Self-adjusting surrogate-assisted optimization techniques for expensive constrained black box problems

Bagheri, S.

Chapter 2

Black Box Optimization Methods

2.1 Is There Any Free Lunch in Optimization?

A function is called black box if no assumption about its type (linear, quadratic, ...) can be made due to the lack of any prior knowledge. Optimization tasks in real-world applications are often black box as very little to no information about their objective function is available. The value of a black box function at an arbitrary point can be measured through an evaluation procedure in its search space. This evaluation procedure can be costly in terms of time and expenses which means a very limited number of points can be evaluated in practice. When designing black box optimizers often it is assumed that the objective function has a structure although unknown, since fully random functions are not realistic examples of problems in real-world, therefore tackling them is not of our interests.

Although numerous derivative-free algorithms are developed to solve black box problems, searching for a universal optimizer which performs best for all possible black box problems continues. One might question the existence of such universal algorithm thinking of the famous no free lunch theorems (NFL) for optimization, introduced by Wolpert and Macready in 1997 [181]. These theorems prove that any two optimizers perform the same if their performance are averaged over all possible combinatorial problems in finite discrete domain. For very long time since the introduction of the no free lunch theorems not much effort was devoted to extend these theorems for continuous domain. This lack of interest was perhaps due to the assumption that any continuous optimization problem being optimized with a computer practically has still a discrete domain. However, Droste et al. in 1999 [49] and Igel et al. [86] in 2005 show that NFL is not relevant for real-world problems. In 2010 Auger and Teytaud [10] show that NFL does not hold true for optimization problems in continuous domains. Additionally, the authors bring some arguments for the optimality of surrogate-assisted and estimation of distribution algorithms.

Corn and Knowles in [41] also claim that there are free lunches in multi-objective optimization.

Despite the fact that no universal optimizer is designed yet, many strong optimization frameworks are developed which clearly outperform an exhaustive random search. We will briefly introduce and categorize a handful of unconstrained, constrained and surrogate-assisted optimizers in the following sections.

2.2 Unconstrained Optimization

The main focus of this dissertation is about efficient black box constrained optimizers. However, many constrained optimizers are built on top of an unconstrained optimizer in one or other way, therefore, it is important to give a brief overview about the existing unconstrained optimizers. A black box optimization problem can be simply defined as minimization of an unknown objective function f defined in a d dimensional parameter space:

$$\text{Minimize} \qquad f(\vec{x}), \qquad \vec{x} \in [\vec{l}, \vec{u}] \subset \mathbb{R}^d, \tag{2.1}$$

where $\vec{l}$ is the lower bound of the search space $\mathbb{S} \subseteq \mathbb{R}^d$ and the $\vec{u}$ is the upper bound. Some literature refer to Eq. (2.1) as optimization with bound or box constraints if the lower and upper bounds have finite values. Vector $\vec{x} = [x_1, x_2, \cdots, x_d]$ has a length of d. The goal is to find $\vec{x}^*$ which minimizes the fitness function f in the search space $\mathbb{S}$. Maximization problems can be transformed to minimization by negating the function without loss of generality.

In absence of the derivative information in black box optimization problems, the gradient based techniques lose their functionality. Gradient based optimization algorithms including gradient descent, Newton method, conjugate gradient, etc. require first or even second order derivatives. In case of black box optimization these derivatives cannot be directly measured. Also it is often too expensive or even impossible to approximate these derivatives. Derivative-free algorithms, assuming that black box functions have an unknown and complex structure, try to approach the optimal solution by learning about the hidden structure of the function iteratively, in one or another way.

In order to gain some overview about the existing numerous derivative-free algorithms, we try to categorize them by a few of their features and briefly discuss them in this section. However, a detailed review about such algorithms can be found in [147]. Black box optimizers can be *deterministic* or have some randomness and be

stochastic. A large group of stochastic heuristics, categorized as *nature-inspired*, are motivated by a variety of natural processes. Also, we can categorize all derivative-free algorithms under *population based* or *point based* methods. Expensive optimization problems are often addressed by *surrogate-assisted* solvers. *Estimation of distribution algorithms* aka model-based optimizers are a group of optimizers which aim to model a distribution and learn about the problem iteratively. This class of optimizers have shown promising contributions solving black-box problems. Fig. 2.1 shows a taxonomy of derivative free unconstrained optimization algorithms with examples and references for each category.

Deterministic optimizers are a class of optimizers which, starting with a fixed initial configuration, will always generate the same solution after the same amount of iterations. No sort of randomness appears in these algorithms except for some of them that need to be randomly initialized by a starting point or a population of points. In practice often one or several initial solutions are known and the goal is to improve the existing solution(s), so there is no need for a random start. The taxonomy in Fig. 2.1 lists several well-known deterministic optimizers including direct search aka pattern search [24], Lipschitzian [163], BOBYQA [130], etc.

