-4. doi: 10.1007/978-3-319-91479-4_40. URL https://doi.org/10.1007/978-3-319-91479-4_40.

[85] Bas van Stein, Hao Wang, and Thomas Bäck. Automatic configuration of deep neural networks with parallel efficient global optimization. In *2019 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2019. ISBN 2161-4407. doi: 10.1109/IJCNN.2019.8851720.

[86] Felipe A. C. Viana, Raphael T. Haftka, and Layne T. Watson. Efficient global optimization algorithm assisted by multiple surrogate techniques. *Journal of Global Optimization*, 56:669–689, 6 2013. ISSN 0925-5001. doi: 10.1007/s10898-012-9892-5. URL https://doi.org/10.1007/s10898-012-9892-5.

[87] Hao Wang, Bas van Stein, Michael Emmerich, and Thomas Bäck. A new acquisition function for Bayesian optimization based on the moment-generating function. In *2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pages 507–512, 2017. doi: 10.1109/SMC.2017.8122656. URL https://doi.org/10.1109/SMC.2017.8122656.

[88] Hu Wang, Fan Ye, Enying Li, and Guangyao Li. A comparative study of expected improvement-assisted global optimization with different surrogates. *Engineering Optimization*, 48:1432–1458, 8 2016. ISSN 0305-215X. doi: 10.1080/0305215X.2015.1115645.

[89] Jialei Wang, Scott C. Clark, Eric Liu, and Peter I. Frazier. Parallel Bayesian global optimization of expensive functions. *Operations Research*, 68:1850–1865, 6 2020. ISSN 0030-364X. doi: 10.1287/opre.2019.1966. URL `https://doi.org/10.1287/opre.2019.1966`. doi: 10.1287/opre.2019.1966.

[90] Paul Westermann and Ralph Evins. Surrogate modelling for sustainable building design – a review. *Energy and Buildings*, 198:170–186, 9 2019. ISSN 03787788. doi: 10.1016/j.enbuild.2019.05.057. URL `https://www.sciencedirect.com/science/article/pii/S0378778819302877`.

[91] David H. Wolpert and William G. Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1: 67–82, 1997. ISSN 1941-0026. doi: 10.1109/4235.585893.

[92] Furong Ye, Carola Doerr, and Thomas Back. Interpolating local and global search by controlling the variance of standard bit mutation. *2019 IEEE Congress on Evolutionary Computation (CEC)*, Jun 2019. doi: 10.1109/cec.2019.8790107. URL `http://dx.doi.org/10.1109/CEC.2019.8790107`.

[93] Pengcheng Ye and Guang Pan. Global optimization method using adaptive and parallel ensemble of surrogates for engineering design optimization. *Optimization*, 66:1135–1155, 7 2017. ISSN 0233-1934. doi: 10.1080/02331934.2016.1266627. URL `https://doi.org/10.1080/02331934.2016.1266627`. doi: 10.1080/02331934.2016.1266627.

[94] Martin Zaefferer, Daniel Gaida, and Thomas Bartz-Beielstein. Multifidelity modeling and optimization of biogas plants. *Applied Soft Computing*, 48:13–28, 11 2016. ISSN 1568-4946. doi: 10.1016/j.asoc.2016.05.047. URL `https://www.sciencedirect.com/science/article/pii/S1568494616302575`.

[95] Dawei Zhan and Huanlai Xing. Expected improvement for expensive optimization: a review. *Journal of Global Optimization*, 78:507–544, 2020. ISSN 1573-2916. doi: 10.1007/s10898-020-00923-x. URL `https://doi.org/10.1007/s10898-020-00923-x`.

[96] Antanas Zilinskas. A review of statistical models for global optimization. *Journal of Global Optimization*, 2:145–153, 1992. ISSN 1573-2916. doi: 10.1007/BF00122051. URL `https://doi.org/10.1007/BF00122051`.