

University of Warwick institutional repository: <http://go.warwick.ac.uk/wrap>

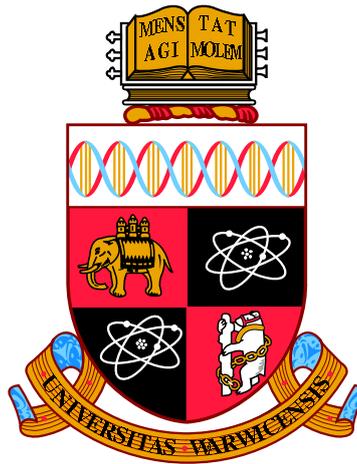
**A Thesis Submitted for the Degree of PhD at the University of Warwick**

<http://go.warwick.ac.uk/wrap/62123>

This thesis is made available online and is protected by original copyright.

Please scroll down to view the document itself.

Please refer to the repository record for this item for information to help you to cite it. Our policy information is available from the repository home page.



**Efficient sequential sampling for global optimization  
in static and dynamic environments**

by

**Sergio Morales Enciso**

**Thesis**

Submitted to the University of Warwick

for the degree of

**Doctor of Philosophy**

**Complexity Sciences and Operational Research**

May 2014

THE UNIVERSITY OF  
**WARWICK**

# Contents

<b>List of Tables</b>	<b>iv</b>
<b>List of Figures</b>	<b>v</b>
<b>Acknowledgments</b>	<b>vii</b>
<b>Declarations</b>	<b>viii</b>
<b>Abstract</b>	<b>ix</b>
<b>List of Abbreviations</b>	<b>x</b>
<b>Chapter 1 Introduction</b>	<b>1</b>
<b>Chapter 2 Global optimization of expensive-to-evaluate black-box functions</b>	<b>4</b>
2.1 Global optimization of expensive-to-evaluate black-box functions . . .	5
2.2 Initial design of experiments for sequential sampling optimization . . .	7
2.3 Gaussian processes as response surfaces for optimization . . . . .	9
2.3.1 Response surface methodologies . . . . .	10
2.3.2 Gaussian processes . . . . .	10
2.4 Sequential optimization . . . . .	13
2.4.1 Sequential sampling policies . . . . .	14
2.4.2 Efficient Global Optimization . . . . .	16
2.5 Expected improvement maximization . . . . .	18
2.5.1 Expected improvement maximization techniques . . . . .	18
2.5.2 Testing the need for good quality EI optimizers . . . . .	19

2.6	Selecting an EI maximization technique using the Black-Box Optimization Benchmark . . . . .	24
2.6.1	The Black-Box Optimization Benchmark . . . . .	24
2.6.2	Performance measures . . . . .	25
2.6.3	EI maximization algorithms comparison . . . . .	27
2.6.4	Practical implementation details . . . . .	33
2.7	Results and concluding notes . . . . .	36

**Chapter 3 Space partitioned EGO for accelerated response surface modeling** **39**

3.1	EGO in partitioned spaces . . . . .	41
3.1.1	Survey on multi-point sequential sampling . . . . .	42
3.1.2	Related space partitioning algorithms . . . . .	44
3.1.3	Space partitioning for accelerating efficient global optimization	45
3.1.4	Non-disjoint space partitioning efficient global optimization	49
3.2	Numerical implementation decisions . . . . .	50
3.2.1	Initial DoE . . . . .	52
3.2.2	Partitioning dimension selection criteria . . . . .	53
3.2.3	Partition size threshold . . . . .	56
3.2.4	Comparing SPEGO and NSPEGO . . . . .	59
3.3	SPEGO in context . . . . .	64
3.3.1	SPEGO vs. EGO . . . . .	64
3.3.2	SPEGO against the state of the art . . . . .	67
3.4	Conclusions . . . . .	72

**Chapter 4 Tracking global optima in dynamic environments with efficient global optimization** **75**

4.1	Related work . . . . .	76
4.2	Adaptations of EGO to the dynamic case . . . . .	78
4.2.1	Random strategy . . . . .	79
4.2.2	Reset strategy . . . . .	79
4.2.3	Ignore strategy . . . . .	79
4.2.4	Reset* sampling strategy . . . . .	80

4.2.5	Discounted information through noise sampling strategy (DIN)	80
4.2.6	Time as an additional dimension sampling strategy (TasD+1)	81
4.2.7	Previous surface mean prior sampling strategy (PSMP)	84
4.3	Experiment setup for comparing dynamic optimizers	85
4.3.1	The moving peaks benchmark	86
4.3.2	Performance measures for dynamic optimization	86
4.3.3	Implementation details	87
4.4	Numerical results and model comparison	89
4.4.1	Experimental procedure	89
4.4.2	Parameter analysis	95
4.4.3	Statistical comparison of model performance	98
4.5	Discussion and conclusions	99
<b>Chapter 5 Conclusions and future work</b>		<b>102</b>
<b>Appendix A SPEGO against the state of the art, separated by BBOB</b>		
	<b>type of function</b>	<b>107</b>

# List of Tables

3.1	SPEGO - Initial DoE . . . . .	54
3.2	SPEGO - Partitioning dimension selection criteria . . . . .	56
3.3	SPEGO - Partitioning threshold as a function of problem dimension	58
3.4	SPEGO - Effect of partitioning threshold size . . . . .	58
3.5	SPEGO vs. NSPEGO . . . . .	64
3.6	SPEGO vs. EGO . . . . .	65
3.7	SPEGO vs. CMA-ES vs. SMAC . . . . .	71
4.1	Parameters for the moving peaks benchmark . . . . .	88
4.2	Performance of the proposed strategies for the dynamic optimization problem . . . . .	100

# List of Figures

2.1	Expected improvement illustration . . . . .	17
2.2	Comparison of EGO with different fixed solution qualities for the expected improvement . . . . .	20
2.3	Comparison of EGO with random solution qualities for the expected improvement . . . . .	21
2.4	Number of local maxima in EI as a function of sample size . . . . .	23
2.5	Fixed target vs. fixed cost paradigms . . . . .	26
2.6	Illustration of the random multi-start and hyper-box methods . . . . .	27
2.7	Comparison of the random multi-start, hyper-box, and genetic algorithm optimization methods to optimize the expected improvement . . . . .	32
2.8	Comparison for restricted EI function evaluations budget of the three proposed EI maximization techniques . . . . .	34
3.1	Space partitioning illustration . . . . .	48
3.2	SPEGO - Initial DoE . . . . .	55
3.3	SPEGO - Dimension selection criteria . . . . .	57
3.4	SPEGO - Threshold partitioning size - Performance . . . . .	60
3.5	SPEGO - Threshold partitioning size - Running time . . . . .	61
3.6	SPEGO vs. NSPEGO - Performance . . . . .	62
3.7	SPEGO vs. NSPEGO - Running time . . . . .	63
3.8	SPEGO vs. EGO - Fixed time comparison . . . . .	66
3.9	SPEGO vs. EGO - Fixed budget comparison . . . . .	68
3.10	SPEGO vs. EGO - Running time . . . . .	69
3.11	SPEGO vs. CMA-ES vs. SMAC . . . . .	73

4.1	Sequential sampling using DIN strategy illustration (low noise) . . .	82
4.2	Sequential sampling using DIN strategy illustration (high noise) . . .	83
4.3	Noise level optimization for DIN sampling strategy . . . . .	90
4.4	Offline and average errors for all strategies - 1D . . . . .	91
4.5	Offline error convergence curves . . . . .	93
4.6	Current error for each compared strategy - Individual plots . . . . .	94
4.7	Offline error for the parameter analysis - 1D . . . . .	96
4.8	Offline error for the parameter analysis - 2D . . . . .	97
A.1	SPEGO vs. CMA-ES vs. SMAC - Separable functions . . . . .	108
A.2	SPEGO vs. CMA-ES vs. SMAC - functions with low or moderate conditioning . . . . .	109
A.3	SPEGO vs. CMA-ES vs. SMAC - functions with high conditioning and unimodal . . . . .	110
A.4	SPEGO vs. CMA-ES vs. SMAC - multi-modal functions with ade- quate global structure . . . . .	111
A.5	SPEGO vs. CMA-ES vs. SMAC - multi-modal functions with weak global structure . . . . .	112

# Acknowledgments

I would like to express my deepest appreciation to Professor Juergen Branke, my main supervisor, for his wise advice, bright ideas, continuous encouragement, and unrestricted availability, all of which made this accomplishment possible.

I am also extremely grateful to Professor Robin C. Ball, my co-supervisor, for his invaluable insights, and all the help and support provided throughout my years at the Centre for Complexity Science.

For encouraging me to consider the path of research, I would like to thank Dr. Enrique Garza Escalante.

For their unconditional support, I am ever grateful to my family. In particular, to my mother for instilling into me the passion for creativity, and showing me that imagination is more powerful than knowledge. And to my father, who through his example, taught me the value of rightness, hard work, and importance of humanity as a value.

# Declarations

This thesis is submitted to the University of Warwick in support of my application for the degree of Doctor of Philosophy. It has been composed by myself and has not been submitted in any previous application for any degree. Every effort has been made to credit the authors of the published work which provides the foundations upon which this research is built.

The work presented (including data generated and data analysis) was carried out by the author.

Parts of this thesis have previously been published, or have been submitted for publication, by the author:

- Morales-Enciso, S. and Branke, J. Response Surfaces with Discounted Information for Global Optima Tracking in Dynamic Environments. In *Nature Inspired Cooperative Strategies for Optimization (NICSO 2013)*, vol. 512 of *Studies in Computational Intelligence*, pp. 57–69. Springer International Publishing, 2014. doi: 10.1007/978-3-319-01692-4
- Morales-Enciso, S. and Branke, J. Tracking global optima in dynamic environments with efficient global optimization. Submitted to the *European Journal of Operational Research*, November 2013.

The present thesis has not been submitted for consideration at any other institution.

# Abstract

Many optimization problems involve acquiring information about the underlying process to be optimized in order to identify promising solutions. Moreover, in some cases obtaining this information can be expensive, which calls for a method capable of predicting promising solutions so that the global optimum can be found with as few function evaluations as possible. Another kind of optimization problem arises when dealing with objective functions that change over time, which requires tracking of the global optima over time. However, tracking usually has to be quick, which excludes re-optimization from scratch every time the problem changes. Instead, it is important to make good use of the history of the search even after the environment has changed.

This thesis revolves around the topic of response surface based sequential sampling for global optimization of expensive-to-evaluate black-box functions under static and dynamic scenarios. Regarding the former scenario, it addresses the high computational cost inherent to Efficient Global Optimization (EGO), a global search algorithm that is known to work well for expensive black-box optimization problems where only few function evaluations are possible, and which uses surrogate models of the fitness landscape for deciding where to sample next. The proposed variant is based on partitioning the space and building local models to accelerate the selection of future sampling locations with a minimal impact on the optimization performance. The linear computational complexity as a function of the number of observations of this extension is shown, and its performance benchmarked to both the original algorithm it extends, and state of the art algorithms. For the latter scenario, we propose and compare four methods of incorporating old and recent information in the surrogate models of EGO in order to accelerate the search for the global optima in a dynamically changing environment. As we demonstrate, exploiting old information as much as possible significantly improves the tracking behavior of the algorithm.

# List of Abbreviations

ANN	Artificial Neural Networks
APRS	Adaptive partitioning random search
BBOB	Black-box optimization benchmark
CMA-ES	Covariance matrix adaptation for evolution strategies
DoE	Design of experiments
EA	Evolutionary algorithm(s)
ECDF	Empirical cumulative density function
EGO	Efficient Global Optimization
EI	Expected improvement
GA	Genetic algorithm
GP	Gaussian process(es)
HB	Hyper-box multi-start with local hill climbers
IAGO	Informational approach to global optimization
KGCB	Knowledge gradient-policy for correlated beliefs
KGCP	Knowledge gradient for continuous parameters policy
LHS	Latin hypercube sampling
LIP	Least interpolating polynomials

MARS Multivariate adaptive regression splines

NP Nested partitioning

NSPEGO Non-disjoint space partitioning efficient global optimization

PR Polynomial regression

PRS Pure random sampling

RBF Radial Basis Functions

RMS Random multi-start (with local hill climbers)

RS Random sampling

SA Simulated annealing

SMAC Sequential model-based algorithm configuration procedure

SPEGO Space partitioning efficient global optimization

SS Stratified sampling

SVM Support Vector Machines

# Chapter 1

## Introduction

Optimization is the process of finding the best alternative, out of all feasible alternatives, with respect to previously defined criteria. However, finding the best alternative (here called *solution*) is rarely trivial. Optimization is ubiquitous, but its presence is often overlooked. For instance, parcel delivery, air traffic control systems, search engines, timetable assignments, and unmanned aerial vehicles, all have in common that they rely on solving optimization problems.

In order to tackle optimization problems algorithmically, it is convenient to state them in a mathematical way, where the decisions to be made are modeled as independent variables or parameters. When evaluating the problem at a given set of parameters, the observed outcome represents a dependent variable, or response value. The pair formed by this outcome, together with the parameter setting used to obtain it, is referred to as a sample or an observation, and the function mapping the space of all parameters to their corresponding response value, as objective function.

When the objective function is a simulation model, or represents an unknown natural process, an analytic representation is usually not available. Such optimization problems are often referred to as black-box optimization problems. The lack of analytic representation restricts the use of gradient based optimization methods. Instead, during optimization, it is necessary to learn about the characteristics of the objective function via sampling. The learning process can be implemented in different ways, one of which is to use statistical techniques capable of inferring and exploiting correlations with the aim of building a surrogate model which can provide

an approximation of the value of a function at a given location.

In some cases, obtaining information about the system is costly. We refer to these cases as expensive-to-evaluate due to the amount or type of resources needed to acquire new information, which most often involves large investments, long waiting times, human resources, or disruptions to a system. Expensive-to-evaluate optimization problems are likely to arise, for example, when dealing with complex simulations or in engineering design [Burl and Wang, 2009]. For instance, improving the design of a wing, a nano-pipe, or a car frame, which is controlled by a set of parameters that are to be optimized, usually involves running computational fluid dynamics simulations that may take hours to run. In this example, the cost is measured in time. Another common scenario is the identification of promising regions for mining, where data collection involves sending out workers for excavations, chemical analyses or data processing. Not only is this time consuming but it also is expensive in monetary terms.

Shan and Wang [2009] conclude that current research does not focus on trying to directly model and understand black-box functions, but rather focuses on improving sampling strategies and finding clever uses of the scarce observed data in order to determine promising areas to explore. This coincides with the main topic of the current thesis which is to determine efficient sequential sampling policies and to study their practical implementation under different situations. The sampling strategies presented throughout this thesis are based on the well-studied field of sequential Bayesian optimization, which is a general probabilistic approach for estimating unknown probability density functions in a recursive manner over time, using data to update the estimated model progressively, as it becomes available.

The main contributions of this thesis are, first, a quantitative study showing the benefits of finding the global maximum of the expected improvement, an auxiliary function that arises as part of the selected response-surface-aided sequential optimizer. Second, the conception, design, analysis, and benchmark of SPEGO, a response-surface-aided sequential sampling optimization algorithm that extends the well established Efficient Global Optimization (EGO) algorithm. The proposed extensions rely on the partition of the search space, reducing EGO's cubic compu-

tational complexity as a function of the number of samples, to linear, with only a minor impact on the quality of the solutions it finds. And third, we propose the first approach to track the global optima of dynamically changing optimization problems by means of a surrogate-based global search algorithm.

This thesis is structured as follows. Chapter 2 starts by presenting a literature survey on response surface modeling, which reveals that the static version of the black-box optimization problem has been extensively studied. Then, the necessary terminology and notation to approach this problem is introduced, providing the theoretical background and fundamental building blocks required to treat more sophisticated versions of the problem in later chapters. Besides, the methodology selected to benchmark the algorithms throughout the rest of the thesis is explained, demonstrating its use by exploring the impact of an important parameter governing the chosen sequential optimization algorithm.

In Chapter 3 we propose a new algorithm, based on space partitioning, so as to accelerate the response surface aided sequential optimization process. After introducing the main concepts for partitioning the space, different enhancements and implementation decisions are tested through numerical experimentation. Then, the proposed algorithm is benchmarked against both the original algorithm it extends, and state of the art algorithms encountered in the literature.

Chapter 4 is devoted to the study of dynamic environments where, rather than finding the global optimum, the goal is to track the changing optimum over time. This chapter starts by presenting a literature review on dynamic optimization of black-box functions, followed by the detailed presentation of several methods to adapt the response surface based optimization techniques introduced in previous chapters. Then, the performance of these methods is evaluated using a benchmark specifically designed for dynamic environments, and a statistical study of the results obtained through numeric experimentation is presented.

Finally, Chapter 5 provides some concluding notes and proposes future research areas that would extend the work here presented.

## Chapter 2

# Global optimization of expensive-to-evaluate black-box functions

As outlined in the introductory chapter, global optimization is a very broad term which tackles the problem of finding the best solution, in the solution set, of an objective function. Best, in this context and without loss of generality, refers to either the minimum or the maximum of such an objective function, depending on the nature of the problem at hand.

The field of global optimization is vast, and the problems it addresses vary so much that many sub-fields have emerged. A good survey on recent advances in some sub-fields of global optimization can be found in Floudas and Gounaris [2008].

Since the main focus of the current thesis is to study efficient strategies for finding the global optima of deterministic functions of which the structure is not known, and which are expensive to evaluate, we devote this first chapter to introduce the basic concepts that arise in the context of this specific sub-field of optimization. More specifically, we present a review of the state of the art of the existing methodologies, we establish the terminology that will be used throughout the remainder of the thesis, and we justify the selection of a good response surface methodology coupled with a sequential algorithm to be used as a base to develop new ideas that are proposed in the following chapters. In particular, Section 2.1 sets the

context of the current research with respect to the whole field of global optimization, and points to related techniques that have been applied to solve similar problems. In Section 2.2, the concept of design of experiments is outlined, which, together with the response surface methodologies introduced and studied in Section 2.3, serve as a starting point to understand the idea behind sequential sampling for optimization. Since sequential sampling for optimization is at the core of the ideas discussed all throughout this thesis, it is explained in detail in Section 2.4, where different approaches are compared, and justifications for the selected method are provided. Then, the subproblem arising when choosing new locations to sample is thoroughly addressed in Sections 2.5 and 2.6, the latter serving as well to introduce a widely accepted benchmark and useful performance measures. Finally, some concluding notes and remarks are given in Section 2.7.

## **2.1 Global optimization of expensive-to-evaluate black-box functions**

In a recent article, Sörensen [2013] proposed that algorithms for global optimization can be classified into three categories (exact algorithms, heuristic algorithms, and approximation algorithms), explaining that exact algorithms guarantee to find an optimal solution in a finite amount of time, whereas heuristics do not provide this guarantee at all, and approximation methods can only guarantee to find a solution within some arbitrary precision of the real optimum. While this classification is accurate, it fails to consider algorithms that guarantee finding optimal solutions in infinite time, which are usually stochastic methods (e.g. simulated annealing [Kirkpatrick, Gelatt, and Vecchi, 1983]), or space filling iterative methods such as the one that is the subject of study in this thesis and for which Locatelli [1997] presented a proof of convergence in infinite time. However, care must be taken when considering infinite time horizons, and for practical purposes, algorithms falling in this category shall be considered heuristics.

The term black-box describes a system of which the internal functioning and implementation are not known, and can therefore only be explored in terms of the

responses obtained for the provided inputs. Due to the lack of an analytic expression for black-box objective functions, methods requiring an analytic expression of the objective function or of its gradient can not be applied. This leaves heuristics and approximation methods as the only approaches for optimizing black boxes. There is a plethora of heuristics and approximation methods that have been proposed. Below only some of the most widespread and commonly found in literature are briefly explained, and a deeper review on approximation methods is left for Section 2.3.

- Pure random sampling (PRS): Points within the feasible space are sampled uniformly at random. This is applicable to most problems since no assumptions need to be made about the search space. Nevertheless, this approach tends to be very slow since no adaptation nor learning about the structure of the problem at hand is done. For a review on these methods and proof of convergence, see Solis and Wets [1981].
- Simulated annealing (SA): First proposed by Kirkpatrick, Gelatt, and Vecchi [1983], this algorithm, which mimics the annealing process of solids, explores the search space by always accepting local movements towards states with better solutions and accepting local movements towards states with worse solutions with a positive probability that decreases according to a predefined schedule. Being able to accept movements with worse solutions is what prevents the algorithm from getting trapped in local optima, and allows a more extensive search of the optimal solution. As in PRS, no explicit adaptation nor learning is done during the search process, although local structure is exploited through local movements towards better states.
- Evolutionary algorithms (EA): As explained by Spears, de Jong, and Bäck [1993], the term EA is a term used to refer to a variety of heuristic algorithms inspired by the processes of natural evolution, and amongst others, includes genetic algorithms, evolutionary programming, and genetic programming. Although they differ in the details and in many aspects, all these algorithms share the basic idea of keeping a population of individuals which evolves according to some rules of selection such as mutations and crossovers. With the aim of optimizing a given problem, the individuals of the population are evaluated

according to a fitness function so that the fittest individuals under such a measure are somehow given preference to survive and/or reproduce. Mutations and crossover, being random, help avoiding premature local convergence.

- Response surface modeling: Unlike all the previous outlined methods which at best use some sort of memory to keep track of good regions within the feasible space, creating a response surface using the available data allows learning of the structure of the landscape by studying the existing correlations amongst observations. Different techniques for doing so exist, and in general they involve some type of regression which is used for the inference part (fitting a response surface to the data), and a criterion to select where the new observations shall be made based on the inferred surface. Exploiting the structure of the landscape provides an advantage to this technique when compared to the previously mentioned optimization techniques, which is a great advantage when dealing with a limited number of samples. However, this advantage comes at the cost of having to spend more time and computational effort in both building the response surface, and selecting the locations for the next observations. Since response surface based optimization is a central topic of this thesis, a more thorough discussion can be found in Section 2.3.

## **2.2 Initial design of experiments for sequential sampling optimization**

In a broad sense, design of experiments (DoE) is a systematic method for selecting which data should be collected and how it should be collected during an experiment, usually with the goal of inferring as much information as possible from the responses obtained subject to a fixed budget in terms of number of samples. More traditional DoE focuses on selecting the data collection points before the experiment starts (off-line) while sequential sampling dynamically selects the location of the next sample each time new data is available (on-line). However, even the latter requires a few samples before it can actually build a useful model to start the on-line process for selecting the next most promising location.

Methods for selecting this initial set of samples have been widely studied under the umbrella of DoE. These methods assume a budget, measured for instance in number of samples to be taken, that we shall call  $\lambda$ , and aim to distribute them throughout the feasible space so as to obtain as much information as possible. Depending on how information is characterized, this leads to different criteria to be optimized. However, since most of the response surface enhanced optimization models require only a reduced number of initial samples to start the infill process<sup>1</sup>, these methods do not have much influence on the final result. So we describe only three settings popular in both the DoE and the sequential optimization communities, but point the interested reader to Pukelsheim [2006] for a thorough enumeration of DoE criteria with detailed definitions.

- Random sampling (RS): Points are drawn uniformly at random from the feasible space, without any restriction. This might lead to some samples being close to each other and to leave some regions unexplored. No partitioning of the space is required, and it is straightforward to implement.
- Stratification, or stratified sampling (SS): The feasible space is first partitioned into disjoint strata (or regions), and then random sampling is applied in each stratum with a local sampling budget proportional to the size of the region. Using this technique, all regions of the feasible space will be represented. In some cases, the partitions might be meaningful and arise naturally from the problem at hand, however, when dealing with generic black-box functions the partitions might seem arbitrary. Raj [1968] offers a deeper explanation on this technique. SS designs can require a very large number of samples when dealing with a large number of dimensions.
- Latin hypercube sampling (LHS): Introduced by McKay, Beckman, and Conover [1979], the idea is to divide each input variable (or dimension) into ranges so as to create partitions of the feasible space, and then choose in which regions to get the samples. The selection of the regions to be sampled must satisfy the

---

<sup>1</sup>In this context, infill process refers to the action of selecting the locations at which observations are to be made, one by one, in order to complement the initial DoE. This process is used when performing sequential optimization with the goal of finding the global optimum with as few samples as possible, which is formally addressed in Section 2.4

Latin hypercube property, which means that exactly one sample of each range will be randomly taken. This is the generalization to many dimensions of the Latin square sampling method [Raj, 1968]. In doing so, LHS ensures not only that —when considering a given single dimension— all portions of the feasible space are sampled, but also that each of the ranges from all the input variables are represented. Since LHS does not aim to sample all the regions created by the intersection of the chosen ranges in all dimensions simultaneously —unlike SS—, the number of samples required to create a sampling design remains relatively low even for a large number of dimensions.

In general, DoE techniques aim to distribute the samples in order to get the best approximation of the whole space, but this does not necessarily help when seeking to optimize an expensive-to-evaluate black-box function since the interest is rather in obtaining good approximations only around the good regions, and ideally, samples should not be wasted in accurately modeling bad regions. This is why we focus on sequential sampling strategies which, given an initial design, provide a location for the next most promising sample. So, in the remainder of this chapter, the initial design is obtained using random sampling, with a budget for this initial stage large enough only to allow for fitting a response surface model, and the use of LHS as an initial DoE is delayed to Chapter 3.

### **2.3 Gaussian processes as response surfaces for optimization**

Response surfaces (or surrogate models) are approximations of an objective function created using available data, and are the output of some sort of regression. These models are used when a direct measurement of the function is not practical, for instance, if each measurement is expensive to obtain in time, money, or any other cost unit.

Many different techniques to build response surfaces have emerged in the recent years, so in this section we first review some of the most widely used response surface techniques, and then present the one of our choice, Gaussian processes (GP), in detail.

### 2.3.1 Response surface methodologies

Response surface optimization has now been studied for a long time, and many techniques have been proposed, some of which have been enhanced, refined, and tailored for specific problems throughout the years. However, the basic techniques for fitting response surfaces remains the same and in most of the cases falls within one of the following classes.

Polynomial regression (PR) is perhaps the oldest response surface method, dating back to 1805 and 1809 when Legendre and Gauss independently proposed it, but it was not until the 20<sup>th</sup> century that it was popularized as a DoE method [Smith, 1918]. Other techniques include, but are not limited to, least interpolating polynomials (LIP) [Boor and Ron, 1990], multivariate adaptive regression splines (MARS) [Friedman, 1991], artificial neural networks (ANN) [Papadrakakis, Lagaros, and Tsompanakis, 1998], support vector machines (SVM) [Drucker, Burges, Kaufman *et al.*, 1997; Cristianini and Shawe-Taylor, 2000], radial basis functions (RBF) [Dyn, Levin, and Rippa, 1986; Fang and Horstemeyer, 2006], and GP (also known as kriging) [Jones, Schonlau, and Welch, 1998].

Comprehensive and detailed surveys on response surface modelling for global optimization are provided by Jones [2001], Wang and Shan [2007], Shan and Wang [2009], and Lim, Jin, Member *et al.* [2010].

### 2.3.2 Gaussian processes

One reason to choose GP as a method to build response surfaces is that it is a non-parametric regression technique, which is an advantage over methods like PR, LIP, and MARS when dealing with black-box functions. Furthermore, GP provide analytical tractability not only for the predictions but also for the confidence on its predictions, an essential requirement for the Efficient Global Optimization algorithm to work (to be introduced in Section 2.4.2), which SVM and ANN can not provide. Finally, GP set a natural framework to incorporate old information for the dynamic case as is proposed in Section 4.2.

GP, just like all the other techniques, take as an input a set of observations (the dataset) that we shall denote as  $\mathcal{D}$ , which is composed by  $N \in \mathbb{N}$  pairs of a  $D$ -dimensional ( $D \in \mathbb{N}$ ) vector of independent variables ( $\mathbf{x}_n \in \mathbb{R}^D$ ) and their

corresponding observed response, or dependent variable,  $y_n \in \mathbb{R}$ . The available observations in  $\mathcal{D}$ , which is defined as

$$\mathcal{D} = \{(\mathbf{x}_n, y_n)_{n=1}^N\} = \{\mathbf{X}, \mathbf{Y}\}, \quad (2.1)$$

are considered a training set, from which the model learns. As an output, the model aims to provide a prediction or estimate of the dependent variable  $\hat{y}_p \in \mathbb{R}$  at any other test point  $\mathbf{x}_p \in \mathbb{R}^D$ .

Formally, a GP is fully defined by a mean function which allows introduction of any prior information available into the model, and a covariance function (or kernel) which expresses the scaled correlation between the data points [Rasmussen and Williams, 2006]. As a result of applying GP for regression to a dataset, we obtain a random function  $f_\ell$  distributed as a Gaussian process with mean function  $m(\mathbf{x})$ , and covariance function  $k(\mathbf{x}, \mathbf{x}')$ , which is denoted as

$$f_\ell \sim \mathcal{GP}(m(\mathbf{x}), k(\mathbf{x}, \mathbf{x}')). \quad (2.2)$$

Unless otherwise stated, throughout this thesis, a zero mean prior function

$$m(\mathbf{x}) = 0 \quad (2.3)$$

and the squared exponential covariance function

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\sum_{d=1}^D \frac{(x_d - x'_d)^2}{2\ell_d^2}\right) + \sigma_n^2 \delta(\mathbf{x}, \mathbf{x}') \quad (2.4)$$

are used, where  $\delta(\mathbf{x}, \mathbf{x}')$  is the Kronecker delta function, and is defined as

$$\delta(\mathbf{x}, \mathbf{x}') = \begin{cases} 0 & \text{if } \mathbf{x} \neq \mathbf{x}' \\ 1 & \text{if } \mathbf{x} = \mathbf{x}'. \end{cases} \quad (2.5)$$

The zero mean prior function is chosen —without loss of generality— to simplify the equations, however this is not a requirement, and arbitrary mean prior functions can be used, as it is done in Section 4.2.7. The kernel function can be viewed as a measure of distance or correlation between data points, which is to be

calculated for every possible pair of samples and gives rise to the covariance matrix  $\mathbf{K}$ . However, the covariance matrix will typically have some free parameters that, lacking any prior knowledge, can only be estimated from the data. Considering Equation (2.4), there are  $D + 2$  parameters to be learnt. These parameters include the signal variance  $\sigma_f^2 \in \mathbb{R}$  which determines the maximum covariance of the process, the measurement noise variance  $\sigma_n^2 \in \mathbb{R}$  which quantifies how noisy the process generating the samples is and can be set to zero for the deterministic case, and a characteristic length-scale for each dimension  $\boldsymbol{\ell} = [\ell_1, \dots, \ell_D] \in \mathbb{R}^D$  which dictates in what measure two data samples correlate to each other as a function of the distance separating them. For convenience, we aggregate all the parameters in the vector

$$\boldsymbol{\theta} = [\sigma_f^2, \sigma_n^2, \boldsymbol{\ell}] \in \mathbb{R}^{D+2}. \quad (2.6)$$

A common way for estimating  $\boldsymbol{\theta}$  is to choose the set of parameters  $\boldsymbol{\theta}^*$  that maximize the probability of the data being generated by a GP given such a set of parameters. This can be done by maximizing the marginal likelihood, or equivalently, the marginal log-likelihood of the parameters given the data, which is given by

$$\log(\mathcal{L}(\boldsymbol{\theta}|\mathcal{D})) = -\frac{1}{2}\mathbf{Y}^T\mathbf{K}(\boldsymbol{\theta})^{-1}\mathbf{Y} - \frac{1}{2}\log|\mathbf{K}(\boldsymbol{\theta})| - \frac{N}{2}\log(2\pi), \quad (2.7)$$

where special emphasis is put on how the covariance matrix depends on the chosen parameters by explicitly writing the dependency down.  $\boldsymbol{\theta}^*$  can then be defined as

$$\boldsymbol{\theta}^* = \underset{\boldsymbol{\theta}}{\operatorname{argmax}} (\log(\mathcal{L}(\boldsymbol{\theta}|\mathcal{D}))). \quad (2.8)$$

Once the parameters have been estimated by solving Equation (2.8), we can now calculate the kernel matrix that characterizes the process as

$$\mathbf{K} = \begin{bmatrix} k(\mathbf{x}_1, \mathbf{x}_1) & \cdots & k(\mathbf{x}_1, \mathbf{x}_N) \\ \vdots & \ddots & \vdots \\ k(\mathbf{x}_N, \mathbf{x}_1) & \cdots & k(\mathbf{x}_N, \mathbf{x}_N) \end{bmatrix}. \quad (2.9)$$

To estimate the objective function value at a new point  $\mathbf{x}_p$ ,  $\mathbf{K}$  is augmented

to include this additional data point, leading to

$$\mathbf{K}_p = \begin{bmatrix} \mathbf{K} & k(\mathbf{x}_1, \mathbf{x}_p) \\ & \vdots \\ k(\mathbf{x}_p, \mathbf{x}_1) & \cdots & k(\mathbf{x}_p, \mathbf{x}_p) \end{bmatrix}. \quad (2.10)$$

Then, the expected value for our prediction  $\hat{y}_p$  is given by

$$\hat{y}_p = m(\mathbf{x}) + k(\mathbf{x}_p, \mathbf{X})\mathbf{K}^{-1}\mathbf{Y}, \quad (2.11)$$

and the confidence about that estimate by

$$\sigma^2 = \text{Var}[y_p] = k(\mathbf{x}_p, \mathbf{x}_p) - k(\mathbf{x}_p, \mathbf{X})\mathbf{K}^{-1}k(\mathbf{x}_p, \mathbf{X})^T, \quad (2.12)$$

where  $k(\mathbf{x}_p, \mathbf{X})$  is a vector of the kernel function applied to  $\mathbf{x}_p$  paired to each point in  $\mathbf{X}$ . This allows us to characterize the prediction of the outcome  $y_p$  at the test point  $\mathbf{x}_p$  with a normal distribution rather than with a single value, which can be written as

$$y_p \sim \mathcal{N}(\hat{y}_p, \sigma^2). \quad (2.13)$$

A complete and formal description on GP is provided by Rasmussen and Williams [2006], and by MacKay [2003].

## 2.4 Sequential optimization

As pointed out in Section 2.2, when looking for the optimum of expensive-to-evaluate black-box functions, it might not be desirable to waste resources getting samples from non-promising areas, but rather to focus on regions that are more likely to contain the global optimum. The sequential sampling paradigm, as opposed to DoE where samples are chosen in one stage, achieves this by following the principle of allocating the resources in the most promising regions, however, it poses the problem of quantifying the goodness for the sample candidates. On the one hand, a candidate sample could be considered good if we anticipate that the response obtained at such

location will be better than what has already been observed, in which case the available knowledge would be exploited. But on the other hand, a candidate sample might be sought as valuable based on how much information about unexplored regions it would contribute, which might lead to future better decisions. As in many other areas of optimization, we face a natural trade-off between exploration and exploitation, which for the problem in question has been addressed by proposing different figures of merit aiming to balance this trade-off.

All of the figures of merit presented here require a way for predicting the response value at all points in the considered space, and most of them require a way for quantifying the level of uncertainty of the predictions made based on the available information. Furthermore, they assume  $\mathcal{D}$  contains enough samples to fit the required model, which we refer to as surrogate model or response surface, so that it can be used to make predictions.

#### 2.4.1 Sequential sampling policies

A naive way to choose the next best (or most promising) sample is to find the global optimum of the surrogate model and choose it as the next sample to be evaluated. This greedy strategy focuses purely on exploitation and fails to explore the solution space which makes it quite likely to get stuck in local optima.

A far better use of the surrogate, as shown by Jones, Schonlau, and Welch [1998], is to sample where the expected improvement (EI) is maximized. This is, for every point in space, the probability of observing a better candidate than the current best known is estimated based on the uncertainty of the available predictor, and is used together with the magnitude of the improvement to calculate the EI at each point. Then, the next sample is proposed to be taken where the EI is maximized. Schonlau [1998] proved that the EI criterion is equivalent to the one-step-ahead optimal sampling policy of the general n-step Bayesian global optimization proposed by Mockus [1994], but which is computationally infeasible. This technique is called Efficient Global Optimization (EGO) and, due to its simplicity in concept and good performance, has become a popular choice in literature with many variations and adaptations. A more detailed and technical treatment of EGO is provided in Section 2.4.2.