Hooke & Jeeves [83], being a special type of pattern search with one evaluation per iteration and Nelder-Mead [122] based on the simplex method are some examples of very strong deterministic solvers for nonlinear problems, although they often face difficulties as the parameter space grows. The convergence conditions of this class of solvers are studied in [174, 105]. Several deterministic black box optimizers are surveyed and compared in [104].

Stochastic optimizers refer to a large category of optimization techniques benefiting from randomness in one or another way. Many of the recently developed modern derivative-free solvers employ randomness directly or indirectly as an exploration tool to avoid getting stuck in local optima. Simulated annealing (SA) [96] is an example for stochastic optimization which is well suited for global optimization of multimodal problems. A great number of the stochastic optimization algorithms can be categorized as **nature-inspired** methods, imitating various processes in nature like evolution, chemical reactions, social behavior in animals, neural networks, etc. [184, 59]. Evolutionary algorithms including evolution strategy, genetic algorithm, etc. are among very successful nature-inspired optimizers which are widely used in many different areas [12, 11, 13].

In general all the sequential derivative-free optimizers fall into two categories of **population based** and **point based** algorithms. The former refers to the algorithms which require evaluation of a population of solutions in each iteration, their performance is designed according to the statistical properties or the interaction

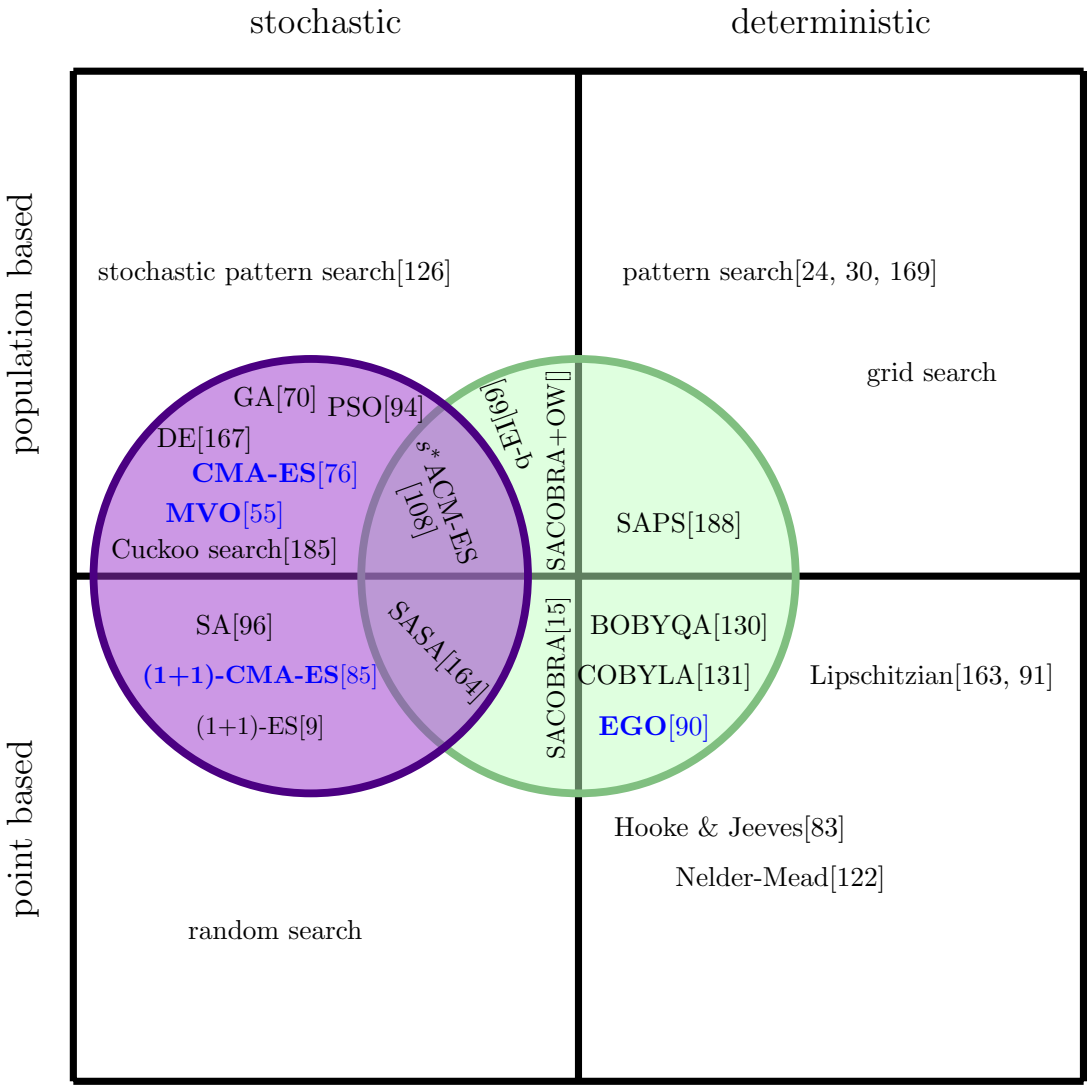

Figure 2.1: Taxonomy of derivative-free unconstrained optimization techniques for black-box optimization problems. The green circle groups the surrogate-assisted techniques together and the purple circle belongs to the nature-inspired algorithms. Algorithms written in blue are from the EDA optimization class.

