Surrogate-based methods for black-box optimization

Ky Khac Vu, Claudia d'Ambrosio, Youssef Hamadi, Leo Liberti

Surrogate-based methods for black-box optimization

Vu Khac Ky, Claudia D'Ambrosio, Youssef Hamadi, Leo Liberti

CNRS LIX, École Polytechnique, F-91128 Palaiseau, France
Email:{vu,dambrosio,youssefh,liberti}@lix.polytechnique.fr

March 14, 2016

Abstract

In this paper, we survey methods that are currently used in black-box optimization, i.e. the kind of problems whose objective functions are very expensive to evaluate and no analytical or derivative information are available. We concentrate on a particular family of methods, in which surrogate (or meta) models are iteratively constructed and used to search for global solutions.

1 Introduction

In this paper, we survey about a special method for solving the following optimization problem:

$$z = \min \{f(x) \mid x \in \mathcal{D}\}, \tag{1}$$

in which $f : \mathbb{R}^d \to \mathbb{R}$ is a continuous black-box and $\mathcal{D}$ is a compact subset of $\mathbb{R}^d$. Here, black-boxes refer to functions that have no analytical form and are often expensive to evaluate. Black-box functions appear in a variety of different settings. The most notable example is computer simulation programs, which may consist of thousands of code lines and take hours, or even days to complete a single run. Since the computer programs implementing black-box functions are often very complicated, it is difficult, impractical, or downright impossible, to obtain an explicit description of these functions. Thus, simulation programs can be treated as black-box functions.

The methods we will present are based on the construction of surrogate models to approximate black-box functions and using them to search for (global) optimal solutions. They are often called surrogate-based method. In this paper, we will mainly focus on continuous variables. The continuity and compactness of f are technical assumptions which ensure the existence of a global minimum of this problem. Later on we will briefly discuss about the case when there are discrete variables.

1.1 Motivational examples

One classic example of real world application of black-box optimization is reservoir engineering, e.g., the so-called wells placement problem. In this problem, we aim at deciding the location and geometry (i.e. trajectory) of wells. The independent variables are: the number of wells, their trajectory, and their type (injector or producer). The main constraints are represented by minimum distance between wells to be respected while the objective is typically to either maximize the oil production or the Net Present Value (NPV) function. For a given solution, the oil production or the NPV is given as result of a simulation that is heavy in practice (minutes for simplified cases to several hours for the more realistic ones), see PumaFlow [21]. In the literature, evolutionary/metaheuristic algorithms can be found, see, for example, Bouzarkouna et al. [11] and Onwunalu et al. [48]. More recently, hybrid approaches have been proposed, see, for example, Lizon et al. [37].

Building design is another field to which black-box optimization is often applied. One of the examples is the SmartBuilding project of Microsoft Research [19], with the aim to devise new methods for optimizing smart buildings design by minimizing energy consumption, construction cost, and so on. The energy consumption of a building (in one year period) is estimated by EnergyPlus, a simulation program that is computationally expensive, which takes up to 1 hour for a single simulation. Decision variables are selected to be both continuous and discrete: the orientation angle, the scaling factors of window sizes, and the choices of material for each inner insulation layer of the external walls. In [19], the problem is solved using an evolutionary algorithm called HYPE. A similar problem in architectural designs [15] is solved using a Radial Basis Functions based method.

1.2 Heuristical methods

There are many difficulties associated with black-box optimization. The most obvious is that the problems has no structure, so we can not exploit their analytical and derivative information. Due to this lack of structure, the most natural way to attack black-box optimization is to used heuristics such as evolutionary algorithms, simulated annealing, tabu search... This approach has the advantage that it only relies on function evaluations and is relatively easy to implement. However, due to the computational cost of function evaluation, these methods are only applicable for problem with small dimension since they often require a large number of function evaluations. Moreover, sometimes they are not desirable since they lack of mathematical convergence guarantees: generally, we do not know how close our solutions are from the optimum.

1.3 Derivative-free methods

The lack of derivatives motivates us to use derivative-free optimization (DFO) methods to solve black-box optimization problems. One of these methods is directional direct-search, which is often studied in a search-and-poll framework [3, 6, 62]. This method makes use of the fact that if a solution is not locally optimal, then we can improve it by moving along one of the directions in some positive bases (or spanning sets). Based on this observation, from a given point x_k, we will look at all vectors u_k in a positive basis and select an appropriate step size α_k such that $f(x_k + \alpha_k u_k) < f(x_k)$. Since the number of vectors in a positive basis is large and feasible step sizes are unknown, in general, it might be expensive to find an updated better point. Moreover, most of the directional direct-search algorithms can only find local solutions. *Mesh adaptive direct search* (MADS), proposed by Audet and Dennis [6], is one of the few algorithms in this class that can converge to local solutions in smooth and non-smooth cases. This algorithm is implemented in NOMAD [1, 35], an open-source software for solving BB optimization, which can also deal with black-box constraints, multiple objectives and parallelism, in both continuous and discrete cases.

Another DFO method is simplicial direct-search. In this method, new points are searched in the direction that is as far as possible from the vertex with the worst function value. One of the most popular methods in this class is proposed by Nelder-Mead [47], in which a simplex consists of $d+1$ points are maintained at each iteration. Then, new points are found by performing either reflection, expansion, or contraction. The worst points in the simplex will be replaced by a point constructed from one of the 3 points. This method converses, but sometimes not to a stationary point. Several additional modifications are taken into consideration to overcome this drawback.

The two above methods may require substantial computing resources since the number of function evaluations is sometimes very large. This occurs because the inherent smoothness of the blackbox function is not exploited appropriately. A method which overcomes this disadvantage to a certain extent is the trust-region algorithms. In trust-region methods, at each step, the objective function is approximated by a linear (or quadratic) model using current data points. The model is solved within a local region (also called trust-region), which is basically a ball around the best current point (for certain norms). The

trust-region is then adaptively changed, either by increasing or decreasing the radius or by moving its center, based on how much improvement of the newly found solution compared to the current best point. With the assumption that the model is fully linear (or fully quadratic), trust-region method converges to stationary solutions [14]. However, similar to the two previous methods, it fails to converge to a global solution.

1.4 Surrogate-based methods

In this paper, we will concentrate on another approach which appears to be very successful at finding global solutions of black-box optimization problems. The main idea behind these methods is to iteratively construct surrogate models to approximate the black-box functions (globally) and use them to search for optimal solutions [58]. A common approach (in its simplest form) for surrogate-based methods is as follows

- Phase 1 (design): Let $k := 0$. Select and evaluate a set S_0 of starting points.
- While some given stopping criteria are not met:
  - Phase 2 (model): From the data $\{(x, f(x)) \mid x \in S_k\})$, construct a surrogate model $s_k(.)$ that approximates the black-box function.
  - Phase 3 (search): Use $s_k(.)$ to search for a new point to evaluate.
    Evaluate the new chosen point, update the data set S_k. Assign $k := k + 1$.

Phase 1 is commonly referred to as sampling or design of experiment (DOE) [20, 55]. Its purpose is to find a set of points uniformly spread over the domain, so that, if we evaluate the function at these points, we might obtain a global picture of its range. To better achieve this a uniformity measure of each point set is employed. Our expected sampling is then found by maximizing uniformity (this is known as the design problem). In Section 2, various such measures as well as methods for solving associated design problems are introduced. We also mention the definition of stratified sample plans. Among those *Latin hypercube* [40] is the most popular.

In **Phase 2**, various models can be used to approximate black-box functions. In Section 3, we will introduce some of the most popular models, including polynomials, radial basis functions (RBF) and kriging. While polynomials are well-studied, their use as global models for high-dimensional black-box functions is only limited to linear and quadratic cases. Other models like RBF and kriging can fit more complicated functions while still being relatively simple to build. For this reason, many recent surrogate-based methods for black-box optimization make use either one of the two as interpolation models. It is also useful to use multiple surrogate models at the same time, because it can prevent us from choosing poorly fitted models [64].

Phase 3 is the crucial step in the procedure. Given the information from the current surrogate model, we need to decide which point(s) should be evaluated in the subsequent step. There are many strategies to do that, and indeed they represent the main feature distinguishing different surrogate-based optimization methods. The most common idea of these strategies is to define a merit (or cost) function of candidate point that predicts the objective and/or model accuracy improvement if we evaluate the black-box there. We select the next point for evaluation to be the one that maximizes (or minimizes) the merit function (resp. cost function). Different merit functions and methods for solving associated subproblems will be introduced in Section 4.

With the emphasis on *expensive* functions, the whole process for solving black-box optimization problems is often dominated by the number of function evaluations. Therefore, the performance of an algorithm is often measured by the number of evaluations needed until an acceptable (global) solution is found. In fact, it may also be used as a stopping criterion in black-box optimization: we stop when we reach a certain number of evaluations.

The above procedure is only a simplification of various surrogate-based methods for black-box optimization. In the literature, there are more complex frameworks. For example, surrogate models can be used in the poll step of a directional direct-search: Before the poll trial points are evaluated on the true functions, they are evaluated on the surrogates. This list of points is sorted according to the surrogate values so that the most promising points are evaluated first during the true evaluations [10, 13, 35]. This is called surrogate management framework [10], various formulations for it is proposed in [61]. In [54], a quasi-multistart framework called AQUARS is proposed, which runs a multistart method to identify the local minima of the current surrogate model. At every iteration, AQUARS checks if a local minimum has been "fully explored", if this is the case, we do not need to carry out any further exploration in its vicinity. AQUARS also decides on which type of local search (breadth first or depth first) should be performed. Another framework that combines local and global methods is presented in [29]. It consists of two procedures: the first constructs a global model to approximate the black-box function and explores an unvisited area to search for a global solution; the other identifies a promising local region and conducts a local search to ensure local optimality. At each iteration, we divide all existing data points into 2 sets: a *full information* set, which consists of all evaluated points so far, and a *partial information* set, which is the set of only evaluated points since the most recent *Restart*. Surrogate models are used to fit both these two types of sets to serve different purposes: we use full information model to switch back and forth between local and global searches and use partial information model to generate candidate points for global search.

1.5 Dealing with constraints and multiple objectives

Black-box optimization problems become much more difficult with the present of multiple black-box functions, such as in the case of multiple black-box objectives or constraints (we remark that constraints having inexpensive evaluation, such as when the analytical form is given, are implicitly included in the feasible set $\mathcal{D}$). The most natural idea to deal with multiple expensive constraints is to combine them all into a weighted penalty term added to the objective function. Another idea is to aggregate all these nonlinear constraints into a single constraint violation function, and possibly even treat the optimization problem as an unconstrained bi-objective problem [4] or treat it as a single constraint [5]. However, a more direct approach for solving these problems is to construct one surrogate model for each black-box function [2, 36]. Using surrogate models for constraint functions helps us to predict feasible points, so we can avoid spending time to evaluate points that might be infeasible.

The rest of this paper is constructed as follows: In Section 2, we give an overview of the experimental design problem. In Section 3, we introduce some popular surrogate models used in practice. In Section 4, we review common choices of merit functions, associated search domains and their uses for solving black-box optimization. Section 5 concludes the paper by summarizing the idea of surrogate-based methods and suggesting a few possible research directions.