An information theory approach which accounts for the overall information gain on the optimizer obtained from a new evaluation has also been presented. This is known as informational approach to global optimization (IAGO), and uses conditional entropy as a measure for information [Villemonteix, Vazquez, Sidorkiewicz *et al.*, 2008; Villemonteix, Vazquez, and Walter, 2008]. The main advantage of IAGO is that it is able to quantify the amount of information obtained by a candidate sample, considering every possible outcome of the observation while taking into account the possible correlations with the rest of the samples, which allows it to select the point which maximizes the information acquisition. However, two main disadvantages must be highlighted. First, it requires the sample space to be discretized and then to evaluate the conditional entropy at each possible location, which makes it computationally infeasible for high dimensional problems. And second, by maximizing the information acquisition it concentrates mainly in exploration, which makes it suitable for obtaining a good approximation of the entire landscape but not necessarily for finding the global optimum in a reduced number of samples.

More recently, a generalization of EGO based on a dynamic programming approach known as knowledge gradient-policy for correlated beliefs (KGCB) was proposed by Frazier, Powell, and Dayanik [2008, 2009], and then extended for the continuous case by Scott, Frazier, and Powell [2011] under the name of knowledge gradient for continuous parameters policy (KGCP). Before taking any new sample, KGCP estimates a new response surrogate for a given possible sample (one step look ahead), and then compares the maximum of this anticipated estimation with the current maximum of the initial surface (as opposed to the maximum observed value used in EGO) in order to calculate the expected gain that would have been achieved had a particular sample been taken. Even with the approximations introduced in Scott, Frazier, and Powell [2011], this technique is computationally very intensive due to the necessity of having to fit a new GP for each potential candidate sample.

The remaining work on this thesis is based on EGO because it is well established in the field and has proven to be useful in a wide variety of applications, requires far less computational resources than KGCP, and provides a more analytically tractable framework than IAGO.

### 2.4.2 Efficient Global Optimization

EGO looks for the sample that maximizes the expected improvement with respect to the currently best known sample, which is possible to calculate because the GP provides an analytic expression of the probability distribution for each predicted value.

Assuming we face a minimization problem, if we follow the notation from Section 2.3.2, and given  $\mathcal{D}$ , we are able to make a prediction of a response value at any input  $\mathbf{x}_n \in \mathbb{R}^D$  in the form of a normal distribution (Eq. 2.13). In order to calculate the expected improvement  $\mathbb{E}[I_{x_p}(y)]$  (Eq. 2.16) at the test point  $\mathbf{x}_p$ , the best observed value so far  $y^* = \min_{i=1}^N(y_i)$ , is taken as a reference. Then, the EI is given by the probability of the predicted value

$$\mathbb{P}_{y_p}(y) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{1}{2}\left(\frac{y - \hat{y}_p}{\sigma}\right)^2\right) \quad (2.14)$$

times the obtained improvement

$$I_{x_p}(y) = \max(y^* - y, 0) \quad (2.15)$$

integrated over all possible values better than  $y^*$ . Put together, this gives rise to

$$\begin{aligned} \mathbb{E}[I_{x_p}(y)] &= \int_{-\infty}^{y^*} I_{x_p}(y) \mathbb{P}_{y_p}(y) dy \\ &= (y^* - \hat{y}_p) \Phi\left(\frac{y^* - \hat{y}_p}{\sigma}\right) + \sigma \phi\left(\frac{y^* - \hat{y}_p}{\sigma}\right), \end{aligned} \quad (2.16)$$

where  $\Phi$  denotes the cumulative density function of the normal distribution, and  $\phi$  is its probability density function. The next sample  $\mathbf{x}_{n+1}$  is finally taken where the expected improvement is maximized, i.e. following

$$\mathbf{x}_{n+1} = \operatorname{argmax}_{\mathbf{x}_p \in \mathbb{R}^D} (\mathbb{E}[I_{x_p}(y)]) \quad (2.17)$$

and, together with the observed response, is added to  $\mathcal{D}$ . Figure 2.1 illustrates the concept of EI calculated at two different points for the 1 dimensional case.

This sampling strategy has proven to be successful in a variety of applications

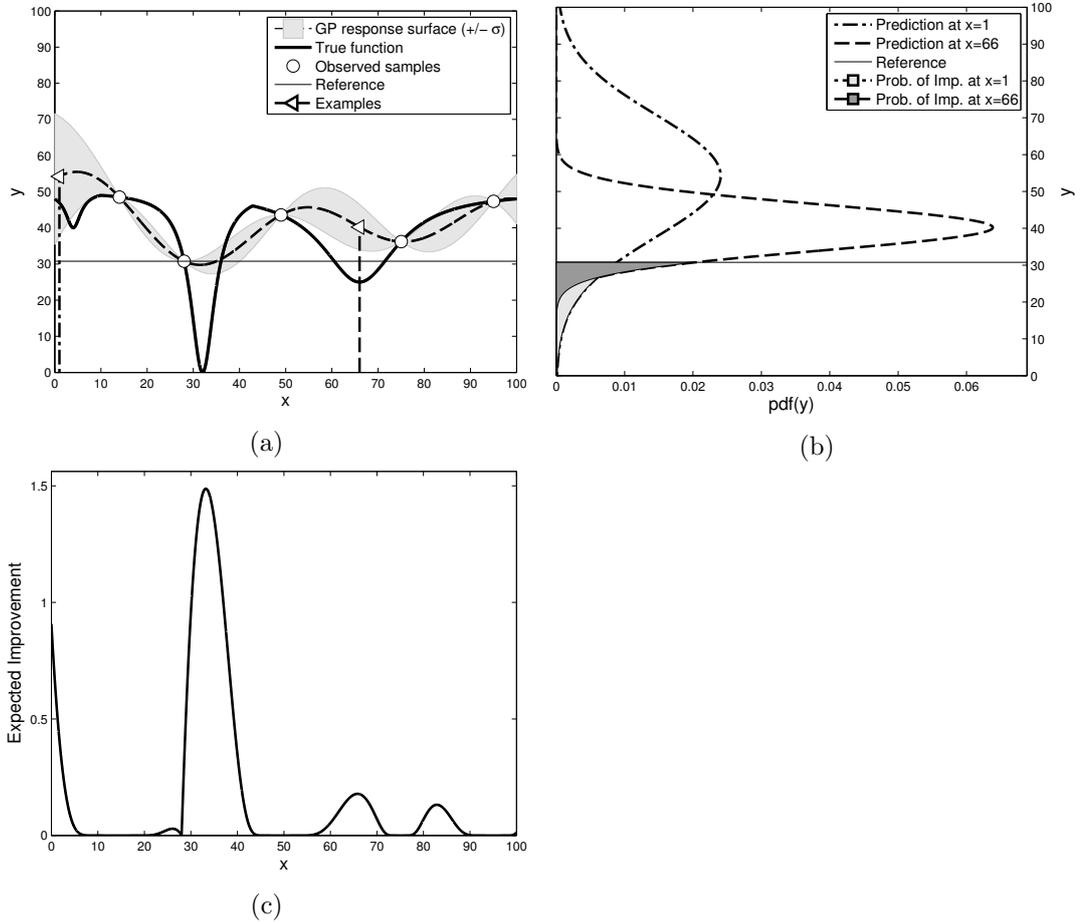


Figure 2.1: Expected improvement illustration for a 1D function. Fig. 2.1a shows an unknown function (bold continuous line) from which 5 samples are observed (white circles) and are used to fit a GP, whose mean and  $\pm 1$  standard deviation confidence interval are represented by the shadowed area with a dashed thick line. Two locations (white triangles) are chosen to illustrate the EI calculation. First, the best observed sample (minimum in this case) is taken as a reference (thin horizontal line), and then for each candidate, the probability of improvement is calculated. For the two selected points for this example, this probability of improvement is shown in Fig. 2.1b as two different overlapping shaded regions. Finally, the expected improvement (Eq. 2.16) is presented for all  $x$  in Fig. 2.1c, where it can be seen that the EI is maximized for  $x^* = 32$ , which should be considered as the next most promising sample.

for static problems [Biermann, Weinert, and Wagner, 2008]. Nevertheless, finding the global maximum of the EI is a challenging task, since it can be a highly multi-modal function, so in the next section we propose how to do so in a practical manner.

## 2.5 Expected improvement maximization

So far, it would seem that all EGO does is to replace the original problem of finding the global optimum of a (potentially multi-modal) function with a series of problems of the same kind, since the EI needs to be maximized at each iteration. However, challenging as it might be to maximize the EI, it is important to keep in mind that evaluations of the EI improvement function are relatively very cheap as compared to the original expensive-to-evaluate black-box function, which means that more traditional methods can be applied.

### 2.5.1 Expected improvement maximization techniques

Jones, Schonlau, and Welch [1998] originally proposed to use a branch-and-bound algorithm<sup>2</sup> to maximize the EI for low dimensional problems, and a limited memory version of the same algorithm for higher dimension problems. Later, Jones [2001] proposed to use a multi-start local maximization method where the starting points are chosen according to an heuristic based on finding midpoints of line segments connecting pairs of sampled points. This method was first mentioned by Schonlau [1998] but not explained.

More recent implementations of EGO usually either do not mention the procedure used to solve this auxiliary problem, or just mention the technique. For example, Ranjan, Bingham, and Michailidis [2008] uses a genetic algorithm fine-tuned with a local hill climber, and Kleijnen, Beers, and Nieuwenhuyse [2011] use either “a space-filling design with candidate points or a global optimizer such as the genetic algorithm in Forrester, Sobester, and Keane [2008] (p. 78.)”.

Perhaps, this omission is due to the fact that Mockus [1994] stated that “...there is no need for exact minimization of the risk function. We can use some simplest methods such as Monte Carlo...”, where risk function refers to a generalized version of the EI. This statement was repeated then by Schonlau [1998]. In order

---

<sup>2</sup>The details of the bounding are well detailed in the paper, but they are not sketched for the branching part of the algorithm

to explore whether this statement is true, in the remaining of this section we devise a simple experiment to test this hypothesis.

### 2.5.2 Testing the need for good quality EI optimizers

In order to test the impact of finding the global optimum of the EI function, we devise a simple experimental set-up, where the goal is to maximize a 1-dimensional multi-modal objective function through sequential optimization starting with  $\lambda = 4$  initial random samples, and implementing the EGO algorithm to progressively choose the next sampling locations.

Two sets of experiments are considered. The first set consists of three experiments using different quality solutions for the EI according to the ranking. To do this, at each iteration of the EGO algorithm, when the EI shall be maximized, all the local maxima are found, ranked in decreasing order, and called the  $k^{\text{th}}$  best solution ( $k = 1$  being the global optimum). The three experiments in the first set are for  $k = \{1, 2, 3\}$ .

The second set has only two experiments of which one is the reference experiment ( $k = 1$ ), and the second one alternates randomly, with equal probability, between  $k = 1$  and  $k = 2$ , so as to simulate what would happen with a maximization method that finds the global optimum half of the time, while the other half still returns a good solution (second best local maximum).

Unlike in higher dimensions, finding all the local maxima of a function and ranking them can be done by exhaustive search using a discretization of the space whose resolution is fine enough to capture all the characteristics of the EI function, which for this experiment can safely be assumed to be smooth, given that it is based on a GP with squared exponential kernel (Eq. 2.4). We consider a local maximum to be any  $x$  for which the associated EI value is strictly greater than both of its neighboring points in the discretized space. This is why we restrict this experiment to the 1 dimensional case.

The chosen objective function is

$$f(x) = \max_{p=[1..P]} \left( \frac{h_p}{w_p(x - x_p^*)^2 + 1} \right), \quad x \in [0, 100], \quad (2.18)$$

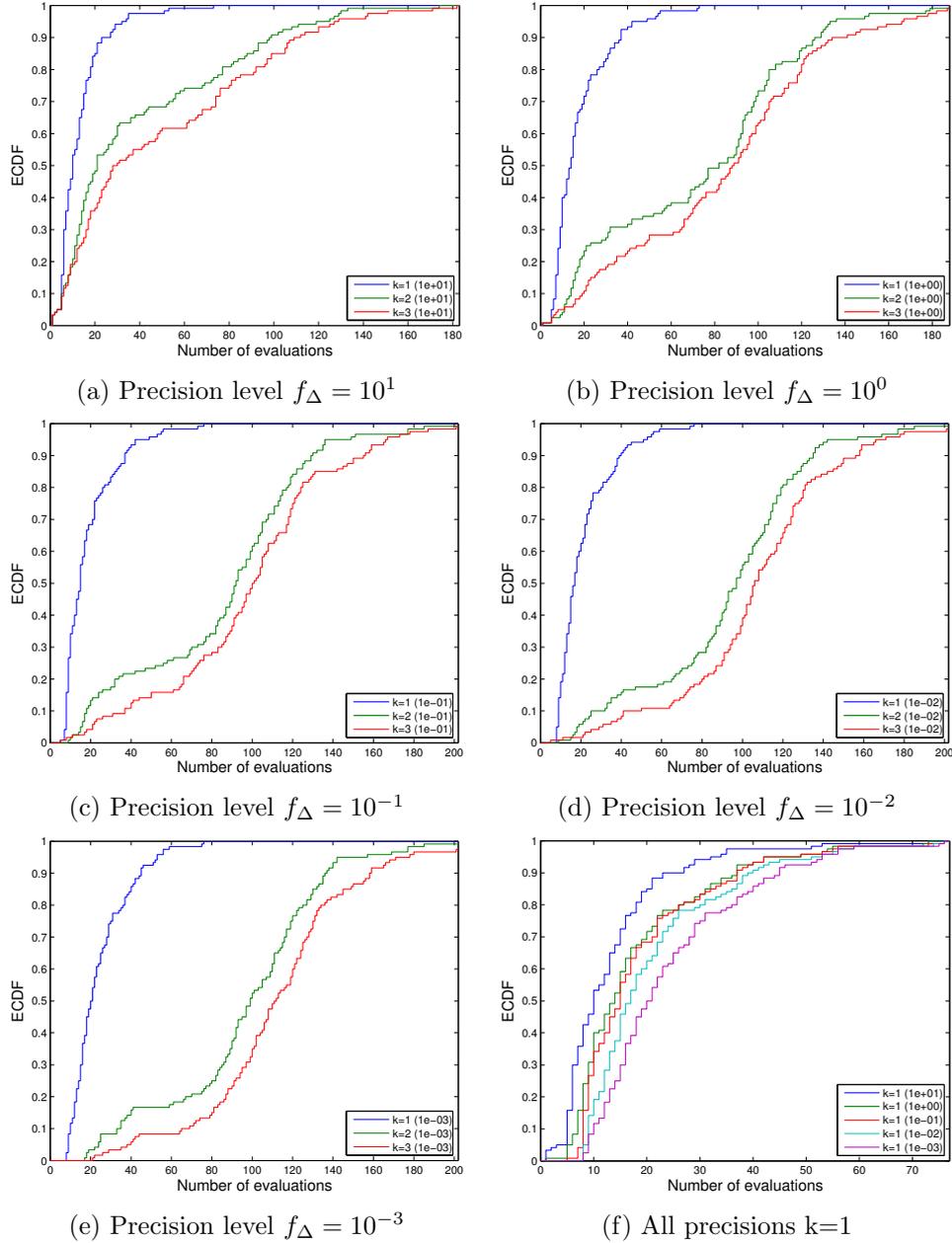
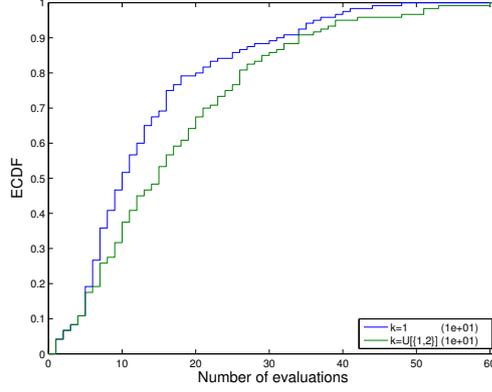
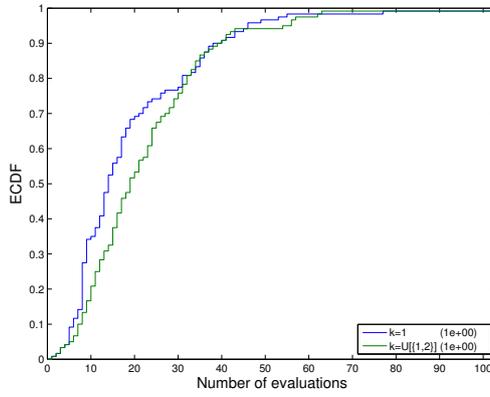


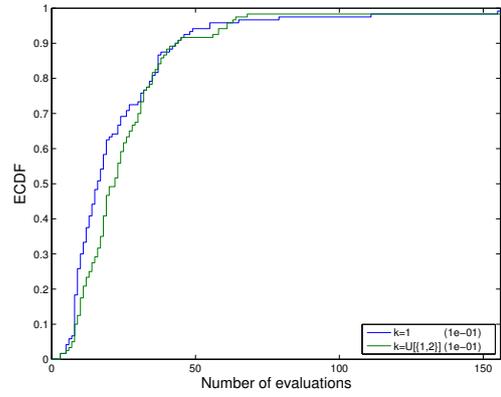
Figure 2.2: Figs. 2.2a through 2.2e compare the performance of three implementations of the EGO algorithm using the ECDF (see Section 2.6.2 for a full explanation on this error measure). The blue line stands for the implementation using the global maximum of the EI ( $k = 1$ ), the green line for the one using the second best ( $k = 2$ ), and the red line for the one using the third best local maximum of the EI ( $k = 3$ ). Each sub-figure presents the results for different precision levels ( $f_{\Delta} = \{10^1, 10^0, 10^{-1}, 10^{-2}, 10^{-3}\}$ ). For all considered cases, the implementation where  $k = 1$  largely outperforms the rest. Fig. 2.2f compares the performance of the reference implementation ( $k = 1$ ) through all the considered precision levels.



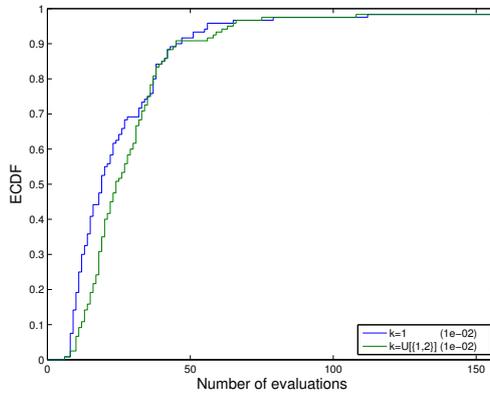
(a) Precision level  $f_{\Delta} = 10^1$



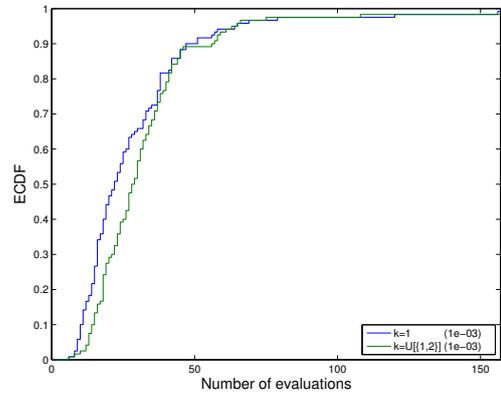
(b) Precision level  $f_{\Delta} = 10^0$



(c) Precision level  $f_{\Delta} = 10^{-1}$



(d) Precision level  $f_{\Delta} = 10^{-2}$



(e) Precision level  $f_{\Delta} = 10^{-3}$

Figure 2.3: ECDF plots (see Section 2.6.2 for a full explanation on this error measure) comparing the EGO implementation for the reference case (blue line), where the best peak of the EI is used ( $k = 1$ ), and the case where either the best or second best local maximum of the EI are used with probability  $\frac{1}{2}$  respectively ( $k \sim U(\{1, 2\})$ ) (green line) for different precision levels ( $f_{\Delta} = \{10^1, 10^0, 10^{-1}, 10^{-2}, 10^{-3}\}$ ). At all considered precision levels, using the best solution for the EI provides a noticeable advantage.

a combination of  $P$  inverse squared exponentials, each contributing with a peak (or local maximum) and parametrized by its corresponding width ( $w_p \in \mathbb{R}^+$ ), height ( $h_p \in \mathbb{R}^+$ ), and peak location ( $x_p^* \in \mathbb{R}$ ). This function is also used by the Moving Peaks Benchmark, which is explained in detail in Section 4.3.1 and was first introduced in Branke [1999].

To ensure the results are statistically significant,  $R = 20$  replications are run for each number of peaks  $P \in [1 \dots 6]$ . We refer to each individual combination of number of peaks and random realization of parameters as an instance, and to the aggregate of all replications for all number of peaks as an experiment. At each instance, the function parameters are drawn from a uniform random distribution, ensuring that the same instances are presented across different experiments, as follows:

$$h_p \sim U[30, 70], \quad (2.19)$$

$$w_p \sim U[0.01, 1], \text{ and} \quad (2.20)$$

$$x_p^* \sim U[0, 100]. \quad (2.21)$$

To evaluate the performance of each implementation, we use a precision target  $f_\Delta^*$ , defined as the real global maximum of the objective function  $f^*$  minus an acceptable precision level  $\Delta_f$ . For a given precision level, we can then measure the number of evaluations it took the optimizer to find the global optimum of the objective function up to this precision level. By tracking the number of function evaluations required to achieve a precision target for every instance, we can then build the empirical cumulative density function (ECDF), which enables for fair and straightforward comparisons among different optimization methods. Further explanations and justifications for this performance method are given in Section 2.6.2.

Both experiments were run for different precision levels ( $f_\Delta = \{10^1, 10^0, 10^{-1}, 10^{-2}, 10^{-3}\}$ ). The result of the first set of experiments is presented in Figure 2.2 where Figure 2.2a through 2.2e show how the optimizer using the global optimum of the EI at each iteration ( $k = 1$ ) largely outperforms those using the second and third best by having found the global optima at all times in less than 80 function evaluations even for  $f_\Delta = 10^{-3}$ , while the two others can not find it in some oc-

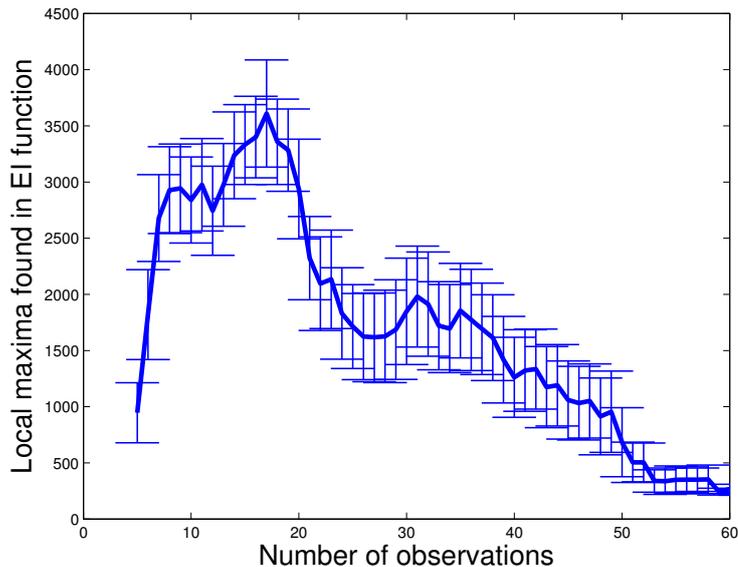


Figure 2.4: Number of local maxima found in the EI function with a discretization grid of step  $10^{-5}$  throughout the second set of experiments as a function of the number of observations used to build the response surface. Error bars represent  $\pm 1$  standard error. The decreasing number of local maxima in the EI function towards the end of the experiment is due to a finite size effect.

casions even after 180 function evaluations. Such a large difference occurs because the optimizer is pushed away from the region where the global maximum is, by not allowing it to sample ever at the most promising region. This bias is what motivated the second set of experiments, whose results are presented in Figure 2.3 where the difference between the reference experiment and the one choosing randomly, with equal probability, between the best and second best local maximum of the EI, is less pronounced, yet still noticeable across the tested precision levels.

As a reference, Figure 2.4 shows the number of local maxima found in the EI function throughout the second set of experiments as a function of the number of observations used to build the response surface, using a discretization grid one order of magnitude finer ( $10^{-5}$ ) than the stopping criterion precision of  $10^{-4}$ . The decreasing number of local maxima in the EI function towards the end of the experiment is due to a finite size effect, since simulations converge at different rates.

In an uncontrolled environment and for higher dimensions, it is unlikely that an optimizer using a local hill climber with multiple starting locations chosen at random would find the global optimum (or even the second best), given the highly

multi-modal nature of the EI function (see Figure 2.6 for an illustration).

These simple experiments demonstrate that the EI maximization is an important part of the EGO algorithm, and suggest that enough care must be placed during this task when implementing the EGO algorithm. This is why we devote Section 2.6 to compare the impact of using different maximization techniques for the EI in a more elaborated and widely accepted benchmark.

## 2.6 Selecting an EI maximization technique using the Black-Box Optimization Benchmark

As shown in Section 2.5.2, it is important to find the global maximum of the EI, however, allocating too many resources to this task would be time consuming. The purpose of this section is to determine which optimization technique shall be used throughout the remainder of this thesis when confronted with the EI maximization auxiliary problem so as to take the most advantage out of it. To achieve a thorough and fair comparison, we first introduce the black-box optimization benchmark (BBOB), which provides a comprehensive set of functions and performance measures to quantify and compare numerical optimization methods. Then, we describe in detail the algorithms to be compared along with the numeric implementation details associated to them. Finally, the results of the experiments are presented and analysed.

### 2.6.1 The Black-Box Optimization Benchmark

The BBOB provides a well-motivated single-objective set of benchmark functions, designed to test different aspects of numeric global optimizers along with some suggested performance measures to facilitate the comparison of different algorithms. Both, a deterministic (noiseless), and a stochastic (noisy) version of the functions are proposed by BBOB, but only the former is considered for the experiments described hereafter. The complete description of the benchmark, including the motivation, implementation details, and justification for each function in the testbed can be found in Hansen, Auger, Finck *et al.* [2010].

All the functions in the benchmark are scalable with the dimension and are

defined for  $\mathbf{x}_n \in \mathbb{R}^D (D > 2)$ , but they have an artificially chosen global optimum ( $f^*$ ) within the compact  $[-5, 5]^D$ , which serves as bounds for the search space. By providing an instance index to the functions ( $i \in \{1, \dots, 15\}$ ), different parametrizations of the functions can be instantiated, allowing for the optimizers to be tested on different versions of the same function so as to provide statistically relevant results. BBOB considers 24 functions ( $f_1^i, \dots, f_{24}^i$ ), all of which are thoroughly explained in Finck, Hansen, Ros *et al.* [2010], each contributing with a unique feature so that valuable information about the tested algorithms can be extracted.

### 2.6.2 Performance measures

There are typically two methods of measuring the performance of the implementation of a numeric optimizer.

The first one is to fix a budget (measured in running time, function evaluations, computational time, or any other cost unit), and then measure the quality of the best outcome returned by the optimizer with respect to the real global optimum. This seems close to real life needs, where it is often the case that there is a limited budget to solve a particular problem of which the best possible value is not known<sup>3</sup>.

The second common option is to fix a desired quality of a solution to be considered as optimal (target), and then measure the required budget to achieve it, which requires to know in advance the global optimum of the function.

When solving an unknown problem, the former method is preferred, since the latter would prove impractical due to the requirement of knowing the solution in advance. However, the latter provides a quantitative interpretation of the performance by allowing direct comparisons among algorithms. This is better sketched in Figure 2.5, where the error (measured as the difference between the real global optimum  $f^*$  and the currently best known observation) is plotted as a function of the number of evaluations for three different algorithms trying to optimize the same function  $f$ . In this figure, fixing a target means choosing some acceptable level of error, tracing a horizontal line at such level, and reading in the horizontal axis the number of function evaluations required to reach the chosen target. For example,

---

<sup>3</sup>It could be the case that the optimal value of the objective function is known (e.g. from theoretical reasons), but not the arguments of the function for which the function takes this optimal value, so an optimization would still be required.

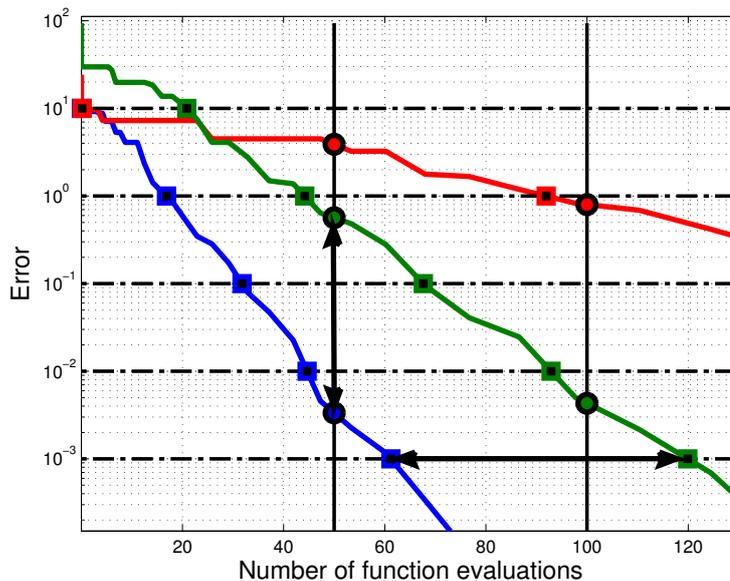


Figure 2.5: Fixed target vs. fixed cost performance measurement.

if we were to compare the performances of the blue and the green optimizers at a precision level of  $10^{-3}$  —as shown by the thick horizontal line with arrows—, we could read that the blue one requires 60 function evaluations to reach it, while the green one needs 120, concluding that it took twice as many function evaluations to the green one as compared to the blue one. This provides a quantitative comparison that can not be achieved with the fixed budget performance measure. For instance, say we fix the budget at 50 function evaluations (by tracing a vertical line), and that we want to compare the same two optimizers. We would then read that, for such a budget, the optimizers achieved an error of 0.003 and 0.5 respectively, which does not give much insight in the relative performances. So, for benchmarking purposes, since the global optimum is known, the fixed target method is preferred for performance comparison.

The goal of evaluating the optimizers in a benchmark is to determine which method should be preferred under unknown conditions. By collecting the resulting performances after running each optimizer for several instances for each of the proposed functions, we can aggregate the results and build the empirical cumulative density function (ECDF). The ECDF is built by counting the number of instances that reached a specified target at each number of function evaluations, which results

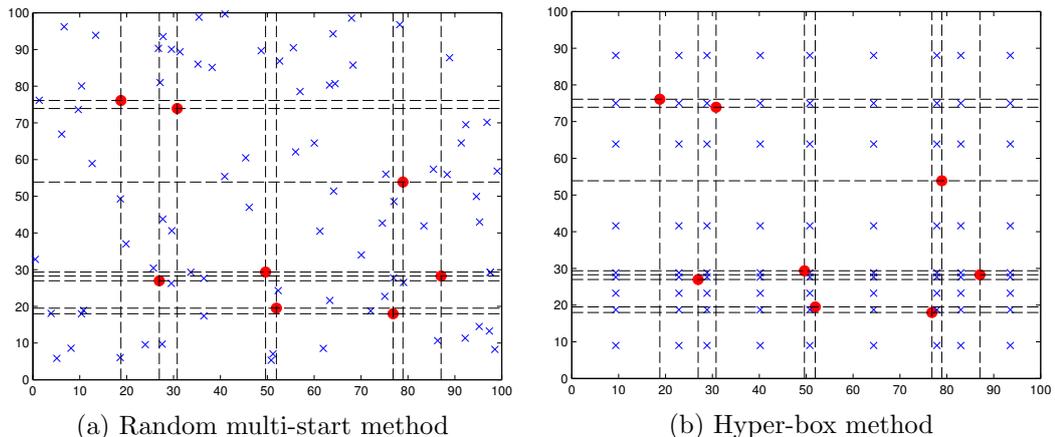


Figure 2.6: Eight random samples in  $2D$  (circles) with dashed lines showing how the (hyper-)boxes are created. Figure 2.6a shows  $\zeta = (|\mathcal{D}| + 1)^D = 81$  randomly allocated starting points (crosses), many of which fall in the same box, leaving other boxes unexplored. Figure 2.6b allocates the same 81 starting points at the center of each (hyper)-box generated by every two neighbor samples, according to the method described in Section 2.6.3.

in a step function that jumps by  $\frac{1}{\eta}$  at each of the  $\eta$  collected results. The ECDF is a useful method for aggregating the observed performances because it can be interpreted as the probability for the optimizer of finding the fixed target value. As its name suggests, the ECDF is built on sample observations, and it is an unbiased estimator of the fraction of successful runs at each time point (or number of function evaluations). Moreover, it provides a practical and reliable way of comparing mean performance of several algorithms. Figure 2.2f is an example of the ECDF used to display the performance of one numeric optimizer for several precision targets, while Figure 2.2e illustrates how the ECDF can be used to compare two different numeric optimizers for a fixed precision target.

### 2.6.3 EI maximization algorithms comparison

Based on Section 2.5.1, the selected methods to be compared for maximizing the EI criteria at each iteration of the EGO algorithm are random multi-start local hill climbers, genetic algorithms, and the hyper-box method, all of which are described below.

- The random multi-start with local hill climbers (RMS) method is a standard and generic approach used in global optimization when no information about the function to be optimized is available [Solis and Wets, 1981]. This method selects  $\zeta$  locations at random, uniformly distributed throughout the domain space, to be used as starting points of local optimizers. Each local optimizer implements the Nelder-Mead method (or simplex algorithm) [Nelder and Mead, 1965]. A disadvantage of this method to maximize the EI is that, as observations are incorporated to  $\mathcal{D}$ , new local maxima can be created between neighboring samples. Jones, Schonlau, and Welch [1998] explicitly discourage the use of this method for the same reason. Even with a large number of starting points for the local hill climbers, the probability of leaving unexplored at least one hyper-box (defined by a pair of neighboring samples), increases exponentially with the cardinality of  $\mathcal{D}$ , as is illustrated in Figure 2.6a.
- The hyper-box multi-start with local hill climbers (HB) method exploits the structure of the EI function by selecting the starting points at the center of each hyper box created by every pair of neighbor samples, which requires  $\zeta = (|\mathcal{D}| + 1)^D$  starting points. Their initial step size is chosen such that each local hill climber, implementing the Nelder-Mead method as well, does not leave its corresponding hyper-box in the first step, forcing it to explore the locality. This method is still a heuristic and does not guarantee to find the global maximum of the EI, but it shows better performance than random multi-start locations. An illustration of the HB method for a 2 dimensional problem is shown in Figure 2.6b. The HB sampling method is loosely based on a method originally proposed by Jones [2001], which is shortly sketched in the Appendix, however, such a method is based on finding the midpoints of line segments connecting pairs of sampled points, and then applying a clustering method, while the HB method generates a starting point in between any possible pair of samples.
- The genetic algorithm (GA) method starts by generating an initial set of solutions called population, which are selected uniformly at random from the whole search space. The initial population is then evaluated using the objec-

tive function (EI in this case), and the individuals of this initial population are ranked according to their performance (or fitness). The algorithm then creates a sequence of new populations, which are called generations. At each step, the new population is created based on the population of the previous generation by creating three different types of children. The first type is elite based, and is performed simply by selecting the 2 best individuals of the previous population. The second type, generated by crossover, accounts for 80% of the population and is the result of randomly selecting pairs of parents out of the entire pool of individuals from the previous generation, with a distribution reflecting the individual's fitness, and combining their vector components (genes) through a weighted average with random weights. This procedure is known as scattered crossover. The final type of children are generated by mutation, which accounts for the remainder of the population. This is done by randomly selecting single parents from the same distribution as in the crossover operation, and adding Gaussian noise to them with zero mean and standard deviation 1. The whole process is then repeated for several generations.

In order to make the number of EI function evaluations simple to compare, the population size and the number of generations for which the process is repeated are kept constant for the experiments described in this chapter. The number of generations and the population size are calculated so that the total number of EI function evaluations used by the RMS method can be attained. From the experiments performed, it was calculated that 400 generations, each of 400 individuals, satisfy this requirement. Nevertheless, for further chapters, we propose to dynamically adjust these parameters depending on the size of the problem to be solved. So, in Chapter 3, instead of using a constant size for the population and the number of generations, these parameters are set to  $\lceil v\sqrt{ND} \rceil$ , where the user defined EI maximization budget constant  $v$  is inferred from the data collected during the experiments performed at the current stage (comparison between HB, RMS, and GA).