of a population of solutions, e.g., differential evolution (DE) [167], evolution strategy [13, 160]. On the contrary, **point based** algorithms evaluate one solution at a time. Examples for such algorithms are the stochastic (1+1)-ES [11, 9] and the deterministic Nelder-Mead [122]. The population based methods often demand too many function evaluations. This makes them sometimes unaffordable for expensive real-world optimization problems. However, these algorithms can benefit from a suitable parallelization approach [35, 69, 68].

The optimizers which make use of any mathematical modeling technique (surrogate models) to assist the optimization procedure and save some function evaluations, belong to the category of **surrogate-assisted** optimizers. Real-world optimization tasks which are black box and expensive to evaluate are the main motivation for the development of many surrogate-assisted optimization techniques in recent years. As in practice it is an absolute necessity to be thrifty with the amount of function evaluations, surrogates are used to model the hidden structure behind the black box objective functions in order to reduce the number of real function calls by as many as possible. A wide range of modeling techniques are used to solve expensive optimization problems efficiently, e.g., linear local models in Cobyla [131], quadratic modeling in BOBYQA [130], radial basis function interpolation in SACOBRA [19], probabilistic modeling in EGO [90], random forest in [21], recurrent neural networks in [36, 37], etc. Many surrogate-assisted solvers employ a regression technique as a surface fitting approach to replace the objective function, but surrogate-assisted optimization is not in general limited to regression techniques. As the focus of this dissertation is on solving expensive constrained optimization problems, surrogate-assisted constrained optimization is discussed in more details in Sec. 2.4.

Estimation of Distribution Algorithms (EDAs) aim at estimating a distribution of solutions which are likely to improve the current solutions [106]. This estimated distribution is often updated iteratively. Covariance matrix adaptation evolution strategy (CMA-ES) [76] and Mean Variance optimization (MVO) [55] are examples for EDAs which show very strong performance on a large set of benchmarks [75, 55].

2.3 Constraint Handling Techniques

Real-world optimization problems can be more demanding than simply minimizing a function as in Eq. (2.1), when the feasible solutions are restricted by multiple constraints coming from many various sources. In this thesis we focus on solving black box constrained optimization problems (COPs) with expensive objective and

constraint functions. Depending on the application of the COP, the evaluation of the constraint functions outputs differently. Evaluating explicit constraints outputs real numbers indicating the level of the constraint violation. But in case of implicit constraints the evaluation can only reveal feasibility or infeasibility of the evaluated point. An optimization problem with explicit constraint functions can be defined by the minimization of an objective function f subject to inequality constraint function(s) g_j and equality constraint function(s) h_k :

$$\begin{array}{lll} \text{Minimize} & f(\vec{x}), & \vec{x} \in [\vec{l}, \vec{u}] \subset \mathbb{R}^d, \\ \text{subject to} & g_j(\vec{x}) \leq 0, & j = 1, 2, \ldots, m, \\ & h_k(\vec{x}) = 0, & k = 1, 2, \ldots, r, \end{array} \tag{2.2}$$

where $\vec{l}$ is the lower bound of the search space $\mathbb{S} \subseteq \mathbb{R}^d$ and the $\vec{u}$ is the upper bound. $\vec{x} = [x_1, x_2, \cdots, x_d]$ is a vector with the length of the parameter space size d. The variable x_i refers to the i-th element of the vector $\vec{x}$. The goal is to find $\vec{x}^*$ which minimizes the fitness function $f(.)$ in the feasible space $\mathbb{F} \subseteq \mathbb{S} \subseteq \mathbb{R}^d$. Similar to the unconstrained case (Eq. (2.1)), a constrained maximization problem can be transformed to a minimization problem by negating the fitness function without loss of generality.

The constraint functions of the real-world optimization problems can be grouped with different factors into different categories. A real-world COP can have either a set of *black box* or *white box* constraints, meaning that with some COPs no prior knowledge about the constraints is available and for some other the constraints can be formulated analytically. For solving COPs with a black box objective function but known constraints several algorithms are developed [5, 4, 166].

Depending on the application of a COP, its constraint functions can be categorized as *cheap* or *expensive* to evaluate regardless of the type of the objective function. This means that in some cases a black box COP is only expensive in its objective function and the constraints evaluations are not as expensive as the objective function.