2 Overview of experimental designs

The first step in any surrogate-based method is to find a set of starting points for evaluation. This step is commonly referred as *design of experiments* (DOE) [20, 30, 40, 55]. Function values given at these points provide us the very first information about the black-box function, which is then used to construct suitable surrogate models. Good choices of experimental designs can help us to identify inadequacies in proposed models, as well as to avoid model bias [56].

In the following, such a set of starting points will be called *a design* and the entire domain $\mathcal{D}$ will be called the *design space*. Moreover, we will mostly discuss computer experiments. The main difference between a physical and a computer experiment is that the latter is *deterministic*, i.e. replicate observations at the same input yield the same results [55, 56]. Computer experiments therefore lack random errors. Because of this, each input corresponds to a unique response. A word of caution about this statement is

in order, however: in practice, computers can (and usually do) behave nondeterministically, as hardware glitches provide a constant source of errors — only the error correction codes save us from total chaos.

If no detail about the functional behavior of the response is available, we need to explore the entire design space. Therefore, we will need to find a design set so that a variety of models can be fitted and it must spreads over the experimental region [56]. Moreover, if one of the input parameters has little influence on the function value, two points that differ only in this parameter should be considered the same. To avoid this situation, we require that no two design points share common value in any dimension.

From the above discussion, we summarize the two most important requirements for a good experimental design [27, 59]:

- **space-fill**: The design points should be uniformly spread over the entire design space.
- **Non-collapse**: Two design points should not share any coordinate value if we do not know a priori which dimensions are important.

Note that when the number of variables is high, preprocessing the problem to reduce its dimension is crucial. If all the parameters in the problem are selected to be *important*, the non-collapse requirement becomes less significant. In this case, designs with only*space-fil* are easy to construct. However, many practical applications are often subject to several constraints, therefore, we list an additional requirement (which is also hard to deal with) for a design set

- **Constraints-fil**: The design points have to satisfy a set of linear and/or nonlinear constraints.

The concept of space-fill is based on spreading points uniformly over the design space. There are many definitions which result in different quality measures for a design [56]. Some of the most popular measures in the literature, often based on either geometrical or statistical properties, are *minimax, maximin, IMSE, maximum entropy, Audze-Eglais,* and so on. The associated *design problems* consist in maximizing (or minimizing) these measures.

There is a trade-off between space-fill and non-collapse properties: a design with good space-fill properties is often collapsing (a striking example is *full factorial designs*). To obtain a good design in terms of both space-fill and non-collapse, we often solve the design problem over a reduced class of non-collapsing designs. Among such classes, *Latin hypercube designs* (LHD) [38] are the best known. On the other hand, some point generation techniques, such as Sobol' sequence, avoid this trade-off by uniformly filling the space with non-collapsing point sets.

In fact, finding an optimal design is a very complicated problem, especially in high dimension. For example, with a relatively simple measure among them, the *maximin* LHD problem is believed to be NP-hard [18] (to the best of our knowledge, the question is still open).

In the next subsection, we will introduce various criteria to measure the space-fill and/or non-collapse of a designs. After that, we overview common methods to find optimal designs with respect to such criteria.

2.1 Space-filling and non-collapsing designs

In this subsection, for simplicity, we assume that the design space $\mathcal{D}$ is a compact subset of $\mathbb{R}^d$ and is defined by box constraints. More specifically, let $\mathcal{D} = \{x \in \mathbb{R}^d | \, L_i \leq x_i \leq U_i\}$. In most cases, we assume that all design parameters are equally important, therefore we can scale each interval $[L_i, U_i]$ linearly to $[0, 1]$ for all dimensions i. This subsection reviews the most common space-filling and non-collapsing designs over $\mathcal{D}$. Various criteria based on the same idea for selecting optimal designs are introduced: first, define a measure $\Phi(.)$ of space-fill and then optimize $\Phi(S)$ over all $S \subseteq \mathcal{D}$ with $|S| = n$. When we

restrict the search space to a certain class of non-collapsing designs, we can find designs that both satisfy the two requirements. There are many choices for $\Phi(.)$ and each choice corresponds to a different type of designs. We categorize them into two main groups: geometrical designs and statistical designs.

2.1.1 Geometrical designs

Factorial designs: *Full factorial designs* might be the simplest and most straightforward way to sample points in the uniform manner [20]. They are constructed by dividing each dimension i of the design space $\mathcal{D} \subseteq \mathbb{R}^d$ into n_i equal intervals and then selecting centers of all $n = n_1 n_2 \dots n_d$ resulting cells. Obviously, full factorial designs are highly uniform but lack of the non-collapse property. Another drawback is their inflexibility regarding the number of sampling points. In particular, a full factorial design can only be used if the number of sampling points is of the form $n = n_1 n_2 \dots n_d$, which is often very large ($n \geq 2^d$). This almost prohibits the use of full factorial designs for sampling sets of size that is small or given in advance. Sometimes, we use a variation called *fractional factorial* design, in which only a fraction of the design points specified by the full factorial designs are considered.

Latin hypercube designs: *Latin hypercube designs* (LHD) [38] are the most popular non-collapsing designs in the literature. An LHD with n points is constructed as follows: we first divide each axis into n equal intervals: now the design space $\mathcal{D}$ will consist of n^d identical cells. An LHD is then obtained by assigning n points to centers of these n^d cells such that there are no two points with the same coordinate in any dimension. One point can be conveniently represented as an element in $\{1, 2, \dots, n\}^d$ (not to be confused with its coordinate). Therefore, we can interpret an LHD as a set of n points $x_1, \dots, x_n \in \{1, 2, \dots, n\}^d$ with the property that, for each $j \in \{1, \dots, d\}$, the set $\{x_{1j}, \dots, x_{nj}\}$ is a permutation of $\{1, 2, \dots, n\}$. Formally, let $S = \{x_1, \dots, x_n\}$ be a set of n points in $\{1, 2, \dots, n\}^d$, we define the *design matrix* of S as follows:

$$M(S) := \begin{bmatrix} x_{11} & x_{12} & \dots & x_{1d} \\ x_{21} & x_{22} & \dots & x_{2d} \\ \dots & \dots & \dots & \dots \\ x_{n1} & x_{n2} & \dots & x_{nd} \end{bmatrix}.$$

Then S is an LHD if each column of $M(S)$ is a permutation of $\{1, 2, \dots, n\}$. Note that an LHD is not necessary space-filling. In Figure 1, two different LHDs are presented but in the first design, all points are distributed along the main diagonal (so they do not spread uniformly over the design space).

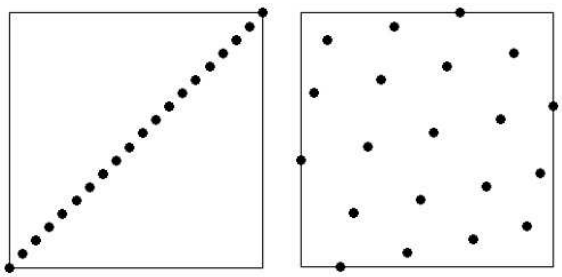

Figure 1: Latin square examples (reproduced from [Viana13]).

Minimax (mM) distance design [30]: Assume that $(\mathcal{D}, d)$ is a metric space for some metric $d : \mathcal{D}^2 \to \mathbb{R}_+$. For any $x \in \mathcal{D}$ and $S \subseteq \mathcal{D}$, denote $d(x, S) = \min_{y \in S} d(x, y)$. We define a *minimax distance* design of cardinality n to be the set S^* that solves

$$\min_{|S|=n} \max_{x \in \mathcal{D}} d(x, S).$$

The definition of minimax distance designs is very intuitive: we want each point in $\mathcal{D}$ to be presented (i.e. be close to) by at least one of the selected points. Therefore, the information about that point can be exploited through its representative. In particular, in the minimax design problem, we want to choose a minimal radius $r > 0$ and a set of centers $x_1, \ldots, x_n$ such that the design space $\mathcal{D}$ is covered by the union of all the spheres $B(x_i, r)$, $i = 1, \ldots, n$. Therefore, minimax distance designs closely relate to the covering problems. They are, however, are computationally difficult to find, because for a given design, an optimization step is required to calculate the value of the measure $\max_{x \in \mathcal{D}} d(x, S)$ [59].

Maximin (Mm) distance design [30]: Similarly, a *maximin distance design* S^* of cardinality n is the one that solves

$$\max_{|S|=n} \min_{x,y \in S} d(x, y).$$

The definition of maximin distance designs is also very intuitive: since the number of design points is often relatively small and the design space is often very large, we do not want to evaluate two points that are too close to each other. The maximin distance design problem is similar to the packing problem, in which we need to choose a maximal radius $r > 0$ and a set of centers $x_1, \ldots, x_n$ such that all the spheres $B(x_i, r)$, $i = 1, \ldots, n$ are pairwise disjoint. Moreover, the space-fill measure in this case, i.e. $\min_{x,y \in S} d(x, y)$, is quite simple. For that reason, Mm distance designs are the most well-studied among all experimental designs [20, 27, 59].

Φ_p designs [40]: One problem with maximin distance designs is it only considers the smallest distance of a point set and neglects other distances. Another design [40], makes all distances of the point set counted. In this design, we have to minimize:

$$\Phi_p(S) = \Big(\sum_{x,y \in S} \frac{1}{d(x, y)^p} \Big)^{1/p},$$

where p is a parameter specified by users. Note that the choice of p is very important. When p is large enough, the contributions of largest distances become the dominating factors and Φ_p designs are equivalent to minimax distance designs. However, when p gets larger, the optimization problem becomes more difficult. For lower values of p, the measure Φ_p might not exactly be the same as the minimax design, but it is more amenable to optimization methods [20]. Note that when $p = 2$, we have the Audze-Eglais distance design (also called potential energy) [7]:

$$S^* \in \arg \min_{|S|=n} \sum_{x,y \in S} \frac{1}{d(x, y)^2}.$$

This design is favorable because when dealing with euclidean metrics, the quantities $d(x, y)^2$ will cancel out all square roots. Therefore, the associated objective function becomes analytically simple. Moreover, when minimizing this measure over the reduced class of LHDs, we can obtain designs with very good properties and useful for many applications [8, 27, 60].

2.1.2 Statistical designs

In statistical designs, we treat a deterministic response of a black-box function as if it were a realization of a stochastic process X. We assume X has mean zero and correlation function R with the property that for all $x_i, x_j \in \mathcal{D}$, $R(x_i, x_j)$ increases when $d(x_i, x_j)$ decreases (i.e. when x_i and x_j are close, their correlation is high and vice versa). Many other functions of this kind can be found in [30]. One example of such a correlation function is $R(x_i, x_j) = \exp\big(-\theta \|x_i - x_j\|\big)$ for some $\theta > 0$. In [55], a different choice of R is given by

$$R(x_i, x_j) = \exp\Big(\sum_{k=1}^{d} -\theta_k |x_{ik} - x_{jk}|^t \Big), \qquad 1 \le i, j \le n,$$

for some $\theta_k > 0$ and $1 \leq t \leq 2$. These functions are more convenient than the former, since we can express them as products of one-dimensional correlations, while still provide enough flexibility for adequate prediction in most cases. We will explain more about this function later.