Since the real interest is not in maximizing the auxiliary problem, but rather in finding the technique which best enhances the search of the main problem at

a reasonable cost, a direct comparison of the performance of these three methods would not provide much insight. Besides, comparing absolute values of the EI across different instances or different algorithms is not meaningful since the EI is a function relative to the course of action taken (observed samples). Furthermore, as opposed to the functions of the benchmark, the global maximum of the EI at each iteration is not known, which discards the possibility of using fixed target performance measures.

Instead, we propose to use an indirect measurement to assess the performance of the maximization methods in question by fixing the number of evaluations of the EI function allowed, and measuring the performance achieved by the whole EGO optimizer on the BBOB functions. In this way, the fixed target performance measure explained in Section 2.6.2 can be used.

For the RMS method, the number of starting points can be arbitrarily selected, and this is chosen to match the number of starting points required by the HB method, which has a fixed number of starting points at each iteration. Yet, for either of these methods, the total number of function evaluations<sup>4</sup> of the EI can not be fixed, since it depends on how many of them are required by each local hill climber to converge.

To achieve a fair comparison, we keep track of how many EI function evaluations are used, on average, throughout an entire run of the BBOB functions by the RMS and the HB methods, and use the maximum of this number to set the maximum budget for the GA method. This turns out to be equivalent to a population size of 400 individuals and 400 generations, on average, which are the values used for these experiments. With an equivalent maximum budget now specified, the whole experiment can be run. The selected functions from the BBOB for this experiment are  $f_1^i, f_2^i, f_5^i, f_7^i, f_8^i, f_9^i, f_{14}^i, f_{19}^i, f_{21}^i$ , and  $f_{22}^i$  ( $i \in \{1, \dots, 15\}$ ). The reason for restricting the study to these functions is provided in Section 2.6.4. All the optimizers start with a random uniform set of  $\lambda = 4$  observations, and are allowed to run for up to  $N = 80$  samples. The complete description of the benchmark, including the motivation, implementation details, and justification for each function

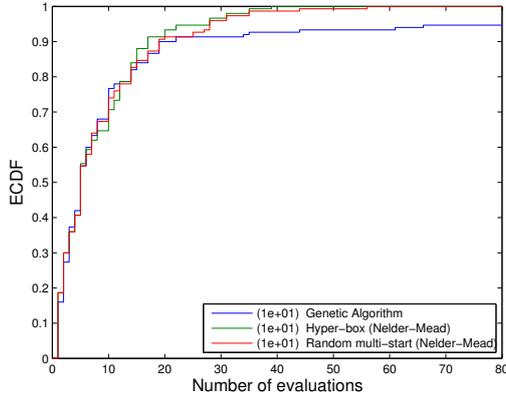
---

<sup>4</sup>The total number of EI function evaluations is the sum of the EI function evaluations performed throughout the full sequential sampling experiment.

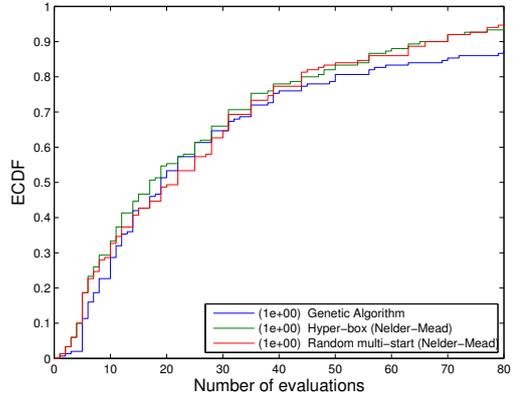
in the testbed can be found in Hansen, Auger, Finck *et al.* [2010].

The results of the experiment comparing the three proposed methods for maximizing the EI within the EGO algorithm are presented in Figure 2.7 for six precision levels. It is observed that in all cases the HB method outperforms the others, with the RMS coming in second place, and the GA consistently in last place. However, it is to be highlighted that, even if the GA method was allowed as many function evaluations as the other two methods, an additional stopping criteria of the GA algorithm, which prevents the algorithm from continuing if the relative improvement over the best fitness value from one generation to the next is less than the specified tolerance (here  $10^{-6}$ ), decided to stop before, saving a significant amount of computational effort. More precisely, the GA method used less than 10% of the total number of EI function evaluations with respect to the number used by either the RMS or HB methods. Furthermore, by tracking the number of EI function evaluations performed at each EI maximization iteration, we can fit the EI maximization budget to be used in later experiments, which from the obtained data was found to be  $v = 25$ .

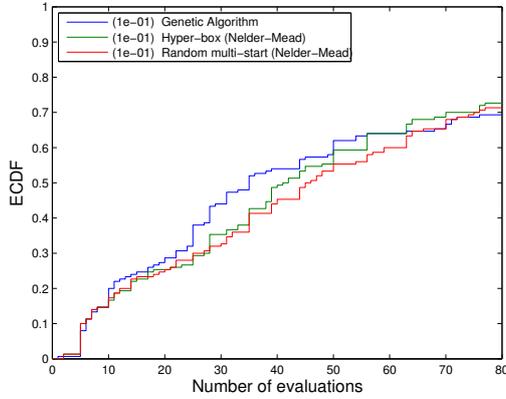
In order to verify these results, an additional experiment, in which the budget of EI function evaluations is limited by the number required by the GA method, is proposed as a control experiment. To impose the limit on the number of EI function evaluations to match the one used by the GA method by the RMS and HB methods (on average), we track the average number of EI function evaluations performed by each local hill climber, and then determine the number of allowed starting points assuming each local hill climber would require the average number of EI function evaluations. For the RMS method, the number of starting locations can be naturally set. However, the HB method requires a fixed number of starting points. This leads to two different possible solutions, both of which are tested. The first one picks a fixed number of starting locations for the local hill climbers at random from the full set of initial starting points as determined by the original HB method. The second one, uses the full set of starting locations, but limits the total number of steps that each local hill climber is allowed. The results of this experiment are shown in Figure 2.8, which shows that the best strategy is the HB method with limited number of EI function evaluations but using all the required



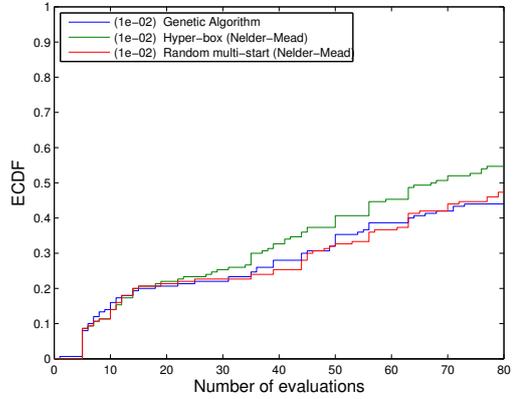
(a) Precision level  $f_{\Delta} = 10^1$



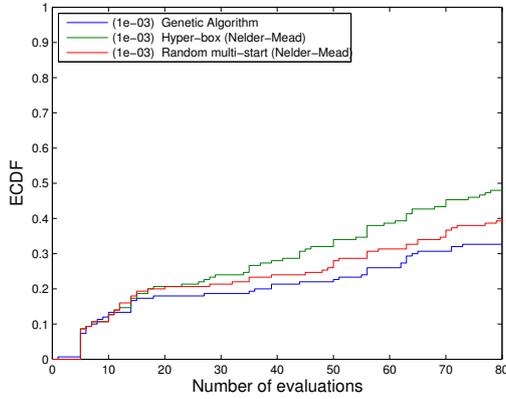
(b) Precision level  $f_{\Delta} = 10^0$



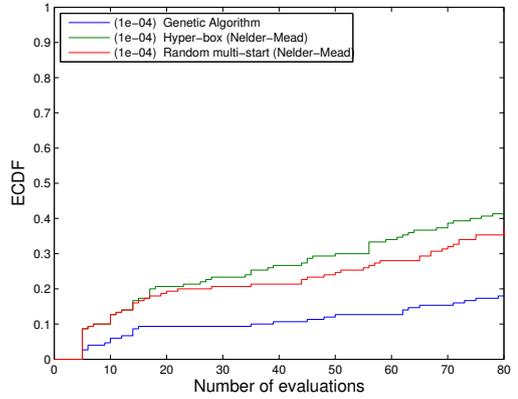
(c) Precision level  $f_{\Delta} = 10^{-1}$



(d) Precision level  $f_{\Delta} = 10^{-2}$



(e) Precision level  $f_{\Delta} = 10^{-3}$



(f) Precision level  $f_{\Delta} = 10^{-4}$

Figure 2.7: ECDF as a function of the number of objective function evaluations comparing, at different precisions, three methods (RMS, HB, and GA) used to optimize the EI auxiliary problem of the EGO algorithm applied to a subset of the BBOB functions, in  $2D$ . The HB method outperforms the others, however, the total number of EI function evaluations used by GA is less than 10% of either of the two other methods.

available starting points. Given that the number of required starting points scales exponentially with the dimensionality of the problem, the HB method—even with restricted budget for each local hill climber—is not recommended for maximizing the EI when dealing with problems of higher dimensionality.

#### 2.6.4 Practical implementation details

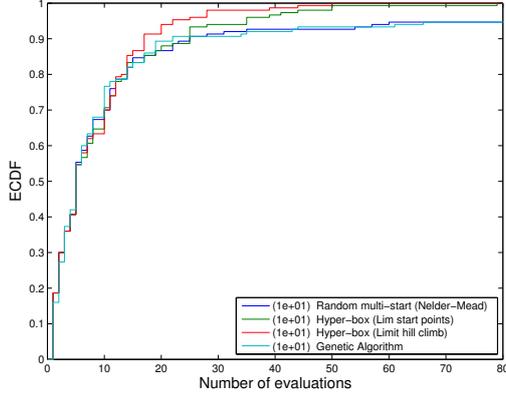
When implementing the algorithms described in the previous sections, some considerations must be taken into account, which we address in the following paragraphs and are related to the GP parameter fitting, to the selection of the initial sample size, to the selection of a subset of functions from the BBOB, and to the reason why we restrict the study in this chapter to the 2 dimensional case.

##### Gaussian process parameter estimation

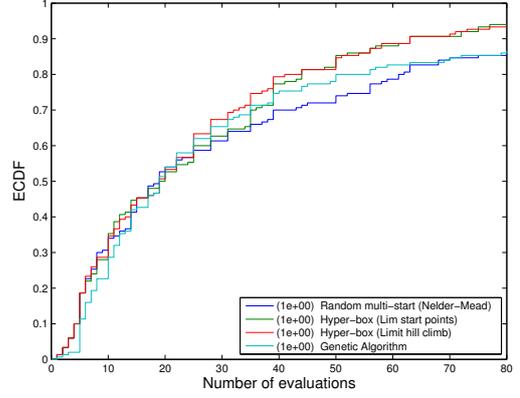
When estimating the parameters of the GP given  $\mathcal{D}$  (Eq. 2.8), the likelihood function (Eq. 2.7), must be evaluated several times. This involves a matrix inversion, which can be numerically unstable. The Cholesky decomposition can be used, which apart from being faster, increases the numerical stability. A detailed algorithm for doing so is presented in Rasmussen and Williams [2006] (Algorithm 2.1).

But even when using a more stable numerical algorithm for calculating the likelihood, it is possible to reach a singularity, for instance, by having a repeated sample, which would make the kernel matrix singular, hence impossible to invert. The case where two identical samples are taken is in theory not possible when using the EGO algorithm, since the EI is exactly zero at locations where an observation has been made. In fact, Locatelli’s proof for the convergence of the EGO algorithm in infinite time relies on this property [Locatelli, 1997]. Yet, in practice, it is very common to have two samples arbitrarily close to each other, rendering the kernel matrix close to singular, and this is specially common when sampling from a good region, since samples tend to be taken close to each other. Some implementations solve this problem by allowing the measurement noise variance ( $\sigma_n^2$ ) to be non zero, however, this parameter has a global impact on the model.

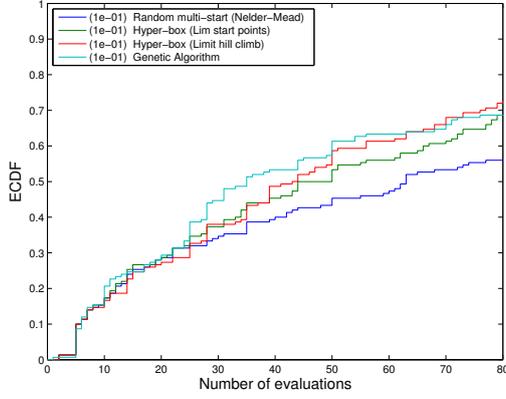
Instead, what we propose, is to detect when the kernel matrix is getting close



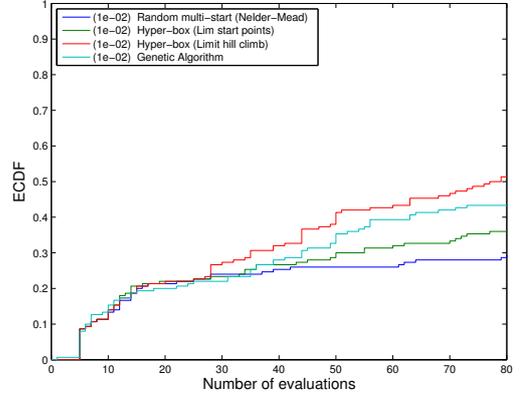
(a) Precision level  $f_{\Delta} = 10^1$



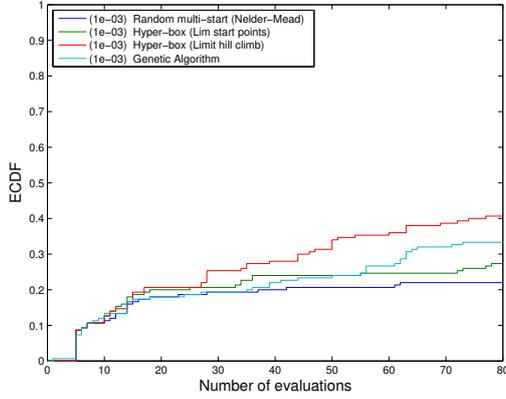
(b) Precision level  $f_{\Delta} = 10^0$



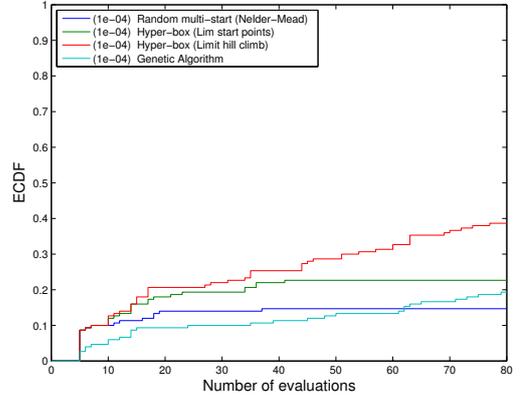
(c) Precision level  $f_{\Delta} = 10^{-1}$



(d) Precision level  $f_{\Delta} = 10^{-2}$



(e) Precision level  $f_{\Delta} = 10^{-3}$



(f) Precision level  $f_{\Delta} = 10^{-4}$

Figure 2.8: ECDF of the control experiment comparing the three proposed EI maximization techniques under similar budget in terms of total number of EI function evaluations allowed throughout the experiment. Two ways of restricting the budget for the HB are tested: restricting the number starting points, and restricting the total number of EI function evaluations each local hill climber is allowed. The latter clearly outperforms all the other tested techniques.

to singular by testing the reciprocal of the condition number<sup>5</sup> against an arbitrary tolerance value ( $10^{-17}$  in this implementation), and if this is the case, removing one of the points from the pair causing this issue, i.e., from those which are the closest to each other. The removed point is the one with the worse response value out of the pair. Furthermore, so that no new samples are taken in the vicinity, a record of this pair of points, along with the spherical region defined by the two points being antipodal to each other, is kept and designated as a taboo region. This means that, within this sphere, the EI is exactly zero to prevent any further sampling within the taboo regions.

The spherical shape of the taboo region was selected to simplify the definition of the region and to accelerate the verification process of points belonging to the region. The choice of the points defining the sphere relies on the assumption that the objective function is monotonic in between the two points defining the region, but makes no assumption about the objective function outside the defined area, which would be the case had one of the samples been used as the center of the sphere (as opposed to lie on the perimeter). This monotonicity assumption leaves a positive probability that the global optimum lies within the taboo region. However, when dealing with engineering systems, local monotonicity and smoothness are reasonable assumptions, especially at such a small scale (relative to the target units).

### **Budget allocation determination**

There are two choices made about the budget of samples allocated for the experiments. First, the initial number of samples taken is decided to be  $\lambda = D + 2$  since it satisfies the required condition of having at least 2 samples to create a response surface—even if perhaps not a very accurate one—and because it is the suggested number by BBOB, without any proper theoretical justification. And second, the maximum allowed budget of  $N = 40D = 80$  (since only two dimensional experiments are considered in this section), was decided to keep the optimizers running within reasonable time, considering that many replications must be run for each experiment.

---

<sup>5</sup> The condition number is defined as the ratio of the largest singular value of the kernel matrix to the smallest. Singular values of a real or complex matrix  $\mathbf{M}$  are the square roots of the non-zero eigenvalues of  $\mathbf{M}\mathbf{M}^*$ , where  $\mathbf{M}^*$  is the conjugate transpose of  $\mathbf{M}$ . Large condition numbers indicate a nearly singular matrix. So, when the reciprocal of the condition number approaches 1, the kernel matrix is well conditioned, and when it approaches 0, it is close to singularity.

## Selected testbed

The selected subset of functions from the complete BBOB set of functions in which the experiment described in this section was performed was chosen so that the EGO algorithm performed sufficiently well to capture any useful measurements, i.e., for the functions left out from the experiment, the measured performance in terms of probability of finding the global optima within the specified tolerance level, was close to zero. This does not mean that the presented algorithm is bad as compared to other algorithms tested in the BBOB found in the literature, since most of them require thousands of evaluations to find good solutions. At this stage, handling so many samples would take too much time using the standard version of EGO, and this is precisely the problem addressed in detail in Chapter 3. The exact same reason holds for the chosen precision levels, which in general are set as low as  $10^{-8}$  in the competitions using the BBOB. Furthermore, it is important to keep in mind that the presented algorithm is meant to exploit the structure and correlations of the available data as much as possible, which is computationally expensive, but proves useful when dealing with expensive black boxes where the number of samples available is limited.

## 2.7 Results and concluding notes

This chapter introduced basic concepts such as global optimization, expensive black-box functions, and heuristic methods in order to set in context the ideas discussed in this thesis as well as to provide the basis upon which these ideas rely. In particular, the difference between DoE which mainly aims to get good approximations of an entire search space, and sequential sampling which focuses rather on detecting promising regions was highlighted. Then, the concepts of response surface building for optimization coupled with sequential sampling policies were introduced and reviewed. This allowed to justify the choice of GP as the preferred response surface building technique throughout this work together with the EI as utility function as a criterion to decide where observations should be made, which lead to the EGO algorithm, all of which were treated in detail.

The subproblem of maximizing the EI, which arises as part of the EGO

algorithm, was explored and some experiments were performed, first to test whether it is worth spending resources trying to obtain close-to-optimal solutions for the EI, and then to compare the performance achieved by three strategies when optimizing such a function, since the former experiments indicated it is worth investing in this task. The second set of experiments called for a more sophisticated benchmark and performance measures allowing to compare the output of the proposed optimizers. The BBOB was then introduced to provide a wide set of functions. Furthermore, two paradigms to measure the performance of the optimizers, fixed budget and fixed target, were explained, both of which are useful in different contexts. However, due to the quantitative interpretation it provides, and the fact that the global optima of the benchmark functions are known, the fixed target performance measure was preferred. Following the fixed target performance measure, the ECDF was presented as a way of transforming the output to a probability empirically calculated from the available sample experiments, simplifying the comparison of different algorithms.

From the outcome of the second set of experiments, we concluded that the HB method outperformed the RMS and GA methods in all tested cases, all of which were done in  $2D$  given the high computational cost of the method. This high computational cost is due to the fact that both the HB and the RMS methods require a number of starting points that increases exponentially with the number of samples in the dataset and the dimensionality of the problem, which makes it difficult to use for hard problems that require many observations, or for high dimensional problems. Furthermore, it was noted that the GA method, even if achieving a lower performance as compared to HB and RMS, still provides acceptable solutions while only requiring 10% of the budget. Given the bad scaling of either the HB or RMS algorithms in higher dimensions, and the relatively low compromise in performance achieved by the GA, we conclude that, for harder problems, using a GA is a good option to maximize the EI, which is why this will be the case all throughout Chapter 3.

Finally, we presented some remarks about the numerical implementation of the algorithms discussed in this chapter and explained some choices regarding the budget allocation and the subset of functions selected from the BBOB, which otherwise might have seemed arbitrary, and we pointed out that the apparently bad

performance of the EGO algorithm when tested in these functions is not necessarily bad, since this algorithm is tailored for expensive black-box functions.

In the next chapter, some modifications to the EGO algorithm are proposed in order to adapt it to harder problems, such as those already introduced by the BBOB.

## Chapter 3

# Space partitioned EGO for accelerated response surface modeling

Given the recent success of GP generated response surfaces to optimize expensive-to-evaluate black-box functions, it is natural to wonder whether these techniques can be applied to more general cases. As explained in the previous chapter, the main drawback of these methods is the high computational time they require as compared to other global optimization heuristics that do not exploit information about the landscape, such as those presented in Section 2.1.

In this chapter, we slightly relax the *expensive* part of the objective function, and investigate how to accelerate the use of GP as a response surface for global optimization so that it is applicable to problems which require a larger number of samples than is currently manageable. Relaxing the assumption of expensive samples requires that the modified model can propose a new next best location to evaluate in a reasonably short time. What an acceptable duration is depends on the real objective function to be optimized and the resources that we are willing to invest in the optimization process. However, the main goal is still to find the global optimum with as few samples as possible.

Fitting a GP is slow since it involves inversions of matrices whose size depends on the number of observations. This motivates the idea of partitioning the

space and fitting a GP to each region in order to make predictions. The main advantage of partitioning the space is that the number of samples used to fit a model is significantly reduced, which is the major driver for increasing the computational time when dealing with GP. The compromise, however, is that only local information is used in each model. The number of points in the regions (and not the size of the regions) directly controls the trade-off existing between very slow but global response surfaces and fast but local models.

The idea of using local approximations of the landscape in order to accelerate the fitting process part of the response surface generation process has been applied to surrogate assisted EA based optimization. For instance, Zhou, Ong, Nair *et al.* [2007] use a set of local RBF to generate cheap surrogate models through on-line learning, structured in a hierarchical manner, to replace the exact computationally expensive objective functions during the filtering part of an evolutionary search.

The main contributions of this chapter are first, the concept of partitioning the space for the EGO algorithm in order to accelerate the response surface stage together with a detailed explanation of its implementation. Second, a set of experiments that helps comparing different design choices, all with the objective of accelerating the response surface aided optimization process while preserving as much as possible the quality of the results. And third, the benchmarking of the proposed algorithm against the state of the art heuristics that have been tested using the BBOB.

This chapter is organized as follows. In Section 3.1, we start by introducing the main idea of partitioning the space and propose two adaptations of EGO to work in partitioned spaces. Section 3.2 is devoted to implementing, testing, and improving different aspects of the algorithms described in Section 3.1, which requires numerical experimentation. All the experiments are performed using the BBOB introduced in Chapter 2. The tested enhancements, all resulting in design choices for later stages, include assessing the importance of selecting an adequate initial DoE, choosing a dimension selection criterion to partition the space, and exploring the impact of the maximum number of samples allowed in a partition. Then, in Section 3.3 the fully enhanced version of the space partitioning algorithms proposed for accelerating the response surface model based global optimization are evaluated on the full BBOB

testbed, allowing comparison of their performance, under fair conditions, against other state of the art algorithms and against the standard version of EGO. Finally, Section 3.4 presents some concluding remarks.

### 3.1 EGO in partitioned spaces

The experiments performed in the last sections of Chapter 2 were limited in terms of the number of samples that were collected due to the constantly increasing time that it takes to make a new prediction, i.e., determine the next best sample to take. For every additional available sample in  $\mathcal{D}$  it takes an increasing amount of time, given that the computational complexity of fitting a GP is of order  $O(|\mathcal{D}|^3)$  [Rasmussen and Williams, 2006].

One approach to increase the number of samples to process is to develop faster algorithms for fitting GP, which is an active field of research (e.g. Musizza, Petelin, and Kocijan [2010]). However, the cubic computational complexity of the algorithm, as a function of the number of samples available, means that trying to accelerate the fitting process, or relying on increasingly faster computers, would not really solve the problem of being able to use many more observations than it is currently possible within a reasonable time. Another approach to accelerate the process of fitting a GP is to retain only the most relevant information available in the training dataset, but to reduce the size of the resulting covariance matrix [Quiñonero Candela and Rasmussen, 2005]. Good literature surveys, highlighting these two approaches, have been proposed by Barillec, Ingram, Cornford *et al.* [2011], and by Kocijan [2012]. A third option to accelerate the response surface aided global optimization process is to exploit further, at each iteration, the readily available model by determining the next  $q$  samples, instead of just the next one. This technique is sometimes referred to as parallel sampling or multi-point sequential sampling, and given its importance and the lack of literature concentrating the most relevant existing methods, we present a survey on this topic in Section 3.1.1 for completeness, even if we do not follow this path.

Instead, we propose to attack the problem from a different perspective which relies on reducing the number of samples in  $\mathcal{D}$  used at each time to generate a

response surface by partitioning the space into  $\varrho$  regions. Then, using samples only in each region, a GP can be fitted locally allowing to calculate the EI also per region. Finally, the local EI from each of the  $\varrho$  regions is compared, and the next best sample is taken where the overall largest EI is found.

Assuming the number of samples per region can be controlled, and considering  $|\mathcal{D}|$  samples in the whole space, on average, there will be  $\frac{|\mathcal{D}|}{\varrho}$  samples in each region at all times. The number of regions increases linearly with the number of samples, which means that  $\varrho$  is a linear function of the number of samples. This implies that, on average, only a constant number of samples  $\left(c_1 = \frac{|\mathcal{D}|}{\varrho(|\mathcal{D}|)}\right)$  will be used to fit a local GP, at each of the  $\varrho$  regions. This reduces the computational complexity of the algorithm from  $O(|\mathcal{D}|^3)$  to  $O(\varrho \times c_1^3) = O(\varrho \times c_2)$ , where  $c_2$  is a constant and does not depend on the total number of samples. If we further manage to eliminate the requirement of having to update each of the  $\varrho$  regions at every iteration, the final computational complexity for determining the next best sample of such an algorithm ( $O(c_2)$ ) becomes independent of the total number of samples, i.e., the running time of the sequential sampling process is linear in  $|\mathcal{D}|$ . Therefore, the successful partitioning algorithm needs to:

1. be able to control the number of samples per region, and
2. require only one local response surface to be updated per iteration.

The number of regions can not be arbitrarily large, since sparsely sampled regions lead to bad approximations of the landscape. Besides, since each model accounts only for local features, too much global information would be lost. Nevertheless, such partitioning technique would allow to continue the sampling process with many more observations than it is otherwise possible.

With the aim of creating an approximation algorithm whose computational complexity scales linearly with the number of observations made, in this Chapter we choose to design an algorithm in which the number of partitions can be controlled.

### 3.1.1 Survey on multi-point sequential sampling

Algorithms capable of proposing more than one next location at which to sample are scarce, however some of the existing ones have proven to successfully accomplish the task.

Sobester, Leary, and Keane [2004] attempt parallel updates by using either a gradient-based optimizer with multiple restarts or a GA with clustering and sharing to locate  $q$  local maxima of the EI function in order to sample at those  $q$  locations. The clustering and sharing steps are required to ensure that  $q$  distinct locations are found. The main drawback of this method is that it does not consider any correlations among the selected locations.

Regis and Shoemaker [2007] adapt 2 different RBF methods to do the sequential optimization with the idea of choosing  $q$  next locations to evaluate the objective function, but their model is specific to the two sequential infill methods they employ. The first method (Gutmann-RBF) requires an estimate of the global optimum to work, so the problem of finding  $q$  next locations is solved simply by using  $q$  different values for such an estimate. The second method (CORS-RBF) requires an auxiliary problem to be solved, so following the same principle, they solve  $q$  times the same problem with a different value for the *distance-factor* parameter.

Viana, Haftka, and Watson [2012] fit  $q$  different types of surrogate model to propose as many points to be evaluated. After evaluating these  $q$  points, the observations from the objective function are shared to update all the models. Since some models do not provide confidence intervals, and the utility function they use (EI) to choose the next best candidate requires one, the confidence intervals built from models that support it are shared with those that do not.

In the different —but related— context of approximate dynamic programming, Deisenroth, Rasmussen, and Peters [2009] face a similar problem where the  $q$  most promising candidates must be selected, and “to avoid combinatorial explosion in the selection of the best set of  $\ell$  states”, those states —which correspond to sampling locations in our context— are added sequentially by replacing the observed value with the current estimate. This still requires for the updated utility function to be optimized for every simulated sample, but removes the necessity of updating the surrogate model as often, which is the most computationally intensive part. The utility function in this problem is not the EI, but rather a function that explicitly has 2 parameters to control exploration and exploitation. This good idea can easily be adapted to be used with the EI as utility function since an estimate of the objective function is readily available at any point through the response surface. In fact,

Ginsbourger, Riche, and Carraro [2007, 2010] achieve this in a very elegant manner, proposing a full analytic derivation for the case where  $q = 2$  parallel samples are required, then extending it to the general case, which is acknowledged to be analytically intractable. Overcoming this intractability by using Monte Carlo simulations is explored, however since this method is still computationally expensive, they develop two heuristics, both of which replace the point at which the EI is maximized by a value, and treat it as known. Then this process is repeated until  $q$  points have been selected. This algorithm is known as the  $q$ -EI. The two heuristic methods they present differ in the way the replacement values are used. The first one, called the kriging believer, uses the value predicted by the response surface at the location where the EI was maximized to simulate an observation without actually sampling the objective function, and updates the covariance matrix ( $K$ ) without refitting the parameters, allowing the procedure to be repeated until  $q$  locations have been proposed. However, this method tends to lead to  $q$ -sequences that get trapped in a local area. The second one, called the constant liar, uses an exogenously proposed value, and then proceeds as the former. Three methods for proposing the values are compared: the minimum, the mean, and the maximum of the response values observed so far ( $\mathbf{Y}$ ), and it is consistently found that the best method is to use the minimum, reaching near optimal performances when compared with the equivalent Monte Carlo simulations.

### 3.1.2 Related space partitioning algorithms

There exist some algorithms that take advantage of a partitioning scheme in order to concentrate the search effort in reduced subregions of the search space. One example —perhaps the simplest— is an improvement on the multi-start algorithm which uses an initial number of random samples that serve as starting points for local hill climbers (see Boender, C. and Romeijn [1995] for a review). Since different starting points might lead to the same local optimum, clustering methods attempting to determine which points are close to each other are applied before starting the local searches aiming to reduce the total number of local hill climbers [Rinnooy Kan and Timmer, 1987].

Another idea on how to partition the space could be borrowed from either the

adaptive partitioning random search (APRS) [Bo Tang, 1994] designed for continuous functions, or the nested partition (NP) algorithm proposed by Shi and Ólafsson [2000] designed mainly for combinatorial problems, but also applicable for bounded continuous spaces. These algorithms require a figure of merit (called *promising index*) to quantify how promising each subregion is, and start by partitioning the space into  $\varrho$  subregions. Then, the most promising subregion is preserved, and all the rest are merged into one large region. Finally, the most promising subregion is subdivided into  $\varrho$  subregions so that, at all times, together with the recently merged outer region, there are a total of  $\varrho + 1$  partitions. Both algorithms focus mainly on how to compare the partitions to decide where to sample, and to decide when to merge and subdivide them, but avoid the explanation of a general method for partitioning the space. Specific space partitioning methods are only sketched for the examples provided in the referenced papers, and apply only to a particular class of problems.

Hughes [2003] proposes a partitioning method known as binary search algorithm that does not involve repartitioning the space. This space partitioning technique uses all the available data points to define the bordering regions of the partitions so that the resulting hyper-boxes are empty. Then, a sample is taken towards the center of the largest hyper-box, and the partitioning is updated so that the new sample lies on the most recently added boundary. This space partitioning optimization algorithm explicitly requires a parameter that balances exploration and exploitation, and does not rely on generating a response surface.

### 3.1.3 Space partitioning for accelerating efficient global optimization

Due to the way in which sequential sampling algorithms work, it is common that many observations concentrate in regions with local optima while other regions are only sparsely sampled. So it might be tempting to use clustering as a method to partition the space with the aim of grouping samples close to a same local optimum for them to be used together when generating a local response surface. However, an important feature of the desired partitioning method is that it must preserve a balanced number of samples in each region to avoid two kinds of extreme cases.

The first, in which there is an empty (or nearly-empty) partition, which limits the accuracy of predictions, and the second in which a partition contains almost all the samples, which defeats the original purpose of reducing the computational cost. These extreme scenarios do not satisfy the first necessary condition for the successful partitioning algorithm. We therefore discard any clustering approach.

The APRS and NP algorithms can be adapted to use the EI as figure of merit for the response surface based sequential optimization problem, however, the fact that regions are constantly merged and created would require to fit a GP in every region each time this happens, which could be as often as every time a new observation is made, failing to satisfy the second necessary condition. Furthermore, the number of samples in each subregion is not guaranteed, so similar problems as described for the clustering based method may arise.

The space partitioning technique used in the binary search algorithm does not satisfy the desired partitioning method either, given the nature of the resulting hyper-boxes, which by design are empty. This implies that there is no natural grouping on the samples, which prevents the creation of local models within each region.

The idea of partitioning the space is to reduce the overall complexity of fitting a response surface that exploits the structure of the landscape with the aim of finding the global optimum of a black-box function. In line with the requirements of preserving a balanced number of samples in each region so that building response surfaces is possible, meaningful, and computationally affordable, and for local updates to be required only upon the arrival of a new sample falling in a region, we propose the following space partitioning method for the EGO algorithm (SPEGO).

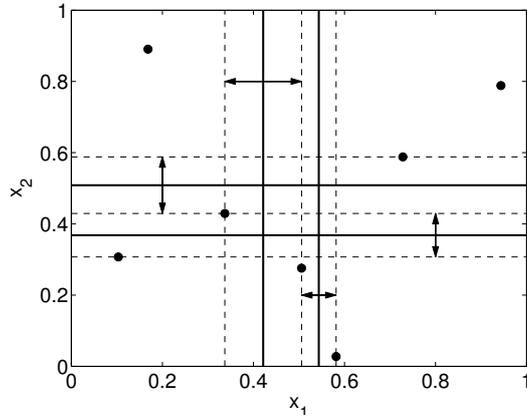
First, select a threshold  $\nu$ , which is the maximum number of samples allowed in a region. This threshold can be determined by considering how long it takes to fit a GP as a function of the sample size, and comparing it to the available computational budget for the whole optimization procedure. Then, define the whole search space as the root region ( $\rho_o$ ) and start the EGO algorithm as it would normally be done, i.e. choose the initial DoE budget  $\lambda$  and a sampling method (see Section 2.2), and draw additional samples sequentially until there are  $\nu$  samples available. Once

the threshold is reached in terms of number of samples, choose a dimension along which to perform a partition and create a dividing hyper-plane, co-planar to the vector expanding the chosen dimension, so that there is an equal number of samples in each resulting region. The dividing hyper-plane can be found by sorting the samples along the chosen dimension and finding the midpoints between the  $\frac{|\mathcal{D}|}{2}$ <sup>th</sup> and the  $\left(\frac{|\mathcal{D}|}{2} + 1\right)$ <sup>th</sup> elements if  $|\mathcal{D}|$  is even, or between the  $\left(\left\lceil \frac{|\mathcal{D}|}{2} \right\rceil - 1\right)$ <sup>th</sup> and  $\left\lceil \frac{|\mathcal{D}|}{2} \right\rceil$ <sup>th</sup> or the  $\left\lceil \frac{|\mathcal{D}|}{2} \right\rceil$ <sup>th</sup> and  $\left(\left\lceil \frac{|\mathcal{D}|}{2} \right\rceil + 1\right)$ <sup>th</sup> elements —whichever gap is the widest— if  $|\mathcal{D}|$  is odd.