Inequality constraints restrict the feasible solutions to a subset of the search space with the same dimensionality. On the contrary, *equality constraints* limit the feasible solutions to a subspace with a smaller dimensionality, in other words the volume of the feasible space is zero for COPs with equality constraints. It is worth to mention that black box equality constraints are more challenging to handle comparing to inequality constraints and they cannot be addressed by many

constrained optimizers. A detailed discussion about the equality constraints and different techniques to handle them can be found in Ch. 4 of this thesis.

A solution, satisfying the constraints, is called a feasible solution and the one which does not lie within the region restricted by constraints is called an infeasible solution. Evaluation of an infeasible solution can have different outputs depending on the application of the COP. In some cases evaluation of an infeasible solution has a very harsh consequence like a software crash, for such problems infeasible solutions must be avoided as much as possible. There are also cases, where the constraint functions output Boolean results saying if the evaluated point is feasible or infeasible but not more. For such problems classification algorithms might be helpful to model the hidden structure of the constraints. The third case is when the constraint function outputs real numbers indicating not only the feasibility or infeasibility but also the level of constraint violation. A more detailed taxonomy of constraint function in real-world applications can be found in [46]. In this dissertation we focus on developing efficient algorithms to tackle fully black box expensive optimization problems subject to real value equality and inequality constraint functions.

The already existing unconstrained optimizers can be extended to constrained optimizers in different fashions. We list the most effective techniques which are widely used. Several approaches concentrate on the feasible region and assume that any feasible solution is better than the infeasible ones [102, 44] unlike some others which try to approach the feasible region by allowing some infeasible points in the population [150, 151]. Additionally, there exists several approaches which benefit from the existence of infeasible solutions by repairing and guiding them to the feasible area [38, 113, 98]. Various constraint handling techniques can be classified as following:

- *Death penalty* rejects the infeasible individuals and re-samples as long a feasible solution is found. This method is used with simulated annealing [164] and evolution strategy (ES) for simple problems [13]. A drawback of this approach is the large amount of imposed function evaluations, especially for COPs with very small feasible region.

- *Penalty functions* are one of the most common ways of handling constraints in optimization. The idea is to change the constrained optimization problem to an unconstrained one by minimizing $\tilde{f}$ a weighted combination of the objective function f and a measure of the constraint function(s) G as follows:

$$\tilde{f} = f(x) + \alpha \cdot G(x), \tag{2.3}$$

where α is the penalty factor. G can be defined in different ways including the sum or product of all constraint violations, maximum constraint violation, etc. The penalty factor can be assigned as a constant or can be adapted during the optimization process. However, a drawback of such techniques is that the penalty factor is very problem sensitive and sometimes time-consuming tunings are required to find the right penalty factor.

- *Stochastic ranking* originally was utilized for an evolution strategy (ES) by Runarsson et al. [150, 151], to assist solving COPs with ES. However, the main idea of stochastic ranking, assigning good ranks to infeasible solutions with some probability in order to benefit from existence of infeasible solutions in the population, can be applied in many different nature-inspired rank-based algorithms.

- *Repair algorithms* try to modify the infeasible solutions and move them toward the feasible region. Gene repair [113] is an example of repair algorithms used in combination to a genetic algorithm. Several repair mutations are also proposed for ES approaches in [25, 166]. Chootinan et al. [38] propose a gradient based repair algorithm embedded in a genetic algorithm. Koch et al. [98] introduce a surrogate-assisted repair.

- *Multiobjective optimization* techniques are used for solving COPs by several authors [88]. This class of constraint handling approaches aim at minimizing the objective function and one or several measures of constraint violations, e.g., sum or max of constraint violation.

The main drawback of the most of the mentioned algorithms is that they are often not applicable to expensive real-world COPs as they demand too many function evaluations. This motivates the coming chapters of this thesis, introducing new surrogate-assisted constrained optimizers suited for expensive COPs.

2.4 Surrogate-Assisted Constrained Optimization

2.4.1 Taxonomy of Surrogate Models

Optimizing an expensive black box function subject to multiple constraints, is a common task in real-world applications. When every single function evaluation imposes significant effort, then a suitable optimizer must be efficient in terms of number

of function evaluations that it requires to find a near-optimal solution. Surrogate-assisted optimizers using mathematical modeling techniques aim to take advantage from the limited information gained about the black box functions during the optimization procedure as much as possible. The term surrogate can refer to any type of modeling technique employed to approximate the hidden structure of the black box functions for an optimization task. To tackle unconstrained optimization problems, a great number of surrogate-assisted solvers applying different modeling approaches were developed [90, 68, 108, 130, 188, 164]. However, less effort is devoted to solve the constrained optimization problems (COPs) by assistance of surrogates. We categorize the surrogate-assisted constrained and unconstrained optimizers into four different groups based on their modeling approaches:

- *Interpolation* is one of the most common modeling techniques used to replace the black box functions in an optimization task. The idea is to fit a surface for the black box function based on the limited evaluated points scattered in the search space. Cobyla [131] is a constrained surrogate-assisted approach making use of *linear models* to approximate objective and constraints function locally. This approach applies the Nelder-Mead technique combined with a penalty function to solve the constrained problem on the surrogates. BOBYQA [130] uses *quadratic approximation* for the objective functions in unconstrained optimization. Wang et al. [178] proposed a variant of a response surface method to approximate the objective function by quadratic modeling. Although they addressed constrained problems, they assumed that constraint functions are cheap to evaluate in order to keep the problems easy to tackle. Quadratic models are also used to solve black box COPs in [40]. Since most of the real-world optimization problems have nonlinear functions, the *radial basis function interpolation (RBF)* attracted a lot of attention in the surrogate-assisted constrained and unconstrained optimization field [145, 140, 144, 143, 82]. This is because the RBF models are reasonably accurate, easy to train and extend to high dimensions. In this thesis (Ch. 3) we introduce an algorithm which works by means of RBF interpolations. Sec. 2.4.2 briefly describes the RBF interpolation technique.

- *Probabilistic Modeling* is commonly employed as a mean to build surrogate models for unconstrained optimization problems and less often for the constrained optimization problems. Gaussian process aka Kriging [103] is a probabilistic modeling approach which is capable of not only providing a model for the black box function but also an uncertainty quantification for the model, given a limited number of evaluated points. Gaussian process armed with an

uncertainty measure is the backbone of the *expected improvement* concept used in Bayesian optimization approaches [116, 117] and the Efficient Global Optimization (EGO) [90]. Schonlau et al. [159] use probabilistic modeling to tackle constrained optimization problems, by modeling the feasibility probability. In this thesis (Ch. 5) a constrained solver equipped with probabilistic models is introduced which outperforms the latter technique [159] in several cases.

- *Classification* approaches are also utilized as surrogates to predict if a solution is feasible in infeasible before being evaluated on the real expensive black box constraint functions. Poloczek and Kramer [128] employed support vector machines as a feasibility classifier combined with the Covariance Matrix Adaptation Evolution Strategy (CMA-ES). Their approach tends to reduce the number of constraint function evaluations, as they assume that only constraint functions are expensive to evaluate. They obtain slight improvements on analytical test functions, but also report negative results on some functions. Arnold and Hansen [7] give reasons for this decreased performance, which is possibly due to the rotation of the CMA-ES mutation distribution. The contribution made by [128] in terms of reducing the required number of constraint function calls is not significant. Arnold and Hansen [7] recommend an alternative approach which yields better results. Later, Basudhar et al. [22] coupled the efficient global optimization algorithm (EGO) as a surrogate for the objective function with a support vector machine classifier as surrogate for constraint functions. They report promising results on an experimental study for low-dimensional COPs.

- *Deep Learning Sequence Models* are used very recently for optimization purposes. Chen et al. in [36] proposed a fundamentally new surrogate-assisted approach. In the mentioned work the surrogates do not learn the black box functions, they rather learn a generic optimizer to solve black box functions by means of sequence models like recurrent neural networks. Later, the work was extended in [37].

2.4.2 Radial Basis Function Interpolation

Radial basis function (RBF) interpolation is one of the strongest methods among the multitude of existing regression techniques for modeling a black box function, given a set of sparse scatter points in any n-dimensional search space. This makes them a very suitable choice as surrogate models: They are utilized to solve expensive optimization problems efficiently. Although the origin of the RBF development goes

back to the invention of a special form of RBF interpolation (Multiquadric RBF) by trial and error for 2-dimensional problems [78], strong theoretical foundations support their advantages over other interpolation techniques like polynomial or Fourier interpolation even for high dimensional problems [112, 182]. In 1990 a topographer called Hardy tried to automatize the generation of contour maps for 2-dimensional topographic surfaces by means of mathematical approaches which can be locally accurate while it can also model the global trend reasonably well. For Hardy's tasks, the Fourier interpolation was a failure due to its aggressive oscillation between the sparse evaluated points and polynomial interpolation had difficulties to model surfaces with large derivatives. Additionally, according to the Haar theorem [74] and Mairhuber-Curtis theorem [43, 109], there are infinitely many sets of points that can cause instability for polynomial or other types of interpolation techniques, but not for RBF.