Now we want to choose the design set S such that the response X can be best predicted. This is the subject of classical *optimal design*. Many different criteria for choosing *optimal* designs which exploit the information about the assumed correlation $R(.,.)$ have been proposed. We will not go into details about all of them, instead we discuss some relatively intuitive criteria being proposed by [30]. These criteria rely on the predictors $\hat{X}(x)$ at each $x \in \mathcal{D}$, which are found based on the assumed data at $\{x_1, \ldots, x_n\}$. Details on how to find these predictors will be presented in Section 3.

I-optimality:
We are given a weight function $\phi(s)$ and let $\operatorname{Var} \hat{X}(s)$ be the variance of the prediction $\hat{X}(s)$ (note that both $X(s)$ and $\hat{X}(s)$ are random). A design S^* is called *I-optimal* if it solves

$$\min_{|S|=n} \int_{\mathcal{D}} \operatorname{Var} \hat{X}(s)\phi(s)\, ds.$$

In [55], the function being minimized is called *integrated mean squared error* (IMSE). When $\mathcal{D}$ is finite, we often select $\phi(s) \equiv 1$, so the objective becomes $\sum_{s_i \in \mathcal{D}} \operatorname{Var} \hat{X}(s_i)$, which is the total error associated with all the prediction points.

G-optimality:
Similar to I-optimality, a design S^* is called *G-optimal* if it solves

$$\min_{|S|=n} \big[\max_{s \in D} \operatorname{Var} \hat{X}(s)\big],$$

i.e. minimizes the maximum error of all predictions. In [55], the function being minimized is called *maximum mean squared error* (MMSE).

D-optimality:
For any design set $S = \{x_1, x_2, \ldots, x_n\}$, let

$$\mathbf{R}(S) = \begin{bmatrix} \text{Corr } (X(x_1), X(x_1)) & \ldots & \text{Corr } (X(x_1), X(x_n)) \\ \ldots & \ldots & \ldots \\ \text{Corr } (X(x_n), X(x_1)) & \ldots & \text{Corr } (X(x_n), X(x_n)) \end{bmatrix}$$

be the correlation matrix of S. A design S^* is called *D-optimal* if it solves

$$\max_S \det \mathbf{R}(S),$$

i.e. maximizes the determinant of the correlation matrix $\mathbf{R}(S)$. Note that large values of $\det \mathbf{R}(S)$ imply the low correlations of all random variables $X(x_i)$. Since $R(x, y)$ decreases when the distance $d(x, y)$ increases, in general, D-optimal designs are uniformly distributed. Moreover, D-optimality is much easier to deal with than G-optimality and I-optimality since it does not involve predictions at any $x \in \mathcal{D}$.

Another way to explain D-optimality is via the concept of *entropy*: we want to find design sets that minimize the expected posterior entropy, i.e. the sets of points at which we have the least information. Shewry and Wynn [57] showed that this is equivalent to minimizing

$$-\log (\det\ \mathbf{R}(\mathbf{S})),$$

which is obviously equivalent to finding D-optimal designs.

2.2 Methods for finding optimal designs

From the previous subsection, we know that a general design problem can be written as

$$\min_{S \subseteq \mathcal{D}, |S|=n} \Phi(S) \tag{2}$$

where $\Phi(.)$ is a measure that quantifies the non-uniformity of a set. To get the balance between space-filling and non-collapsing properties, we also consider *Latin hypercube design problems*, in which the search space for S is restricted to the class $\mathcal{LD}$ of all Latin hypercube designs:

$$\min_{S \subseteq \mathcal{LH},\, |S|=n} \Phi(S). \tag{3}$$

In the previous subsection, we have introduced many design measures $\Phi(.)$ but have not mentioned how to solve the problems (2), (3). Before moving on to discuss methods for solving them, we should note that it is not obvious which measures should be used in design problems. A design found by optimizing one criterion is not necessarily good at the others. Moreover, design problems are often very hard to solve. Moreover, optimality guarantees are not required. Therefore, heuristic methods, such as evolutionary algorithms, may be more practical. For example, in [8], the authors apply a permutation genetic algorithm to find good Audze-Eglais LHDs. Results for eight different combinations of n and k are also reported [8, 27].

In principle, we can apply standard nonlinear programming (NLP) methods to find optimal designs. Most of the previously mentioned design problems are already in the form of NLPs, so we can simply plug them into a solver. For example, in the maximin distance design problem, we exploit the idea of maximizing the radius of n identical balls centered in $\mathcal{D}$ which form a packing to obtain the following NLP:

$$\max \{r \mid x_i \in \mathcal{D},\, d(x_i, x_j) \geq 2r \text{ for all } i, j\}. \tag{4}$$

If n is small, this problem can be handled by powerful NLP solvers. However, when n gets larger, these NLPs become very difficult due to their nonconvexity and the number of constraints [59].

2.2.1 Local improvements

In experimental design problems, we are required to optimize the locations of n points. Therefore, it is natural to use the idea of local improvement: we start with a particular design and iteratively improve it by optimizing the location of one point while fixing $n-1$ others. The problem of nd variables will now be converted into subproblems of only n variables.

This idea can be implemented for many design problems. For example, for Φ_p family, we can consider the following subproblems

$$\min_{x_i \in \mathcal{D}} \sum_{j \neq i} \frac{1}{d(x_i, x_j^*)^p}$$

in which $x_1^*, \ldots, x_n^*$ is the current design set.
In [59], the NLP (4) is replaced by the following NLP subproblems, which are still non-convex, but generally much easier to solve:

$$\max_{x_i \in \mathcal{D}} \{r \mid d(x_i, x_j^*) \geq 2r \text{ for all } j \neq i\}.$$

Note that at each step we have to select a suitable index i to process. We can sequentially select i from any permutation of $\{1, 2, \ldots, n\}$, but a smarter choice will probably better perform. For example, in [59] an index i will be skipped if the previous iteration shows no significant improvement with respect to x_i.

2.2.2 Column exchange

Column exchange is a special local technique applied for Latin hypercube design problems. Instead of locally modifying one point from the current design, we look for improvements by perturbing one or

several columns of the design matrix. Recall that a LHD matrix is a matrix in which each column is a permutation of $\{1, 2, \ldots, n\}$. Therefore if we replace one column by another permutation of $\{1, 2, \ldots, n\}$, we still obtain a LHD matrix.

In [40], the authors use a simulated annealing method to optimize LHDs. The method starts with a certain Latin hypercube design S. At each iteration, a column of the design matrix $M(S)$ is selected and a new design S' is generated from S by interchanging two random entries in that column. If the objective function Φ is improved, S will be replaced by S'. Otherwise, we decide whether to keep S or update S to S' by computing a probability given by $\exp\left(\frac{\Phi(D')-\Phi(D)}{t}\right)$, where t is an algorithm parameter often known as *temperature*. In [23], an Iterated Local Search (ILS) heuristics is proposed, in which column exchanges are used in both local search steps and perturbation steps. In the local search step, we define a set of critical points and the swaps are only made between components of these points. In the perturbation step, cyclic order exchanges are performed to get rid of local optimum.

Since the class of all LHDs is quite large (there are $(n!)^{d-1}$ possibilities), a restricted class of *symmetric Latin hypercube designs* (SLHD) with desirable properties is introduced in [65]. A design S is called *symmetric* if the sums of the i^{th} and $(n+1-i)^{\text{th}}$ row vectors are always equal to $(n+1, \ldots, n+1)$ for all i. SLHDs are claimed to work better than regular LHDs in the cases of entropy or maximin measures [65]. In [65], an columnwise-pairwise algorithm is proposed for finding optimal designs within this class. At each iteration, two pairs of entries (instead of one) are exchanged to ensure that the resulting design is still symmetric. These points are not randomly chosen as in [40], but they are selected as the best two simultaneous exchanges.

3 Surrogate models

This section introduces some of the most popular surrogate models being used in engineering designs. We begin with polynomials, a class of functions that are well-studied. They are, for example, used in trust-region methods to provide approximation of the true function in local areas. However, they are unsuitable as global models for highly nonlinear, multidimensional functions. Other models that are better performed, including radial basis functions and kriging will be introduced in the subsequences.

3.1 Polynomials

Let $\mathcal{P}^m$ be the space of all polynomials of degree at most m on $x \in \mathbb{R}^d$. Let $\{\phi_1, \phi_2, \ldots, \phi_M\}$ be a basis of $\mathcal{P}^m$, then each polynomial on $\mathcal{P}^m$ can be written as

$$p(x) = \sum_{i=1}^{M} a_i \phi_i(x) \tag{5}$$

where $a_i \in \mathbb{R}$, $i = 1, 2, \ldots, M$. An example of such a basis is the natural basis $\phi_\alpha(x) = x^\alpha := x_1^{\alpha_1} \ldots x_d^{\alpha_d}$, where $\alpha_j \in \mathbb{N}$ for all $1 \leq j \leq d$ and $\alpha_1 + \ldots + \alpha_d \leq m$. Assume that we want to find a polynomial of the form (5) that interpolates the data $(x^1, y^1), \ldots, (x^n, y^n)$, then we need to solve the system of equations:

$$p(x^i) = y^i \qquad \text{for all } i = 1, 2, \ldots, n.$$

Denote by

$$M_{\phi,x} = \begin{bmatrix} \phi_1(x^1) & \ldots & \phi_M(x^1) \\ \ldots & \ldots & \ldots \\ \phi_1(x^n) & \ldots & \phi_M(x^n) \end{bmatrix}, \; a = \begin{bmatrix} a_1 \\ \ldots \\ a_M \end{bmatrix} \text{ and } y = \begin{bmatrix} y^1 \\ \ldots \\ y^n \end{bmatrix},$$

then the above system can be written as $M_{\phi,x}\, a = y$. This system has a solution if the matrix M is full-row rank. It is well-known that when $n = M$, the existance and uniqueness of the interpolation does not depend on particular choices of the basis $\{\phi_i\}_i$ (see the proof in [14]).

3.1 Lemma
Let $x^1, \ldots, x^n$ be such that $M_{\phi^*,x}$ is a non-singular square matrix for some basis $\phi^* = \{\phi_1^*, \ldots, \phi_M^*\}$ in $\mathcal{P}^m$. Then for any choice of basis ϕ, the system $M_{\phi,x}\, a = y$ has an unique solution.

The above lemma, which requires $n = M$, offers no flexibility on choosing the number of evaluation points n. In general, when $n < M$, we need to restrict the search to a subspace $\mathcal{V} \subseteq \mathcal{P}^d$ in order to ensure the unique existance of the interpolated polynomial. The theory of polynomial interpolation goes beyond the scope of this paper, so we recommend the survey [22] by M. Gasca and T. Sauer for interested readers. The primary weakness of polynomials are their inability to fit smooth functions of any shapes [26]. Moreover, they require a large number of parameters to be estimated (due to the high dimension of the space $\mathcal{P}^d$). Polynomial model, however, is used for interpolation in practice due to the wide availability of techniques and softwares for computing polynomials.

