Constrained mixed-variable blackbox optimization with applications in the automotive industry

Marie-Ange Dahito

INSTITUT POLYTECHNIQUE DE PARIS

NNT : 2022IPPAS017

Thèse de doctorat

Constrained mixed-variable blackbox optimization with applications in the automotive industry

Thèse de doctorat de l'Institut Polytechnique de Paris
préparée à Télécom SudParis

École doctorale n°626 École doctorale de l'Institut Polytechnique de Paris (EDIPP)
Spécialité de doctorat : Mathématiques et informatique

Thèse présentée et soutenue à Palaiseau, le 27 octobre 2022, par

MARIE-ANGE DAHITO

Composition du Jury :

Pierre-Alain Boucard Professeur des Universités, ENS Paris-Saclay (LMPS)	Président
Alain Faye Maître de conférences, ENSIIE (Cédric)	Rapporteur
Sébastien Le Digabel Professeur, Polytechnique Montréal (GERAD)	Rapporteur
Francesco Rinaldi Professeur, University of Padova (Department of Mathematics)	Examinateur
José Neto Maître de conférences, Télécom SudParis, IP Paris (SAMOVAR)	Directeur de thèse
Laurent Genest Leader Optimisation Numérique, Stellantis	Co-encadrant de thèse
Alessandro Maddaloni Maître de conférences, Télécom SudParis, IP Paris (SAMOVAR)	Invité

Acknowledgements

As this dissertation means the closure of the PhD work, I would like to thank many people who made this possible. This long journey, during which I truly learned a lot about myself, was full of scientific but also something I would call "human" challenges, including facing the overturning engendered by a pandemic.

First of all, my gratitude goes to my supervisors Laurent Genest, Alessandro Maddaloni and José Neto for the precious time they dedicated to me and for all the feedback and advice on my research. In particular, thank you Laurent for your constant follow-up and for tolerating me over these years and special thanks to Alessandro and José for being such considerate supervisors.

I am also sincerely thankful to the PhD thesis committee, Pierre-Alain Boucard, Alain Faye, Sébastien Le Digabel and Francesco Rinaldi for the time allocated to review and evaluate my work. I wish you a pleasant and interesting reading.

I am grateful to Rémy Bompoint, Alexandre Bonet, David Gaudrie, the formerly named COVP team and my other colleagues from Stellantis for their welcome, kindness and answers to my numerous questions. I learned a lot working in these big open spaces and the industrial context added a real-world aspect to my research experience. Moreover, I do not forget the PhD team and the interns, in particular Eléonore, Sandy and Augustin for all the useful and nice talks.

Furthermore, I obviously have to thank my fellows of the CMAP, Alann Cheral, Paul Dufossé, Eugénie Marescaux, Cheikh Touré and Konstantinos Varelas for their support and communicative cheerfulness. I really appreciated our after-work parties.

Finally, I express my deep gratitude to my close relatives, my friends Sonia, Anne, Glwadys, Gwénola, Julie, Tiphaine, Mathieu, Pierre-Yves and many others who will recognize for their encouragement and unwavering support. Thank you also Laura, Jason, Quentin and my badminton partners (and opponents). All the energy released during the sessions really helped me decompressing, as well as the snacks afterwards, of course.

Abstract

Numerous industrial optimization problems are concerned with complex systems and have no explicit analytical formulation, that is they are blackbox optimization problems. They may be mixed, namely involve different types of variables (continuous and discrete), and comprise many constraints that must be satisfied. In addition, the objective and constraint blackbox functions may be computationally expensive to evaluate.

In this thesis, we investigate solution methods for such challenging problems, i.e constrained mixed-variable blackbox optimization problems involving computationally expensive functions.

As the use of derivatives is impractical, problems of this form are commonly tackled using derivative-free approaches such as evolutionary algorithms, direct search and surrogate-based methods.

We investigate the performance of such deterministic and stochastic methods in the context of blackbox optimization, including a finite element test case designed for our research purposes. In particular, the performance of the ORTHOMADS instantiation of the direct search MADS algorithm is analyzed on continuous and mixed-integer optimization problems from the literature.

We also propose a new blackbox optimization algorithm, called BOA, based on surrogate approximations. It proceeds in two phases, the first of which focuses on finding a feasible solution, while the second one iteratively improves the objective value of the best feasible solution found. Experiments on instances stemming from the literature and applications from the automotive industry are reported. They namely include results of our algorithm considering different types of surrogates and comparisons with ORTHOMADS.