We give a sketch of the proofs of these theorems along the lines of the excellent article by Fornberg et al [62], explaining why the mentioned instabilities do not happen for RBF interpolation. Let us assume that any interpolation approach tries to approximate a function f by means of a weighted linear combination of k basis functions F_i as in Eq. (2.4).

$$s(\vec{x}) = \sum_{i=1}^{k} \theta_i F_i(\vec{x}), \tag{2.4}$$

To find the parameters of such interpolant satisfying $s(\vec{x}_j) = f(\vec{x}_j)$ for n given points $\vec{x}_j$, the linear equation system in Eq. (2.5) must be solved.

$$\underbrace{\begin{bmatrix} F_1(\vec{x}_1) & F_2(\vec{x}_1) & \cdots & F_k(\vec{x}_1) \\ F_1(\vec{x}_2) & F_2(\vec{x}_2) & \cdots & F_k(\vec{x}_2) \\ \vdots & \vdots & & \vdots \\ F_1(\vec{x}_n) & F_2(\vec{x}_n) & \cdots & F_k(\vec{x}_n) \end{bmatrix}}_{\mathbf{A}} \underbrace{\begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_k \end{bmatrix}}_{\vec{\theta}} = \underbrace{\begin{bmatrix} f(\vec{x}_1) \\ f(\vec{x}_2) \\ \vdots \\ f(\vec{x}_k) \end{bmatrix}}_{\vec{f}} \tag{2.5}$$

Let us assume that for a set of given points matrix $\mathbf{A}$ has a non-zero determinant meaning that this equation system has a unique solution. If we fix all the points but move two points in the search space and swap them with each other, then the sign of the determinant of matrix $\mathbf{A}$ changes. This being said, it becomes clear that somewhere on the interchanging path the determinant of $\mathbf{A}$ becomes zero. Considering the fact that there are infinite number of possible paths for such an interchange in 2 or higher dimensions, means that there are infinite configurations of points for

which no unique polynomial interpolant can be found. This does not hold true for RBF interpolants, as the basis functions are radial, meaning that they are only dependent on the distance of the points to each other, see Eq. (2.6). In case of the RBF interpolation linear equation system, swapping two points does not change the sign of the determinant as the distances remain unchanged.

As already mentioned, RBF interpolation approximates a function by fitting a linear weighted combination of radial basis functions. Any function which is only dependent on the distance from a specific point (centroid) in the space belongs to the group of radial functions. RBF takes all the evaluated points as the centroids of the basis functions, produces a perfect fit through these points and reasonably approximates the unknown area:

$$\hat{f}(\vec{x}) = \sum_{i=1}^{n} \theta_i \varphi(r_i) = \sum_{i=1}^{n} \theta_i \varphi((||\vec{x} - \vec{x}_i||) \tag{2.6}$$

The distance r is often determined based on the Euclidean norm but this is not the only approach. Some of the commonly used RBFs are shown in Tab. 8.1. The radial basis functions can be parameter-free like the cubic RBF or having a shape parameter α as in Gaussian RBF. In order to determine the weights θ_i, Eq. (2.7) must be solved.

$$\underbrace{\begin{bmatrix} \varphi(r_{11}) & \varphi(r_{12}) & \cdots & \varphi(r_{1n}) \\ \varphi(r_{21}) & \varphi(r_{22}) & \cdots & \varphi(r_{2n}) \\ \vdots & \vdots & \ddots & \vdots \\ \varphi(r_{n1}) & \varphi(r_{n2}) & \cdots & \varphi(r_{nn}) \end{bmatrix}}_{\mathbf{\Phi}} \underbrace{\begin{bmatrix} \theta_1 \\ \theta_2 \\ \vdots \\ \theta_k \end{bmatrix}}_{\vec{\theta}} = \underbrace{\begin{bmatrix} f(\vec{x}_1) \\ f(\vec{x}_2) \\ \vdots \\ f(\vec{x}_k) \end{bmatrix}}_{\vec{f}}, \tag{2.7}$$

where $\mathbf{\Phi} \in \mathbb{R}^{n \times n}$ and r_{ij} is the Euclidean distance between the $\vec{x}_i$ and $\vec{x}_j$ for $i, j = 1, \ldots, n$. Therefore, the weights can simply be computed as

$$\vec{\theta} = \mathbf{\Phi}^{-1} \vec{f}, \tag{2.8}$$

if the matrix Φ is invertible, which is - due to the Haar theorem - more often the case for RBFs than for other interpolants.

Augmented RBF

Although RBF matrices have fewer singularities than other interpolation schemes, the above statements are not a guarantee that $\mathbf{\Phi}$ in Eq. 2.7 is always invertible.