3.2 Radial basis function

A radial function ϕ is the one whose value at each point depends only on the distance between that point and the origin, i.e. $\phi(x) = \xi(\|x\|)$ where ξ is a real function. Examples of such functions are

- Linear: $\phi(x) = \|x\|$
- Cubic: $\phi(x) = \|x\|^3$
- Thin plate spline: $\phi(x) = \|x\|^2 \log \|x\|$
- Multiquadric: $\phi(x) = \sqrt{\|x\|^2 + \gamma^2}$
- Gaussian: $\phi(r) = e^{-\gamma \|x\|^2}$ where γ is a prescribed positive constant.

Given a set of points $S = \{x_1, x_2, \ldots, x_n\},\ x_i \in \mathbb{R}^d$ and their corresponding responses $y_1, y_2, \ldots, y_n \in \mathbb{R}$, we want to interpolate the data $(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)$ by functions of the form

$$s_n(x|S) = \sum_{i=1}^{n} \lambda_i \phi(\|x - x_i\|) + p(x) \tag{6}$$

where p is a polynomial and ϕ is one of the radial basis functions listed above. Let denote by $\mathbf{p} = (p(x_1), \ldots, p(x_n)),\ \mathbf{y} = (y_1, \ldots, y_n),\ \lambda = (\lambda_1, \ldots, \lambda_n)$ and $\Phi = \left[\phi(\|x_i - x_j\|)\right]_{1 \le i,j \le n}$, then the interpolation condition system can be written as $\Phi\lambda + \mathbf{p} = \mathbf{y}$. The polynomial p appears here to ensure the existence of at least one solution for this system (otherwise, when Φ is nonsingular, the system $\Phi\lambda = \mathbf{y}$ has no solution).

In [24], all RBFs listed above are discussed (using the concept of conditionally positive/negative definiteness), but later the author claimed that the multiquadric and Gaussian cases are disappointing (according to [9]). If $\phi(x)$ is either linear, cubic or thin plate spline, the polynomial p can be taken in the form of $p(x) = b^T x + a$ (or just a constant $p(x) = a$ in the case $\phi(x)$ is linear). The unknown parameters $\lambda_1, \ldots, \lambda_n, a, b_1, \ldots, b_d$ are obtained by solving the following system of linear equations

$$\begin{pmatrix} \Phi & P \\ P^T & 0 \end{pmatrix} \begin{pmatrix} \lambda \\ c \end{pmatrix} = \begin{pmatrix} \mathbf{y} \\ 0 \end{pmatrix}, \qquad \text{where } P = \begin{pmatrix} x_1^T & 1 \\ \ldots & \ldots \\ x_n^T & 1 \end{pmatrix} \text{ and } c = \begin{pmatrix} b_1 \\ \ldots \\ b_d \\ a \end{pmatrix}.$$

When $\operatorname{rank}(P) = d+1$, the matrix $\begin{pmatrix} \Phi & P \\ P^T & 0 \end{pmatrix}$ is nonsingular, so the above equation has a unique solution. Let Π_1 be the set of linear polynomials and $\mathcal{V}$ be the space of all $\lambda \in \mathbb{R}^n$ that satisfy

$$\sum_{i=1}^{n} \lambda_i p(x_i) = 0 \ \text{ for all } \ p \in \Pi_1.$$

For all RBFs being listed above, there is a nonnegative integer m_0 such that $(-1)^{m_0}\lambda^T\Phi\lambda > 0$ for all $\lambda \in \mathcal{V}$. Let $\mathcal{A}_\phi$ be the set of functions that are defined as in (6) with $\lambda \in \mathcal{V}$. For any s and u in $\mathcal{A}_\phi$, say

$$s(x) = \sum_{i=1}^{n_1} \lambda_i \phi(\|x - x_i\|) + p(x) \quad \text{and} \quad u(x) = \sum_{j=1}^{n_2} \mu_j \phi(\|x - z_j\|) + q(x),$$

we define a semi-inner product of s and u by

$$\langle s, u \rangle := (-1)^{m_0} \sum_{i=1}^{n_1} \sum_{j=1}^{n_2} \lambda_i \, \mu_j \, \phi(\|x_i - z_j\|).$$

As usual, $\|s\| = \sqrt{\langle s, s \rangle}$ also defines a semi-norm on $\mathcal{A}_\phi$ and we have

$$\|s\|^2 = \langle s, s \rangle = (-1)^{m_0} \lambda^T \, \Phi(x) \, \lambda.$$

When a lower bound on the black-box function is known, the RBF model can be modified to justify this information. If we know that $\underline{y}$ is a lower bound of function $f(x)$, then any surrogate model whose minimum value drops below $\underline{y}$ should be avoided. A procedure for generating lower-bounded interpolants is proposed in [12]. The procedure will augment the sample set S_n by a set of additional points S_f that are discarded once a new point x_{n+1} is selected and $f(x_{n+1})$ evaluated.

Assume that $\overline{x}$ is a global solution for the current model $s_n(x|S_n)$. If $s_n(\overline{x}|S_n) \geq \underline{y}$, then our constructed model satisfies the required lower bound $\underline{y}$ and we can use it without any further modification. Otherwise, $\overline{x}$ is temporarily grouped into our design set S_n. Now, $s_n(\overline{x}|S_n)$ is replaced by a new model, which is obtained as follows: First, we iteratively solve

$$\begin{aligned} \min_{\lambda,c,\epsilon} \quad & (-1)^{m_0} \lambda^T \, \Phi(x) \, \lambda \\ \text{subject to:} \quad & \Phi(x)\lambda + Pc - \begin{bmatrix} 0 \\ \epsilon \end{bmatrix} = \begin{bmatrix} \mathbf{y} \\ \underline{y} \end{bmatrix} \\ & P\lambda = 0 \\ & \epsilon \geq 0. \end{aligned}$$

This quadratic convex problem is well-studied and can be solved efficiently by many current methods. When a new surrogate model is constructed, we can find its minimal value and re-check whether that value is at least $\underline{y}$ or not. If it is not the case, then the process is repeated by inserting an additional point and constraint to increase the lower bound of the interpolation model. We continue this procedure until we reach a maximum number of iterations, and then replace $s_n(\overline{x}|S_n)$ by the last interpolation model.

3.3 Kriging

Kriging is a special case of Gaussian processes. The use of kriging in black-box optimization begins with the famous paper by Sacks et al. [55]. In that paper, each deterministic function value $y(x)$ is treated as a realization of a stochastic regression process $Y(x) = \sum_{j=1}^{h} \beta_j f_j(x) + Z(x)$, in which the random process $Z(.)$ is assumed to have mean zero and covariance

$$\text{Cov } (Z(x), Z(s)) = \Phi^2 R(x, s),$$

where Φ^2 is the process variance and $R(x, s)$ is the correlation. For each untried point s, the value of $Y(s)$ is predicted as the best linear unbiased predictor (BLUP). This is equivalent to minimizing the *mean squared error* (MSE) subject to the unbiasedness constraint, i.e.

$$\begin{aligned} \text{minimize}_{\ c(x)} \quad & E[c(x)^T Y_S - Y(x)]^2 \\ \text{subject to:} \quad & E[c(x)^T Y_S] = E[Y(x)]. \end{aligned}$$

By introducing Lagrange multipliers $\lambda(x)$ for the unbiasedness constraint, we can easily solve this problem. Denote by

$$f(x) = \begin{bmatrix} f_1(x) \\ \vdots \\ f_h(x) \end{bmatrix},\ F = \begin{bmatrix} f_1(x_1) & \dots & f_h(x_1) \\ \vdots & \vdots & \vdots \\ f_1(x_n) & \dots & f_h(x_n) \end{bmatrix},\ r(x) = \begin{bmatrix} R(x_1, x) \\ \vdots \\ R(x_n, x) \end{bmatrix},\ R = [R(x_i, x_j)]_{1 \le i,j \le n},$$

then a convenient representation for the mean squared error of a predictor $\hat{y}(s)$ is given by [55]:

$$\text{MSE } (\hat{y}(s)) = \Phi^2 \left[1 - \left(f(x)^T \ , \ r(x)^T \right) \begin{pmatrix} 0 & F^T \\ F & R \end{pmatrix}^{-1} \begin{pmatrix} f(x) \\ r(x) \end{pmatrix} \right] \tag{7}$$

The correlation $R(x, s)$ has to be specified to compute this MSE. In [55], the authors restrict their attention to a special correlation class of the form

$$R(x, s) = \prod_{j=1}^{d} \exp(-\theta_j |x_j - s_j|^{p_j}) = \exp\big(- \sum_{j=1}^{d} \theta_j |x_j - s_j|^{p_j} \big) \tag{8}$$

where $\theta_j > 0$ and $0 < p_j \le 2$. According to [31], these correlations have the property that if $x = s$ then $R(x, s) = 1$. Similarly, when $\|x_i - x_j\| \to \infty$, they tend to zero. The parameter θ_j determines how fast the correlation decreases in the j^{th} coordinate direction. The parameter p_j determines how smooth the function is in the j^{th} coordinate direction.

In [31], a more natural way that leads to kriging is introduced. In that paper, the authors treats $y(x)$ as a realization of a normally distributed random variable $Y(x)$ with mean μ and variance Φ^2. The correlation $R(x, s)$ is specified the same as in equation (8). However, the parameters $\mu, \Phi^2, \theta_j, p_j$ are chosen to maximize the likelihood of the observed data, which is

$$\frac{1}{(2\pi)^{\frac{n}{2}} (\Phi^2)^{\frac{n}{2}} (\det R)^{\frac{1}{2}}} \exp \left[\frac{-(\mathbf{y} - \mathbf{1}\mu)^T R^{-1} (\mathbf{y} - \mathbf{1}\mu)}{2\Phi^2} \right]$$

where $\mathbf{y} = (y_1, y_2, \dots, y_n)^T$. Taking logarithm and using first order conditions, we can estimate $\hat{\mu}$ and $\hat{\Phi}^2$ as functions of $\det R$ and y. To predict the response at an untried point $x^* \in \mathcal{D}$, we choose a predicted value y^* to maximize the *augmented likelihood function.* It is shown to be equivalent to minimizing

$$\begin{pmatrix} \mathbf{y} - \mathbf{1}\hat{\mu} \\ y^* - \hat{\mu} \end{pmatrix}^T \begin{pmatrix} R & \mathbf{r} \\ \mathbf{r}^T & 1 \end{pmatrix}^{-1} \begin{pmatrix} \mathbf{y} - \mathbf{1}\hat{\mu} \\ y^* - \hat{\mu} \end{pmatrix}.$$

By applying the partitioned inverse formula, this function can be written as a quadratic function of y^*. Therefore we obtain a standard formula for the kriging predictor

$$y^* = \hat{\mu} + \mathbf{r}^T R^{-1} (\mathbf{y} - \mathbf{1}\hat{\mu}). \tag{9}$$

3.4 Support Vector Machine

Another surrogate model that is also used for engineering design and black-box optimization is Support Vector Machine (SVM). SVM is the model used very often in supervised learning to help analyzing classification and linear regression problems. Intuitively, it constructs a hyperplane with maximal margin that separates the training data. The use of SVM for black-box optimization is suggested in [46], in which the authors report promising computational results.