Résumé

Bon nombre de problèmes d'optimisation rencontrés dans l'industrie font appel à des systèmes complexes et n'ont pas de formulation analytique explicite : ce sont des problèmes d'optimisation de type boîte noire (ou *blackbox* en anglais). Ils peuvent être dits "mixtes", auquel cas ils impliquent des variables de différentes natures (continues et discrètes), et avoir de nombreuses contraintes à satisfaire. De plus, les évaluations de l'objectif et des contraintes peuvent être numériquement coûteuses.

Dans cette thèse, nous étudions des méthodes de résolution de tels problèmes complexes, à savoir des problèmes d'optimisation boîte noire avec contraintes et variables mixtes, pour lesquels les évaluations des fonctions sont très coûteuses en temps de calcul.

Puisque l'utilisation de dérivées n'est pas envisageable, ce type de problèmes est généralement abordé par des approches sans dérivées comme les algorithmes évolutionnaires, les méthodes de recherche directe et les approches basées sur des métamodèles.

Nous étudions les performances de telles méthodes déterministes et stochastiques dans le cadre de l'optimisation boîte noire, y compris sur un cas test en éléments finis que nous avons conçu. En particulier, nous évaluons les performances de la variante ORTHOMADS de l'algorithme de recherche directe MADS sur des problèmes d'optimisation continus et à variables mixtes issus de la littérature.

Nous proposons également une nouvelle méthode d'optimisation boîte noire, nommée BOA, basée sur des approximations par métamodèles. Elle comporte deux phases dont la première vise à trouver un point réalisable tandis que la seconde améliore itérativement la valeur de l'objectif de la meilleure solution réalisable trouvée. Nous décrivons des expériences utilisant des instances de la littérature et des applications de l'industrie automobile. Elles incluent des tests de notre algorithme avec différents types de métamodèles, ainsi que des comparaisons avec ORTHOMADS.

Résumé long

L'optimisation des structures est un incontournable dans de nombreuses industries, en particulier dans le secteur automobile. Celui-ci est très concurrentiel et connaît un durcissement des normes en matière de réduction des émissions de CO_2, ce qui se traduit notamment par un objectif d'allègement des véhicules. La réduction des coûts de production fait également partie des objectifs majeurs, surtout avec le développement de nouvelles technologies embarquées et l'électrification des véhicules.

L'atteinte de ces objectifs nécessite l'introduction de nouveaux leviers d'optimisation et la possibilité de traiter des variables de différentes natures. En effet, un problème-type d'optimisation paramétrique rencontré dans l'industrie automobile comprend des paramètres de forme continus, des épaisseurs de tôles discrètes et des matériaux qui sont des variables de catégorie. De tels problèmes sont alors dits "mixtes".

En outre, les quantités intervenant dans ces problèmes d'optimisation proviennent de simulations numériques par éléments finis qui peuvent être particulièrement coûteuses en termes de temps de calcul et de mémoire requise. Leur formulation analytique n'étant pas connue, ces problèmes sont de type boîte noire (ou *blackbox* en anglais) et l'utilisation de dérivées n'est pas envisageable. D'autres part, la tenue des prestations du véhicule doit être prise en compte, ce qui s'exprime par la présence de nombreuses contraintes à satisfaire dans le problème d'optimisation.

En l'absence de dérivées, des approches comme les algorithmes évolutionnaires et les méthodes de recherche directe sont classiquement utilisées. Cependant, les algorithmes d'optimisation sans dérivées pouvant résoudre le type de problèmes considérés sont rares et ils requièrent souvent de nombreuses évaluations des fonctions. L'utilisation de métamodèles permet souvent une réduction du coût numérique de l'optimisation.

Dans cette thèse, nous étudions des méthodes de résolution de tels problèmes complexes, à savoir des problèmes d'optimisation boîte noire avec contraintes et variables mixtes, pour lesquels les évaluations des fonctions sont très coûteuses en temps de calcul. Tout d'abord, nous étudions les performances de méthodes déterministes et stochastiques dans le cadre de l'optimisation boîte noire.

En effet, des algorithmes d'optimisation de la librairie SciPy en Python sont testés sur des problèmes continus de la littérature et disponibles à travers la plateforme COCO, dédiée à la comparaison d'algorithmes en optimisation boîte noire. En particulier, l'efficacité de l'algorithme SLSQP, qui utilise des approximations quadratiques et linéaires, a été mise en lumière.