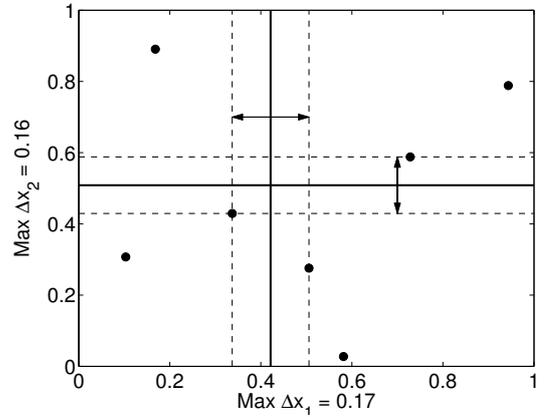
Choosing in which dimension to partition can be done by comparing the possible partitioning hyper-planes across all dimensions and selecting the one where the widest gap would be divided, by selecting the dimension whose associated length-scale is the shortest, or simply at random.

After partitioning the space,  $\rho_o$  is discarded and the observations in  $\mathcal{D}$ , originally belonging to  $\rho_o$  are assigned to the corresponding new region ( $\rho_{o'}$  or  $\rho_{o''}$ ). As a result, there are enough samples in each region to fit a GP and find both the location where the EI is maximized and the magnitude of the EI locally, which should be done for all non-discarded regions. Since the EI is an absolute quantity and not a relative measure when considering one function, it can be compared to other EI values across regions. The next best sample is then chosen where the maximum of all the local EIs of each region is found. The partitioning procedure for a  $2D$  case, using the maximum gap method, is illustrated in Figure 3.1.

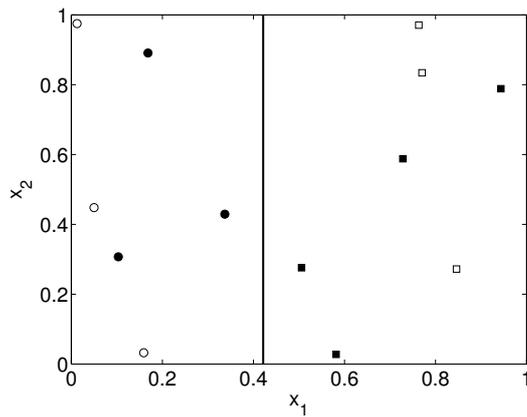
Each partitioning operation splits a region into 2 subregions, leaving unchanged any other existing partitions. So the above procedure can be repeated recursively to incrementally subdivide the search space as required. Every time a new observation is made, there are two possibilities. If the number of samples remains below the partitioning threshold, only the corresponding region must be updated to reflect the new information in its response surface model and its maximum EI value. Otherwise, a partition operation is required and two response surface models along with their corresponding maximum EI need to be calculated. In either case, the size of each local model, and thus the required computational time, are significantly smaller than they would be if considering all the points in  $\mathcal{D}$ .



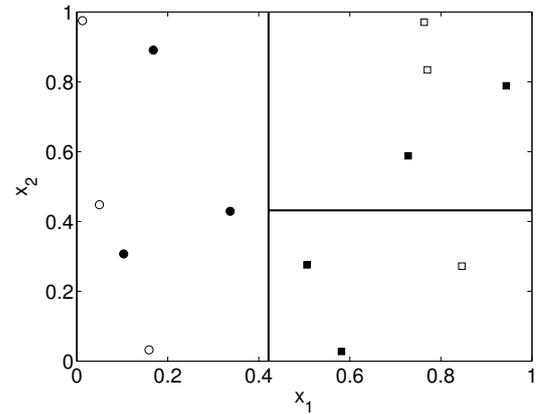
(a) First time the partitioning threshold ( $\nu = 7$ ) is reached, with auxiliary lines.



(b) Maximum gap candidate for each dimension.



(c) First partition done, with new samples (empty markers).



(d) Second partition done after the number of samples in the right subregion reached the threshold ( $\nu = 7$ ).

Figure 3.1: Figures 3.1a through 3.1d illustrate the space partitioning, using the maximum gap method, used by the SPEGO algorithm for 2 dimensions and a partitioning threshold  $\nu = 7$ . First, Figure 3.1a shows how the partitioning candidates (bold lines) are found independently for each dimension, using dashed lines and arrows to compare the gaps in between samples close to the middle. In Figure 3.1b, only the best partitioning candidate for each dimension, according to the maximum gap criteria, is shown, along with the existing gap labeled in the corresponding axis. Figure 3.1c displays the partitioned space with 6 new observations (empty markers), which makes the right partition ready to be sub-partitioned, since it contains now 7 samples. The outcome of the sub-partition is presented in Figure 3.1d.

### 3.1.4 Non-disjoint space partitioning efficient global optimization

GP are known to be good at interpolating, but like most regression techniques, they are not as reliable when extrapolating, i.e., making predictions where there is no data to support them [Zimmermann and Han, 2010; Razavi, Tolson, and Burn, 2012]. This shortcoming motivates a variation of the SPEGO algorithm described in Section 3.1.3 that we will refer to as non-disjoint<sup>1</sup> SPEGO (NSPEGO). The modification aims to reduce potential inaccuracies introduced by the artificial imposition of hard boundaries into the search space that may lead to discrepancies in the predictions at a boundary depending on which of the available models is being used. NSPEGO benefits from the fact that samples from bordering regions are already available at no extra cost.

The idea of NSPEGO is to use these observations in addition to those in the region for which a response surface is being built. The number of additional observations to take into account must not be too large, since this hinders the original purpose of partitioning the space by rapidly increasing the size of the dataset used to fit the model. We propose then to consider only the closest point to the border in each direction, which translates into adding  $2D$  observations to each region for central regions (those whose boundaries are not the search space boundaries). For a given region, the maximization of the EI needs to be done within its boundaries to keep the proposed next best sample in the same region. The major consideration is that whenever a new sample is taken, in addition to updating the region where the new sample lies, the bordering regions must also be checked since they may require to be updated in order to take advantage of the most recent information.

In the next section SPEGO and NSPEGO —the two variations of EGO motivated and proposed in this section— are implemented and tested for different parameter settings.

---

<sup>1</sup>In a partition, the intersection of the subsets of a set must be empty, which makes the outcome of the NSPEGO algorithm formally not a partition. However, we take the liberty of using the “non-disjoint” qualifier and keeping the term “partition” in the name to emphasize the main characteristic of the proposed variation.

## 3.2 Numerical implementation decisions

When implementing SPEGO and NSPEGO, there are some parameters that need to be set and some decisions to be made whose impact on the algorithm performance is often unpredictable due to the complex interactions existing among the algorithms' stages and its random nature. In order to back up these decisions, in this section we test different possible configurations of the proposed algorithms and report on their advantages and disadvantages. Specifically, we address:

- the initial DoE setup
- the selection criteria of the dimension at which the partition should be made
- the tuning of the partitioning threshold  $\nu$

Due to the large number of possible combinations, the studies are made independently from each other, but in a staged manner, i.e., the best outcome of each test is carried on to the next one. The first three experiments implement the SPEGO algorithm, and in Section 3.2.4 SPEGO and NSPEGO are compared using the best parametrization found. All the comparisons are performed in the full BBOB testbed ( $f_j^i$ ,  $j \in \{1, \dots, 24\}$ ,  $i \in \{1, \dots, 15\}$ ) for the 2 and 3 dimensional cases with a total budget of  $N = 200D$  samples. At all times, when maximizing the EI, the GA method with dynamically adapted population and generation sizes is implemented according to the procedure described in Section 2.6.3.

To compare the performances achieved, for each tested configuration, we tracked the number of instances that reached each of the fixed targets along with the number of function evaluations that were required to reach them ( $n_{f_{i,j}}$ ). For the cases where the targets were not attained, the best performance at the end of the total budget ( $N$ ) was recorded instead, in the form of the best achieved error ( $\epsilon_N^{i,j}$ ).

This information is then used in two different ways to display the results. First, in order to allow a quick visual comparison, we present the number of instances that reached the target within the specified budget, up to the specified target level, in the form of ECDF plots, for selected precision levels. Since the full BBOB testbed for  $2D$  and  $3D$  cases is being considered, there are a total of 720 instances, and to provide further context, the number of instances that reached the target

is also displayed in the legend of each plot in parenthesis. Considering only the number of instances that attained the specified target through ECDF plots has some advantages, but its interpretation must be done with care, particularly when most of the instances fail to reach the specified targets, given that information about these non successful instances is disregarded.

Since this is often the case for reduced budgets, we mostly rely on the second performance measure, which is based on a paired signed rank test [Wilcoxon, 1945] looking for evidence to reject the null hypothesis that *“the difference between the number of required function evaluations to achieve a given precision level by two configurations has median zero”*. Applying this statistical test directly to the number of function evaluations required by an optimizer to reach a specified target error, encounters the same problem as the method described above. However, this second method also allows the incorporation of information about instances that did not converge. Since the employed statistical test relies only on the relative ranking of the results, and not on any absolute magnitudes, we can apply an order preserving transformation that uses the negative inverse of the number of function evaluations required to converge ( $-\frac{1}{n_{f_{i,j}}}$ ) if the target was reached, or the best achieved error ( $\epsilon_N^{i,j}$ ) otherwise.

Since  $n_{f_{i,j}} \in [1, \dots, N]$ , then  $-\frac{1}{n_{f_{i,j}}} \in [-1, 0[$ . This transformation ensures that the instances that converged are all negative (in the range  $[-1, 0[$ ), and preserves their ranking, while the rest of the instances are ordered according to their best observed performance, with values closer to zero being those with lower error, thus considered better. After the transformation, the signed rank test is applied to each pair of optimizers being compared, yielding a p-value as the outcome of the test. In this context, the p-value represents the probability of observing a difference of zero between the paired (transformed) performances of two configurations of the algorithm (null hypothesis) by chance. Therefore, low p-values suggest there is strong evidence to reject the null hypothesis, in favor of the alternative hypothesis (one algorithm consistently performed better than the other).

For the compared algorithm settings, we display the resulting p-value of each paired test in a table. Values with a plus sign superscript indicate that the algorithm in a row performed better than the one in its corresponding column, while

the contrary case is indicated with a minus superscript. Furthermore, p-values in bold highlight that the difference was found to be significant at the 95% confidence level (p-value < 0.05).

### 3.2.1 Initial DoE

Due to the assumption that each new sample is expensive, in Chapter 2 the experiments were initialized with a small number of samples ( $\lambda = 2$  or  $\lambda = D + 2$ ), so the sampling method used to acquire these initial points did not play a major role. However, in the current scenario where the high sampling cost assumption has been relaxed, it is reasonable to verify whether by using different DoE configurations, with a larger number of samples, the performance of SPEGO can be improved for a fixed budget of total number of samples.

Following van Dam, Husslage, and Hertog [2009],  $\gamma$ -deep nested designs could also be used for the selected initial set of observations to anticipate for the need of sequential infill samples after the first stage, which —according to the maximin criteria— would result in a better space filling than taking samples independently from each other [van Dam, Husslage, den Hertog *et al.*, 2007; Husslage, Rennen, van Dam *et al.*, 2010], since this method takes into consideration the possible correlations amongst the samples. Nonetheless, nested designs are not considered here first, because the aim of SPEGO is not to obtain the best approximation of the function but to find its global optimum, and second, because such techniques have been only developed for either  $1D$  or for  $\gamma \leq 2$  depth levels.

Therefore, the proposed experiments compare the RS and LHS strategies introduced in Section 2.2 for two DoE budgets each ( $\lambda = 6D$ , and  $\lambda = 18D$ ). At this stage, a fixed partitioning threshold of  $\nu = 24$ , and a total budget of  $200D$  samples are used for the experiments. A detailed list of the parameters along with the results obtained for precision levels of  $f_\Delta = 10^1$  to  $f_\Delta = 10^{-8}$ , is given in Table 3.1. Furthermore, the ECDF plots at selected precision levels are shown in Figure 3.2. Throughout these figures, a different behavior between DoE settings using  $6D$  and  $18D$  can be observed. A performance improvement appears to happen as soon as the sequential infill algorithm starts (i.e., when the samples are sequentially chosen and not left to the initial DoE). Since this happens earlier for the cases using

$6D$ , there is a clear separation between implementations using  $6D$  and those using  $18D$  samples, however, this gap progressively narrows down as the number of total samples becomes larger.

The results in Table 3.1 show that for precisions higher than  $f_{\Delta} = 10^1$ , the LHS with a low budget ( $\lambda = 6$ ) significantly outperforms both of the RS initial DoE strategies. When compared to the LHS strategy with a higher budget, LHS-06 also performs better, even though the statistical significance of the results decreases as the precision level increases. We can conclude that, when using a limited budget of function evaluations, it is best to use a reduced number of samples for the initial design relative to the total budget. Based on these results, the initial design LHS-06 is used for the rest of the experiments.

### 3.2.2 Partitioning dimension selection criteria

As detailed in Section 3.1.3, we propose three selection methods to choose along which dimension to partition the space. The first method is to choose the dimension where, after performing the partition, the widest gap would be split. This is aimed at diminishing the cases where partitions dividing a set of observations close to each other (perhaps hinting at the existence of a local optimum) are created. The second proposed method is to partition along the dimension whose associated lengthscale, as found when fitting the GP, is the shortest. Since short lengthscales indicate low influence of samples located far from each other, this method attempts to minimize the disruption partitions cause by not creating barriers in dimensions where long range information is most valued. The last proposed method is to select the dimension at random in order to guarantee that partitions are done with equal probability along all dimensions, and that the resulting regions are not, in general, disproportionately long in any of the dimensions.

In this section we test whether there is a significant advantage for using either of these procedures by fixing the experiment parameters to the best configuration from Section 3.2.1, and varying only the dimension selection criterion. The details and results are presented in Table 3.2, and the ECDF plots allowing a visual comparison of the convergence rate of the three dimension selection criteria compared are presented in Figure 3.3.

Table 3.1: Test for different **initial DoE** configurations for the SPEGO algorithm with a total budget of  $200D$  samples, implementing the random dimension selection criterion, and a partitioning threshold  $\nu = 24$ . The four compared configurations are represented by rows, and enumerated from 1 to 4 so that they can be matched to their corresponding numbers in the low level headers of each column. At the intersection of each row and column, the resulting p-value after applying the paired sign rank test is displayed. p-values with a plus (+) superscript indicate that the configuration in the row was found to be better than the one in the column, while minus (-) superscripts indicate the opposite case. Furthermore, values in bold indicate that the difference in performance was found to be significant at the 95% significance level. The comparisons are made for different target errors, which are indicated as high level column headers. For practical reasons, the table is split into four parts. Results show that the LHS with  $\lambda = 6D$  outperforms the other strategies almost consistently.

DoE		$10^1$			$10^0$			$10^{-1}$		
		2	3	4	2	3	4	2	3	4
1	RS-06D	<b>0.0008</b> <sup>+</sup>	0.0560 <sup>+</sup>	<b>0.0074</b> <sup>+</sup>	<b>0.0001</b> <sup>+</sup>	<b>0.0055</b> <sup>-</sup>	<b>0.0101</b> <sup>+</sup>	<b>0.0019</b> <sup>+</sup>	<b>0.0051</b> <sup>-</sup>	0.2968 <sup>+</sup>
2	RS-18D		<b>0.0000</b> <sup>-</sup>	0.2252 <sup>-</sup>		<b>0.0000</b> <sup>-</sup>	0.3649 <sup>+</sup>		<b>0.0000</b> <sup>-</sup>	0.0793 <sup>+</sup>
3	LHS-06D			<b>0.0002</b> <sup>+</sup>			<b>0.0000</b> <sup>+</sup>			<b>0.0027</b> <sup>+</sup>
4	LHS-18D									
(cont.)		$10^{-2}$			$10^{-3}$			$10^{-4}$		
		2	3	4	2	3	4	2	3	4
1	RS-06D	<b>0.0061</b> <sup>+</sup>	<b>0.0221</b> <sup>-</sup>	0.3752 <sup>+</sup>	<b>0.0133</b> <sup>+</sup>	<b>0.0299</b> <sup>-</sup>	0.4435 <sup>+</sup>	<b>0.0326</b> <sup>+</sup>	<b>0.0261</b> <sup>-</sup>	0.4312 <sup>+</sup>
2	RS-18D		<b>0.0000</b> <sup>-</sup>	0.0775 <sup>+</sup>		<b>0.0003</b> <sup>-</sup>	0.0992 <sup>+</sup>		<b>0.0011</b> <sup>-</sup>	0.1194 <sup>-</sup>
3	LHS-06D			<b>0.0124</b> <sup>+</sup>			<b>0.0302</b> <sup>+</sup>			0.0586 <sup>+</sup>
4	LHS-18D									
(cont.)		$10^{-5}$			$10^{-6}$			$10^{-7}$		
		2	3	4	2	3	4	2	3	4
1	RS-06D	0.0602 <sup>+</sup>	<b>0.0375</b> <sup>-</sup>	0.2435 <sup>+</sup>	0.1269 <sup>+</sup>	<b>0.0256</b> <sup>-</sup>	0.1568 <sup>-</sup>	0.1071 <sup>+</sup>	<b>0.0277</b> <sup>-</sup>	0.1703 <sup>-</sup>
2	RS-18D		<b>0.0063</b> <sup>-</sup>	0.0780 <sup>-</sup>		<b>0.0112</b> <sup>-</sup>	0.0998 <sup>-</sup>		<b>0.0119</b> <sup>-</sup>	0.0938 <sup>-</sup>
3	LHS-06D			0.1813 <sup>+</sup>			0.2562 <sup>+</sup>			0.2718 <sup>+</sup>
4	LHS-18D									
(cont.)		$10^{-8}$								
		2	3	4						
1	RS-06D	0.1167 <sup>+</sup>	<b>0.0333</b> <sup>-</sup>	0.1601 <sup>-</sup>						
2	RS-18D		<b>0.0128</b> <sup>-</sup>	0.0787 <sup>-</sup>						
3	LHS-06D			0.2873 <sup>+</sup>						
4	LHS-18D									

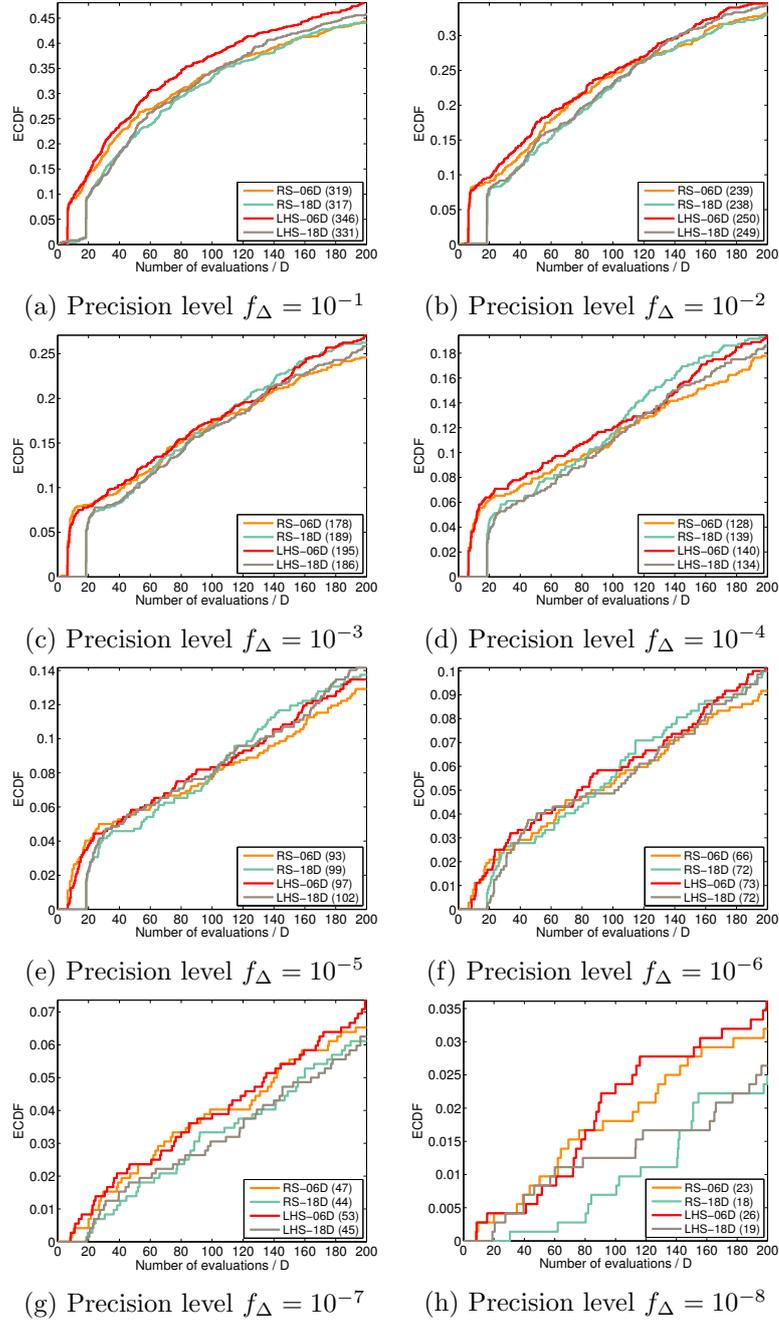


Figure 3.2: ECDF plots showing the proportion of trials that converged to different target errors for four **initial DoE** settings, implemented with the SPEGO algorithm. Values in parenthesis indicate the number of instances that reached the target, out of the total 720. Given enough function evaluations, the plots shown should converge to 1, since the EI is zero wherever a sample exists, preventing any location to be resampled. However, acquiring many more data points than the ones shown is impractical due to the available computational resources.

Table 3.2: **Partitioning dimension selection criteria** test for the SPEGO algorithm with a total budget of  $200D$  samples, LHS with  $\lambda = 6$  as initial DoE, and a constant partitioning threshold  $\nu = 24$ . Results show that the random selection criterion consistently outperformed the other dimension selection criteria.

Criterion		$10^1$		$10^0$		$10^{-1}$		$10^{-2}$		$10^{-3}$	
		2	3	2	3	2	3	2	3	2	3
1	Random	0.2473 <sup>+</sup>	0.2562 <sup>+</sup>	<b>0.0258<sup>+</sup></b>	<b>0.0252<sup>+</sup></b>	<b>0.0073<sup>+</sup></b>	<b>0.0048<sup>+</sup></b>	<b>0.0044<sup>+</sup></b>	<b>0.0033<sup>+</sup></b>	<b>0.0032<sup>+</sup></b>	<b>0.0023<sup>+</sup></b>
2	Max gap		0.2156 <sup>+</sup>		0.2578 <sup>+</sup>		0.1696 <sup>+</sup>		0.1924 <sup>+</sup>		0.2168 <sup>+</sup>
3	Min lengthscale										
(cont.)		$10^{-4}$		$10^{-5}$		$10^{-6}$		$10^{-7}$		$10^{-8}$	
		2	3	2	3	2	3	2	3	2	3
1	Random	<b>0.0024<sup>+</sup></b>	<b>0.0017<sup>+</sup></b>	<b>0.0019<sup>+</sup></b>	<b>0.0019<sup>+</sup></b>	<b>0.0029<sup>+</sup></b>	<b>0.0030<sup>+</sup></b>	<b>0.0027<sup>+</sup></b>	<b>0.0010<sup>+</sup></b>	<b>0.0033<sup>+</sup></b>	<b>0.0014<sup>+</sup></b>
2	Max gap		0.2689 <sup>+</sup>		0.2590 <sup>+</sup>		0.2624 <sup>+</sup>		0.2385 <sup>+</sup>		0.2428 <sup>+</sup>
3	Min lengthscale										

Contrasting the results shown in Figure 3.3 and in Table 3.2 for  $f_\Delta = 10^{-5}$  and  $f_\Delta = 10^{-6}$  illustrates why —for our specific case— the ECDF plots are not considered as reliable as the sum rank tests. The ECDF plots, which only considers 106 and 82 instances out of the 720, respectively, show that more instances using the minimum lengthscale criterion than the instances using other criteria reach the required target errors. On the other hand, the statistical tests also take into account the differences in performance achieved for the rest of the instances, which in this case, are the majority.

The experiment results show that the random dimension selection criterion consistently outperforms the maximum gap and the minimum lengthscale dimension selection criteria, in a statistically significant manner at all target error levels, except for the lowest precision ( $f_\Delta = 10^1$ ). The reason remains unclear, however, one possible explanation is that neither the magnitude of the lengthscale nor the size of the gaps by themselves, characterize the best dimension at which to split. Instead, it is perhaps a combination of both that best identifies it, which could be expressed in the form of the ratio of the maximum gap divided by the minimum length. For the remainder of the experiments, however, the random partitioning dimension criterion is implemented.

### 3.2.3 Partition size threshold

In this section we assess the impact of varying the partition threshold value  $\nu$ , which regulates the trade-off between the amount of shared information (size of partition) and the running time. The fewer samples allowed per partition (low  $\nu$ ), the less

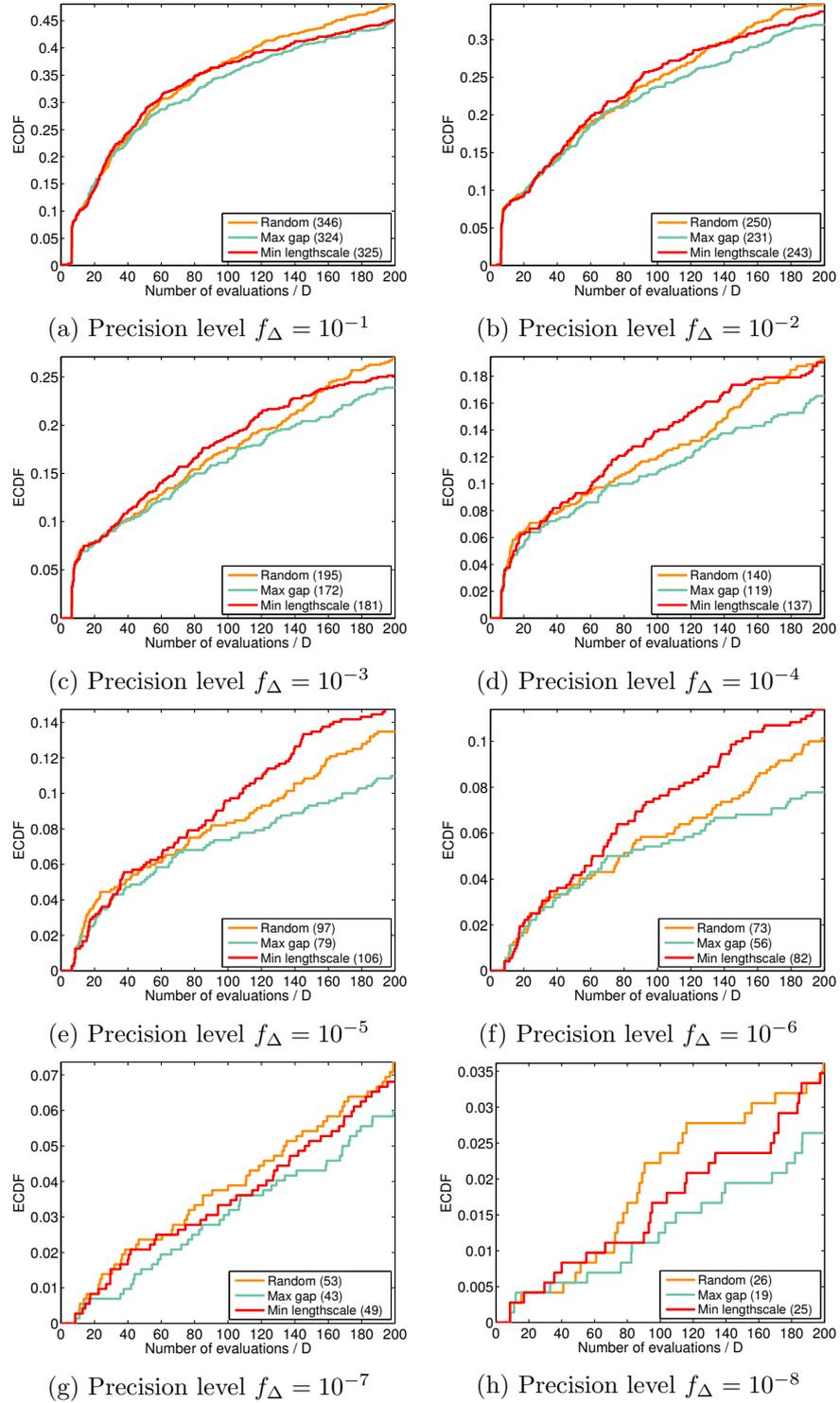


Figure 3.3: ECDF plots comparing the proportion of converged trials for the three proposed **dimension selection criteria** to partition the space, implemented with the SPEGO algorithm, and tested in the BBOB at several target error resolutions.

Table 3.3: **Partitioning threshold ( $\nu$ ) increase** as a function of problem dimensionality on the SPEGO algorithm with a total budget of  $200D$  samples, LHS with  $\lambda = 6$  as initial DoE, and the random partitioning dimension selection criterion. Only  $3D$  instances were considered. Given the low statistical significance of the results, no conclusions can be drawn from this experiment.

$\nu$	$10^1$	$10^0$	$10^{-1}$	$10^{-2}$	$10^{-3}$	$10^{-4}$	$10^{-5}$	$10^{-6}$	$10^{-7}$	$10^{-8}$
	2	2	2	2	2	2	2	2	2	2
1 $\nu = 24$	0.3155 <sup>+</sup>	0.2398 <sup>+</sup>	0.1679 <sup>+</sup>	0.1887 <sup>+</sup>	0.1691 <sup>+</sup>	0.1490 <sup>+</sup>	0.1456 <sup>+</sup>	0.1420 <sup>+</sup>	0.1387 <sup>+</sup>	0.1423 <sup>+</sup>
2 $\nu = 12D$										

Table 3.4: Effect of **partitioning threshold ( $\nu$ ) size**, tested using the same configuration as described in Table 3.3, but using  $2D$  and  $3D$  instances. Using larger partitioning sizes seems to be a consistently better choice.

$\nu$	$10^1$	$10^0$	$10^{-1}$	$10^{-2}$	$10^{-3}$	$10^{-4}$	$10^{-5}$	$10^{-6}$	$10^{-7}$	$10^{-8}$
	2	2	2	2	2	2	2	2	2	2
1 $\nu = 6D$	0.1062 <sup>+</sup>	0.0000 <sup>-</sup>								
2 $\nu = 12D$										

accurate —yet faster— the local models will be, and vice versa. Since more samples are required to obtain appropriate approximations of the landscape as the number of dimensions increases, we propose to define the partitioning threshold as a function of  $D$  in order to provide a fair ground to the optimizers when working in higher dimensions.

Two separate tests are performed. First, Table 3.3 shows the outcome of a comparison between a constant threshold size, and a threshold size that varies with the dimensionality of the problem. By comparing the performance of SPEGO with a constant value of  $\nu = 24$  against an adjusted value of  $\nu = 12D$ , we test the hypothesis that increasing the partition size as a function of the dimension is beneficial. Since  $\nu$  is the same for  $D = 2$ , only functions in  $3D$  are considered. Even though the results seem to indicate that increasing the partition size as a function of the dimension is not beneficial, the significance level is low, so the outcome of this test is considered inconclusive.

Second, Table 3.4 serves to compare the effect of the coefficient dictating the partition threshold value. The results for this test clearly demonstrate that increasing the number of samples in the partition improves the performance of SPEGO.

Figure 3.4 shows the ECDF plots comparing the performance of SPEGO, in terms of its convergence rate, using the three proposed partitioning thresholds

over the  $2D$  and  $3D$  instances together, and Figure 3.5 presents the same output as a function of the computational time required. The latter figure corroborates that increasing the size of the partition has a negative impact on the amount of computational resources required. Since the value of  $\nu = 12D$  is already close to the unaffordable computation times according to the available resources for these experiments, we retain this parameter setting for further simulations, instead of trying to increase it further.

### 3.2.4 Comparing SPEGO and NSPEGO

Having chosen the initial DoE, the partitioning dimension selection criterion, and the threshold size implementation parameters, the two proposed algorithms can then be compared to each other. Considering only the data being used to build the surrogate models, NSPEGO is expected to achieve better than SPEGO since it uses at least the same information as SPEGO plus additional neighboring points outside the regions when available. However, whether this is the case in practice, and whether the additional computational burden could be justified in such case, is not obvious. In order to address this question we propose to run both algorithms on the BBOB testbed using the design parameters found in the previous sections.

The common parameters for the compared algorithms are a total budget of  $200D$  samples, LHS with  $\lambda = 6$  as initial DoE, the random partitioning dimension selection criterion, and a partitioning threshold  $\nu = 12D$ . The results of this experiment are presented in Table 3.5, which shows inconclusive results, disproving that the only factor to consider is the amount of information taken into account by the local models. Figure 3.6 presents the ECDF plots allowing to visualize the proportion of instances that converged at different target errors, for both algorithms. Moreover, we present a comparison of the running times for both algorithms in Figure 3.7, which suggests that implementing NSPEGO instead of SPEGO is not justified, given the additional computational cost it implies, and the lack of improvements in the performance.

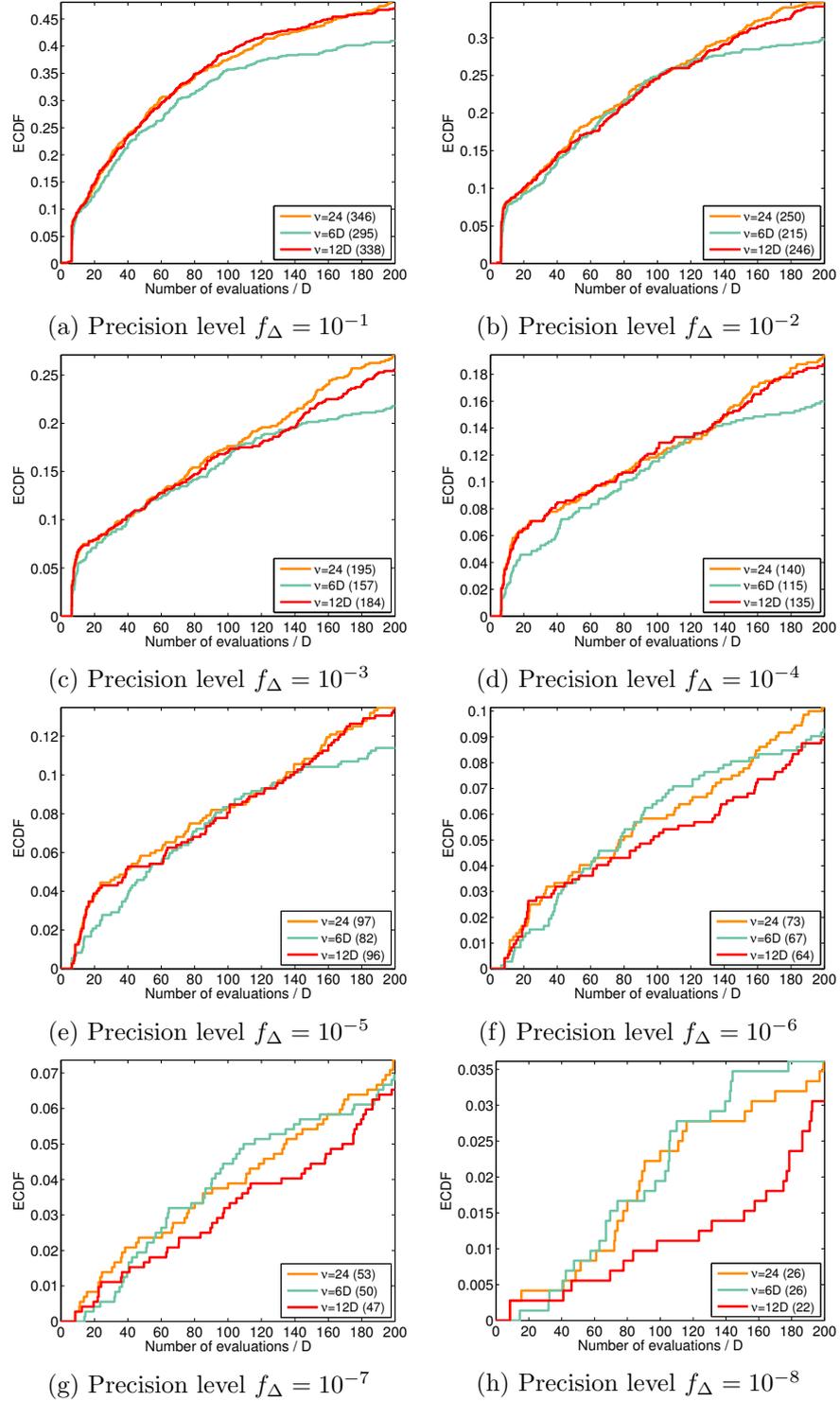


Figure 3.4: ECDF plots comparing the proportion of converged trials for the different sizes of **partitioning threshold**  $\nu$ .