In the simplest case, a SVM model is constructed as the following linear regression:

$$\hat{f}(x) = a^T x + b,$$

in which a, b are the parameters that minimize the quadratic lost function

$$\frac{1}{2}\|a\|^2 + C\sum_{i=1}^{n} \|a^T x_i + b - y_i\|_\varepsilon^2$$

where $\|z\|_\varepsilon := \max\{0, |z| - \varepsilon\}$.

In order to fit nonlinear functions, the data $(x_i)_{1\leq i\leq n}$ are embedded into a higher dimensional space by a mapping ϕ (e.g. polynomials, Gaussian RBF, Hyperbolic tangent, Kriging, ...). Then we can rewrite the SVM problem in the following form:

$$\begin{aligned} \min \quad & \frac{1}{2}\|a\|^2 + C\sum_{i=1}^{n}(\mu_i^2 + \xi_i^2) \\ \text{subject to} \quad & a^T\phi(x_i) + b - y_i \leq \varepsilon + \mu_i \\ & y_i - a^T\phi(x_i) - b \leq \varepsilon + \xi_i \\ & \mu_i, \xi_i \geq 0. \end{aligned}$$

By rewriting this problem in the dual form and using what is called the *kernel trick* $K(x_i, x_j) = \phi(x_i)^T\phi(x_j)$, we obtain the following

$$\begin{aligned} \max \quad & \frac{1}{2}\sum_{i,j=1}^{n}(\alpha_i^+ - \alpha_i^-)K(x_i, x_j)(\alpha_j^+ - \alpha_j^-) - \frac{\varepsilon}{2}\sum_{i=1}^{n}(\alpha_i^+ + \alpha_i^-) + \sum_{i=1}^{n} y_i(\alpha_i^+ - \alpha_i^-) \\ \text{subject to} \quad & \sum_{i=1}^{n}(\alpha_i^+ - \alpha_i^-) = 0 \\ & 0 \leq \alpha_i^+, \alpha_i^- \leq \frac{C}{n}. \end{aligned}$$

This is a quadratic problem and can be solved efficient by current methods such as coordinate descent.

3.5 Mixed surrogate models

The models presented above have their own advantages and disadvantages, depending on the specific applications that we consider. A certain model that is a perfect choice for some problems might perform poorly on others. It is natural to ask, given a black-box problem, what is the best surrogate model to use. In [42], the authors study the influence of combining multiple surrogate models on the performance of methods for solving black-box optimization. They define a mixture surrogate model as a convex combination of individual surrogate models, i.e.

$$s_{\text{mix}}(x) = \sum_{r\in M} w_r s_r(x), \quad \text{where } \sum_{r\in M} w_r = 1, \text{ and } w_r \geq 0,$$

in which M is the set of all available surrogate models. For each $r \in M$, the large w_r means that $s_r(x)$ is a good choice as a surrogate model for the specific problem. They propose to apply Dempster-Shafer theory in order to find the most suitable model as well as the best combination of weights. In [43], they examine various surrogate models and their combinations within the so-called SO-M-c and SO-M-s frameworks. A Matlab toolbox called MATSuMoTo [41] is provided so that users can freely choose various models and sampling strategies from its library to test the algorithm. Moreover, this toolbox also enable users to choose experimental design strategies and make use of MATLAB Parallel Computing Toolbox to perform multiple function evaluations in parallel.

4 Surrogate-based subproblems

We have mentioned in the introduction part that, at each iteration of surrogate-based algorithms, we have to find the next candidate point(s) for evaluation. Their selection is crucial. Even if these points are not better than the previous ones, they provide us more information about the black-box function. This information is of great importance since in black-box optimization, all we know is the values at a very limited number of data points. Thus, the more data points we have, the more we know about the function.

Since we want the function value at each candidate to be improved, we have to focus the search around the current best point. But we also want to search in areas that are far from all previous points to ensure that we do not overlook a global solution. The two types of searches above are respectively called local and global ones. In black-box optimization, they are also mentioned as exploitation and exploration. Both these two processes are indispensable: if we spend too much time on exploitation, we might get trapped at a local solution; if we spend too much time on exploration, we might not reach any optimal solution at all, even a local one. The management of the two processes (i.e. how to balance them), is at the center of surrogate-based algorithms.

There are several ideas for the selection of candidate points to achieve this balance. The first idea is to keep exploiting until there is no sign of improvement and then move to a new unexplored area. The second idea is to separate in advance how many points we will use to search locally and globally at each iteration. We can also look for compromises, i.e. the searches that are neither local nor global, but somewhere "in between".

In black-box optimization, new candidate points are often chosen by solving auxiliary subproblems. We define these problems by introducing a merit function and looking for the maximum or minimum of this function. But an auxiliary subproblem defined by a single merit function over the whole domain $\mathcal{D}$ is usually not enough for searching both locally and globally. In the literature, the most common approach is to use not only one but several merit functions and/or adaptively change the domain of consideration (which we will call *search regions*). However, instead of using different types of merit functions, we can use a unique type and parameterize it as $\sigma(x|\lambda)$, in which a small λ correspond to a local search and when λ increases, the searches become more global. In this section, we will introduce various choices of such (parameterized) merit functions and their associated search regions.

4.1 Choices of merit functions

In the following, we will review some popular merit functions. We first consider the class of merit functions that make use of standard error in statistical models. We then consider RBF-based merit functions like *bumpiness* and its variations. Lastly, general merit functions which do not depend on any particular surrogate model are introduced.

- **Surrogate models as merit functions.** The most intuitive way for choosing a new candidate point is to choose it as a minimizer of the current surrogate model. One advantage of this choice is that the model itself is normally relatively simple to optimize, so we can quickly solve auxiliary subproblems. This choice works well if the model is reliable, but it can be very misleading if the model is biased. In fact, this method does not converge to a global solution. Furthermore, if one of the data points is a minimizer of the current surrogate model, the method might converge to a point that is not even a local optimum. In this case, a minor modification has to be made: in order to require the gradient of surrogate model to agree with that of the black-box function, we need to sample in a small neighborhood of the current solution [31, 33].
  Note that with proper choices of search regions, we can still use surrogate models as merit functions to find global solutions. For example, the CORS algorithm in [50] looks for candidate points by

minimizing surrogate models over search regions of the form:

$$D_n := \mathcal{D} \setminus \bigcup_{i=1}^{n} B(x_i, r_i).$$

In that algorithm, $B(x, r) = [x - r, x + r]$ and all radius r_i have the same value $r_i = \beta_n$. Here β_n are determined by cycles of $N + 1$ iterations where each cycle represent a range of values which starts at a large value close to 1 (global search) and gradually decreases to $\beta_n = 0$ (local search). This method is able to find global minimizers since the iterates generated by the algorithm are proved to be dense [50].

- **Statistical lower confidence bound.** Using kriging, we can measure the standard error between the true black-box and the surrogate functions. In particular, we can associated $s_n(x)$ with a statistical lower confidence bound of the form:

$$\text{Lcb}\,(x) := s_n(x) - b\hat{\sigma}\sqrt{MSE(x)}$$

where $\hat{\sigma}$ is the process variance being estimated as in [31] and b is a factor determined by users. This merit function is used in the so-called *Sequential Design for Optimization* (SDO) algorithm [16,17]. The Lcb has the property that it coincides the black-box function at all data points (since the mean squared errors at those points are all equal to zeros); the values of Lcb at other points become smaller and smaller comparing to the values of the surrogate function as the errors increase (since we hope the black-box function values to be small at uncertain points). Therefore, the Lcb function seems to be more promising than the surrogate model itself.
Since Lcb is quite difficult to minimize, in the SDO algorithm, we only compare the values of Lcb among a finite number of *prediction points*. The algorithm proceeds until a (user-specified) maximum number of function evaluations or when we reach the convergence given by $y_{\min} < \min_i \text{Lcb}\,(x_i)$. Here $y_{\min}$ is the current best function value and the minimum is taken over all possible remaining candidate points [16]. The SDO algorithm, however, concentrates too much on local searches and can potentially delete some regions of search space, so it might not be able to find global minimizers [31]. Therefore, if we want to use it we have to appropriately manage the search regions (as mentioned previously).

- **Probability of improvement.** Assume that T is a real number that is strictly smaller than the current best function value. We associate each $x \in \mathcal{D}$ with the probability that the prediction $Y(x)$ of x (which is a normally distributed random variable) is smaller than T. This is called the *probability of improvement* (PI) at x and is formally defined as follows [34, 39, 66]:

$$PI(x) \;:=\; P[Y(x) \le T] \;=\; \Phi\left(\frac{T - \hat{\mu}(x)}{\hat{\sigma}(x)}\right),$$

here Φ is the normal cumulative distribution function. The next candidate point is chosen to maximize this PI function. More specifically, [39] discusses the general Bayesian approach in which the distribution is taken arbitrarily and [66] introduces an axiomatic treatment of the method, which he calls the P-algorithm.

The key advantage of PI over Lcb is that, under certain mild assumptions, the iterates are dense [31]. The reason is that, when more and more points are sampled around the current best point, the standard error $\hat{\sigma}(x)$ gets smaller, therefore $\Phi\big(\frac{T-\hat{\mu}(x)}{\hat{\sigma}(x)}\big)$ will become so small that we have to switch to unexplored regions. However, this PI function is extremely sensitive to the selection of targets T [31]. If T is too close to the current best value $y_{\min}$ (i.e. T is large), the search will be highly local and only focus in a small neighborhood of the current minimizer. Otherwise, if T is too far from $y_{\min}$ (i.e. T is too small), the search will be highly global. Therefore, proper selections of T will help us to balance between local and global searches. A simple but powerful idea is to use different values of T, ranging from large to $-\infty$, and find all candidate points from resulting subproblems.

- **Expected Improvement.** Assume that we use kriging to interpolate the black-box function. For any $x \in \mathcal{D}$, let $Y(x)$ be the prediction of the black-box function value at x. The improvement at x is then defined by $I(x) = \max\big[y_{\min} - Y(x), 0\big]$, where $y_{\min}$ is the current best value. Therefore, the improvement at each point is a nonnegative random variable, thus we can compute its expectation as follows:

$$\begin{aligned}\mathrm{EI}(x) := E(I(x)) &= E\big[\max\left(y_{\min} - Y(x), 0\right)\big] \\ &= [y_{\min} - \hat{\mu}(x)]\Phi\left(\frac{y_{\min} - \hat{\mu}(x)}{\hat{\sigma}(x)}\right) + \hat{\sigma}(x)\phi\left(\frac{y_{\min} - \hat{\mu}(x)}{\hat{\sigma}(x)}\right),\end{aligned}$$

  where Φ and ϕ are the normal cumulative distribution and density functions, respectively. $\mathrm{EI}(x)$ is called the expected improvement at x [31,32]. The next candidate point is chosen to maximize the EI function.
  Expected improvement is used in the EGO algorithm by Jones et al. [32]. In EGO, auxiliary subproblems are solved using a branch-and-bound algorithm. They use the fact that the EI function is monotonic in $\hat{\mu}$ and in $\hat{\sigma}$ to simplify the computation of its lower and upper bounds in each rectangular. Under mild assumptions, the iterates from EGO are dense [63], so it can be used to find a global optimum. However, the method requires fairly exhaustive search around the initial best point before the algorithm begins to search more globally (because we treat the estimated standard error as if it is correct). Therefore EGO runs relatively slowly compared to modern methods.