En outre, les performances de la variante ORTHOMADS de l'algorithme MADS sont

évaluées sur des problèmes d'optimisation continus et à variables mixtes issus de la littérature. Il s'agit d'un algorithme de recherche directe couramment utilisé en optimisation boîte noire et dont les itérations s'effectuent sur un treillis. Il comporte deux phases dont la première, optionnelle, effectue une recherche globale tandis que la seconde correspond à une recherche locale autour de l'itéré courant. Le type de direction `ORTHO N+1 NEG` s'est révélé performant sur les deux familles de problèmes utilisées tandis que les directions `ORTHO 2N` et `ORTHO N+1 QUAD` se sont montrées efficaces en particulier sur les problèmes continus et à variables mixtes, respectivement. Comparativement à d'autres méthodes, les résultats ont montré un avantage de l'algorithme ORTHOMADS pour les problèmes d'optimisation continus de petite dimension et les problèmes continus multimodaux. Sur les problèmes à variables mixtes, ORTHOMADS figure parmi les meilleurs algorithmes et est particulièrement compétitif pour des budgets d'évaluations limités.

Les études de comparaisons incluent également la conception et l'implémentation d'un cas test en éléments finis, représentant une structure métallique soumise à une force. Trois problèmes d'optimisation boîte noire à variables mixtes ont été résolus, impliquant une contrainte de déplacement sur un nœud du maillage. Les expériences ont été menées avec ORTHOMADS et deux algorithmes évolutionnaires : CMA-ES et NSGA-II. Les résultats montrent une efficacité de l'algorithme ORTHOMADS, notamment dans la minimisation du coût et de la souplesse de la structure considérée. Concernant les autres méthodes testées, CMA-ES est souvent moins coûteux en termes de nombres d'appels à la boîte noire que NSGA-II et concurrence ORTHOMADS pour de grands budgets d'évaluations.

Enfin, nous proposons une nouvelle méthode d'optimisation boîte noire basée sur des approximations par métamodèles et appelée BOA (pour *Blackbox Optimization Algorithm*). Elle comporte deux phases dont la première vise à trouver un point réalisable tandis que la seconde améliore itérativement la valeur de l'objectif de la meilleure solution réalisable trouvée. Une extension de l'algorithme lorsque la parallélisation des évaluations est possible est également présentée. Nous décrivons des expériences utilisant des instances de la littérature et des applications de l'industrie automobile, toutes possédant des contraintes d'inégalité. Des tests utilisant quatre types de modèles de krigeage, des modèles RBF et des modèles MARS dans BOA sont décrits et ces deux derniers types de métamodèles ont généralement obtenu les meilleurs résultats. Les modèles MARS étaient cependant numériquement plus longs à construire. En comparaison avec deux variantes de l'algorithme ORTHOMADS assistées par des RBF et du krigeage, BOA avec des modèles RBF s'est montré globalement plus performant pour des budgets restreints.

Contents

List of Abbreviations

ACO	Ant Colony Optimization
AL-CMA-ES	Augmented Lagrangian-CMA-ES
ANN	Artificial Neural Network
aRT	average RunTime
BBO	BlackBox Optimization
BFGS	Broyden-Fletcher-Goldfarb-Shanno method
BIW	Body In White
CAD	Computer Aided Design
CMA-ES	Covariance Matrix Adaptation - Evolution Strategy
COBRA	Constrained Optimization By RAdial basis function interpolation
COBYLA	Constrained Optimization BY Linear Approximation
COCO	COmparing Continuous Optimizers
CONDOR	COnstrained Discrete Optimization using Response surfaces
ConstrLMSRBF	Constrained Local Metric Stochastic RBF
CS	Coordinate Search
DE	Differential Evolution
DFO	Derivative-Free Optimization
DIRECT	DIviding RECTangles
DOE	Design Of Experiments
EA	Evolutionary Algorithm
EB	Extreme Barrier
EBS	Extended-Box Selection
ECDF	Empirical Cumulative Distribution Functions
EGO	Efficient Global Optimization
EI	Expected Improvement
ES	Evolution Strategy
FE	Finite Element