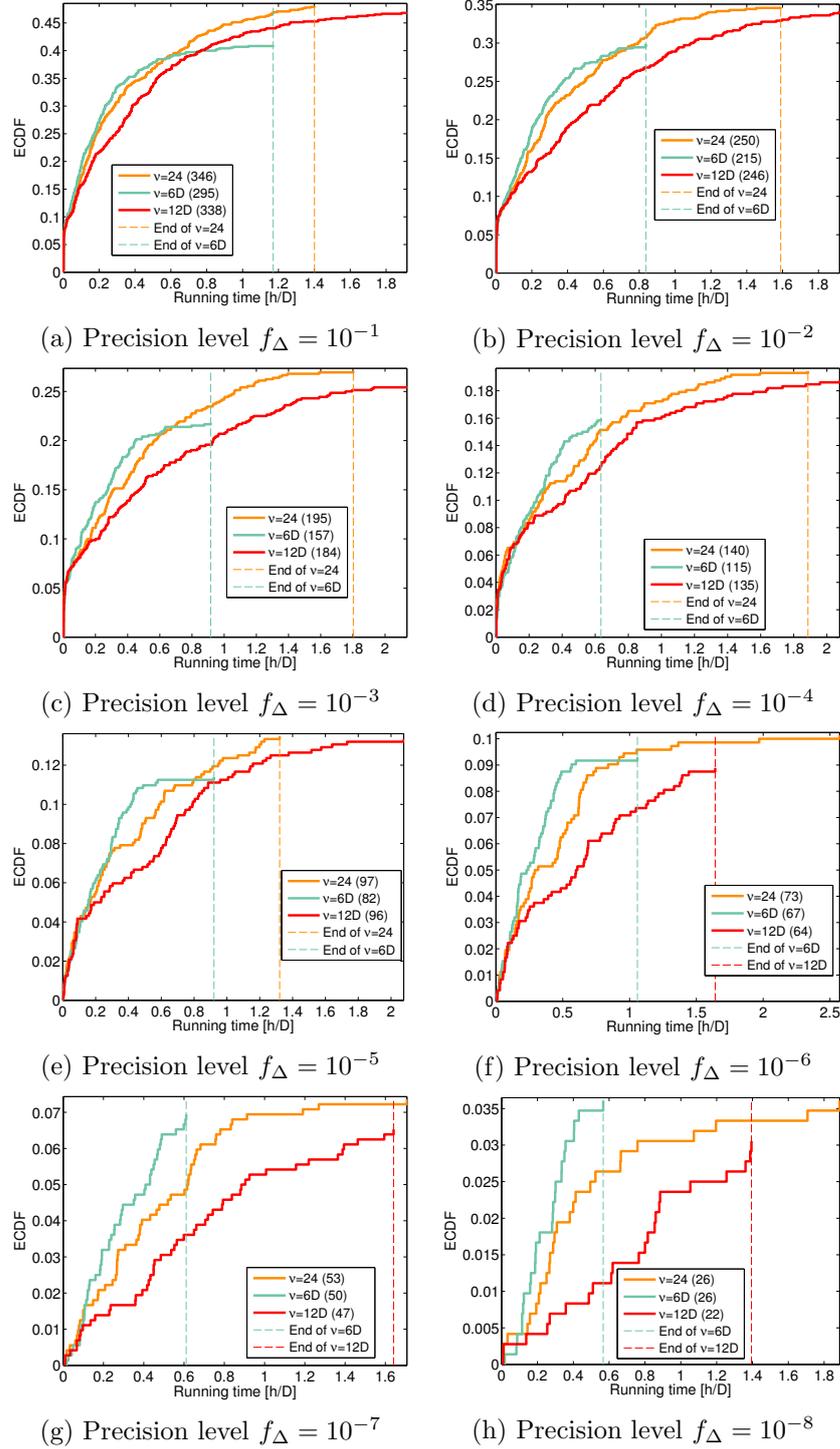


Figure 3.5: ECDF plots comparing the proportion of converged trials as a function of the **running time** required by the different sizes of **partitioning threshold**  $\nu$ . Times reported represent real time in dedicated single Intel<sup>®</sup> Xeon<sup>®</sup> CPU 5160 cores running at 3.00GHz.

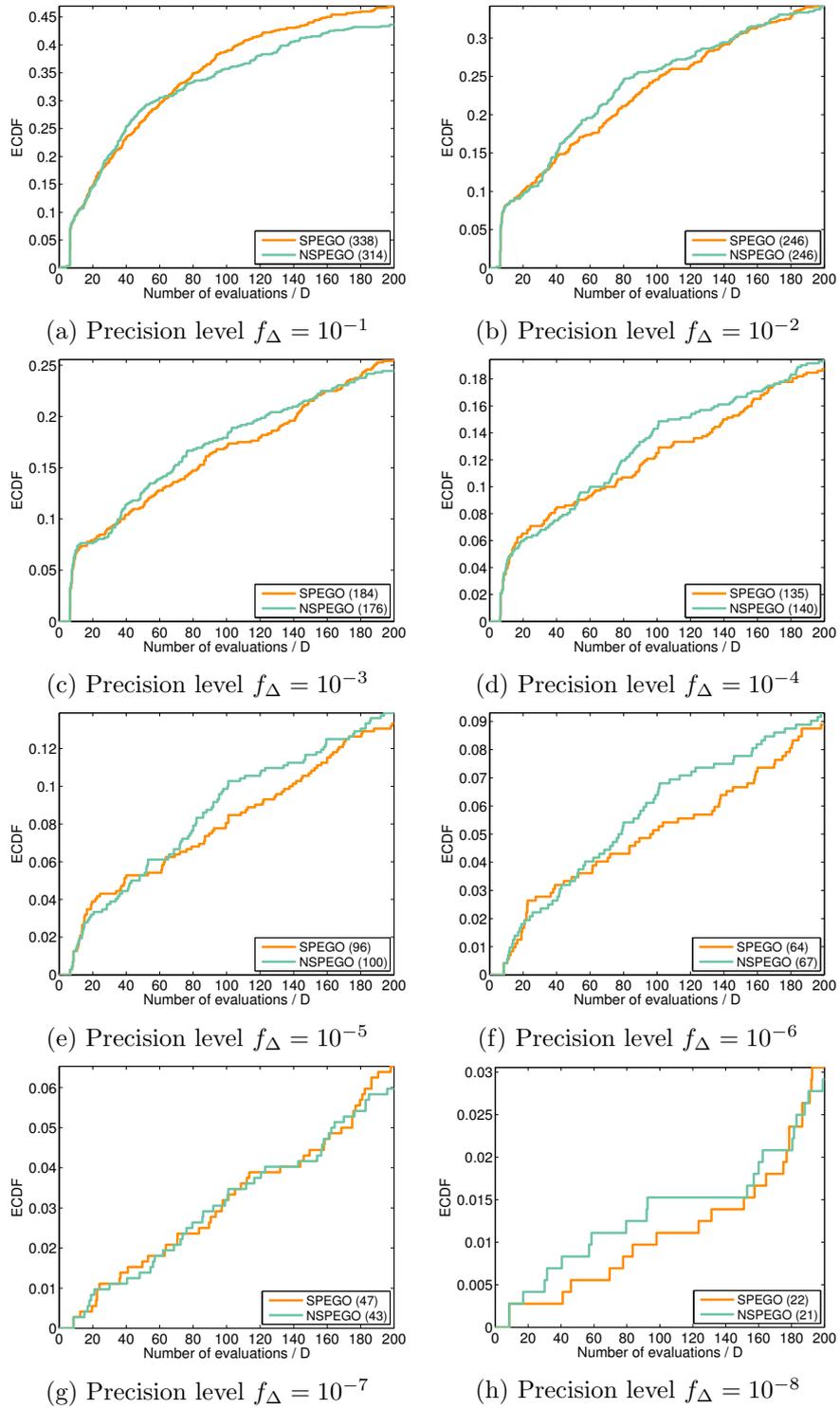


Figure 3.6: ECDF plots comparing the proportion of converged trials of **SPEGO** and **NSPEGO** as a function of the number of evaluations.

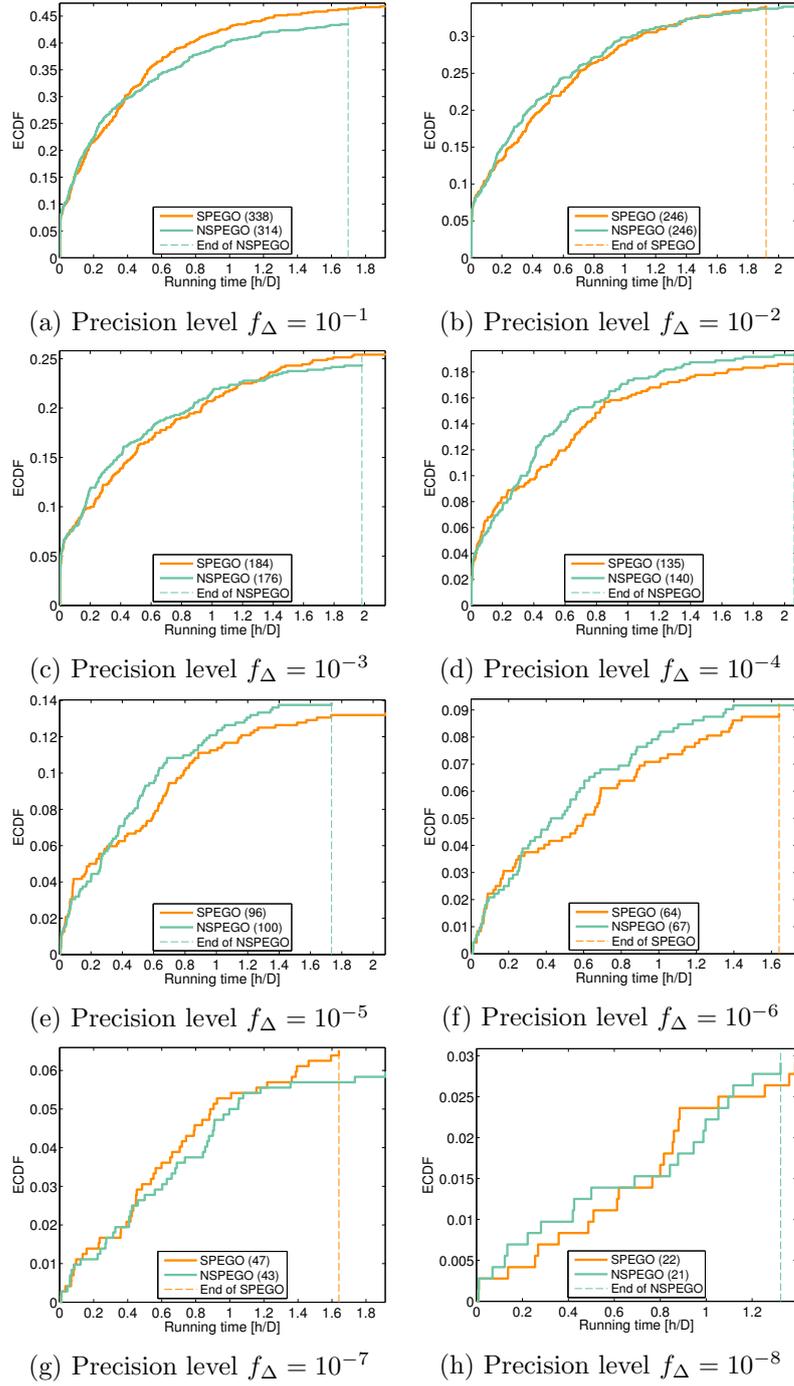


Figure 3.7: **Running time** comparison of **SPEGO** and **NSPEGO**, showing the proportion of converged trials as a function of time, for a fixed number of function evaluations ( $N = 200D$ ). Dashed lines show the end of the budget. Times reported represent real time in dedicated single Intel<sup>®</sup> Xeon<sup>®</sup> CPU 5160 cores running at 3.00GHz.

Table 3.5: **SPEGO vs. NSPEGO**. Tests performed with a total budget of  $200D$  samples, LHS with  $\lambda = 6$  as initial DoE, the random partitioning dimension selection criterion, and a partitioning threshold  $\nu = 12D$ . No statistically significant difference was found between the performances of the compared algorithms.

Algorithm		$10^1$	$10^0$	$10^{-1}$	$10^{-2}$	$10^{-3}$	$10^{-4}$	$10^{-5}$	$10^{-6}$	$10^{-7}$	$10^{-8}$
		2	2	2	2	2	2	2	2	2	2
1	SPEGO	<b>0.0005<sup>+</sup></b>	0.1397 <sup>+</sup>	0.4076 <sup>+</sup>	0.3894 <sup>+</sup>	0.3407 <sup>+</sup>	0.2981 <sup>+</sup>	0.2730 <sup>+</sup>	0.3258 <sup>+</sup>	0.2708 <sup>+</sup>	0.2989 <sup>+</sup>
2	NSPEGO										

### 3.3 SPEGO in context

To understand the advantages and disadvantages of the proposed space partitioning based algorithm, in this section we present two comparisons. First, to assess the effect of partitioning the space, we compare SPEGO against the original version of EGO. And second, to get an idea of the performance of SPEGO with respect to other existing algorithms, we benchmark it against the state of the art algorithms whose performance on the BBOB has been reported.

#### 3.3.1 SPEGO vs. EGO

The original version of EGO struggles to cope with large budgets when used for sequential sampling, since it has a computational complexity that scales as the cube of the number of samples. This is the original motivation for the development of SPEGO, which is an approximation of EGO whose computational complexity is linear with respect to the total number of samples. Given that these two algorithms work at different timescales, to facilitate their comparison, we tested their implementations on the BBOB by fixing the running time budget at  $8D$  hours (for each instance), and then analyzed the results from two perspectives.

The first one focuses on the number of function evaluations that each algorithm is capable of handling for a fixed time. This comparison is shown in Figure 3.8 through plots of the total running time as a function of the number of evaluations. The results for each of the 720 instances of each algorithm are shown in the background (light colors), separating  $2D$  and  $3D$  experiments. The mean is then calculated separately for each dimension and over-imposed in bold. The fact that the simulations stop after  $8D$  hours bias the mean running time (bold lines) towards the end, as only instances which have not exhausted their running time budget are

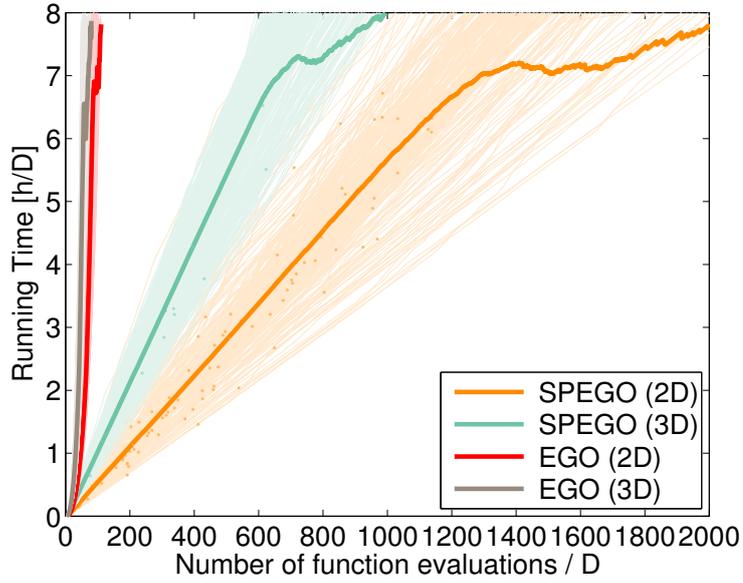
Table 3.6: Results of the sum rank test applied to the fixed number of function evaluations (first row), and the fixed running time (second row) tests for  $N = 48D$  samples and  $8D$  hours budgets respectively, comparing the EGO and the SPEGO algorithms. The p-values with a plus superindex indicate that EGO outperformed SPEGO at the budget shown in each row, measured for the target error displayed in the columns. Values in bold show statistically significant results at the 95% confidence level. For the low fixed number of function evaluations test, EGO consistently outperforms SPEGO, as expected. But when compared at a fixed running time of  $8D$  hours, SPEGO shows a significantly better performance than EGO for error targets lower than  $f_{\Delta} < 10^1$ .

Algorithms	Budget	$10^1$	$10^0$	$10^{-1}$	$10^{-2}$	$10^{-3}$	$10^{-4}$	$10^{-5}$	$10^{-6}$	$10^{-7}$	$10^{-8}$
EGO	$48D$	<b>0.0000<sup>+</sup></b>	<b>0.0282<sup>+</sup></b>	<b>0.0244<sup>+</sup></b>	<b>0.0268<sup>+</sup></b>	<b>0.0236<sup>+</sup></b>	<b>0.0271<sup>+</sup></b>	<b>0.0267<sup>+</sup></b>	<b>0.0282<sup>+</sup></b>	<b>0.0302<sup>+</sup></b>	<b>0.0285<sup>+</sup></b>
> SPEGO	$8Dh$	0.0843 <sup>+</sup>	<b>0.0000<sup>-</sup></b>								

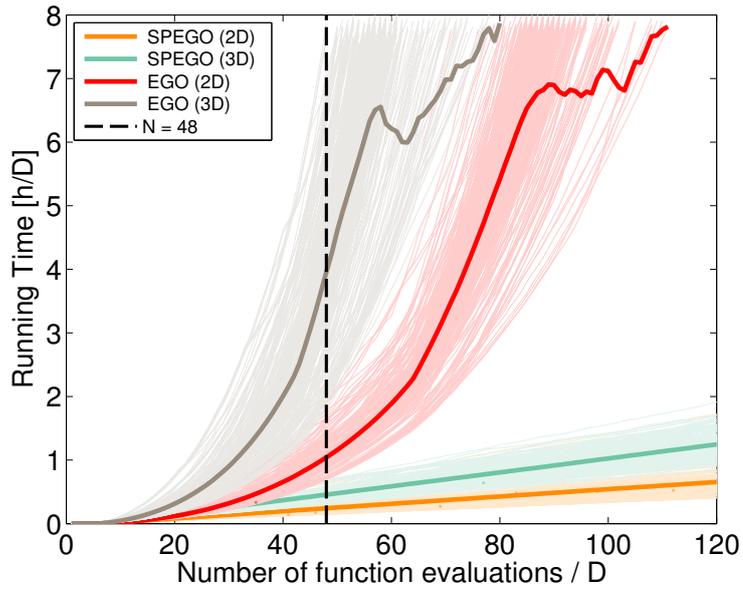
considered. This explains the erratic behavior of the means as less samples are available. Furthermore, the dots at the end of some individual runs represent the locations where some instances converged to the lowest target error ( $f_{\Delta} = 10^{-8}$ ). Given the large difference in computational time required by EGO and SPEGO, two plots are shown. Figure 3.8a highlights the contrast of the computational complexities (cubic for EGO and linear for SPEGO), but it hides most of the information relevant to EGO, which is better appreciated in the zoomed Figure 3.8b.

The second perspective follows the same format as the comparisons used throughout Section 3.2, which compares the rate at which the optimizers reach the targets, and the performance of the algorithms for both a fixed budget of function evaluations, and a fixed computational time budget. The fixed budget comparison takes place at a budget of  $N = 48D$  function evaluations, since this is the largest number of function evaluations reached by all instances of EGO and SPEGO. The fixed time comparison, on the other hand, is performed at the end of the  $8D$  hours of computational time allocated to each instance. The results obtained from this second perspective are presented in Figure 3.9, and in Table 3.6, which shows the outcome of the statistical tests at the two selected budgets. To better identify where the fixed number of function evaluations comparison takes place, the selected budget is also displayed in Figures 3.8 and 3.9 as a dashed vertical line.

As expected —considering the nature of SPEGO— EGO achieves a better performance than SPEGO when measured at  $N = 48D$  function evaluations. When measured for a fixed time, however, SPEGO outperforms EGO in a statistically



(a) Full version



(b) Zoomed version of Figure 3.8a, indicating the budget (dashed vertical line) at which the statistical comparison presented in the first line of Table 3.6 was performed.

Figure 3.8: Running time comparison of **EGO** and **SPEGO**, for a fixed timed budget. Running time in hours per dimension as a function of number of function evaluations, for each of the 720 instances of each algorithm (light colored lines), separating 2D and 3D experiments. Bold lines represent the group mean, and dots at the end of individual runs indicate instances that reached the lowest target error ( $f_{\Delta} = 10^{-8}$ ). Times reported represent real time in dedicated single Intel<sup>®</sup> Xeon<sup>®</sup> CPU 5160 cores running at 3.00GHz.

significant manner. It is to be highlighted that, while EGO already used most of the computational time budget ( $8D$  hours) at  $N = 48D$ , SPEGO only uses a fraction of it (see Figure 3.8).

Another view of the difference in performance between EGO and SPEGO can be appreciated through Figure 3.9, which shows the achieved rate of convergence at different target errors in a logarithmic scale, and is complemented with Figure 3.10, which presents the rate at which the targets are reached, at different error precisions, but this time as a function of required computational time. Using these two figures together, we can conclude that, for extremely expensive objective functions, EGO might be a better option. Otherwise, SPEGO is a better choice.

### 3.3.2 SPEGO against the state of the art

In order to understand how the performance of SPEGO compares to other algorithms found in literature, in this section we benchmark the proposed algorithm against two selected algorithms, one of which is considered the state of the art when dealing with expensive objective functions (CMA-ES) according to the BBOB ranking, and one that is also inspired by EGO (SMAC).

- CMA-ES stands for covariance matrix adaptation for evolution strategies, and is an evolutionary algorithm that uses a sophisticated self-adaptive method to vary the probability distribution governing the mutation operation used to generate new populations. Rather than considering only static mutation steps, the proposed self-adaptation takes into account the evolution path over a number of generations. CMA-ES was first introduced by Hansen and Ostermeier [1996], and since then, many improvements have been proposed. The implementation against which we benchmark SPEGO is the one proposed by Jastrebski and Arnold [2006], due to its good performance in the 2013 version of BBOB, and because the output of its performance is publicly available. This particular variation considers information of unsuccessful offspring, which would otherwise passively decay, in order to actively reduce variances of the mutation distribution in unpromising directions of the search space.

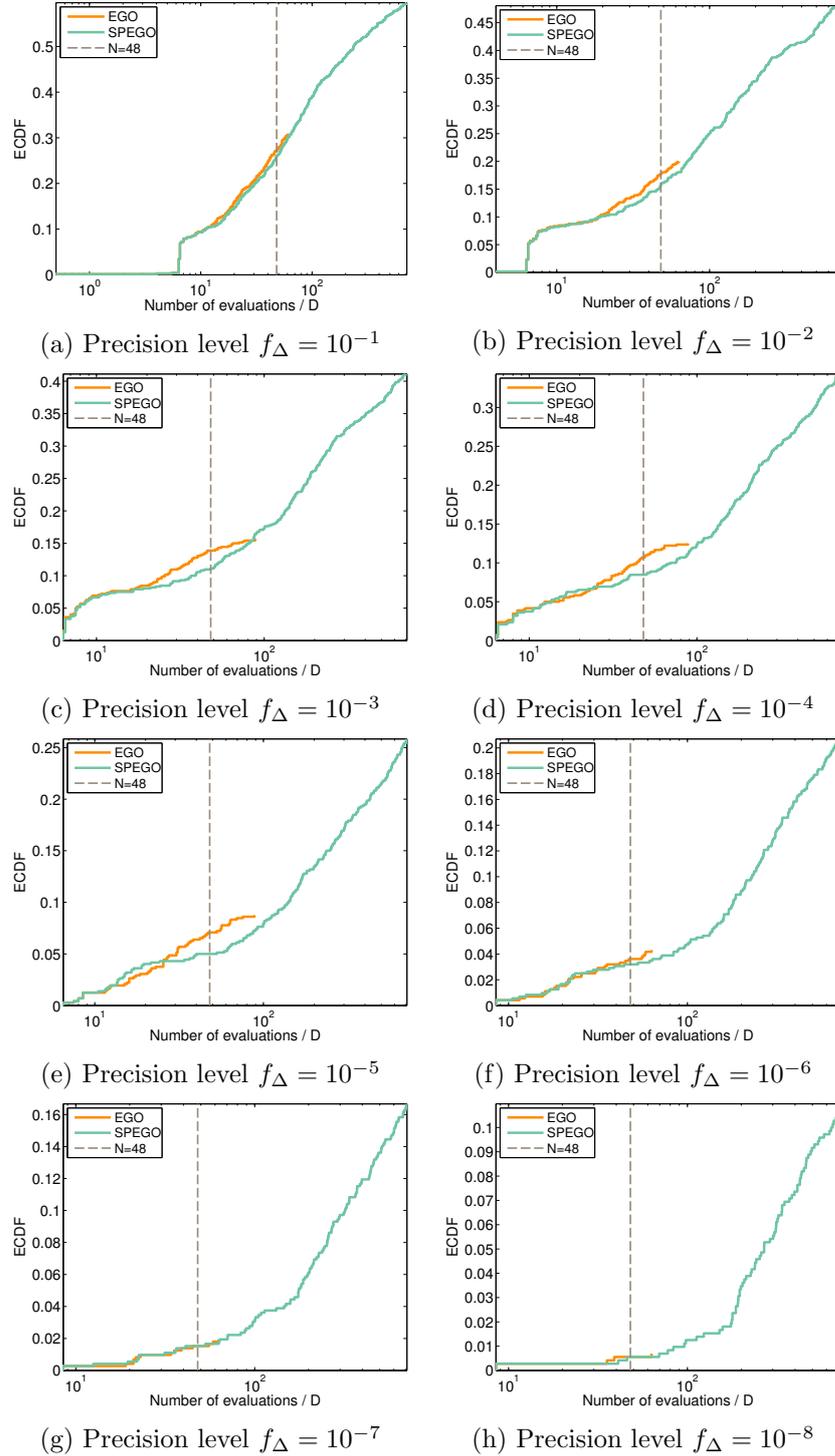


Figure 3.9: ECDF plots comparing the proportion of converged trials of the **SPEGO** and **EGO** algorithms as a function of the number of evaluations, presented in a logarithmic scale.

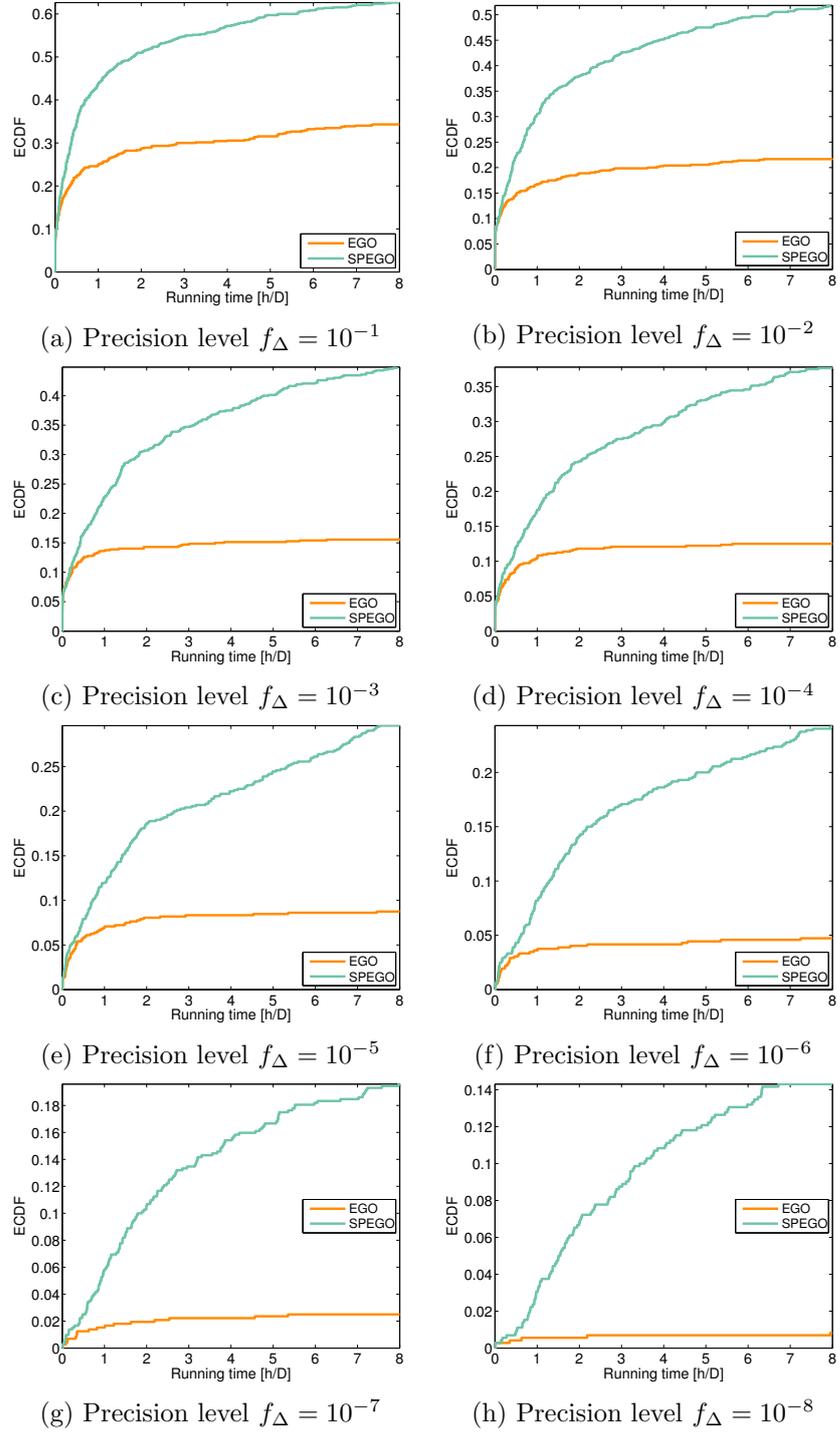


Figure 3.10: **Running time** comparison of **EGO** and **SPEGO**, showing the proportion of converged trials as a function of time, for a total running time budget of  $8D$  hours. Times reported represent real time in dedicated single Intel<sup>®</sup> Xeon<sup>®</sup> CPU 5160 cores running at 3.00GHz.

- SMAC stands for sequential model-based algorithm configuration procedure [Hutter, Hoos, and Leyton-Brown, 2013], and is a modified version of EGO. The only differences are that it uses an isotropic Matérn kernel instead of the anisotropic Gaussian one, and the heuristics used to maximize the EI. A Matérn kernel is defined as in Equation (2.9), but replacing the use of the squared exponential (Gaussian) covariance function (2.4), by the Matérn covariance function, which is defined as

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \frac{1}{\Gamma(\beta) 2^{\beta-1}} \left( \sqrt{2\beta} \frac{\mathbf{x} - \mathbf{x}'}{\alpha} \right)^\beta B_\beta \left( \sqrt{2\beta} \frac{\mathbf{x} - \mathbf{x}'}{\alpha} \right), \quad (3.1)$$

where  $\Gamma$  is the gamma function,  $B_\beta$  is the modified Bessel function of the second kind, and  $\alpha, \beta \in \mathbb{R}^+$  are parameters of the covariance.

A GP with Matérn covariance function has sample paths that are  $\lfloor \beta - 1 \rfloor$  times differentiable, and as  $\beta \rightarrow \infty$ , the Matérn covariance function tends to the squared exponential covariance function [Rasmussen and Williams, 2006]. SMAC justifies the use of a less smooth kernel to achieve better approximations of functions with sharp edges, which are often encountered in the BBOB. Besides, by using the same lengthscale for each dimension (isotropic property), large savings on the computational effort are achieved, and —according to the authors— the performance for highly dimensional problems is improved.

The output data generated by testing CMA-ES and SMAC on the BBOB, is publicly available<sup>2</sup>, and this is the data used to benchmark SPEGO. Given the expensive nature of the optimizers, there is only available data for 103D function evaluations, so even though we present results up to 200D in Figure 3.11 for SPEGO, the budget at which the comparisons are made is restricted to  $N = 103D$ , which is shown with a dashed vertical line. All throughout this figure, the SMAC algorithm reaches the target value at all precisions for all instances of  $f_5$  (the linear function) within only 3 samples, translating in reaching a probability of 0.042 (or 15/720). This explains the large jump observed for the first number of function evaluations. Similarly, CMA-ES finds the target for the same set of functions —although in a more progressive manner—, with a noticeable lower number of function evaluations

---

<sup>2</sup><http://coco.gforge.inria.fr/>

Table 3.7: **Benchmark of SPEGO** against **CMA-ES**, a state of the art algorithm, and **SMAC**, another EGO based algorithm, at a budget of  $N = 103D$ . For precisions  $f_{\Delta} = 10^{-2}$  to  $f_{\Delta} = 10^{-6}$ , SPEGO outperforms CMA-ES and SMAC with a confidence level of 95%.

Algorithm		$10^1$		$10^0$		$10^{-1}$		$10^{-2}$		$10^{-3}$	
		2	3	2	3	2	3	2	3	2	3
1	CMA-ES	<b>0.0000</b> <sup>-</sup>	<b>0.0000</b> <sup>-</sup>	<b>0.0000</b> <sup>-</sup>	<b>0.0000</b> <sup>-</sup>	<b>0.0380</b> <sup>-</sup>	<b>0.0000</b> <sup>-</sup>	0.0540 <sup>-</sup>	<b>0.0000</b> <sup>-</sup>	0.0728 <sup>+</sup>	<b>0.0001</b> <sup>-</sup>
2	SMAC		<b>0.0001</b> <sup>+</sup>		0.3349 <sup>-</sup>		<b>0.0002</b> <sup>-</sup>		<b>0.0001</b> <sup>-</sup>		<b>0.0000</b> <sup>-</sup>
3	SPEGO										
Algorithm		$10^{-4}$		$10^{-5}$		$10^{-6}$		$10^{-7}$		$10^{-8}$	
		2	3	2	3	2	3	2	3	2	3
1	CMA-ES	0.1145 <sup>+</sup>	<b>0.0002</b> <sup>-</sup>	0.1389 <sup>+</sup>	<b>0.0010</b> <sup>-</sup>	0.1437 <sup>+</sup>	<b>0.0022</b> <sup>-</sup>	0.1385 <sup>+</sup>	<b>0.0036</b> <sup>+</sup>	0.1355 <sup>+</sup>	<b>0.0042</b> <sup>+</sup>
2	SMAC		<b>0.0000</b> <sup>-</sup>		<b>0.0001</b> <sup>-</sup>		<b>0.0001</b> <sup>-</sup>		<b>0.0001</b> <sup>-</sup>		<b>0.0001</b> <sup>-</sup>
3	SPEGO										

than for the rest of the functions, which in particular for Figures 3.11g and 3.11h, creates a flattening effect at the same probability level for more than 35 function evaluations.

The statistical tests presented in Table 3.7 were performed at the same budget as shown by the gray dashed line in Figure 3.11 ( $N = 103D$ ). For this comparison, no running time is reported, since this data is not available for the selected algorithms. The selected configuration for SPEGO is the same as in Section 3.3.1 (LHS with  $\lambda = 6$  as initial DoE, the random partitioning dimension selection criterion, and a partitioning threshold  $\nu = 12D$ ). The comparisons take into account simulation runs on the full BBOB testbed for dimensions 2 and 3.

The results displayed in Table 3.7 show that SPEGO outperforms SMAC almost consistently, at a statistically significant level. The two scenarios where this is not the case are for the lowest precision target errors  $f_{\Delta} = 10^1$  and  $f_{\Delta} = 10^0$ . This is explained by two reasons. First, for all dimensions, SMAC converges to the solution at the highest precision level for 15 out of 15 instances of the linear function ( $f_4$ ) with only 3 samples, and second, SMAC does not use an initial design, allowing it to find better low precision solutions at very early stages. When comparing SPEGO against CMA-ES, we observe that SPEGO is outperformed only for the highest precision target errors ( $f_{\Delta} = 10^{-7}$  and  $f_{\Delta} = 10^{-8}$ ), indicating that SPEGO finds good solutions, but perhaps fails in sharing information globally, which might lead to better convergence at high precision targets.