Table 2.1: Commonly used radial basis functions

Type of basis function	$\varphi(r)$
Parameter-free RBF	
Cubic	r^3
Thin plate spline	$r^2 \log r$
RBF with shape parameter	
Gaussian	$e^{-\frac{r^2}{2\alpha^2}}$
Multiquadric (MQ)	$\sqrt{1-(\frac{r}{\alpha})^2}$

Micchelli shows that for some radial basis functions including the cubic RBF there are special configuration of points for which $\mathbf{\Phi}$ becomes singular. In order to assure that the Eq. 2.7 has a unique solution using any type of radial basis functions, Micchelli introduced augmented RBFs [112]. Augmented RBFs are actually RBF functions with a polynomial tail:

$$\hat{f}(\mathbf{x}) = \sum_{i=1}^{n} \theta_i \varphi(||\vec{x} - \vec{x}_i||) + \mu_0 + \sum_{l=1}^{kd+1} \mu_l p_l(\vec{x}), \quad \vec{x} \in \mathbb{R}^d, \tag{2.9}$$

where $\mu_0 + \sum_{l=1}^{kd+1} \mu_l p_l(\vec{x})$ is a k-th order polynomial tail in d-dimensional space with $kd+1$ coefficients.

The augmented RBF model requires the solution of the following linear system of equations:

$$\begin{bmatrix} \mathbf{\Phi} & \mathbf{P} \\ \mathbf{P}^T & \mathbf{0}_{(kd+1)\times(kd+1)} \end{bmatrix} \begin{bmatrix} \vec{\theta} \\ \vec{\mu} \end{bmatrix} = \begin{bmatrix} \vec{f} \\ 0_{(kd+1)} \end{bmatrix} \tag{2.10}$$

Here, $\mathbf{P} \in \mathbb{R}^{n\times(kd+1)}$ is a matrix with $(1, x_{i1}, \cdots, x_{id}, \cdots, x_{i1}^k, \cdots, x_{id}^k)$ in its ith row, where x_{ij} is the jth component of vector $\vec{x_i}$ for $i = 1, \cdots, n$ and $j = 1, ..., d$. $\mathbf{0}_{(kd+1)\times(kd+1)} \in \mathbb{R}^{(kd+1)\times(kd+1)}$ is a zero matrix, $0_{(kd+1)}$ is a vector of zeros. In this work, we use the augmented cubic radial basis function with a second order polynomial tail ($k = 2$).

Fig. 2.2 and 2.3 show a $1D$ interpolation example by means of Gaussian and cubic augmented RBF with a first order polynomial tail ($k = 1$). The goal is to approximate a curve according to the information from the blue points.

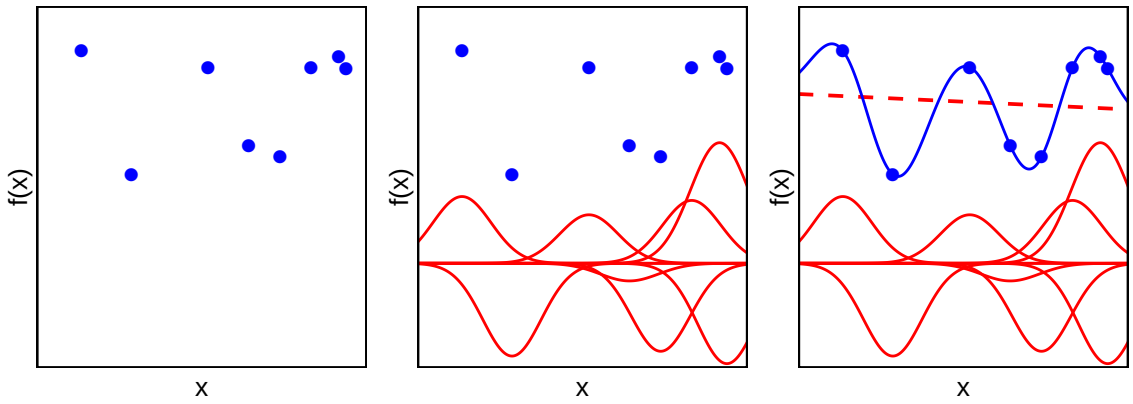

Figure 2.2: Conceptualization of RBF interpolation in $1D$, with Gaussian $\phi(r)$. The goal is to approximate a curve according to the information from the blue points. The red curves are weighted Gaussian radial basis functions with centers of blue points and the red dashed line is the polynomial tail $p(x)$. Summation of all the red curves and the dashed line is the blue curve which interpolates all points and fits a smooth curve through them.