GA	Genetic Algorithm
GIS	Global Intelligence Selection
GOSAC	Global Optimization with Surrogate Approximation of Constraints
GPS	Generalized Pattern Search
GSS	Generating Set Search
KKT	Karush-Kuhn-Tucker
L-BFGS	Limited-memory BFGS method
L-BFGS-B	L-BFGS method with handling of Box constraints
MADS	Mesh Adaptive Direct Search
MARS	Multivariate Adaptive Regression Spline
MISO	Mixed-Integer Surrogate Optimization
MLS	Moving Least-Squares
MSE	Mean Squared Error
MSS	Maxmin Stochastic Sampling
NEWUOA	NEW Unconstrained Optimization Algorithm
NM	Nelder-Mead method
NOMAD	Nonlinear Optimization with the MADS algorithm
NSGA	Nondominated Sorting Genetic Algorithm
ORBIT	Optimization by Radial Basis Interpolation in Trust regions
ORTHOMADS	Instantiation of MADS using orthogonal poll directions
PB	Progressive Barrier
PSO	Particle Swarm Optimization
RBF	Radial Basis Functions
RS	Random Search
RSM	Response Surface Model
SA	Simulated Annealing
SBO	Surrogate-Based Optimization
SE	Squared Error
SLSQP	Sequential Least-SQuares Programming
SO-I	Surrogate Optimization - Integer
SO-MI	Surrogate Optimization - Mixed-Integer
SVR	Support Vector Regression
TPE	Tree-structured Parzen Estimator
TARBF	Trust-region-based Adaptive RBF interpolation algorithm

List of Figures

List of Tables

List of Algorithms

1 Introduction

1.1 Context of the study

Numerical simulation has become a staple in many industrial fields and is used in place or together with physical experiments for diverse reasons including financial ones. Indeed, an experimentation can require the purchase of equipment, the availability of an area with the required conditions for the tests and human resources for the supervision. Besides, the equipment may be altered or even destroyed, in which case it needs to be replaced for each new trial run. Thus, the resources involved can represent substantial expenses.

Another argument to boost the use of numerical simulations is safety since the realization of a physical experiment often requires human intervention which can comprise a risky procedure. Hence, using simulations makes experiments safer for engineers.

Moreover, computer simulations have eased the design optimization of complex systems thanks to algorithms and the need to solve real-world optimization problems involving computer code is in constant growth since decades.

There are several types of simulations. In particular, the partial differential equations modelling physical phenomena are traditionally solved by *finite element* (FE) simulations. The FE approach discretizes a complex system into a finite number of subregions called elements, where ordinal differential equations are solved. This discretization in elements

constitutes a mesh.

When a computer code is used to compute the outputs of a function of an optimization problem, no explicit formulation is generally available. A function that can be called without knowledge of its formulation is called a *blackbox* and the resolution of such optimization problems lies in the frames of blackbox optimization (BBO) and derivative-free optimization (DFO). BBO refers to the optimization of blackbox functions in particular whereas DFO is a bit more general and focuses on optimization techniques that do not use any derivative information.

In industry, physical, industrial or regulation obligations often have to be satisfied and are expressed as constraint functions of the optimization problems. These constraints are also often blackbox functions, computed from computer simulations.

Finally, real-world optimization problems may involve different types of variables and the computing time can be significant. All these specificities require the use of optimization algorithms that can handle constraints and mixed types of variables while giving reasonable good solutions with limited evaluation budgets. The automotive industry is an illustrative field where such challenges are encountered.

1.2 Motivation

The motivation to solve mixed-variable optimization problems under constraints in the frame of this PhD arises from the problems encountered in the automotive group Stellantis. The automotive industry is a very competitive sector subject to different challenges.

First, the evolving and often more restrictive regulations force the optimizers to be agile and farsighted. In particular, global warming is a more and more concerning issue caused by the emissions of greenhouse gases in the atmosphere and, notably, carbon dioxide. In this respect, automotive vehicles are imposed maximal allowed CO_2 emissions that are decreasing with years. In order to reduce its environmental impact and meet the different regulations, Stellantis has committed to a reduction of its consumption, especially by decreasing the weights of its vehicles.

Furthermore, an increase of the costs is induced by the development of new embedded technologies, such as advanced driver assistance systems and power trains, but also the electrification of the vehicles or the need of greater mileage capacities. Besides, the negotiation of prices has become less practicable with the grouping of suppliers.

As a result, numerical optimization is commonly used at Stellantis as a decision making tool to enhance the performance of the vehicles, such as crashworthiness or acoustic comfort, while respecting regulations and maintaining reasonable production costs. The

optimization occurs at different phases of the vehicle design process and uses FE models of the body-in-white (BIW) structure, that is the skeleton of the vehicle. The numerical costs are an important aspect to take into account in the optimization as the FE simulations are time-consuming.