A detailed comparison of the performance of the three algorithms, separated by family of functions (according to the classification defined in the BBOB), is provided in Appendix A.

### 3.4 Conclusions

Motivated by the good performance EGO has shown when optimizing expensive-to-evaluate objective functions, and by its inability to handle large numbers of samples due to its high computational cost, in this chapter we proposed SPEGO, a variation of EGO based on the idea of partitioning the search space. The rationale to partition the space is to fit local response surface models at each region and calculate the local EI. Then, the best next location to obtain a sample is determined by comparing the local EI, given its absolute nature. By partitioning the space, the cubic computational complexity of EGO as a function of the total number of samples is shown to be reduced to linear by SPEGO.

Furthermore, three design choices, required for the implementation of SPEGO, were put to the test. First, we evaluated the impact of the initial DoE in the overall performance of the algorithm, from where we concluded that using LHS with low budgets is beneficial. Second, we evaluated three criteria to select a dimension along which to partition the space, including the maximum gap, minimum length-scale, and random. The experiments provided evidence to state that the random criterion, which ensures that on average there are no partitions of disproportionate length in different dimensions, contributed more to the overall performance of SPEGO. And third, we assessed the impact of varying the size of the partition, however for this test the results were inconclusive. Moreover, NSPEGO, an extension to SPEGO that includes points from neighboring regions to improve the accuracy of local response surface models near the arbitrarily defined borders, was compared against SPEGO. The improvement in performance of NSPEGO with respect to SPEGO was not found to be statistically significant, so the simpler model (SPEGO) was selected as a better choice.

In order to understand the real advantages of SPEGO against EGO —its non-partitioned counterpart—, we performed a comparison of the two algorithms under fair conditions, using the BBOB, by allocating a fixed running time budget

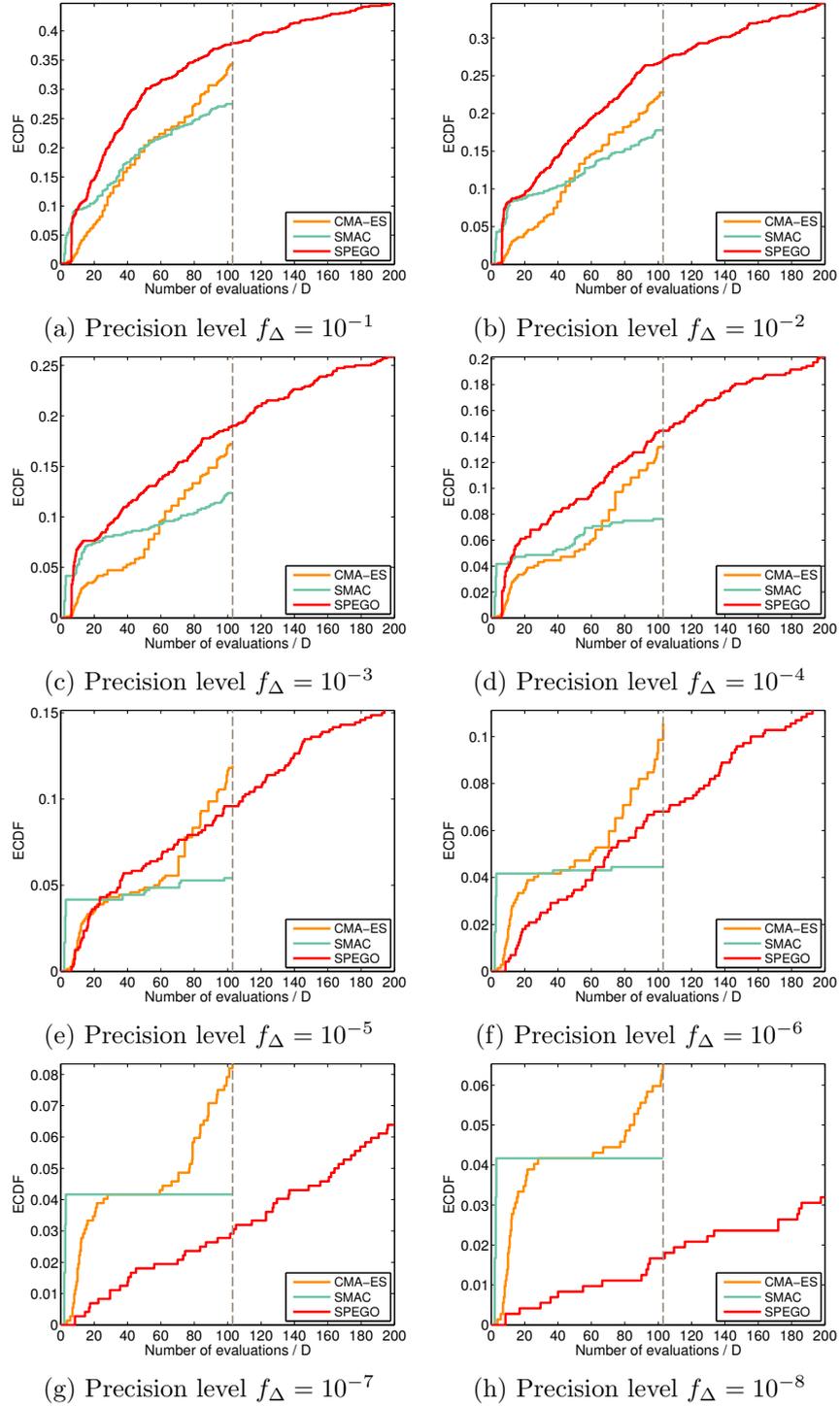


Figure 3.11: ECDF plots comparing the proportion of converged trials of **SPEGO**, **CMA-ES**, and **SMAC** as a function of number of evaluations. Lines are plotted only for the regions where information is available. All comparisons are performed at a budget of  $N = 103D$ , indicated by a dashed vertical line.

to each. The results were analyzed from two perspectives, by fixing the number of function evaluations, and by fixing the running time. The numeric experiments agreed with the theory, which states that for a given number of function evaluations, EGO should perform better than SPEGO, since it uses all the information available, while SPEGO uses only local information. The results obtained also showed how the number of function evaluations that SPEGO can handle can be orders of magnitude larger than the one EGO can manage in the same amount of time, agreeing with the computational complexity analysis. From these results we take that for function evaluations that are extremely expensive, it might be worth considering EGO, but otherwise, SPEGO is a better choice.

Finally, in order to understand how SPEGO compares against other algorithms found in literature, we benchmarked it against the state of the art (CMA-ES) and another EGO based algorithm (SMAC) at a low function evaluation budget. The result was encouraging, showing that SPEGO consistently outperformed SMAC, and that only at the two highest precision target errors CMA-ES outperformed SPEGO. These promising results indicate that it is worth performing an in-depth analysis of the convergence properties of SPEGO, perhaps studying its behavior for each class of functions available in the comprehensive set provided by the BBOB, in order to further improve its performance.

## Chapter 4

# Tracking global optima in dynamic environments with efficient global optimization

In this chapter we consider the problem of finding and tracking the global optima of an expensive unknown function that changes over time. This is similar to the problem treated in earlier chapters with the added complication of the changes in the objective function. This twist calls for a paradigm shift: instead of looking for a global optimum, it must be tracked ; the effectiveness of an optimizer is not longer measured with respect to a fixed value, but rather with respect to the changing goal ; and —amongst others— when fitting a response surface, the age of the observations must be taken into account. However, some similarities remain, which make it possible for the techniques introduced in Chapter 2 to be adapted for the dynamic problem.

This chapter starts by presenting a literature review on dynamic optimization of black-box functions in Section 4.1. In Section 4.2 we propose several methods to adapt the surrogate assisted optimization techniques based on GP introduced earlier in this thesis in order to approach such a problem, while complying with the requirements imposed by its dynamic nature. Then, in Section 4.3 the moving peaks benchmark (MPB) and some performance measures specifically designed for this problem are introduced, so that fair comparisons amongst the simulations explained

in Section 4.4 can be done. Finally, the results are critically analyzed in Section 4.5 along with some concluding remarks.

## 4.1 Related work

The dynamic version of the problem discussed in previous chapters deals with tracking the global optimum of an expensive black-box function changing over time. This calls for a more sophisticated exploration strategy than in the static case capable of keeping track of promising solutions that might become useful at later times.

In the general case, the changes of the objective function can happen each time the function is evaluated, after a given number of evaluations, or after a given period of time. The frequency of the changes depends on the nature of the problem to be solved, for instance, after a given number of performed experiments, or at the beginning of every season. Some studies focus on change detection [Eberhart, 2002; Richter, 2009], but in this thesis we assume the frequency of changes is known in advance, and that it is measured in terms of function evaluations rather than in time.

The nature and severity of the changes are of paramount importance. If the problem changes completely and there are no similarities between the objective function before and after the change, the best one can do is re-start optimization after every change. In most real world scenarios, however, changes are subtle, and thus it should be possible to transfer some useful information from the search process so far to the search after a change. On the other hand, care must be taken to maintain the search capabilities of the algorithm and not overly rely on outdated information that may be misleading.

The idea of using stochastic processes to model a dynamically changing function with the aim of tracking its global optimum was first introduced by Kushner [1962] for a one dimensional problem, and did not receive much attention by the scientific community thereafter. Apart from Morales-Enciso and Branke [2014], which presents part of the work discussed in this Chapter, we are not aware of any other work that employs stochastic processes to build response surfaces to perform sequential sampling in order to track the global optima of a dynamically changing

objective function. On the other hand, optimization in dynamic environments has been a hot research topic in the area of evolutionary computation over the last years, and surveys can be found, e.g., in Jin and Branke [2005] and Nguyen, Yang, and Branke [2012]. Typically, approaches are classified into four categories:

- Introducing diversity after a change. To restore the exploration capability of the algorithm, after a change, specific measures are taken to increase diversity. The most prominent example is hypermutation [Cobb, 1990] which temporarily increases the mutation rate after a change in the environment.
- Maintaining diversity. Rather than introducing diversity after a change, other papers propose to maintain diversity in the population throughout the run. Typical examples in this category are random immigrants [Grefenstette, 1992] where some individuals are generated randomly in every generation, and the thermodynamical genetic algorithm [Mori, Kita, and Nishikawa, 1996] that tries to strike a balance between quality and diversity in the selection step.
- Memory. These approaches maintain a memory of good solutions found in the past, either implicitly by using multiploidy<sup>1</sup> [Hadad and Eick, 1997], or explicitly see e.g. [Yang, 2008].
- Multi-population approaches. The main idea in this category is to split the population into multiple sets which can be used to concurrently search in several promising areas of the search space, see, e.g., [Branke, Kaufler, Schmidt *et al.*, 2000; Yang and Li, 2010; du Plessis and Engelbrecht, 2012].

In most of the previous work, independent of the category, the information transferred from one search stage to the next is in the form of previously found good solutions. Models which build a response surface using old samples updated with new coming information are not found in the literature. So, in the next section, four techniques to track the global optima of a dynamic expensive black-box function based on a response surface are described and compared.

---

<sup>1</sup>Multiploidy refers to the existence of more than one gene expression (set of parameters) for each individual, all of which can be dominant or recessive depending on the current conditions of the environment.

## 4.2 Adaptations of EGO to the dynamic case

The following sequential sampling strategies for parameter optimization of dynamic black-box problems build on the principles of the static version explained in detail in Section 2.4.2. So, each time a new sample is obtained, the response surface is rebuilt by updating the GP with the new observation. Once the response surface has been built, the EGO mechanism is used to determine where to sample next.

The key difference when building response surfaces in dynamic environments is that data of different age is available. Old data should be considered less reliable than recent data because of the changes that have taken place since the time the data was collected.

As stated in Section 4.1, we address the problem where the objective function changes after a known number of evaluations  $c_f \in \mathbb{N}$  (change frequency), and the periods in between changes are referred to as epochs ( $t \in \mathbb{N}$ ) numbered in increasing order. However, it is not the epoch at which each sample was obtained that is relevant to discount the reliability of the sample, but rather how long ago it was taken with respect to the current epoch ( $t_c \in \mathbb{N}$ ). So, instead of using the epoch number, it is the age of a sample with respect to the present ( $\tau = t_c - t$ ) that is considered to reduce reliability of the samples.

Below, seven strategies are described. The first three are simple strategies used as benchmarks. First, a *random* sampling strategy is proposed to compare against the completely uninformed case. Then, two limiting cases are presented: the *reset* strategy as a memoryless model that starts solving the problem from scratch after every change, and the *ignore* strategy that disregards all the changes and considers all the information equally reliable. The last four sampling strategies are proposed as different ways of reusing and transferring information from old epochs to the new ones, exploiting different properties of GP, and constitute the main contributions of this chapter. These seven methods are compared through numerical experiments in Section 4.4.

In order to build a response surface using a GP, it is necessary to start with at least  $\lambda = 2$  ( $\lambda \in \mathbb{N}$ ) data samples to be able to estimate the length-scale parameters of the process. So, for the first epoch it is assumed that there are at least  $\lambda \geq 2$  observations previously obtained following any of the initial DoE techniques

introduced in Section 2.2.

Let  $\mathcal{D}$  be the set of all the samples collected throughout the history of the experiment, and  $\mathcal{D}_\tau \subset \mathcal{D}$  the set of data points of age  $\tau$ .

#### 4.2.1 Random strategy

The *random* sampling strategy explores the parameter space  $\mathbf{x} \in \mathbb{R}^D$  by independently drawing a random number from a uniform distribution for each dimension. This technique serves only as a benchmark in order to set a reference to assess the improvements of the other techniques, and there is no response surface built.

#### 4.2.2 Reset strategy

This strategy discards all the previously obtained samples every time a change on the objective function happens. This is equivalent to a re-start, as if this were a new problem. So, at the current epoch ( $\tau = 0$ ), the response surface will be estimated using only current information in  $\mathcal{D}_0$ . Since previous samples are not considered, at the beginning of each epoch  $\lambda$  observations need to be sampled in order to start building the response surface one more time.

The *reset* strategy also serves as a reference to measure the improvement obtained by other sampling strategies. Besides, it is useful in the presence of very drastic changes where there is no similarity between the objective function before and after each change.

#### 4.2.3 Ignore strategy

As its name suggests, the *ignore* strategy overlooks the fact that a change has happened, which means that all the available samples in  $\mathcal{D}$  are used to fit the response surface. Not only is this a bad strategy to find the global optima of a changing function because old information is taken to be as valid as new one, potentially misleading the search, but also because it unnecessarily increases the computational cost of generating the GP. This is the opposite extreme to the *reset* strategy and serves as another benchmark. The *ignore* strategy is useful when the magnitude of the changes is negligible and the problem is thus similar to a static problem.

#### 4.2.4 Reset\* sampling strategy

*Reset\** differs from *reset* (Section 4.2.2) only in the way the first samples of a new epoch (other than the first one) are taken. Instead of taking  $\lambda$  initial observations at the beginning of a new epoch ( $\tau = 0$ ), *reset\** looks for the best response found in the immediate previous epoch ( $\tau = 1$ ) and resamples at the same place where this previously best response was obtained. Furthermore, the length-scale parameters ( $\ell$ ) found at the end of the immediate previous epoch are reused in order to overcome the inability of fitting a GP with only one data point and allow to take a second sample. Once the second sample has been obtained, the sampling process continues as the *reset* strategy (i.e. refitting the GP parameters from the available data ( $\mathcal{D}_0$ ) every time a new sample becomes available) until the next function change.

#### 4.2.5 Discounted information through noise sampling strategy (DIN)

The idea behind this strategy is to consider newly obtained samples as deterministic—as it has been done throughout all the thesis—, but to introduce some artificial measurement noise in order to discount the old samples. The recent observations, being treated as deterministic (no noise added), force the response surface to go exactly through the measured sample, while the old observations, treated as noisy observations, allow the response surface to pass within some distance of the actually observed response values (proportional to the magnitude of the introduced noise) but not necessarily through them. By considering old information but discounting its accuracy, the search is guided to the regions where there used to be good responses in order to explore if that is still the case, but it is acknowledged that the landscape might have changed.

GP provide a natural way of introducing noise in different magnitudes for each data sample through the noise measurement term ( $\sigma_n^2$ ) in Equation (2.4). Furthermore, the introduced noise can be a function of the age of the observations. This modification gives rise to

$$k(\mathbf{x}, \mathbf{x}') = \sigma_f^2 \exp\left(-\sum_{d=1}^D \frac{(x_d - x'_d)^2}{2\ell_d^2}\right) + \sigma_n^2(\tau)\delta(\mathbf{x}, \mathbf{x}'), \quad (4.1)$$

which is used to calculate the covariance matrix needed to generate the response

surface for the DIN model. An illustration of this model is provided in Figures 4.1 and 4.2, which show the behavior of DIN for two different noise values. The two cases are provided for illustration purposes, but no clear conclusion can be drawn from these two cases regarding the value of the noise to be chosen.

$\sigma_n^2(\tau)$  is now a function of the age of the samples and no longer a constant as in Equation (2.4), and can be any strictly increasing function in  $\tau$  such that  $\sigma_n^2(\tau_c) = 0$ , for instance

$$\sigma_n^2(\tau) = \tau s^2, \quad (4.2)$$

where  $s \in \mathbb{R}$  is some constant noise level.

The introduced noise  $\sigma_n^2(\tau)$  increases as a function of the age of the samples following a predefined functional form which is user defined rather than learnt.

Since DIN uses samples from previous epochs, it is not necessary to generate any random sampling nor to reuse the GP parameters from previous epochs other than for the first epoch. Nonetheless, the first sample of each epoch is taken where the best response was obtained at the previous epoch, following the same procedure as in *reset\** (Section 4.2.4).

#### 4.2.6 Time as an additional dimension sampling strategy (TasD+1)

Another way of using the temporal information is by considering time as an additional dimension ( $D + 1$ ). The advantage of this method is that it learns the time correlations from the data instead of relying on arbitrarily chosen functions or noise levels as it is done in the DIN method explained in Section 4.2.5. The learning process takes place when estimating the introduced parameter  $\ell_{D+1}$  and it is done by maximizing its likelihood. Nevertheless, having an additional parameter to estimate during the likelihood maximization increases the difficulty of finding the optimal parameters as compared to the other models.

Introducing an additional dimension is naturally done through the covariance function (2.4). Consider an observed data point  $\mathbf{x} \in \mathbb{R}^D$ , then let  $\tilde{\mathbf{x}} \in \mathbb{R}^{D+1}$  be the vector containing  $\mathbf{x}$  augmented by the age of the samples  $\tau$ . Then, the correlation

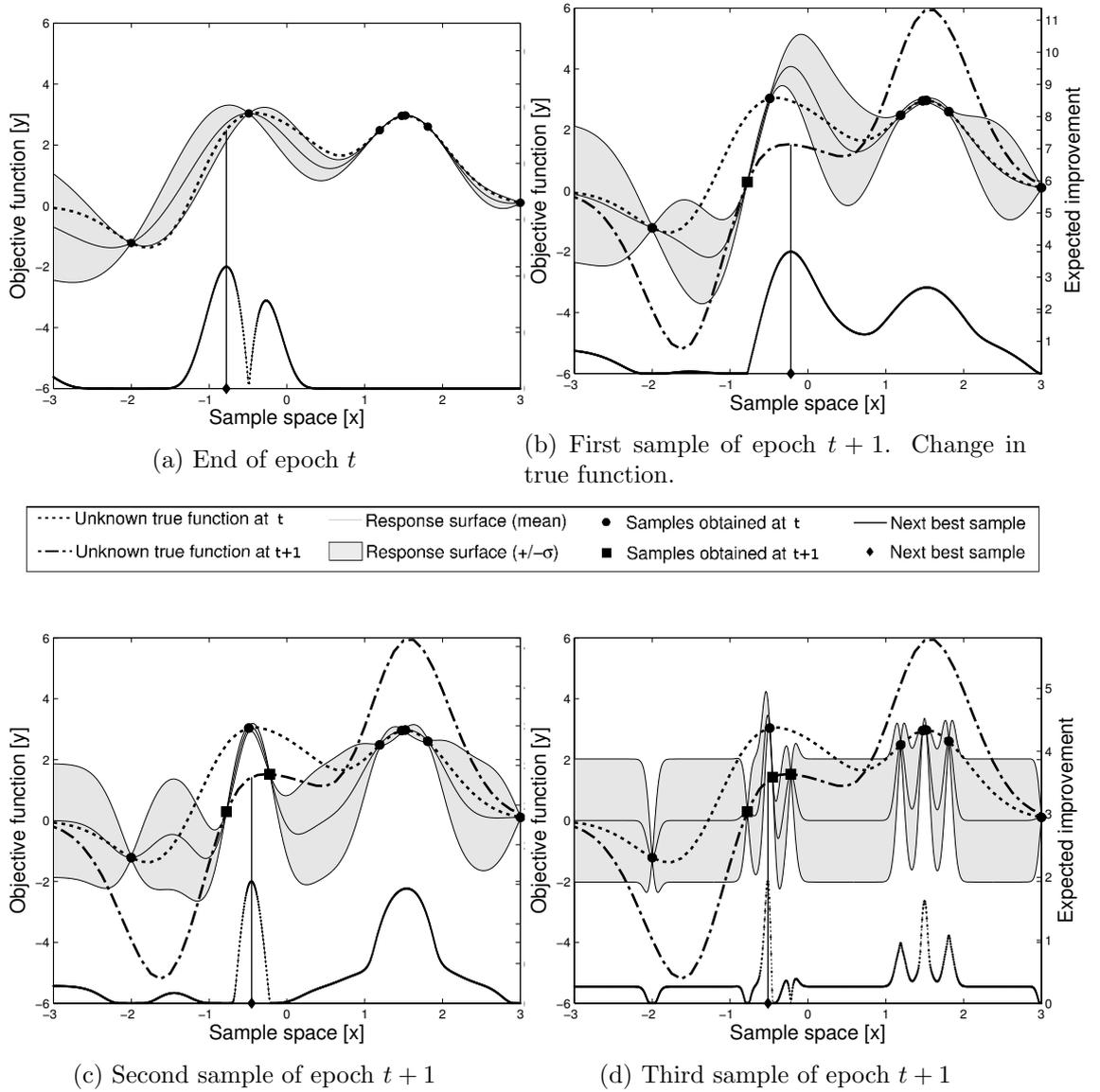


Figure 4.1: Sequential sampling using increased measurement noise information decay  $s^2 = 0.1$  (Eq. 4.2). 4.1a shows the end of an epoch, along with the GP response surface generated using samples obtained at  $t$ . The confidence interval presented for the response surface corresponds to  $\pm\sigma$  away from the predicted mean. The vertical line shows where the next best sample should be taken according to EGO. In 4.1b the new sample has been taken (square), but the true objective function has changed. When fitting the GP, the response surface passes exactly through the new observation even if there are other old samples in the region. But in the absence of recent information in other regions, old data is used to guide the response surface, resulting in wider confidence intervals in those regions due to the introduced noise. 4.1c and 4.1d show the next two samples taken. 4.2a through 4.2d show the same procedure for a noise level  $s^2 = 2.0$ .

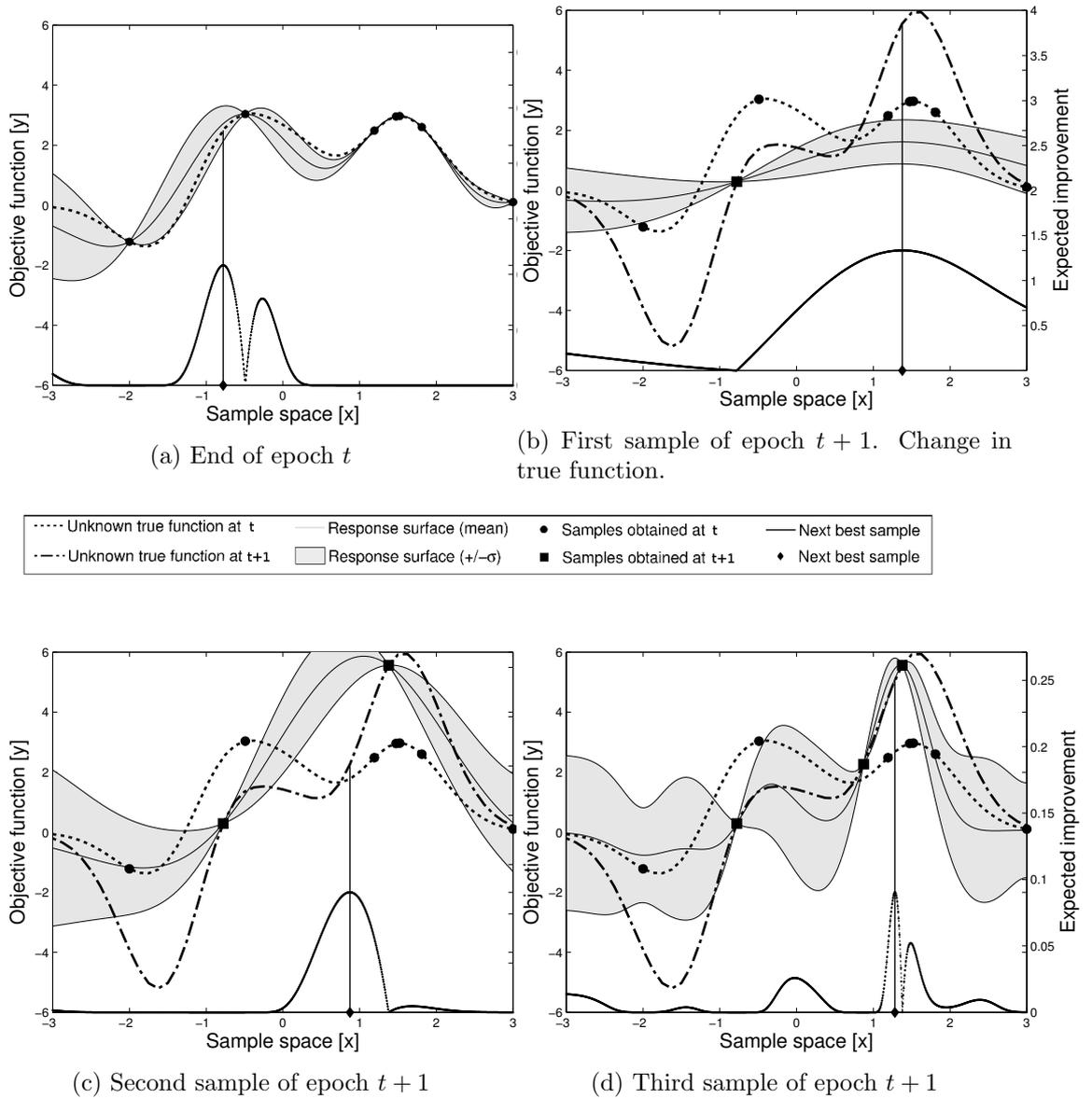


Figure 4.2: Figures 4.2a through 4.2d illustrate the same sequential sampling following the DIN strategy as shown in Figure 4.1 with an increased measurement noise level of  $s^2 = 2.0$ .

function between two augmented samples can be written as

$$k(\tilde{\mathbf{x}}, \tilde{\mathbf{x}}') = \sigma_f^2 \exp\left(-\sum_{d=1}^{D+1} \frac{(\tilde{x}_d - \tilde{x}'_d)^2}{2\ell_d^2}\right). \quad (4.3)$$

The procedure for estimating the response surface is not changed. When evaluating the response surface from the surrogate model,  $\tau$  must be set to zero so as to make valid predictions for the current time (i.e.  $\tilde{x}'_{D+1} = 0$ ).

The first sample of each epoch is chosen exactly as in the DIN strategy (Section 4.2.5).

#### 4.2.7 Previous surface mean prior sampling strategy (PSMP)

The models presented so far have all assumed a zero mean prior GP, as proposed in Equation (2.3). However, since in the regions where there is no data to support any prediction the GP tends to the mean prior, the mean prior of a GP can be used to introduce any information available. The rate at which the response surface tends to the mean prior in each dimension depends on the corresponding inferred length-scale ( $\ell_d$ ). An example of this behavior can be seen in Figure 4.1 throughout all the subfigures in the sample space interval  $[-3, -2]$ , where the predicted response surface tends to zero lacking data to support any other predictions.

PSMP exploits this property of GP in order to transfer information about good regions found at previous epochs to the current epoch through the mean prior by providing a tailored estimate of the expected value at each point. The information is introduced into the current model through the mean prior function in the shape of the response surface available at the end of the previous epoch, which is a GP as well.

This gives rise to a recurrent definition where the mean prior of the GP used to generate the response surface at any epoch is itself the GP that was used at the end of the immediate previous epoch. Since more than one response surfaces are now in play, it becomes necessary to explicitly identify the age of the data being used to generate them, which we do through a time subindex that explicitly indicates this. With this notation,  $\mathbf{X}_t$  is the vector of data points observed at time  $t$ , and  $\mathbf{K}_t$  is the covariance matrix generated by applying the kernel function to  $\mathbf{X}_t$  (Eq. 2.4).

Following this notation, let  $m_t(\mathbf{x}_p)$  be the expected value of the prediction at point  $\mathbf{x}_p$  made using a GP fitted to all the data points from epoch  $t$  and a mean prior which embeds the most recent information from the previous epoch  $t - 1$ . Then, following Equation (2.11) which provides the expected value of a GP,

$$m_t(\mathbf{x}_p) = \begin{cases} m_{t-1}(\mathbf{x}_p) + k(\mathbf{x}_p, \mathbf{X}_t)\mathbf{K}_t^{-1}\mathbf{Y}_t & \text{if } t > 0 \\ \bar{m} + k(\mathbf{x}_p, \mathbf{X}_t)\mathbf{K}_t^{-1}\mathbf{Y}_t & \text{if } t = 0 \end{cases}, \quad (4.4)$$

where

$$\bar{m} = \frac{\sum_{i=1}^{\lambda} y_i}{\lambda} \quad (4.5)$$

is used as a mean prior for the initial epoch, since there is no past information to rely on. This means that for the first epoch ( $t = 0$ ), PSMP uses the mean of the first  $\lambda$  samples as a constant mean prior (Eq. 4.5).

The first observation of any other epoch ( $t > 0$ ) is taken where the best response was obtained at  $t - 1$ , following the procedure explained for *reset\** (Section 4.2.4) including the reuse of the parameters.

Before fitting the parameters ( $\theta$ ) by MLE using Equation (2.7), care must be taken to subtract the mean prior from the vector of observations  $\mathbf{Y}_t$ . Equation (2.3) in the definition of the GP is then replaced by Equation (4.4). Finally, when using the fitted GP to maximize the expected improvement—or for making any predictions—the previously removed mean must be added back, which is already considered in Equation (4.4).

### 4.3 Experiment setup for comparing dynamic optimizers

In this section, the various methods proposed in the previous section are compared empirically using the MPB, a standard benchmark used for dynamic global optimization. A thorough explanation of such benchmark along with two performance measures are provided, followed by a detailed description of the experiments performed to test the performance of each of the seven presented models. The statistical analysis of the experiments is left for Section 4.4.

### 4.3.1 The moving peaks benchmark

Even though there are many real examples of objective functions evolving over time, it is not easy to find cases which are both complex enough to present a challenge and simple enough to analyze and make an interpretation of the tuned parameters. The moving peaks benchmark provides a framework bridging this gap between very complex, hard to understand real-world problems and all too simple toy problems [Branke, 1999].

This benchmark consists of a  $D$  dimensional continuous function defined in a given interval with  $N \in \mathbb{N}$  peaks of different height and width. Each peak is defined by its position  $\mathbf{x}_i \in \mathbb{R}^D$ , height  $h_i \in \mathbb{R}$ , and width  $w_i \in \mathbb{R}$  ( $i \in [1, \dots, N]$ ). At every change, each of the peaks suffers a slight random variation in position, height, and width. For the position of the peaks, the magnitude of the change ( $v_L \in \mathbb{R}$ ) is fixed, but the direction is random. The changes in height and width of the peaks are each independent, normally distributed, and scaled according to their corresponding severity parameter ( $h_s \in \mathbb{R}$  and  $w_s \in \mathbb{R}$  respectively). Besides, these three components are bounded by their corresponding upper and lower real-valued bounds:  $(\mathbf{x}_l, \mathbf{x}_u)$ ,  $(h_l, h_u)$ , and  $(w_l, w_u)$ . In all cases, the boundary conditions are reflective, which means that after a change, if the updated parameter falls outside the bounds by some magnitude, such parameter would bounce back with the same magnitude.

The benchmark is one of the classic benchmarks for dynamic optimization problems in the evolutionary computation area. To be successful, an algorithm has to be able to track a moving peak, but also to jump from one peak to another if the heights change such that another peak becomes the highest peak.

### 4.3.2 Performance measures for dynamic optimization

A standard performance measure for dynamic optimization problems is the offline error [Branke, 1999]. The offline error is the time-averaged error of the best solution found so far in the epoch. Since the objective function changes after a number of function evaluations, we require two time indices to keep track of both the total number of function evaluations, and the function evaluations that belong to a same epoch. Let  $j_t \in [1, \dots, c_f]$  denote the number of function evaluations performed so far

within epoch  $t$ . Then, the total number of function evaluations performed, across all epochs, is given by  $T = c_f t + j_t$ , since  $c_f$  is a known constant. Let  $\epsilon_{j,t}$  denote the evaluation error between the  $j$ -th solution evaluated during epoch  $t$  ( $y_{j,t}$ ) and the global minimum  $f_t^*$  at the current epoch  $t$ , so that

$$\epsilon_{j,t} = f_t^* - y_{j,t}. \quad (4.6)$$

Then, the error of the best solution evaluated so far since the last change is called the current error and calculated as

$$\epsilon_{j,t}^c = \min_{i=1}^j \epsilon_{i,t}. \quad (4.7)$$

The offline error is then just the average over all current errors:

$$\epsilon_T^o = \frac{1}{T} \sum_{i=1}^T \epsilon_i^c, \quad (4.8)$$

where  $T$  is the total number of solutions evaluated so far. The offline error assumes that evaluations are done offline (hence the name), i.e. the best known solution found so far since the last change is actually implemented in the real world while the search for a better solution continues in a separate process.

Another performance measure we look at is the average error, defined as

$$\bar{\epsilon}_T = \frac{1}{T} \sum_{i=1}^T \epsilon_i, \quad (4.9)$$

which measures the average deviation from the global optimum of each function evaluation performed so far. For both performance measures, offline and average errors, if the  $T$  index is dropped, we refer to the error measured at the end of the run.

### 4.3.3 Implementation details

The implementation of the MPB simulates the sequential sampling process applying the seven different strategies described in Section 4.2 in the attempt of tracking the global optima. The parameters governing the dynamics of the objective function

are detailed in Table 4.1.

As discussed in Chapter 3, fitting a GP has computational complexity of  $O(|\mathcal{D}|^3)$ , so the process slows down with each new sample incorporated to the data set. This has an indirect implication on the scalability of the presented technique to problems with a large number of dimensions given that the number of samples required would rapidly increase. For efficiency purposes, only data from the immediate previous epoch (i.e.  $\tau = 1$ ) was considered.

All the simulations start with an initial number of  $\lambda = 4$  samples, and when applicable the same number of initial samples is used at the beginning of later epochs. Then, one of the proposed strategies is followed to fit a GP to the available data.

The EGO policy is followed, so the expected improvement function (Eq. 2.16) needs to be maximized. Since only the 1 and 2 dimensional cases are considered, the EI maximization is performed using the HB method as explained in Section 2.6.3.

Table 4.1: Parameters governing the dynamics of the moving peaks. Whenever more than one value is presented for a parameter, values in bold show the default parameters (base case) and the remaining values are variations used to study the behavior of the proposed strategies under different conditions.