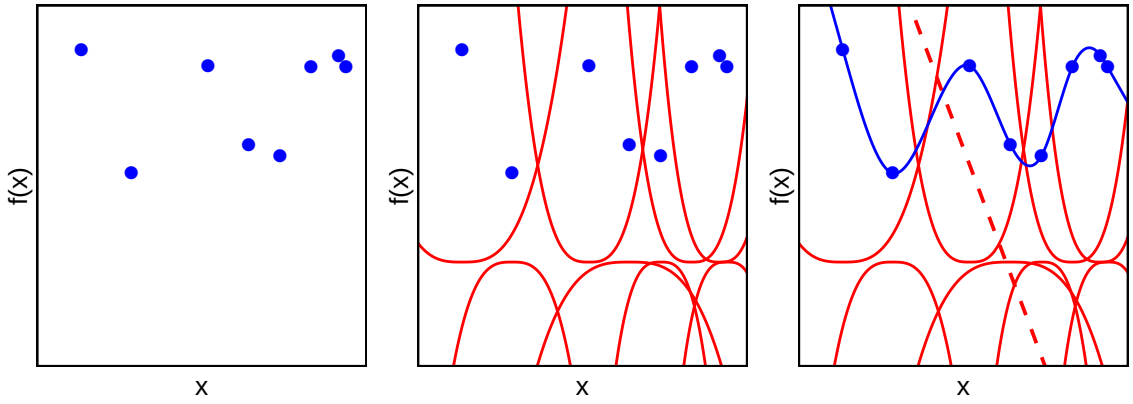

Figure 2.3: Conceptualization of RBF interpolation in $1D$, with cubic $\phi(r)$. Otherwise the same as Fig. 2.2.

In this thesis we use augmented RBF to develop a strong surrogate-assisted constrained optimizer, the so-called SACOBRA optimization framework. We initially use the cubic basis functions in Ch. 3 and 4. Furthermore, we develop an online model selection technique to choose between different types of basis functions during the optimization process in Ch. 8. In the same chapter, we will use a cubic RBF in conjunction with multiquadric RBFs with different shape parameters. We compare

RBF interpolation with Kriging, a probabilistic modeling technique, in Ch. 7. We show that although RBF interpolation unlike Kriging does not have an uncertainty quantification by its nature, it is possible to compute an uncertainty measure for any arbitrary RBF basis function.

2.5 Visualization Methods in Optimization

In many papers in the field of optimization the strength of a technique is measured by comparing the final solution achieved by different algorithms [150]. This approach only provides the information about the quality of the results and neglects the speed of convergence which is a very important measure for expensive optimization problems. Comparing the convergence curve over time (number of function evaluations) is also one of the common benchmarking approaches [141]. Although a convergence curve provides good information about the speed of convergence and the final quality of the optimization result, it can be used to compare performance of several algorithms only on *one* problem. It is often interesting to compare the overall capability of a technique on solving a group of problems. The data and performance profiles developed by Moré and Wild [118] are good approaches to analyze the performance of any optimization algorithm on a whole test suite and are now used frequently in the optimization literature [32, 142, 14, 15, 19].

Performance profiles

Performance profiles are defined with the help of the performance ratio

$$r_{p,s} = \frac{t_{p,s}}{\min\limits_{\forall s' \in \mathbb{S}} \{t_{p,s'}\}}, \qquad p \in \mathbb{P} \tag{2.11}$$

where $\mathbb{P}$ is a set of problems, $\mathbb{S}$ is a set of solvers and $t_{p,s}$ is the number of iterations solver $s \in \mathbb{S}$ requires to solve problem $p \in \mathbb{P}$. A COP problem is said to be *solved* if a feasible solution $\vec{x}$ is found whose objective value $f(\vec{x})$ deviates from the best known objective value $f(\vec{x^*})$ less than a given tolerance τ:

$$f(x) - f_L \leq \tau \tag{2.12}$$

Smaller values are more desirable for the performance ratio $r_{p,s}$. When using the best solver s to solve problem p then $r_{p,s} = 1$. If a solver s cannot solve problem p the performance ratio is set to infinity. The performance profile ρ_s is now defined as

a function of the steerable performance factor α:

$$\rho_s(\alpha) = \frac{1}{|\mathbb{P}|} \left| \{p \in \mathbb{P} : r_{p,s} \leq \alpha\} \right|. \tag{2.13}$$

In performance profile plots the relative performance of each algorithm is shown by a curve of the performance profile over the performance factor. The higher and more to the left this curve is the better the algorithm.

Data profiles

Data profiles are suitable for evaluating optimization algorithms on expensive problems. They are defined as

$$d_s(\alpha) = \frac{1}{|\mathbb{P}|} \left| \{p \in \mathbb{P} : \frac{t_{p,s}}{d+1} \leq \alpha\} \right|, \tag{2.14}$$

with $\mathbb{P}, \mathbb{S}$ and $t_{p,s}$ defined as above and d as the dimension of problem p.

Although performance profiles and data profiles share many common properties with each other, in this dissertation we prefer data profiles because the performance factor α has a more intuitive meaning for data profiles: If we allow for each problem with dimension d a budget of $B_\alpha = \alpha \cdot (d+1)$ function evaluations, then the value $d_s(\alpha)$ can be interpreted as the fraction of problems which solver s can solve within this budget B_α. In other words, data profile curves show the percentage of the solved problems after $\alpha \cdot (d+1)$ function evaluations.