- **Bumpiness of functions.** The RBF-based algorithm proposed by [24] is one of the most popular methods in black-box optimization. It is based on the observation that, among different surrogate functions to approximate a black-box, in practice, the smoothest one is often the most accurate. Assume that we use certain radial basis functions for interpolation. Given data points $(x_1, y_1), \ldots, (x_n, y_n)$, we have for each additional $(x^*, y^*) \in \mathcal{D} \times \mathbb{R}$ a different radial basis function. The idea is, if we assign y^* to a fixed value T, we will look for x^* such that the resulting function is the "least bumpy". Here T can be regarded as an estimate of the optimal value and we expect that it is attained at x (i.e. $f(x^*) = T$).

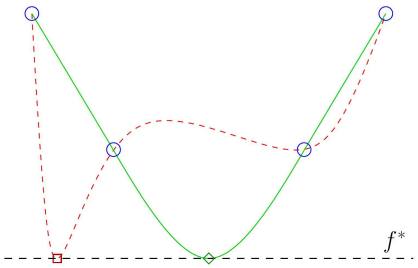

Figure 2: An example of two surrogate models that interpolate 4 data points. The solid-line model is more likely than the dashed-line one since it is less bumpy [15].

For each $w \notin \{x_1, \ldots, x_n\}$, let s_w be the radial basis function defined by the following interpolation conditions:

$$\begin{cases} s_w(x_i) = y_i \text{ for all } i = 1, 2, \ldots, n. \\ s_w(w) = T. \end{cases} \tag{10}$$

Then the next candidate point is selected by

$$x_{n+1} \in \arg\min\big\{\sigma(s_w) \,|\, w \notin \{x_1, \ldots, x_n\}\big\},$$

where σ is a measure of the bumpiness of functions, which is defined by the semi-norm $\sigma(s_w) = \|s_w\| = \sqrt{\langle s_w, s_w \rangle}$ (see Section 3.2). In [24], the bumpiness function $\sigma(s_w)$ is simplified to

$$g_n(w) = (-1)^{m_0}[T - s_n(w)]^2 \mu_n(w),$$

where $\mu_n(w)$ is the coefficient corresponding to variable w of the Lagrangian function L, which satisfies $L(x_i) = 0$, for $i = 1, \ldots, n$ and $L(w) = 1$. We can see that μ_n and g_n are not defined at $x_1, \ldots, x_n$ and $\mu_n(w) = \infty$ if $w \in \{x_1, \ldots, x_n\}$, so there might be computational difficulties if μ_n is evaluated at any points closed to $x_1, \ldots, x_n$. Therefore, instead of minimizing $g_n(w)$, we maximize a differentiable everywhere function $h_n(w)$ which is defined by

$$h_n(w) = \begin{cases} \frac{1}{g_n(w)} & \text{if } w \neq x_1, \ldots, x_n \\ 0 & \text{else.} \end{cases}$$

It is not too difficult to solve this auxiliary subproblem, but we need to be careful when selecting the target T since the problem is very sensitive to the choice of T. If T is large and close to $y_{\min}$, the search is highly local; when T gets smaller, the search will be more global. To handle it, we select T by performing cycles of iterations, in which each cycle starts with a low value (global search) and ends with a high value close to $\min s_n(x)$ (local search). Then we go back to a global search, starting the cycle again [24]. With certain choices of the targets T, the iterates from this algorithm are dense [24].
rbfSolve [9] is a Matlab implementation of the RBF-Gutmann method that uses several strategies for selecting the targets T. The first strategy is to perform a cycle of length $N+1$, starting at some $n = \tilde{n}$ and for $\tilde{n} \leq n \leq \tilde{n} + N - 1$, we choose

$$T = \min_{w \in \mathcal{D}} s_n(w) - W_n \big[\max_{i \in S_n} f(x_i) - \min_{w \in \mathcal{D}} s_n(w) \big],$$

in which $W_n = \left(\frac{N - n + \tilde{n}}{N} \right)^2$ and S_n is form by $\{x_1, \ldots, x_n\}$ after removing $n - n_{max}$ points with largest function values. Note that in a cycle, the values of W_n are decreasing, therefore the corresponding target gradually get closer to the minimum value s_n^* of $s_n(x)$. The second strategy is to choose T as the optimal value of the following auxiliary problem:

$$\begin{aligned} \min_{x \in \mathcal{D}} \quad & T(x) \\ \text{subject to: } & \mu_n(x) \big[s_n(x) - T(x) \big]^2 \leq \alpha_n^2 \end{aligned}$$

and then perform a cycle of length $N+1$ on the choice of α_n (in the paper, the authors use Cholesky factorization to get update for the next interpolations. It helps us to reduce computation time for solving linear system of equations). Another adaptive choice of the targets T is introduced in the ARBF method [25], in which a large set of targets T_j are used:

$$T_j = s_n^{min} - \beta \, . W_j \, . f_\triangle,$$

where β is an adaptive factor and the range $f_\triangle$ is selected as in [25]. If for different target values, we obtain optimal solutions that are very close to each other, we need to increase β. On the other extreme, if the optimal solution regarding the second target is too far from the one obtained with the first target, we need to decrease β because it seems like the target values are too spread out.

- **Metric Response Surface (MRS) weighted score.** The *Metric Response Surface* (MRS) weighted score is a merit function proposed by Regis et al. [53]. In order to find the next candidate for evaluation, the authors do not minimize (or maximize) that merit function over a continuous set. Instead, they select the best among a finite set of N randomly generated points. Merit functions are made up from two criteria: the value of the surrogate model at each point and its minimum distance to existing data points. Explicitly, let Ω_n be the set of N randomly generated points at the iteration n. Let $s_n^{\max}$, $\triangle_n^{\max}$ (corr. $s_n^{\min}$, $\triangle_n^{\min}$) be the maximum (corr. minimum) values of $s_n(x)$ and $\triangle_n(x)$ over Ω_n, where

$$\triangle_n(x) := \min_{1 \leq i \leq n} d(x, x_i).$$

For each $x \in \Omega_n$, we identify two scores

$$V_n^R(x) = \begin{cases} \frac{s_n(x)-s_n^{\min}}{s_n^{\max}-s_n^{\min}} & \text{if } s_n^{\max} \neq s_n^{\min} \\ 1 & \text{otherwise} \end{cases}$$

and

$$V_n^D(x) = \begin{cases} \frac{\triangle_n^{\max}-\triangle_n(x)}{\triangle_n^{\max}-\triangle_n^{\min}} & \text{if } \triangle_n^{\max} \neq \triangle_n^{\min} \\ 1 & \text{otherwise.} \end{cases}$$

The MRS weighted-score merit function is then defined by:

$$W_n^\lambda(x) := \lambda V_n^R(x) + (1-\lambda)V_n^D(x),$$

where $\lambda_n \in [0,1]$ is the weighted score specified by the user. The new candidate point will be found by minimizing W_n^λ. Note that when $\lambda_n = 1$, the subproblem is equivalent to minimizing $s_n(x)$ (a local search). On the other hand, when $\lambda_n = 0$, the subproblem is equivalent to maximizing $\triangle_n(x)$, i.e. to force x to stay away from all existing data points (a global search). Different choices of λ_n will help us to manage the balance of local and global searches. Under mild assumptions, the MSRS method is shown to convergent almost surely.
In [53], two specific implementations are discussed: *Global Metric Stochastic Response Surface* (GMSRS) and *Local Metric Stochastic Response Surface* (LMSRS). In GMSRS, the set Ω_n of test points are uniformly generated throughout $\mathcal{D}$. To achieve a balance between global and local search, the weight λ_n is allowed to cycle through the iterations, starting from a high value (global search) and ending with a low value (local search). In LMSRS, the set of test points is generated by adding random perturbations to the current best solution x_n^*. These perturbations are chosen to be normally distributed with zero mean and with covariance matrix $\sigma_n^2 I_d$, in which σ_n is the step size defined by the user. Global search can be done by restarting the algorithm whenever it appears to have converged to a local solution, i.e. the number of consecutively failed attempts is large enough.

An extension of the LMSRS method when some of the constraints are black-boxes is given in [49]. In this case, the authors use multiple surrogate models to approximate the objective and expensive constraint functions. Starting with a design having at least one feasible point, we then perform the following loop: First, we construct surrogate models for all these black-box functions. Next, test points are generated randomly by perturbing the current best feasible point. These test points are then filtered so that they consist of only the elements that do not violate (or violate the minimum number of) surrogate-constraints. The next candidate point is selected by minimizing the MRS weighted-score function. During the process, we continuously update counters for consecutive failures and successes. The purpose is to check whether we are trapped at a local minimum or not, so we can appropriately adjust the step-size σ_n (see LMSRS method) to switch back and forth between local and global search.

- **Weighted Model Uncertainty.** In the *qualSolve* algorithm, proposed by Jakobsson et al. [28], the strategy for selecting new points for evaluation is to minimize the total model uncertainty weighted against the surrogate function value. Similar to [50, 53], the measure of uncertainty at point x is defined as

$$U_{S_n}(x) := \min_{x_i \in S_n} d(x, x_i).$$

Let $\omega(S(x))$ be a weight function which gives a point x a low weight value when $S(x)$ is high and gives x a high weight value when $S(x)$ is low. Then we consider the following *quality function*

$$Q(y) := \int_{\mathcal{D}} (U_{S_n}(x) - U_{S_n \cup \{y\}}(x))\omega(S(x))dV(x).$$

The point x_{n+1} to evaluate next is obtained by solving

$$x_{n+1} \in \arg\max_{y \in \mathcal{D}} Q(y).$$

The weight function ω should satisfy the following demands

– no area are completely disregarded, i.e. $\omega(z) > 0$ for all $z \in \mathbb{R}$.
– areas with low surrogate function values are weighted higher than areas with high surrogate function values, i.e ω is strictly decreasing.

In [28], the following weight function ω is proposed

$$\omega(x) = \gamma \exp\big(-\frac{z - s_n^{min}}{s_n^{max} - s_n^{min}}\big),$$

where $\gamma \geq 1$ is the parameter which controls the balance between local and global search.

4.2 Choices of search regions

Search regions are the domains where we solve auxiliary subproblems, i.e. to minimize or maximize the merit functions. In many cases, they are simply chosen to be the whole design space $\mathcal{D}$. However, proper choices of search regions can also help us to manage the balance between local and global searches.