As an optimal balance between weight and cost reductions is sought, the introduction of new optimization levers seems necessary. Until recent years, the choice of the material of each part of the BIW was set before any optimization phase and according to historical reasons, experience or intuitions of the experts or physical experiments. Besides, alternative parts were not integrated in the optimization either. Adding materials as design variables and taking into account design alternatives in the optimization problem can lead to lighter configurations and lower costs.

Indeed, the mass needed to meet the demanded performance of a part with a certain material may be reduced by using another one. In addition, a different choice of material could make a reinforcement optional and, thus, reduce the total mass of the vehicle, and possibly the cost. An example is presented in Figure 1.1 showing the possible removal of the right reinforcement of the instrument panel support of the vehicle. Moreover, the inclusion of design alternatives makes the range of possible configurations wider.

(a) Right reinforcement (purple part) (b) Absence of the right reinforcement

Figure 1.1: Finite element model of the right reinforcement of the instrument panel support (source: Stellantis).

Furthermore, the allowed thicknesses of the materials take discrete values for manufacturing and financial reasons and are considered continuous in the optimization process. They are rounded to the closest admissible discrete values after each iteration of the solver used. Hence, considering an optimization algorithm that can directly handle discrete variables could accelerate the convergence.

Finally, among the algorithms often used for the optimization at Stellantis are evolutionary algorithms. However, the latter often need numerous function evaluations to give a good approximation of the solution. Since the last decade, surrogate models such as kriging

are also used in the automotive industry to predict computationally expensive outputs. As an example, kriging is used in [Binois, 2015] for Bayesian multi-objective optimization applied to an automotive crashworthiness problem. The construction of substitute functions can however be expensive when dealing with high dimensions so they have to be cleverly used. Thus, as the number of calls to the FE models has to be restricted, possessing a method designed for limited evaluation budgets would be of great interest.

1.3 Design optimization of a vehicle

The design optimization of a vehicle is done on the BIW structure from FE models. The general BBO problem can be written as follows:

$$\begin{aligned} &\min_{x \in \mathcal{X}} f(x) \\ &\text{s.t. } g_j(x) \leq 0, \forall j \in J \end{aligned} \tag{1.1}$$

where f is the objective function to minimize, typically the weight of the structure, $(g_j)_{j \in J}$ are the constraint functions representing the different services to conform to, with J a finite set of integer indices, and $\mathcal{X} \subset \mathbb{R}^n$ is a closed bounded subspace. We consider that f and $(g_j)_{j \in J}$ are expensive blackbox functions and all the constraints are assumed quantifiable. The formulation of Equation (1.1) is general and also takes into account maximization problems since it is equivalent to minimizing the opposite of the objective. Further formulations with explicit handling of discrete variables are considered in this manuscript.

Several types of optimization studies are performed on the BIW. The latter represents the core structure of the vehicle and mainly consists of sheet metals. An example of BIW is depicted in Figure 1.2.

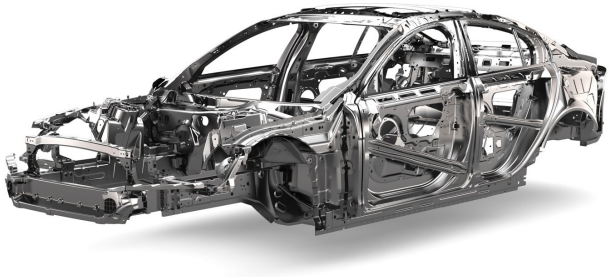


Figure 1.2: Body in white of an automotive vehicle (source: Car Bench).

The design of the BIW is essential since it has to support all the other parts of the vehicle and impacts most of its performances. Besides, it is the heaviest part of the vehicle, which makes it a good candidate for weight reduction. Numerical optimization methods were successfully applied to the BIW in [Genest, 2016] on a crashworthiness problem with gradients approximations of the conception parameters and in [Maliki, 2016] on reliability-based design problems from the automotive industry.

The types of optimization performed on a BIW can be classified into two families that are conceptual and sharp optimization. They have to take into account the required specifications on the vehicles.

1.3.1 Sharp optimization

Sharp optimization includes size and shape optimizations and is used for fine-tuning the geometry of the structure.

First, size optimization consists in minimizing the key morphing parameters of a computer aided design (CAD) like the width of a sheet metal, the angle between two parts or a bend radius for instance. It is often used in industry and presented in Figure 1.3 for the thickness optimization on a truss structure. At Stellantis, size optimization is done to find optimal thicknesses for each part of the BIW structure.