Parameter	Value		Description
	1D	2D	
Number of dimensions $D$	1D	2D	Dimensionality of the parameter space
Number of peaks	5	5	Number of peaks in the objective function
Min coordinate ( $\mathbf{x}_l$ )	0.0	0.0	Minimum coordinate for each dimension
Max coordinate ( $\mathbf{x}_u$ )	100.0	100.0	Maximum coordinate for each dimension
Min peak height ( $h_l$ )	30.0	30.0	Minimum possible height of the peaks
Max peak height ( $h_u$ )	70.0	70.0	Maximum possible height of the peaks
Height stdev	0.0	0.0	Starting value for the height of the peaks. 0 for uniform random within ( $h_l, h_u$ )
Min peak width ( $w_l$ )	1.5	0.05	Minimum width of a peak
Max peak width ( $w_u$ )	2.5	0.15	Maximum width of a peak
Width stdev	2.0	0.1	Starting value for the width of the peaks. 0 for uniform random within the Min and Max width interval
vLength ( $v_L$ )	<b>0.25</b> , 0.5	<b>0.25</b> , 0.5	Distance a single peak moves when a change happens
Height severity ( $h_s$ )	<b>7.0</b> , 15.0	15.0	Intensity of the changes made to the height in one function change
Width severity	0.01	0.01	Standard deviation of the changes made to the width in one function change
Basis function used	false	false	Whether a static basis function is used or not
Correlation lambda	0.0	0.0	Correlation between consecutive movements of a single peak (0 for no drift, > 0 for drift)
Change frequency ( $c_f$ )	25	50	Number of evaluations after which a change takes place
Epochs	80	20	Number of function changes
Peak function	Inverse squared		Function describing the shape of the peaks
Change step size	Constant		Allows alteration of vLength parameter

## 4.4 Numerical results and model comparison

### 4.4.1 Experimental procedure

In order to compare the different models presented in Section 4.2, a set of experiments with different parameter settings was run. Table 4.1 shows the parameters for the MPB. Values in bold are the default parameters used for the base case whenever more than one value was tested for a parameter. In this section only the base case is considered, and the parameter analysis is left for Section 4.4.2. The parameters chosen for the base case are  $D = 1$ ,  $v_L = 0.25$ ,  $h_s = 7.0$ , and  $c_f = 25$ . The simulations are run for 80 epochs, and all results are averaged over  $R = 64$  independent runs.

Since the DIN sampling strategy (Section 4.2.5) requires parameter tuning for the noise level, each experiment has to be run in two steps. The first step is to find out the optimal noise level  $s^*$  by running a first set of simulations of the optimizer using the DIN sampling strategy with different noise discount values, and empirically choosing the one with best performance. Given the high computational cost that running a full set of experiments entails, we restrict the full analysis to only one performance measure. In this case, offline error is chosen as the preferred measure of performance since this reflects a real life scenario where the best known solution would be implemented, while the search for a better parameter configuration carries on. So the remainder of the experiments focus mainly on this performance measure, but the same procedure would apply for the average error. Since the changes of the objective function are stochastic, several replications are required to provide statistical significance to the interpretation of the results. So,  $R = 64$  replications were run in this first part of the experiment.

Figure 4.3 shows how the performance of the DIN strategy changes according to the chosen discount noise. A discount noise level of  $s = 0$  (no discount at all) corresponds to a version of the *ignore* strategy with the first solution evaluated after a change being the best found solution from the previous epoch, while an infinite discount of the old samples ( $s \rightarrow \infty$ ) makes the approach more similar to the *reset\** sampling strategy but does not take the initial  $\lambda$  samples at the beginning of new epochs. For comparison purposes, the performance of three other strategies

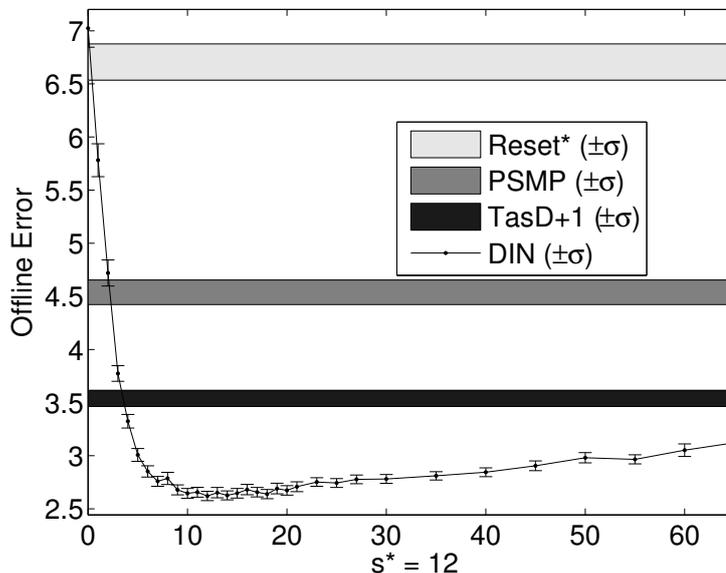
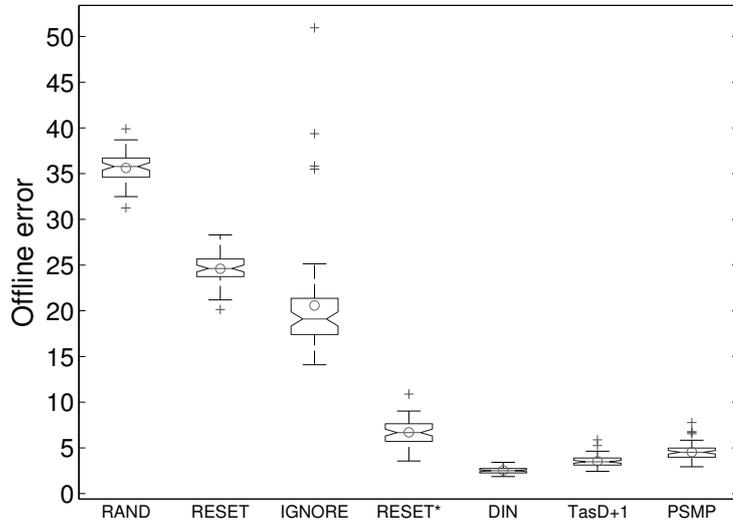


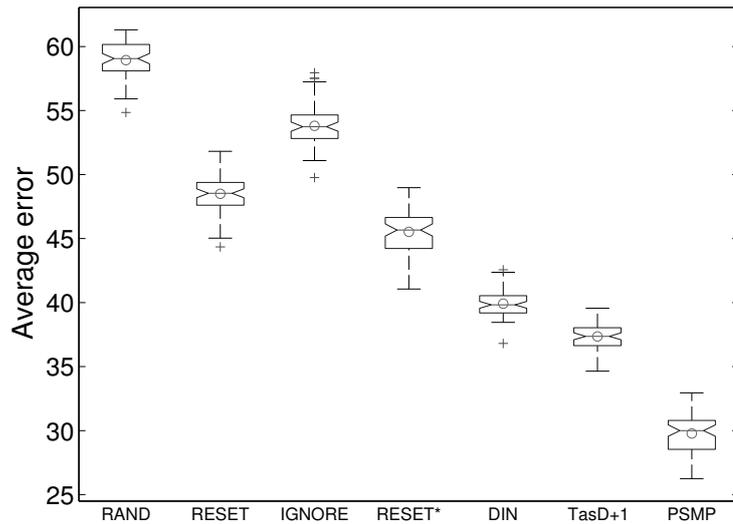
Figure 4.3: Parameter sweep of different noise levels for the DIN sampling strategy considering the offline error (Eq. 4.8). The optimal noise level,  $s^* = 12$  (Eq. 4.2), can be read from this plot and is the value used for the second part of the experiment. Furthermore, the offline error for three other sampling strategies (reset\*, TasD+1, and PSMP) are shown for reference as shaded areas.

(reset\*, TasD+1, and PSMP) is displayed in the background as well. We observe that for some values of  $s$ , TasD+1 and PSMP outperform the DIN strategy, but for well tuned values of  $s$ , the DIN strategy outperforms the others. This reveals the importance of choosing an appropriate value for the discount noise, although an exhaustive tuning might not be required since values close to the optimum do not vary drastically in their performance.

Once the DIN strategy has been tuned, all the strategies can be run to assess their performance. For this part of the experiment another  $R = 64$  replications were run using common random numbers across strategies. These replications do not share common random numbers with the previous step. For each of the seven sampling strategies, the results at the end of the 80 epochs are shown in Figure 4.4 using box plots, which allows an easy visual comparison, although it is not as powerful as the statistical tests performed in Section 4.4.3. We observe that the dominating strategy depends on the performance measure selected. For instance, if we consider



(a) Offline error  $\epsilon_o$



(b) Average error  $\bar{\epsilon}$

Figure 4.4: 4.4a shows the offline error (Eq. 4.8), and 4.4b the average error (Eq. 4.9) of the seven sampling strategies for the base case simulation as described in Table 4.1 (1D case), averaged over 64 replications. For each box, the central horizontal line is the median, the round marker is the mean, the horizontal edges represent the 25<sup>th</sup> and 75<sup>th</sup> percentiles, and the whiskers show the  $\pm 2.7\sigma$  intervals beyond which the extreme cases are displayed individually. Finally, the notches are useful to compare whether the medians from two models are significantly different at the 5% level (if their intervals do not overlap). However, this statistical comparison does not exploit the use of common random numbers (see Section 4.4.3).

offline error, DIN dominates all the other strategies, whereas if we consider average error, PSMP is the best. Nevertheless, the advantage of reusing information in a discounted manner is consistent across performance measures, which is evidenced by the fact that *reset\**, DIN, TasD+1, and PSMP—all reusing information from previous epochs—outperform the strategies that either do not use it (*reset*) or do not discount it (*ignore*).

In terms of computational cost, strategies that use data only from the current epoch (*reset*, *reset\**, and PSMP) to fit the GP are much faster than those using data from previous epochs (*ignore*, DIN, and TasD+1). The difference is due to the computational complexity of fitting a GP. Comparing DIN and TasD+1, the latter has an additional parameter to fit, and so TasD+1 has a larger computational cost when maximizing the likelihood. However, this is negligible when compared to the parameter tuning for DIN that requires several replications, each requiring to fit a large number of GP, for each attempted parameter value.

In order to better understand how the offline error behaves throughout the simulation and to verify that the comparison of the final values happens after convergence, the whole evolution across time is visualized in Figure 4.5.

Finally, the current error plots, displayed in Figure 4.6, are useful to understand the effect of each sampling strategy. We can see that, at the beginning of each new epoch, strategies dismissing old samples (*random* and *reset*) start with a similarly poor solution in every epoch, and require some time to find good solutions. The *ignore* strategy benefits of old samples at the beginning of later epochs, but because the information it relies on is outdated, it does not manage to improve much after that and its performance seems to deteriorate from epoch to epoch. The first sample of the last 4 strategies is taken at the location where the best observation was made in the previous epoch, which explains the fact that they start with a large advantage in terms of current error. However, they all treat old data in different ways which accounts for the difference in performance. A better comparison of the convergence to the global optimum at each epoch can be seen with the overlapped current errors presented in Figure 4.6, where it is shown that even if *reset\** has a good starting point, it gets stuck in good—yet not optimal—regions, which is perhaps due to the lack of memory about good regions from previous epochs. DIN,

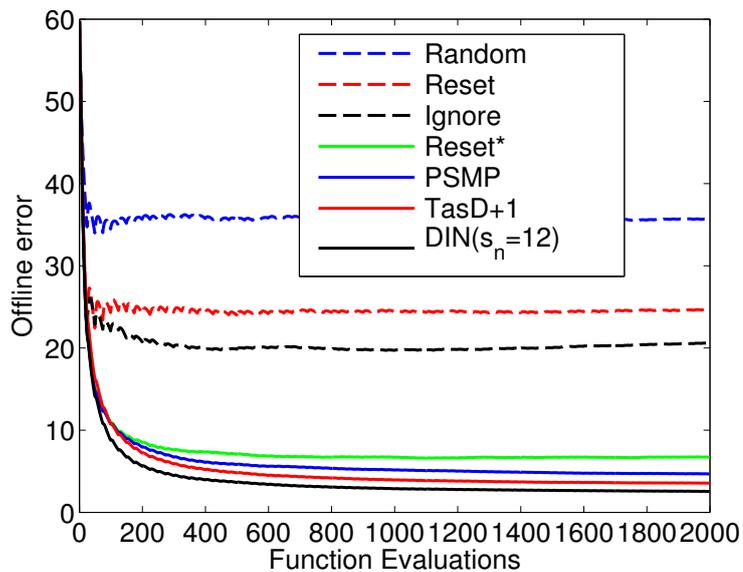


Figure 4.5: Offline error (Eq. 4.8) for the seven compared models throughout the 80 epochs of the simulation. During the first 25 function evaluations all the models display the same behavior, but after the first objective function change, the paths start to diverge since each sampling strategy uses old information differently.

TasD+1 and PSMP do not seem to get stuck, although they seem to have different convergence rates, with the DIN strategy being the fastest to find the global optima.

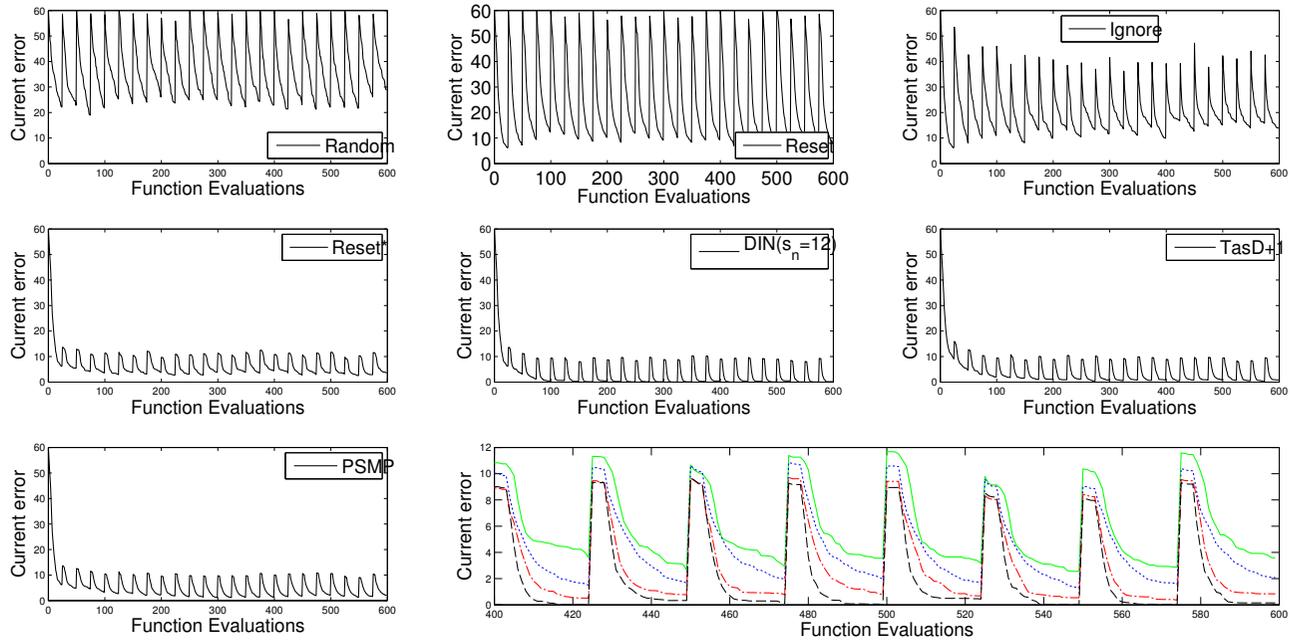
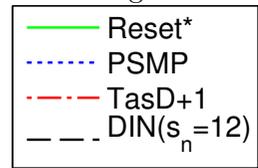


Figure 4.6: Average over 64 replications of the current error (Eq. 4.7) displayed individually for the seven compared models (24 first epochs), and then overlapped for the 4 best performing strategies in a zoom of 8 epochs. The current error plots help in understanding the behavior of the strategies.

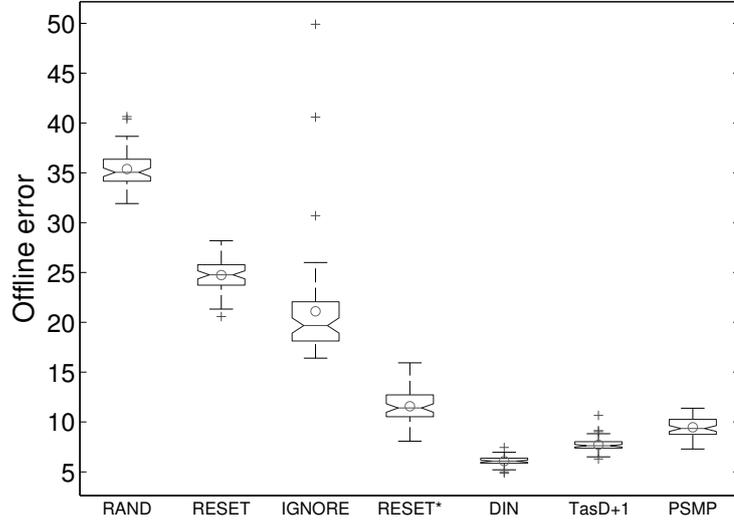


#### 4.4.2 Parameter analysis

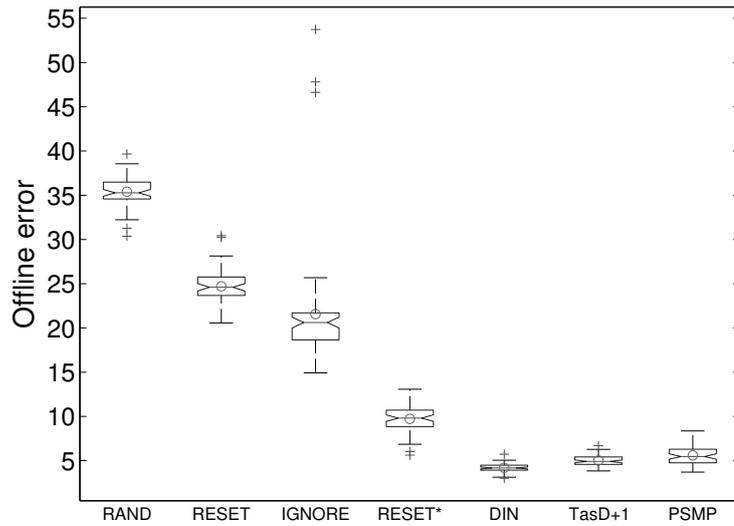
The sampling strategies make different use of the available information, so we expect their performance to depend on the magnitude of the variations suffered by the dynamic objective function. In order to explore this, two more experiments were run. First, the configuration ( $v_L = 0.5, h_s = 7.0$ ) was used to simulate an increase in the distance for which the locations of the peaks change, and second the parameters were set to ( $v_L = 0.25, h_s = 15.0$ ) so as to simulate a scenario where the peaks move as fast as the reference, but the global maximum is more likely to change from one peak to another. Both scenarios make historic information less reliable than the base case. The offline errors measured at the end of these simulations for each strategy are summarized in Figure 4.7, where we observe that even if the ranking is preserved, for an increased  $v_L$ , all the strategies using old information perform worse than in the base case, which is expected since old information is less reliable due to the greater magnitude of the changes in between epochs. However, when increasing  $h_s$ , only the performance of *reset\** significantly deteriorates, while the other strategies with memory preserve a similar performance with respect to the base case. Given that an increased height severity means that the global maximum jumps from peak to peak more frequently, it makes sense that strategies preserving information about different good regions in previous epochs (and not only about the best region as *reset\**) are better at tracking the global optimum in this case.

To verify whether these results extend to higher dimensions, two more experiments were run, this time for the 2 dimensional case, including the base case, and the increase in the distance for which the locations of the peaks change. Except for the length of the simulations and the frequency of changes, all the other parameters were kept as for the 1D base case. Since the difficulty of finding the global maximum of a function increases with the dimensionality of the function, the frequency of changes was decreased to allow more function evaluations in between changes. For these experiments, there are  $c_f = 50$  function evaluations in each epoch, the experiments were run for 20 epochs, and  $R = 64$  replications of each experiment were performed. The results obtained are compiled in Figure 4.8.

In the experiments performed, the DIN sampling strategy outperforms TasD+1. This might be due to the fact that TasD+1 has to learn an additional

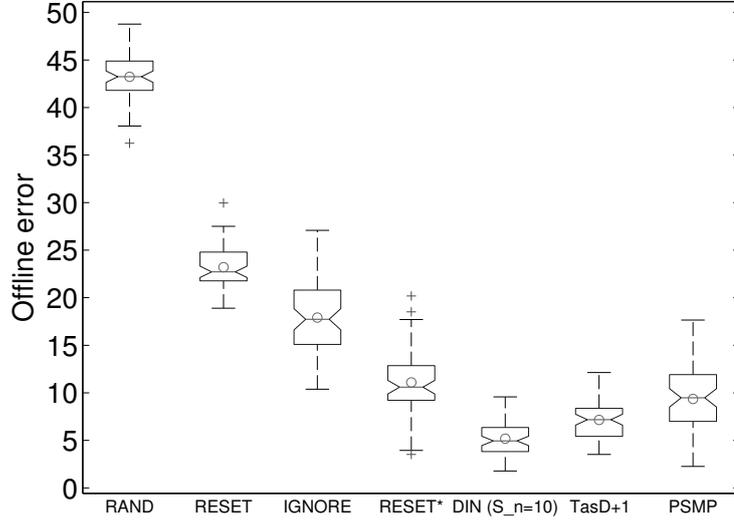


(a) Increased  $v_L$

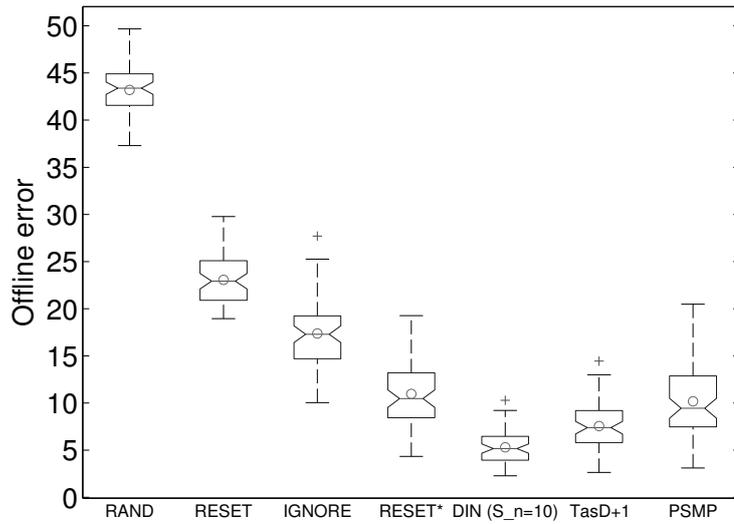


(b) Increased  $h_s$

Figure 4.7: Box plots showing the offline error (Eq. 4.8) averaged across 64 replications of the seven sampling strategies at the end of 80 function changes, each happening after 25 function evaluations. In 4.7a, an increased distance for peak location changes was used ( $v_L = 0.5, h_s = 7.0$ ) while in 4.7b an increased severity of peak height changes was used ( $v_L = 0.25, h_s = 15.0$ ), all with respect to the base case shown in Figure 4.4.



(a) 2D, base case



(b) 2D, increased  $v_L$

Figure 4.8: Box plots showing the offline error (Eq. 4.8) of the seven sampling strategies for the 2 dimensional version of the MPB, averaged across 64 replications. Performance measures taken at the end of 20 function changes, each happening after 50 function evaluations. 4.8a sets the base case for comparison with ( $v_L = 0.25, h_s = 15.0$ ), and 4.8b shows an increased speed for peak location changes ( $v_L = 0.5, h_s = 15.0$ ).

parameter (time scale). However, since DIN has a parameter that requires manual tuning (through an extensive set of simulations), TasD+1 might be preferred unless similar problems are solved repeatedly, in which case the effort of tuning parameters may be justified.

PSMP loosely outperforms the *reset\** sampling strategy in the 1 dimensional case, but the effective difference for the 2 dimensional case is small. This is probably related to the number of samples used to approximate the mean prior, which for the 1 dimensional case is 25 and seems to be enough to provide an accurate approximation of the previous epoch’s landscape. Nevertheless, since the number of required samples to create a good approximation of the landscape is expected to increase as a power of the dimension, 50 samples might not be enough to create a good approximation of the previous epoch in 2 dimensions.

These simulations show that the DIN model outperforms all others for the tested scenarios, but the fact that a previous tuning needs to be made to find the optimal discount noise level must not be discarded, since it requires a lot of computation. So, using the DIN strategy only makes sense if similar problems are solved repeatedly, and for practical purposes or under the constraints of limited resources the TasD+1 strategy might be preferred.

#### 4.4.3 Statistical comparison of model performance

The box plots shown above provide a visual guide on the individual performance of the presented sampling strategies for each experiment, nevertheless they do not exploit the advantage of using common random numbers for the simulations. To compare the performance of the different models taking advantage of the pairing, we use the Wilcoxon signed-rank statistical test [Wilcoxon, 1945; Fay, Proschan, and Others, 2010].

Consider the seven models compared, and number them in ascending order as they were presented so that the *random* sampling strategy is 1, *reset\** is 2, and so on. For a given experiment, let  $\tilde{\epsilon}_i$  ( $i \in [1, \dots, 7]$ ) denote the median offline error of the  $R$  replications of strategy  $i$ . And let the null hypothesis ( $H^0$ ) be that *the median difference between the pairs of observations is zero*. So the alternative hypothesis

( $H^1$ ) is that *the median difference of the pairs is not zero*, which we can write as

$$\begin{aligned} H_{ij}^0 &: \tilde{\epsilon}_i = \tilde{\epsilon}_j, \quad i, j \in [1, \dots, 7] \\ H_{ij}^1 &: \tilde{\epsilon}_i \neq \tilde{\epsilon}_j, \quad i, j \in [1, \dots, 7]. \end{aligned} \tag{4.10}$$

After performing the set of tests and calculating the corresponding p-values ( $p_{ij}$ ),  $H_{ij}^0$  can be rejected—in favour of  $H_{ij}^1$ —at the  $\alpha_{ij} = 1 - p^c$  significance level if  $p_{ij} < p^c$ .

These null hypotheses were tested for all of the configurations described in order to rank the strategies according to their performance in offline error. Furthermore, since several hypotheses are being tested with the same experiments, the problem of multiplicity arises, due to the fact that the likelihood of witnessing a rare event increases with the number of tests performed. To account for this fact, we use the Bonferroni correction [Dunnett, 1955]. We found that for all the  $1D$  cases  $H_{ij}^0$  can be rejected using  $p^c = 0.001$ , or—equivalently— at the 99.9% significance level. For the  $2D$  scenarios the same is true for  $p^c = 0.002$ . A ranking of the different strategies evaluated according to the median offline error for all the experiments described is presented in Table 4.2, where we can see that the ranking of the strategies does not change across different settings.

## 4.5 Discussion and conclusions

An adaptation of EGO, the sequential sampling strategy presented and used in Chapters 2 and 3, to track global optima in dynamic environments is proposed and tested in this chapter. Specifically, *reset\**, DIN, TasD+1, and PSMP, four sequential sampling strategies relying on GP to build a surrogate model are described which—to the author’s best knowledge—constitutes the first approach to dynamically changing optimization problems by means of a surrogate-based global search algorithm.

Different properties of GP are exploited by each of the proposed strategies to construct the response surface using both old and new information in order to enhance tracking of the global optima for dynamic expensive black-box objective functions. These four new sampling strategies, together with three other benchmark

Table 4.2: Ranking in increasing order of median offline error of the different strategies according to the median offline error achieved at each experiment  $\tilde{\epsilon}$ . The p-value shown for each row is the probability of the difference in medians between the strategy in the same row and the one in the row immediately below being different from zero by chance, according to the Wilcoxon signed-rank test performed between each pair of sampling strategies, considering Bonferroni correction.

(a) 1D experiments

$v_L = 0.25, v_s = 7.0$		$v_L = 0.5, v_s = 7.0$		$v_L = 0.25, v_s = 15.0$	
Strategy	p-value	Strategy	p-value	Strategy	p-value
DIN	< 0.01	DIN	< 0.01	DIN	< 0.01
TasD+1	< 0.01	TasD+1	< 0.01	TasD+1	< 0.01
PSMP	< 0.01	PSMP	< 0.01	PSMP	< 0.01
Reset*	< 0.01	Reset*	< 0.01	Reset*	< 0.01
Ignore	< 0.01	Ignore	< 0.01	Ignore	< 0.01
Reset	< 0.01	Reset	< 0.01	Reset	< 0.01
Random		Random		Random	

(b) 2D experiments

$v_L = 0.25, v_s = 15.0$		$v_L = 0.5, v_s = 15.0$	
Strategy	p-value	Strategy	p-value
DIN	< 0.01	DIN	< 0.01
TasD+1	< 0.02	TasD+1	< 0.02
PSMP	< 0.01	PSMP	< 0.01
Reset*	< 0.01	Reset*	< 0.01
Ignore	< 0.01	Ignore	< 0.01
Reset	< 0.01	Reset	< 0.01
Random		Random	

sampling strategies are compared through numeric simulations. The comparisons use an implementation of the widely accepted moving peaks benchmark and the offline error as performance measure. To better understand the advantages and disadvantages of each proposed strategy, different scenarios are tested. And to ensure the significance of the conclusions drawn, a statistical analysis is performed on the numerical results obtained from the experiments.

The poor performance of the *random* strategy throughout the different experiments confirms the advantages of using informed selection of the points to be sampled. The simple trick of re-evaluating the previous best found solution at the beginning of an epoch highly improves the performance in the experiments considered. This idea has been used in all the newly proposed strategies. The experiments also show that sampling strategies using old information in an explicit way (DIN, TasD+1, and PSMP) systematically perform significantly better than those which either discard it (*reset* and *reset\**) or treat it in the same way as recent information (*ignore*).

## Chapter 5

# Conclusions and future work

The need for optimizing expensive-to-evaluate objective functions often arises when dealing with systems where the resources needed to acquire new observations involve large investments, long waiting times, human resources, or disruptions to systems in production. In these cases, it is important to exploit the expensively obtained information as much as possible in order to reach the desired goal with as few samples as possible. One approach to deal with this type of problem is to build a model of the system, given the available information, in the form of a response surface, and then use this model to anticipate promising configurations of the system.

We start this work by presenting a literature review of response surface aided global optimization methods, which leads to the justification of the efficient global optimization algorithm (EGO) as the preferred method on which the rest of the thesis is based, not only because it is well established in the sequential optimization field and it has proven to be useful in a wide variety of applications, but also because it requires less computational resources than other available methods, and the analytical tractability it provides. Since EGO relies on the optimization of the expected improvement (EI), an auxiliary function that is cheap to evaluate but which exhibits an increasing number of local maxima as the number of available samples increases, the impact of using suboptimal solutions for the EI was empirically investigated. A set of numerical experiments revealed the importance of finding the global optimum for the EI, which motivated the need to look for more efficient methods of finding it. Therefore, we proposed the hyper-box method to select starting points to be

used when maximizing the EI, which proved to be beneficial when tested for two dimensional cases, but due to its bad scaling with the dimensionality of the problem, it was not used for higher dimensions. Instead, a genetic algorithm with population and generation sizes that adapt to the size of the dataset and the dimensionality of the problem to be solved, was devised and suggested for optimizing the EI.

Due to the mechanics of sequential sampling optimization, it is common that a large number of samples concentrate in good regions, separated only by arbitrarily small distances, which can induce numerical instabilities during matrix inversions. This problem was also addressed by introducing a taboo method, capable of removing the numerical instabilities by keeping the most informative set of samples, ensuring that new observations are not made within the identified unstable vicinity.

Given the expensive nature of Gaussian processes (GP), which increases as the cube of the number of samples available and directly affects EGO, we proposed SPEGO, a variation of EGO that partitions the search space and fits local response surfaces in order to determine the best next sampling location. We showed that the main advantage of SPEGO is its linear computational complexity as a function of the total number of available observations, as compared to the cubic computational complexity of EGO. Furthermore, we empirically demonstrated that this improvement in number of operations required per objective function evaluation has low impact on the quality of the solutions found, while allowing to handle larger sampling budgets in a fraction of the time. From this comparison we conclude that, for extremely costly objective functions, EGO might be a good choice, yet, for cases where the objective function is moderately expensive, SPEGO is a better option.

We also investigated the impact of three important implementation choices that need to be made when using SPEGO through numerical experimentation. The analysis of the results obtained highlighted the importance of using an initial design of experiment of reduced size, relative to the total available budget, and preferably generated using the Latin hypercube sampling method. In this way, the infill process starts as soon as possible, which is especially beneficial under reduced total sampling budgets. These results also showed that randomly selecting the dimension along which to create new partitions has a positive impact on the overall performance of the optimizer. Furthermore, they corroborated that increasing the size of the

partitions improves the performance of SPEGO.

In order to minimize the impact of the partitions arbitrarily imposed by SPEGO, we extended this algorithm to incorporate information from neighboring regions when fitting local response surfaces, which resulted in NSPEGO. Our experiments showed that the additional computational burden this modification entails is not necessarily justified, given the low improvement in performance observed.

Moreover, to understand how SPEGO compares against other algorithms found in literature, we selected as a benchmark CMA-ES, one of the state of the art algorithms as ranked in the 2013 version of the black-box optimization benchmark (BBOB), and SMAC, a slightly modified implementation of EGO. The results were encouraging, showing that SPEGO consistently outperformed SMAC, and that only for the two highest precision target errors considered, CMA-ES beat SPEGO.

Finally, we presented another adaptation of EGO, this time tailored to dynamic environments, where global optima need to be tracked. Four algorithms capable of incorporating old and new information were proposed, three of which use different properties of GP —upon which EGO relies. The first method consists of discarding all the old information but the location where the best observation was made before a change happened, and together with two other naive strategies that either discard the old information or just disregard that a change occurred, serves as a benchmark. The second method discounts old information by introducing some artificial measurement noise. The caveat of this method is that it requires tuning of the amount of noise to be used, which can be computationally intensive. The third proposed method considers time as an additional dimension, which naturally learns the dynamics of the objective function during the estimation of the corresponding timescale parameter. Unlike all the other methods that use a zero mean GP, the fourth method uses the response surfaces generated at previous epochs as a mean prior, in a recursive manner. By doing so, information reflecting previous states of the landscape is assumed and updated upon the arrival of new evidence. These four new GP based response surface building methods constitute the first approach to dynamically changing optimization problems by means of a surrogate-based global search algorithm.

In order to assess the performance of the different strategies under diverse conditions, we used the moving peaks benchmark, a specially designed benchmark for dynamic optimization. The numeric experiments showed that sampling strategies using old information in an explicit way consistently performed significantly better than those which either discard it or treat it in the same way as recent information.

Several promising research directions were identified throughout the development of this thesis. For instance, performing a detailed analysis of SPEGO, for each of the functions in the BBOB testbed, could lead to further improve its performance by identifying specific points of failure. Also, evaluating the impact of different methods to accelerate the response surface generation step, such as re-using the kernel lengthscales for more than one iteration, only re-fitting a subset of the lengthscales at each iteration, or adapting SPEGO to implement multi-point sequential sampling strategies, might lead to interesting results. Another highly promising improvement for the SPEGO algorithm would be to vary the partitioning threshold as a function of the lengthscales observed in each dimension. By dynamically adjusting the number of samples allowed in a partition, better convergence properties can be expected. In relation to NSPEGO, it might be worth quantifying the potential improvement in performance when using more than one neighboring sample.

Regarding the dynamic case, future work might focus on how to combine the different sampling strategies presented. For example, building a more accurate response surface from old samples using the TasD+1 method and using it as mean prior for the PSMP strategy. Or perhaps, trying to remove the tuning component for the DIN strategy by learning the amount of noise to be introduced online. Another direction could be to extend this work for dynamic objective functions with continuous changes, as opposed to changes happening only after a given number of function evaluations. Although TasD+1 could naturally cope with this case, individual levels of noise might be required for each sample if using DIN, and additional modifications would be required for PSMP to work.

A more ambitious goal that can be pursued in either the static or the dynamic

case, would be to adapt the methods exposed to make them capable of dealing with stochastic objective functions. For the dynamic case, this proves particularly challenging due to the difficulty in distinguishing old from noisy information.

Finally —and perhaps more interesting—, an online self-evaluation of the performance of the presented algorithms can be done by evaluating predictions, prior to obtaining a new observation, made with several models differing, for instance, only by the kernel used. This could provide information to internally rank the performance of each model, allowing the sequential optimizer to use the most suited configuration for each case, rendering it more versatile.