There are several ways to select a search region. For example, we can choose it as a neighborhood of some promising point (e.g. a minimizer of the surrogate function, the current best point). Normally, we use closed balls (with l_1 norm, for simplicity) to define a neighborhood. In [51], the authors realize that the original implementation of RBF-Gutmann algorithm can sometimes converge slowly to the global minimum on some test problems due to its failure to perform local search. This is because the candidate point for evaluation might be far from previous evaluated points and the global minimizers of $s_n(x)$. Another situation might cause the failure is when a global minimizer of $s_n(x)$ coincides or is too close to one of previously evaluated points. When the above cases occur, the strategy of choosing small W_n in [24] does not guarantee local search. Therefore, they suggest that, whenever W_n is small, we restrict the global minimization of the bumpiness function to a small hyper-rectangle centered at a global minimizer of $s_n(x)$, i.e. we solve

$$\min_{[x_n^* - r_n, x_n^* + r_n] \cap \mathcal{D}} B_n(x),$$

where x_n^* is a global minimizer of $s_n(x)$, $B_n(x)$ is the bumpiness function. Radius r_n is an increasing function of the parameter W_n, which tends to 0 when W_n is small and vice versa. In particular, they use the following choice for r_n

$$r_n = \begin{cases} v\sqrt{W_n}(b-a) & \text{if } 0 \leq W_n \leq U \\ b - a & \text{otherwise,} \end{cases}$$

where $0 < v < 1$ and $U > 0$ are parameters to be specified and a, b are given from the definition of design space $\mathcal{D} = \{x \in \mathbb{R}^n |\ a \leq x \leq b\}$.

We can also define a search region of certain distance from existing data points. The idea is to push the next candidate point away from the points that have already been evaluated, thus enable us to move to unexplored regions. In this case, the search regions can be selected as

$$D_n := \mathcal{D} \setminus \bigcup_{i=1}^{n} B(x_i, r_i)$$

One of the examples, the CORS algorithm, proposed by Regis and Schoemaker [50], uses these search regions. Let S_n be the set of all existing data points, then the next candidate point x_{n+1} is chosen to minimizes the surrogate model over the set

$$\Big\{x \in D,\ \|x - S_n\| \geq \beta_n \triangle_n\Big\},$$

where β_n is decided by the user and $\triangle_n$ is defined by

$$\triangle_n = \max_{x \in D} \|x - S_n\|.$$

The parameters β_n are set by performing cycles of $N+1$ iterations where each cycle employs a range of values for β_n, starting with a high value close to 1 (global search) and ending with $\beta_n = 0$ (local search). The iterates generated by the algorithm are dense therefore the method are able to find a global solution.

4.3 Solving black-box optimization problems

We know that by minimizing (or maximizing) a merit function over search regions, we will obtain next candidate points for evaluation. In the previous subsections, we have introduced various choices for such merit functions and search regions. Now we will discuss how to solve these auxiliary subproblems.

In principle, these auxiliary subproblems can be written explicitly as NLPs, therefore we can solve them by a generic solver. Moreover, we can exploit derivative information, which is available in most of cases, to make use of classical methods. However, due to the high nonlinearity and multi-modality of some merit functions (e.g. Expected Improvement), these subproblems are sometimes still difficult to solve. Moreover, it is questionable whether we should solve these problems to optimality or only approximate solutions are sufficient. Merit functions are constructed based on approximate models of the true function, which might be inaccurate. For this reason, it is risky if we spend too much time to solve the auxiliary subproblems, since in many cases, it is the quality of the model that matters. Therefore, approximate solutions (even solutions that are *fairly good*) are sufficient.

Taking advantages of the fact that merit functions are given explicitly and very cheap to evaluate, we can use heuristics like genetic algorithms, simulated annealing and so on, to solve them. The simplest way is to sample a large number of points in the search region, and select the point with the best merit function values. This idea is implemented in [53], in which test points are randomly generated using either uniform and normal distributions and the MRS weighted score is used as the merit function. Under mild assumptions, the corresponding method converges to a global minimum *almost surely*, provided that the algorithm is allowed to run indefinitely.

In many real-life problems, the black-box function can be affected by noises. This makes the surrogate models far from accurate and it may lead to many difficulties for finding optimal solutions. Therefore, we have to adjust surrogate models to deal with the present of noises before going further. In [28], a surrogate model that deviates from the data points is chosen by solving the following subproblem:

$$\begin{aligned} \min \quad & \mu\|S\|^2 + (1-\mu)\|e\|_2^2 \\ s.t \;\; & S(x_i) = f_i + e_i \\ & e_i \in \mathbb{R}^n. \end{aligned}$$

This problem is formed by adding a penalty term to the objective. In [28], the authors show that the optimal error term e can be obtained by solving an explicit linear system:

$$\left(\frac{\mu-1}{\mu} I^{n\times n} - B^T A B\right) e = B^T A B \tilde{f}.$$

A similar approach to deal with noise is introduced in [15]. In this paper, the range within which function values are allowed to vary is required to specify. Assume that in addition to the black-box $f(x)$ we have access to $\hat{f}(x)$ such that $f(x) = \hat{f}(x)(1+\epsilon_r)+\epsilon_a$, where ϵ_r, ϵ_a are random variables with bounded support and unknown distribution. To determine the surrogate model, the authors introduce a vector of slack variables $\xi \in \mathbb{R}^k$ and solve the problem:

$$\begin{aligned} \min \quad & (-1)^{d_{\min}+1}\, \lambda^T \Phi \lambda \\ s.t \;\; & \Phi\lambda + Ph + \xi = F \\ & P^T\lambda = 0 \\ & -\epsilon_r|\hat{f}(x_i)| - \epsilon_a \le \epsilon_i \le \epsilon_r|\hat{f}(x_i)| + \epsilon_a \text{ for all } i \in L \\ & \epsilon_i = 0 \text{ for all } i \notin L. \end{aligned}$$

There are also cases when there is no noise in the black-box, but some points might have much larger function values comparing to others. In these cases, the surrogate model tends to oscillate strongly. It might also happen that min $s_n(y)$ is much lower than the best known function value, which leads to a choice of control parameters that overemphasizes global search. To handle these problems, it is suggested that we replace large function values at each iteration by the median of all computed function values [9] (see more discussion about it in [24, 29]).

Since black-box functions are often expensive to evaluate, we can use parallel computation to speed up the algorithms. The idea of parallel computing is to break the main computation task into independent tasks so that each can be executed in a separate worker computer (or processing unit) simultaneously with the others. In black-box optimization, it is relatively easy to design a parallel algorithm. At each iteration, we find different candidate points (by solving multiple auxiliary subproblems) and assign each point to one worker computer for evaluation. In [51], parallel versions of RBF-Gutmann and CORS-RBF methods are implemented. While in RBF-Gutmann, the set of candidate points are obtained by solving subproblems for different values of the targets T_n, in CORS-RBF, they are obtained by choosing different factors β_n. All these subproblems are solved by the master computer, however we can modify the algorithm to empower worker computers to do that. It is easy to notice that parallel computation can be implemented for many other surrogate-based methods in the same way. For example, different auxiliary subproblems can be found by choosing different values for the radius r_n (in RBF-CG method), the score λ_n (in MSRS method), the control parameter γ (in qualSolve) and so on.

4.4 Dealing with discrete variables

In many applications, we need to deal with integer and binary variables, in addition to continuous ones. It is suggested that we can modify the surrogate-based methods to address these cases. In [45], a method called SO-MI for solving black-box optimization problems with mixed integer variables is proposed. It starts with some feasible point and try to iteratively improve it. In order to do that, four potential new points are chosen for expensive black-box evaluations. They are selected from 4 different groups of points which are generated by perturbing the current solutions in different ways (either continuous or discrete variables or both) or by sampling at unexplored regions. By constructing RBF surrogate models, we determine the best point from each group with respect to the MRS weighted scores. A similar method called SO-I is developed in [44] to deal with pure integer problem. Note that both of them start with a symmetric Latin hypercube design and round to the nearest integers. Moreover, they can handle black-box constraints by incorporating them into the objective functions under penalty terms. The starting feasible point, in those cases, are found by iteratively solving some auxiliary problems.

5 Conclusions

In this paper, we present surrogate-based methods for solving black-box optimization problems. In these methods, the *key to success* is the ability to construct good approximations for the black-box functions. As long as we have a good approximation of the objective, in principle, it is not so difficult to find global minimal solutions. It is because approximate models are often quite simple and easy to work with. Moreover, even if they are analytically complicated, they are much faster to evaluate and we can apply heuristics such as genetic algorithms to solve.

However, it is often impractical to expect that we can accurately approximate the black-box function everywhere in the domain: such an approximation requires us to evaluate of a huge number of data points, which is extremely expensive. In fact, for our purpose, we only need to find the models that approximate the black-box fairly well in the neighborhoods of (local or global) optima. Therefore, we should concentrate more on areas where optima are likely to locate. This observation is fundamental to all the black-box algorithms we introduced in the previous sections. A good model in this case needs not to be a good approximation everywhere but has to fit the black-box function well in the areas that

are intensively evaluated. This is very intuitive since the more we explore in one area, the better we understand the function there. Kriging and radial basis functions are two such models that are very popular in the literature.

Since it is unknown where optima are located, we will look at the two types of potential areas: (i) neighborhoods of the best data points so far and (ii) areas that are least explored. Surrogate-based methods work by iteratively updating data points from one of these such areas. One idea to do that is to construct merit functions so that their optima are either close to the best current points or not explored (or both). Many such merit functions have been introduced in the previous sections.

Another idea is to properly manage the search regions, i.e. areas where we look for candidate points. Some authors have combined the both two ideas in their algorithms, but a systematic test of all combinations (models + merit functions + search regions) has not yet implemented. It might be a good idea to do such a test to compare all current black-box algorithms.

Another possible research direction is to devise parallel implementations for black-box optimization algorithms. This is already done in [52] for RBF with relatively simple ideas. Modifications of this method to apply for other merit functions are possible. Moreover, hopefully more complicated designs, in which the relations between candidate points found by different worker computers are taken into consideration, will speed up the algorithms.

Acknowledgements

The first author of this work is supported by a Microsoft Research PhD Fellowship. The second and fourth authors are partly supported by the FP7 Marie-Curie Initial Training Network "MINO". We thank an anonymous referee for many useful suggestion.

References

[1] M.A. Abramson, C. Audet, G. Couture, J.E. Dennis, S. Le Digabel, and C. Tribes. The NOMAD project. Software available at `https://www.gerad.ca/nomad/`.

[2] T. Akhtar and C.A. Shoemaker. Multi objective optimization of computationally expensive multimodal functions with rbf surrogates and multi-rule selection. *To appear in Journal of Global Optimization*.

[3] C. Audet. A survey on direct search methods for blackbox optimization and their applications. In Panos M. Pardalos and Themistocles M. Rassias, editors, *Mathematics Without Boundaries*, pages 31–56. Springer New York, 2014.

[4] C. Audet and J.E. Dennis. A pattern search filter method for nonlinear programming without derivatives. *SIAM Journal on Optimization*, 14(4):980–1010, 2004.

[5] C. Audet and J.E. Dennis. A progressive barrier for derivative-free nonlinear programming. *SIAM Journal on Optimization*, 20(1):445–472, 2009.

[6] C Audet and J.E. Dennis. Mesh adaptive direct search algorithms for constrained optimization. *SIAM Journal on Optimization*, 17:188–217, 2006.

[7] P. Audze and V. Eglais. New approach for planning out of experiments. *Problems of dynamics and strengths*, 35:104–107, 1977.

[8] S. Bates, J. Sienz, and V.V. Toropov. Formulation of the optimal latin hypercube design of experiments using a permutation genetic algorithm. *5th AIAA/ASME/ASCE/AHS/ASC structures, structural dynamics and materials conference*, pages 1–7, 2004.

[9] M. Björkman and K. Holmström. Global optimization of costly nonconvex functions using radial basis functions. *Optimization and Engineering*, 1:373–397, 2000.

[10] A.J. Booker, J.E. Dennis, P.D. Frank, D.B. Serafini, V. Torczon, and M.W. Trosset. A rigorous framework for optimization of expensive functions by surrogates. *Structural optimization*, 17(1):1–13.

[11] Z. Bouzarkouna, D.Y. Ding, and A. Auger. Using evolution strategy with meta-models for well placement optimization. *Proceedings of the XII European Conference on the Mathematics of Oil Recovery*, 2010.

[12] A. Cassioli and F. Schoen. Global optimization of expensive black box problems with a known lower bound. *Journal of Global Optimization*, 57:177–190, 2013.

[13] A.R. Conn and S. Le Digabel. Use of quadratic models with mesh-adaptive direct search for constrained black box optimization. *Optimization Methods and Software*, 28(1):139–158, 2013.

[14] A.R. Conn, K. Scheinberg, and L.N. Vicente. *Introduction to derivative-free optimization*. SIAM, 2009. MPS/SIAM Series on Optimization.

[15] A. Costa and G. Nannicini. RBFOPT: an open-source library for black-box optimization with costly function evaluations. *Optimization Online*, paper 4538, 2014. Under review.

[16] D.D. Cox and S. John. SDO: A statistical method for global optimization. *Multidisciplinary Design Optimization: State of the Art*, SIAM, Philadelphia:315–329, 1997.

[17] D.D. Cox and S. John. A statistical method for global optimization. *Proceedings of the IEEE international conference on Systems, Man and Cybernetics*, 2:1241–1246, Oct 18-21, 1992.

[18] K.T. Fang, C.X. Ma, and P. Winker. Centered ℓ_2-discrepancy of random sampling and latin hypercube design, and construction of uniform designs. *Mathematics of Computation*, 71:275–296, 2002.

[19] A. Fialho, Y. Hamadi, and M. Schoenauer. A multi-objective approach to balance buildings construction cost and energy efficiency. In *ECAI'12*, pages 961–966, 2012.

[20] A.I.J. Forrester, A. Sóbester, and A.J. Keane *Engineering Design via Surrogate Modelling, a practical guide*. A John Wiley and Sons, 2008. MPS/SIAM Series on Optimization.

[21] B. Franlab. Pumaflow reservoir simulator reference manual. 2012.

[22] M. Gasca and T. Sauer. Polynomial interpolation in several variables. *Advances in Computational Mathematics*, 12:377–410, 2000.

[23] A. Grosso, A. Jamali, and M. Locatelli. Finding maximin latin hypercube designs by iterated local search heuristics. *European Journal of Operational Research*, 197:541–547, 2009.

[24] H.M. Gutmann. A radial basis function method for global optimization. *Journal of Global Optimization*, 19:201–227, 2001.

[25] K. Holmström. An adaptive radial basis algorithm (arbf) for expensive black-box global optimization. *Journal of Global Optimization*, 41(3):447–464, 2007.

[26] M.F. Hussain, R.R. Barton, and S.B. Joshi. Metamodeling: Radial basis functions versus polynomials. *European Journal of Operations Research*, 138:142–154, 2002.

[27] B.G. Husslage, G. Rennen, E.R. van Dam, and D.D. Hertog. Space-filling latin hypercube designs for computer experiments. *Optimization and Engineering*, 12:611–630, 2011.

[28] S. Jakobsson, M. Patriksson, J. Rudholm, and A. Wojciechowski. A method for simulation based optimization using radial basis functions. *Optimization and Engineering*, 11:501–532, 2010.

[29] Y. Ji, S. Kim, and W.X. Lu. A new framework for combining global and local methods in black box optimization. *Optimization Online*, paper 3977, 2013.

[30] M.E. Johnson, L.M. Moore, and D. Ylvisaker. Minimax and maximin distance designs. *Journal of Statistical Planning and Inference*, 26:131–148, 1990.

[31] D.R. Jones. A taxonomy of global optimization methods based on response surfaces. *Journal of Global Optimization*, 21:345–383, 2001.

[32] D.R. Jones, M. Schonlau, and W.J. Welch. Efficient global optimization of expensive black-box functions. *Journal of Global Optimization*, 13:445–492, 1998.

[33] J. Koehler and A. Owen. Computer experiments. *In S. Ghosh and C.R. Rao, editors, Handbook of Statistics, 13: Design and Analysis of Experiments*, pages 261–308, 1996.

[34] H.J. Kushner. A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise. *Journal of Basic Engineering*, 86:97–106, 1964.

[35] S. Le Digabel. Algorithm 909: NOMAD: Nonlinear optimization with the MADS algorithm. *ACM Transactions on Mathematical Software*, 37(4):1–15, 2011.

[36] S. Le Digabel. Efficient global optimization algorithm assisted by multiple surrogate techniques. *Journal of Global Optimization*, 56(2):669–689, 2013.

[37] C. Lizon, C. D'Ambrosio, L. Liberti, M. Le Ravalec, and D. Sinoquet. A mixed-integer nonlinear optimization approach for well placement and geometry. *Proceedings of the XIV European Conference on the Mathematics of Oil Recovery*, 2014.

[38] M.D. McKay, R.J. Beckman, and W.J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21:239–245, 1979.

[39] J. Mockus. Application of bayesian approach to numerical methods of global and stochastic optimization. *Journal of Global Optimization*, 4:347–365, 1994.

[40] M.D. Morris and T.J. Mitchell. Exploratory designs for computational experiment. *Journal of Statistical Planning and Inference*, 43:381–402, 1995.

[41] J. Mueller. Matsumoto: The matlab surrogate model toolbox for computationally expensive black-box global optimization problems. *arXiv preprint arXiv:1404.4261*, 2014.

[42] J. Müller and R. Piché. Mixture surrogate models based on dempster-shafer theory for global optimization problems. *Journal of Global Optimization*, 51(1):79–104, 2011.

[43] J. Müller and C.A. Shoemaker. Influence of ensemble surrogate models and sampling strategy on the solution quality of algorithms for computationally expensive black-box global optimization problems. *J. of Global Optimization*, 60(2):123–144, October 2014.

[44] J. Müller, C.A. Shoemaker, and R. Piché. So-i: a surrogate model algorithm for expensive nonlinear integer programming problems including global optimization applications. *Journal of Global Optimization*, 59(4):865–889, 2014.

[45] J. Müller, C.A. Shoemaker, and R. Piché. So-mi: A surrogate model algorithm for computationally expensive nonlinear mixed-integer black-box global optimization problems. *Computers & Operations Research*, 40(5):1383–1400, 2013.

[46] H. Nakayama, M. Arakawa, and K. Washino. Using support vector machines in optimization for black-box objective functions. In *Neural Networks, 2003. Proceedings of the International Joint Conference on*, volume 2, pages 1617–1622 vol.2, July 2003.

[47] J.A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, 7:308–313, 1965.

[48] J.E. Onwunalu and L.J. Durlofsky. Application of a p article swarm optimization algorithm for determining optimum well location and type. *Computational Geosciences*, 14:183–198, 2010.

[49] R.G. Regis. Stochastic radial basis function algorithms for large-scale optimization involving expensive black-box objective and constraint functions. *Computers and Operations Research*, 38:837–853, 2011.

[50] R.G. Regis and C.A. Shoemaker. Constrained global optimization of expensive black-box functions using radial basis functions. *Journal of Global Optimization*, 31:153–171, 2005.

[51] R.G. Regis and C.A. Shoemaker. Improved strategies for radial basis function methods for global optimization. *Journal of Global Optimization*, 37:113–135, 2007.

[52] R.G. Regis and C.A. Shoemaker. Parallel radial basis function methods for the global optimization of expensive functions. *European Journal of Operational Research*, 182:514–535, 2007.

[53] R.G. Regis and C.A. Shoemaker. A stochastic radial basis function method for the global optimization of expensive functions. *INFORMS Journal on Computing*, 19:497–509, 2007.

[54] R.G. Regis and C.A. Shoemaker. A quasi-multistart framework for global optimization of expensive functions using response surface models. *Journal of Global Optimization*, 56:1719–1753, 2013.

[55] J. Sacks, W.J. Welch, T.J. Mitchell, and H.P. Wynn. Design and analysis of computer experiments. *Statistical Science*, 4:409–423, 1989.

[56] T.J. Santner, B.J. Williams, and W.I. Notz. *The Design and Analysis of Computer Experiments*. Springer, 2003.

[57] M.C. Shewry and H.P. Wynn. Maximum entropy sampling. *Journal of Applied Statistics*, 14:165–170, 1987.

[58] A. Sóbester, A.I.J. Forrester, D.J.J. Toal, E. Tresidder, and S. Tucker. Engineering design applications of surrogate-assisted optimization techniques. *Optimization and Engineering*, 15(1):243–265, 2012.

[59] E. Stinstra, P. Stehouwer, D. den Hertog, and A. Vestjens. Constrained maximin designs for computer experiments. *Technometrics*, 45:340–346, 2003.

[60] R. Stocki. A method to improve design reliability using optimal latin hypercube sampling. *Computer Assisted Mechanics and Engineering Sciences*, 12:393–412, 2005.

[61] B. Talgorn, S. Le Digabel, and M. Kokkolaras. Statistical surrogate formulations for simulation-based design optimization. *Journal of Mechanical Design*, 137(2):1–18, 2015.

[62] V. Torczon. On the convergence of pattern search algorithms. *SIAM Journal on Optimization*, 7:1–25, 1997.

[63] E.L. Vazquez and J. Bect. Convergence properties of the expected improvement algorithm with fixed mean and covariance functions. *Journal of Statistical Planning and Inference*, 140:3088–3095, 2010.

[64] F. Viana, R. Haftka, and V. Steffen. Multiple surrogates: how cross-validation errors can help us to obtain the best predictor. *Structural and Multidisciplinary Optimization*, 39(4):439–457, 2009.

[65] K.Q. Ye, W. Li, and A. Sudjianto. Algorithmic construction of optimal symmetric latin hypercube designs. *Journal of Statistical Planning and Inference*, 90:145–159, 2000.

[66] A. Žilinskas. A review of statistical models for global optimization. *Journal of Global Optimization*, 2:145–153, 1992.