## Appendix A

# SPEGO against the state of the art, separated by BBOB type of function

In this appendix we present the result of SPEGO, compared to CMA-ES and SMAC, separated by type of function. The five types of function considered, as defined by the BBOB [Finck, Hansen, Ros *et al.*, 2010] are

- separable functions
- functions with low or moderate conditioning
- functions with high conditioning and unimodal
- multi-modal functions with adequate global structure
- multi-modal functions with weak global structure.

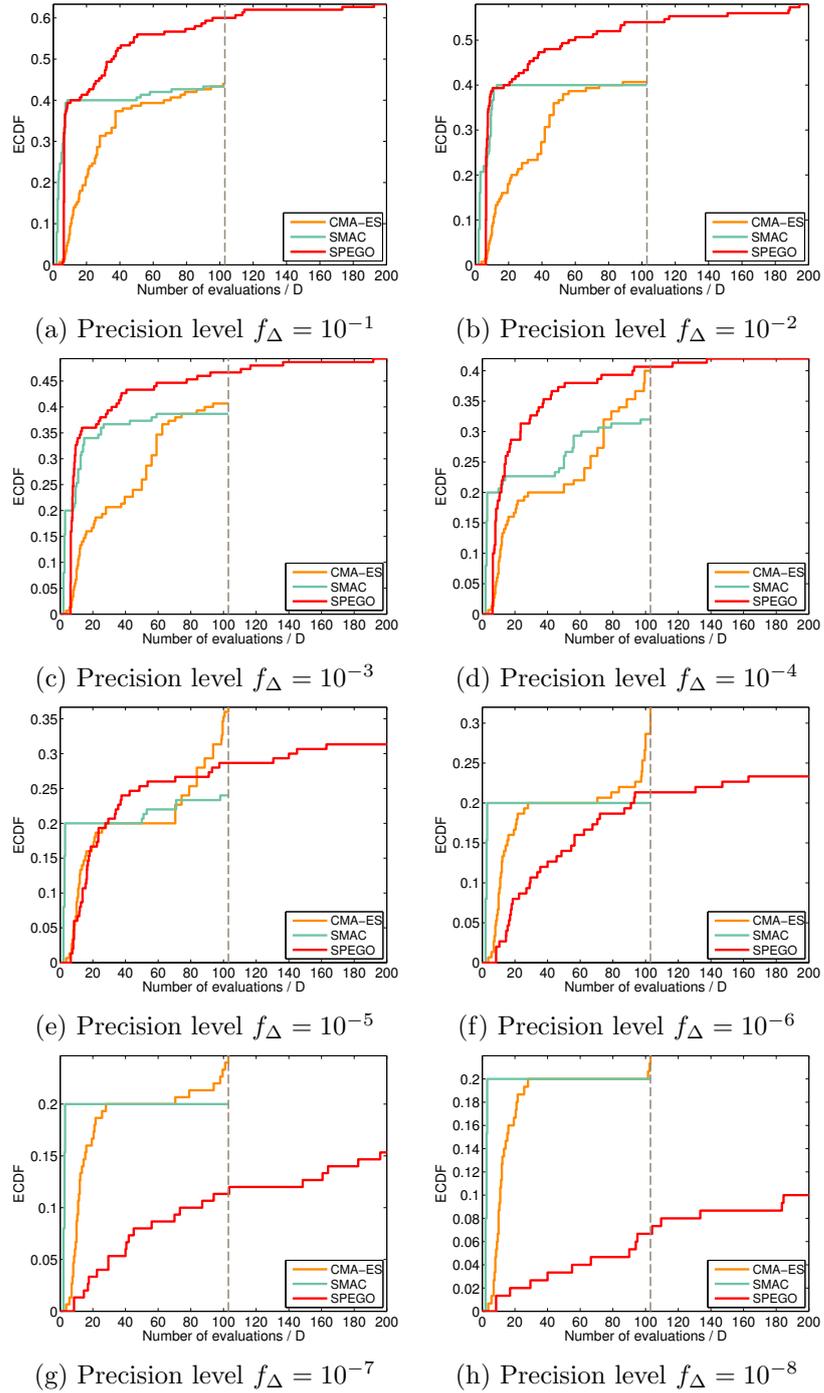


Figure A.1: ECDF plots comparing the proportion of converged trials in the **separable functions** of the BBOB testbed of **SPEGO**, **CMA-ES**, and **SMAC** as a function of number of evaluations. Lines are plotted only for the regions where information is available. All comparisons are performed at a budget of  $N = 103D$ , indicated by a dashed vertical line.

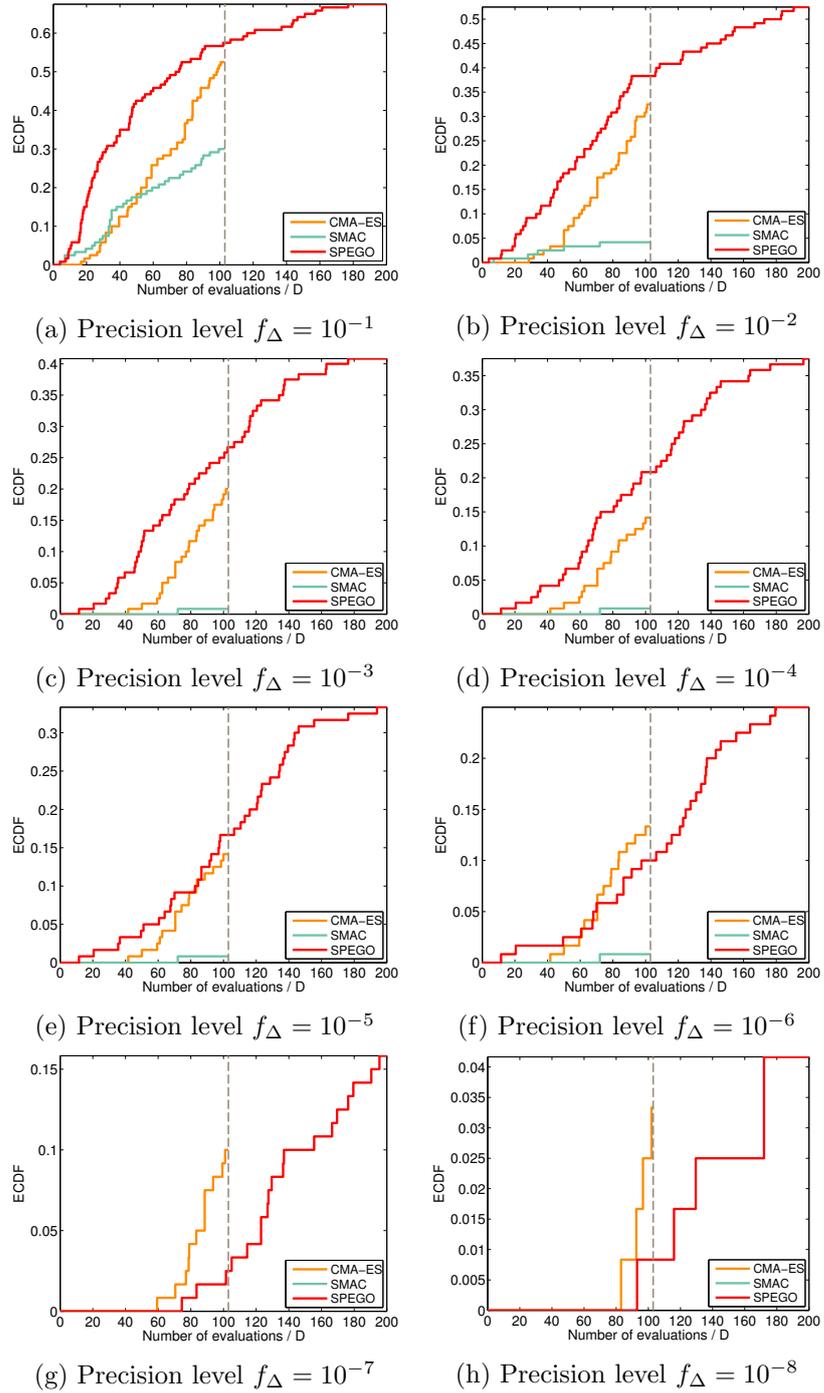


Figure A.2: ECDF plots comparing the proportion of converged trials in the **functions with low or moderate conditioning** of the BBOB testbed of **SPEGO**, **CMA-ES**, and **SMAC** as a function of number of evaluations. Lines are plotted only for the regions where information is available. All comparisons are performed at a budget of  $N = 103D$ , indicated by a dashed vertical line.

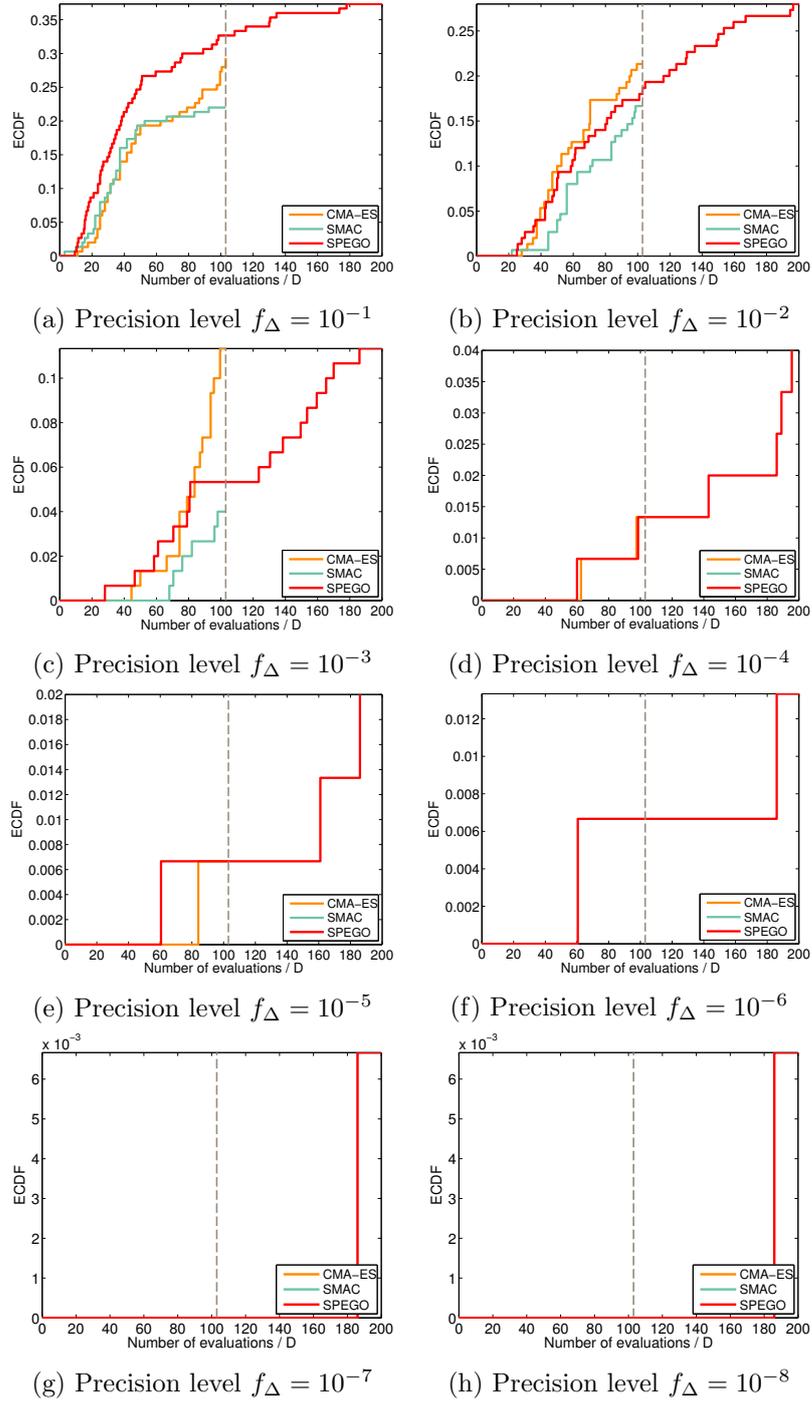


Figure A.3: ECDF plots comparing the proportion of converged trials in the **functions with high conditioning and unimodal** of the BBOB testbed of **SPEGO**, **CMA-ES**, and **SMAC** as a function of number of evaluations. Lines are plotted only for the regions where information is available. All comparisons are performed at a budget of  $N = 103D$ , indicated by a dashed vertical line.

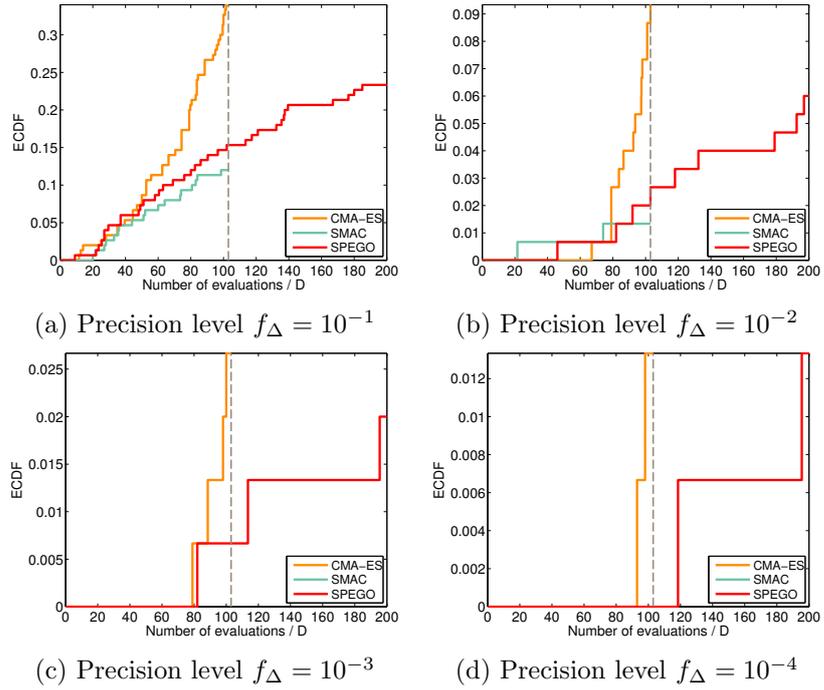


Figure A.4: ECDF plots comparing the proportion of converged trials in the **multi-modal functions with adequate global structure** of the BBOB testbed of **SPEGO, CMA-ES, and SMAC** as a function of number of evaluations. Lines are plotted only for the regions where information is available. All comparisons are performed at a budget of  $N = 103D$ , indicated by a dashed vertical line. (Only cases where the specified precision target was reached are shown.)

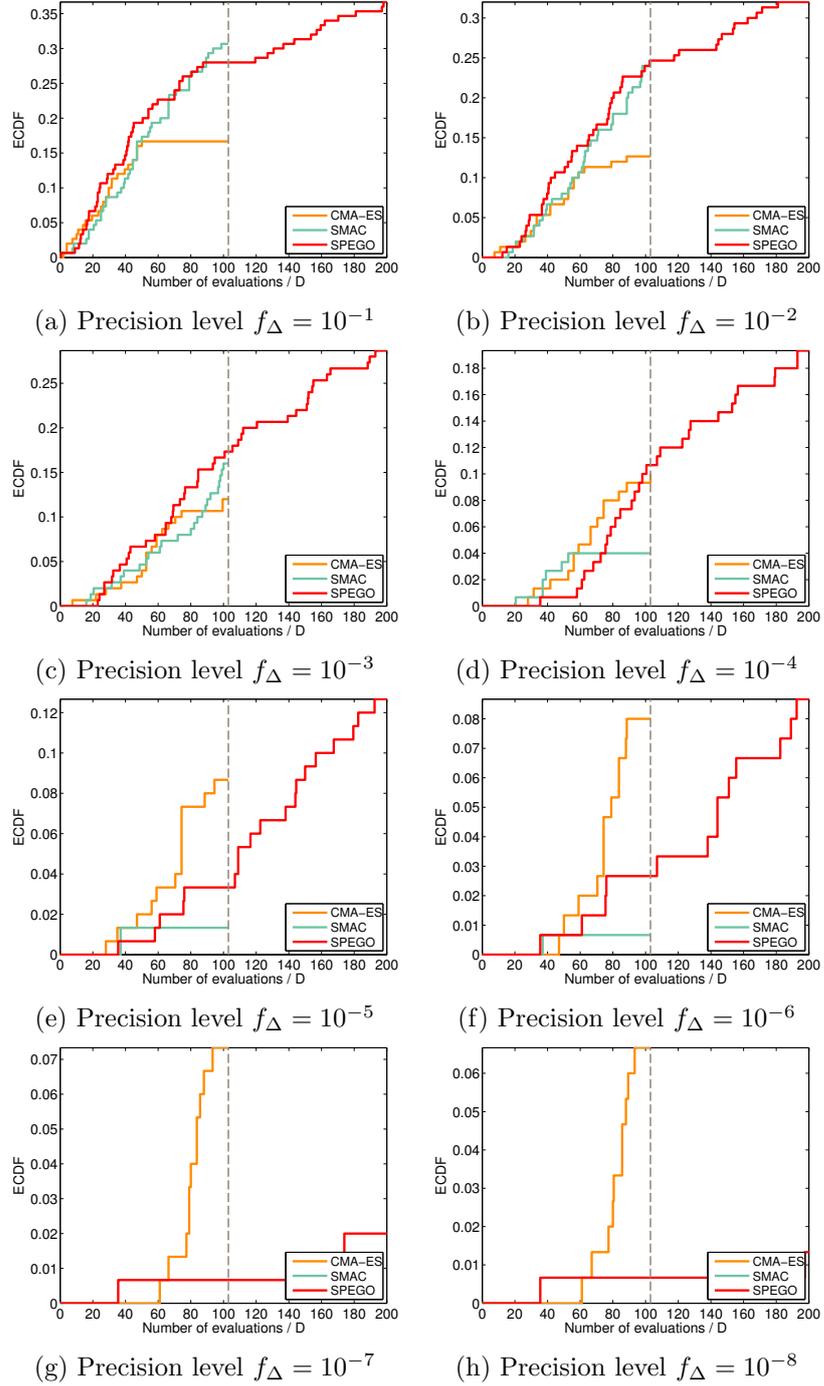


Figure A.5: ECDF plots comparing the proportion of converged trials in the **multi-modal functions with weak global structure** of the BBOB testbed of **SPEGO**, **CMA-ES**, and **SMAC** as a function of number of evaluations. Lines are plotted only for the regions where information is available. All comparisons are performed at a budget of  $N = 103D$ , indicated by a dashed vertical line.

# Bibliography

- Barillec, R., Ingram, B., Cornford, D., *et al.* Projected sequential Gaussian processes: A C++ tool for interpolation of large datasets with heterogeneous noise. *Computers & Geosciences*, **37**(3):295–309, Mar 2011. doi: 10.1016/j.cageo.2010.05.008.
- Biermann, D., Weinert, K., and Wagner, T. Model-based optimization revisited: Towards real-world processes. *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*, pp. 2975–2982, Jun 2008. doi: 10.1109/CEC.2008.4631199.
- Bo Tang, Z. Adaptive partitioned random search to global optimization. *IEEE Transactions on Automatic Control*, **39**(11):2235–2244, 1994. doi: 10.1109/9.333768.
- Boender, C. and Romeijn, H. Stochastic Methods. In Horst, R. and Pardalos, P. M., eds., *Handbook of Global Optimization*, vol. 2 of *Nonconvex Optimization and Its Applications*, chap. 15, pp. 829–869. Springer US, Boston, MA, 1995. doi: 10.1007/978-1-4615-2025-2.
- Boor, C. and Ron, A. On multivariate polynomial interpolation. *Constructive Approximation*, **6**(3):287–302, Sep 1990. doi: 10.1007/BF01890412.
- Branke, J., Kaußler, T., Schmidt, C., *et al.* A multi-population approach to Dynamic Optimization Problems. In *Adaptive Computing in Design and Manufacturing*, pp. 299–307. Springer, 2000. doi: 10.1007/978-1-4471-0519-0\\_24.
- Branke, J. Memory enhanced evolutionary algorithms for changing optimization problems. In *Proceedings of the 1999 Congress on Evolutionary Computation-*

*CEC99 (Cat. No. 99TH8406)*, vol. 49, pp. 1875–1882. IEEE, 1999. doi: 10.1109/CEC.1999.785502.

Burl, M. C. and Wang, E. Active learning for directed exploration of complex systems. *Proceedings of the 26th Annual International Conference on Machine Learning - ICML '09*, pp. 1–8, 2009. doi: 10.1145/1553374.1553386.

Cobb, H. G. An Investigation into the Use of Hypermutation as an Adaptive Operator in Genetic Algorithms Having Continuous, Time-Dependent Nonstationary Environments. Technical Report AIC-90-001, Naval Research Laboratory, Washington, USA, 1990.

Cristianini, N. and Shawe-Taylor, J. *An introduction to support Vector Machines: and other kernel-based learning methods*. Cambridge University Press, New York, NY, USA, 2000.

Deisenroth, M. P., Rasmussen, C. E., and Peters, J. Gaussian process dynamic programming. *Neurocomputing*, **72**(7-9):1508–1524, Mar 2009. doi: 10.1016/j.neucom.2008.12.019.

Drucker, H., Burges, C., Kaufman, L., *et al.* Support vector regression machines. *Advances in neural information processing systems*, pp. 155–161, 1997.

du Plessis, M. C. and Engelbrecht, A. P. Using Competitive Population Evaluation in a differential evolution algorithm for dynamic environments. *European Journal of Operational Research*, **218**(1):7–20, Apr 2012. doi: 10.1016/j.ejor.2011.08.031.

Dunnnett, C. W. A multiple comparison procedure for comparing several treatments with a control. *Journal of the American Statistical Association*, **50**(272):1096–1121, 1955. doi: 10.1080/01621459.1955.10501294.

Dyn, N., Levin, D., and Rippa, S. Numerical Procedures for Surface Fitting of Scattered Data by Radial Functions. *SIAM Journal on Scientific and Statistical Computing*, **7**(2):639–659, Apr 1986. doi: 10.1137/0907043.

Eberhart, R. Adaptive particle swarm optimization: detection and response to dynamic systems. In *Proceedings of the 2002 Congress on Evolutionary Compu-*

- tation. CEC'02 (Cat. No.02TH8600)*, vol. 2, pp. 1666–1670. IEEE, 2002. doi: 10.1109/CEC.2002.1004492.
- Fang, H. and Horstemeyer, M. F. Global response approximation with radial basis functions. *Engineering Optimization*, **38**(4):407–424, Jun 2006. doi: 10.1080/03052150500422294.
- Fay, M. P., Proschan, M. A., and Others. Wilcoxon-Mann-Whitney or t-test? On assumptions for hypothesis tests and multiple interpretations of decision rules. *Statistics surveys*, **4**:1–39, Jan 2010. doi: 10.1214/09-SS051.
- Finck, S., Hansen, N., Ros, R., *et al.* Real-parameter black-box optimization benchmarking 2010: Presentation of the noisy functions. pp. 1–102, 2010.
- Floudas, C. A. and Gounaris, C. E. A review of recent advances in global optimization. *Journal of Global Optimization*, **45**(1):3–38, Aug 2008. doi: 10.1007/s10898-008-9332-8.
- Forrester, A., Sobester, A., and Keane, A. *Engineering Design via Surrogate Modelling: A Practical Guide*. Wiley, 1 ed., Sep 2008.
- Frazier, P., Powell, W., and Dayanik, S. The Knowledge-Gradient Policy for Correlated Normal Beliefs. *INFORMS Journal on Computing*, **21**(4):599–613, 2009. doi: 10.1287/ijoc.1080.0314.
- Frazier, P. I., Powell, W. B., and Dayanik, S. A Knowledge-Gradient Policy for Sequential Information Collection. *SIAM Journal on Control and Optimization*, **47**(5):2410–2439, Jan 2008. doi: 10.1137/070693424.
- Friedman, J. J. H. Multivariate Adaptive Regression Splines. *The Annals of Statistics*, **19**(1):1–67, 1991. doi: 10.2307/2241837.
- Ginsbourger, D., Riche, R. L., and Carraro, L. A Multi-points Criterion for Deterministic Parallel Global Optimization based on Gaussian Processes. In *Non-Convex Programming*. Rouen, France, 2007.
- Ginsbourger, D., Riche, R. L., and Carraro, L. Kriging Is Well-Suited to Parallelize Optimization. In Tenne, Y. and Goh, C., eds., *Computational Intelligence in Ex-*

*pensive Optimization Problems*, chap. 6, pp. 131–162. Springer Berlin Heidelberg, 2010. doi: 10.1007/978-3-642-10701-6\\_6.

Grefenstette, J. J. Genetic algorithms for changing environments. In Maenner, R. and Manderick, B., eds., *Parallel Problem Solving from Nature 2*, pp. 137–144. North Holland, 1992.

Hadad, B. S. and Eick, C. F. Supporting polyploidy in genetic algorithms using dominance vectors. In *Evolutionary Programming VI*, pp. 223–234. Springer, 1997.

Hansen, N. and Ostermeier, A. Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. In *Proceedings of IEEE International Conference on Evolutionary Computation*, pp. 312–317. IEEE, May 1996. doi: 10.1109/ICEC.1996.542381.

Hansen, N., Auger, A., Finck, S., *et al.* Real-parameter black-box optimization benchmarking 2010: Experimental setup. pp. 1–20, 2010.

Hughes, E. Multi-objective binary search optimisation. In *Evolutionary Multi-Criterion Optimization*, vol. 2632 of *Lecture Notes in Computer Science*, pp. 102–117. Springer Berlin Heidelberg, 2003. doi: 10.1007/3-540-36970-8\_8.

Husslage, B. G. M., Rennen, G., van Dam, E. R., *et al.* Space-filling Latin hypercube designs for computer experiments. *Optimization and Engineering*, **12**(4):611–630, Nov 2010. doi: 10.1007/s11081-010-9129-8.

Hutter, F., Hoos, H., and Leyton-Brown, K. An evaluation of sequential model-based optimization for expensive blackbox functions. In *Proceeding of the fifteenth annual conference companion on Genetic and evolutionary computation conference - GECCO '13 Companion*, pp. 1209–1216. ACM Press, 2013. doi: 10.1145/2464576.2501592.

Jastrebski, G. and Arnold, D. Improving Evolution Strategies through Active Covariance Matrix Adaptation. In *2006 IEEE International Conference on Evolutionary Computation*, pp. 2814–2821. IEEE, 2006. doi: 10.1109/CEC.2006.1688662.

- Jin, Y. and Branke, J. Evolutionary Optimization in Uncertain Environments—A Survey. *IEEE Transactions on Evolutionary Computation*, **9**(3):303–317, Jun 2005. doi: 10.1109/TEVC.2005.846356.
- Jones, D. R., Schonlau, M., and Welch, W. J. Efficient Global Optimization of Expensive Black-Box Functions. pp. 455–492, 1998. doi: 10.1023/a:1008306431147.
- Jones, D. A taxonomy of global optimization methods based on response surfaces. *Journal of global optimization*, pp. 345–383, 2001. doi: 10.1023/A:1012771025575.
- Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. Optimization by Simulated Annealing. *Science*, **220**(4598):671–680, May 1983. doi: 10.1126/science.220.4598.671.
- Kleijnen, J. P. C., Beers, W., and Nieuwenhuysse, I. Expected improvement in efficient global optimization through bootstrapped kriging. *Journal of Global Optimization*, **54**(1):59–73, Jun 2011. doi: 10.1007/s10898-011-9741-y.
- Kocijan, J. Dynamic GP models: an overview and recent developments. In *Proceedings of the 6th international conference on Applied Mathematics, Simulation, Modelling*, ASM’12, pp. 38–43. Jun 2012.
- Kushner, H. J. A versatile stochastic model of a function of unknown and time varying form. *Journal of Mathematical Analysis and Applications*, **5**(1):150–167, Aug 1962. doi: 10.1016/0022-247X(62)90011-2.
- Lim, D., Jin, Y., Member, S., *et al.* Generalizing Surrogate-Assisted Evolutionary Computation. *Evolutionary Computation, IEEE Transactions on*, **14**(3):329–355, 2010.
- Locatelli, M. Bayesian Algorithms for One-Dimensional Global Optimization. *Journal of Global Optimization*, pp. 57–76, 1997. doi: 10.1023/A:1008294716304.
- MacKay, D. J. C. *Information Theory, Inference and Learning Algorithms*. Cambridge University Press, 1st ed., Jun 2003.
- McKay, M. D., Beckman, R. J., and Conover, W. J. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, **21**(2):239–245, May 1979. doi: 10.2307/1268522.

- Mockus, J. Application of Bayesian approach to numerical methods of global and stochastic optimization. *Journal of Global Optimization*, **4**(4):347–365, Jun 1994. doi: 10.1007/BF01099263.
- Morales-Enciso, S. and Branke, J. Response Surfaces with Discounted Information for Global Optima Tracking in Dynamic Environments. In *Nature Inspired Cooperative Strategies for Optimization (NICSO 2013)*, vol. 512 of *Studies in Computational Intelligence*, pp. 57–69. Springer International Publishing, 2014. doi: 10.1007/978-3-319-01692-4.
- Mori, N., Kita, H., and Nishikawa, Y. Adaptation to a Changing Environment by Means of the Thermodynamical Genetic Algorithm. In Voigt, H.-M., ed., *International Conference on Parallel Problem Solving from Nature, PPSN*, no. 1141 in *Lecture Notes in Computer Science*, pp. 513–522. Springer Verlag Berlin, 1996. doi: 10.1007/3-540-61723-X\\_1015.
- Musizza, B., Petelin, D., and Kocijan, J. Accelerated learning of Gaussian process models. In *7th Congress on Modelling and Simulation, EUROSIM 2010*. Prague, 2010.
- Nelder, J. A. and Mead, R. A Simplex Method for Function Minimization. *The Computer Journal*, **7**(4):308–313, Jan 1965. doi: 10.1093/comjnl/7.4.308.
- Nguyen, T. T., Yang, S., and Branke, J. Evolutionary dynamic optimization: A survey of the state of the art. *Swarm and Evolutionary Computation*, **6**:1–24, Oct 2012. doi: 10.1016/j.swevo.2012.05.001.
- Papadrakakis, M., Lagaros, N. D., and Tsompanakis, Y. Structural optimization using evolution strategies and neural networks. *Computer Methods in Applied Mechanics and Engineering*, **156**(1-4):309–333, Apr 1998. doi: 10.1016/S0045-7825(97)00215-6.
- Pukelsheim, F. *Optimal Design of Experiments*. Classics in Applied Mathematics. Society for Industrial and Applied Mathematics, 2006.
- Quiñonero Candela, J. and Rasmussen, C. E. A Unifying View of Sparse Ap-

proximate Gaussian Process Regression. *Journal of Machine Learning Research*, **6**:1939–1959, 2005.

Raj, D. *Sampling theory*. McGraw-Hill series in probability and statistics. McGraw-Hill, 1968.

Ranjan, P., Bingham, D., and Michailidis, G. Sequential Experiment Design for Contour Estimation From Complex Computer Codes. *Technometrics*, **50**(4):527–541, Nov 2008. doi: 10.1198/004017008000000541.

Rasmussen, C. E. and Williams, C. *Gaussian Processes for Machine Learning*. Adaptive Computation and Machine Learning. MIT Press, Cambridge, MA, USA, 2006.

Razavi, S., Tolson, B. A., and Burn, D. H. Review of surrogate modeling in water resources. *Water Resources Research*, **48**(7), Jul 2012. doi: 10.1029/2011WR011527.

Regis, R. G. and Shoemaker, C. a. Parallel radial basis function methods for the global optimization of expensive functions. *European Journal of Operational Research*, **182**(2):514–535, Oct 2007. doi: 10.1016/j.ejor.2006.08.040.

Richter, H. Detecting change in dynamic fitness landscapes. In *2009 IEEE Congress on Evolutionary Computation*, pp. 1613–1620. IEEE, May 2009. doi: 10.1109/CEC.2009.4983135.

Rinnooy Kan, A. H. G. and Timmer, G. T. Stochastic global optimization methods part I: Clustering methods. *Mathematical Programming*, **39**(1):27–56, Sep 1987. doi: 10.1007/BF02592070.

Schonlau, M. *Computer experiments and global optimization*. 1998.

Scott, W., Frazier, P., and Powell, W. The Correlated Knowledge Gradient for Simulation Optimization of Continuous Parameters using Gaussian Process Regression. *SIAM Journal on Optimization*, **21**(3):996–1026, Jul 2011. doi: 10.1137/100801275.

- Shan, S. and Wang, G. G. Survey of modeling and optimization strategies to solve high-dimensional design problems with computationally-expensive black-box functions. *Structural and Multidisciplinary Optimization*, **41**(2):219–241, Aug 2009. doi: 10.1007/s00158-009-0420-2.
- Shi, L. and Ólafsson, S. Nested Partitions Method for Global Optimization. *Operations Research*, **48**(3):390–407, Jun 2000. doi: 10.1287/opre.48.3.390.12436.
- Smith, K. On the Standard Deviations of Adjusted and Interpolated Values of an Observed Polynomial Function and its Constants and the Guidance they give Towards a Proper Choice of the Distribution of Observations. *Biometrika*, **12**(1/2):1, Nov 1918. doi: 10.2307/2331929.
- Sobester, A., Leary, S., and Keane, A. A parallel updating scheme for approximating and optimizing high fidelity computer simulations. *Structural and Multidisciplinary Optimization*, **27**(5):371–383, Jun 2004. doi: 10.1007/s00158-004-0397-9.
- Solis, F. and Wets, R. Minimization by random search techniques. *Mathematics of operations research*, **6**(1):19–30, 1981.
- Sörensen, K. Metaheuristics - the metaphor exposed. *International Transactions in Operational Research*, (September):1–20, 2013. doi: 10.1111/itor.12001.
- Spears, W., de Jong, K., and Bäck, T. An overview of evolutionary computation. *Machine Learning: ECML-93*, 1993. doi: 10.1007/3-540-56602-3\\_163.
- van Dam, E. R., Husslage, B., den Hertog, D., *et al.* Maximin Latin Hypercube Designs in Two Dimensions. *Operations Research*, **55**(1):158–169, Feb 2007. doi: 10.1287/opre.1060.0317.
- van Dam, E. R., Husslage, B., and Hertog, D. One-dimensional nested maximin designs. *Journal of Global Optimization*, **46**(2):287–306, May 2009. doi: 10.1007/s10898-009-9426-y.
- Viana, F. A. C., Haftka, R. T., and Watson, L. T. Efficient global optimization algorithm assisted by multiple surrogate techniques. *Journal of Global Optimization*, **56**(2):669–689, Mar 2012. doi: 10.1007/s10898-012-9892-5.

- Villemonteix, J., Vazquez, E., Sidorkiewicz, M., *et al.* Global optimization of expensive-to-evaluate functions: an empirical comparison of two sampling criteria. *Journal of Global Optimization*, **43**(2-3):373–389, Jun 2008. doi: 10.1007/s10898-008-9313-y.
- Villemonteix, J., Vazquez, E., and Walter, E. An informational approach to the global optimization of expensive-to-evaluate functions. *Journal of Global Optimization*, **44**(4):509–534, Sep 2008. doi: 10.1007/s10898-008-9354-2.
- Wang, G. G. and Shan, S. Review of Metamodeling Techniques in Support of Engineering Design Optimization. *Journal of Mechanical Design*, **129**(4):370, 2007. doi: 10.1115/1.2429697.
- Wilcoxon, F. Individual Comparisons by Ranking Methods. *Biometrics Bulletin*, **1**(6):80–83, Dec 1945. doi: 10.2307/3001968.
- Yang, S. Genetic algorithms with memory- and elitism-based immigrants in dynamic environments. *Evolutionary Computation*, **16**(3):385–416, 2008.
- Yang, S. and Li, C. A Clustering Particle Swarm Optimizer for Locating and Tracking Multiple Optima in Dynamic Environments. *IEEE Transactions on Evolutionary Computation*, **14**(6):959–974, Dec 2010. doi: 10.1109/TEVC.2010.2046667.
- Zhou, Z., Ong, Y. S., Nair, P. B., *et al.* Combining global and local surrogate models to accelerate evolutionary optimization. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, **37**(1):66–76, 2007.
- Zimmermann, R. and Han, Z. Simplified cross-correlation estimation for multi-fidelity surrogate cokriging models. *Advances and Applications in Mathematical Sciences*, 2010.