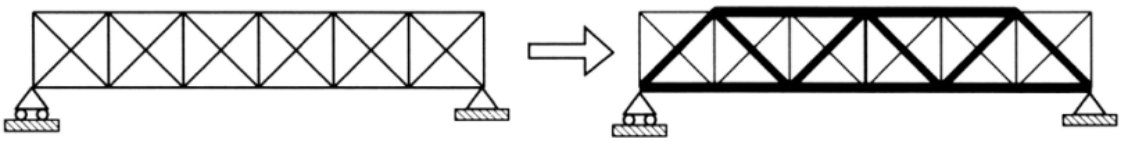
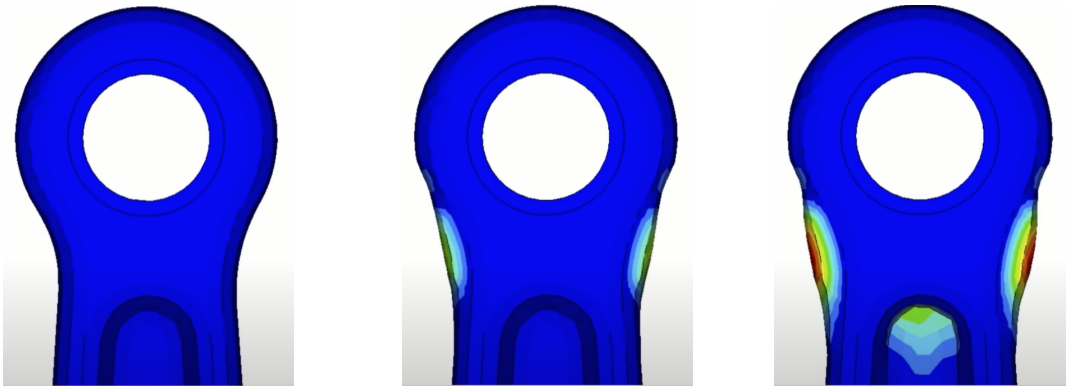


Figure 1.3: Size optimization (source: [Gebisa and Lemu, 2017]).

Shape optimization, depicted in Figure 1.4, is commonly used on structures that are subject to high concentrations of stresses. The latter are therefore evenly distributed during the optimization to prevent breakage. Shape optimization modifies the existing geometry by looking for the optimal displacement field for each node of the FE mesh. It optimizes the parameters characterizing the shape of structural parts such as heights or radii. Examples can be found in [Allaire et al., 2014] where a new approach for shape derivatives is introduced and shape optimization is applied to 2-dimensional cantilevers.

1.3.2 Conceptual optimization

Conceptual optimization gathers topology, free-size and free-shape optimizations.



(a) Initial shape (b) Intermediate shape (c) Final shape

Figure 1.4: Shape optimization (source: ALTAIR).

Topology optimization is a well-known technique used by engineers for the allocation of mass on a structure and is illustrated in Figure 1.5 on a two-dimensional cantilever. It enables to generate optimal forms that can be easily manufactured and allows alterations such as the creation of holes in the structure. For each part, it solves a constrained minimization problem in a fixed volume.

(a) Initial topology (b) Evolved topology

Figure 1.5: Topology optimization (source: Shape Optimization Group at CMAP).

Regarding free-size optimization, it is employed to continuously distribute thicknesses of every element of the FE mesh of the structure and enables to identify local reinforcements for example. It is depicted in Figure 1.6 where the colours indicate where additional mass is allocated.

Finally, free-shape optimization is used by design engineers to generate innovative shapes and stamping. Similarly as shape optimization, free-shape optimization is used to deal with structures subject to high concentrations of stresses but offers more flexibility than shape optimization as one can target specific areas for stress reduction and where new geometries are generated. Unlike topology optimization, free-shape optimization does

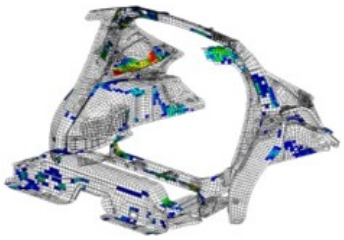


Figure 1.6: Free-size optimization (source: Stellantis).

not change the topology of the structure: holes can be modified but not created. An example of free-shape optimization result is presented in Figure 1.7 where the non-blue areas represent new generated stamping.

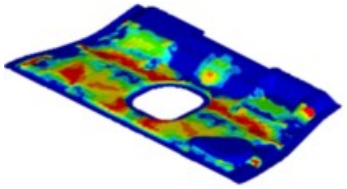


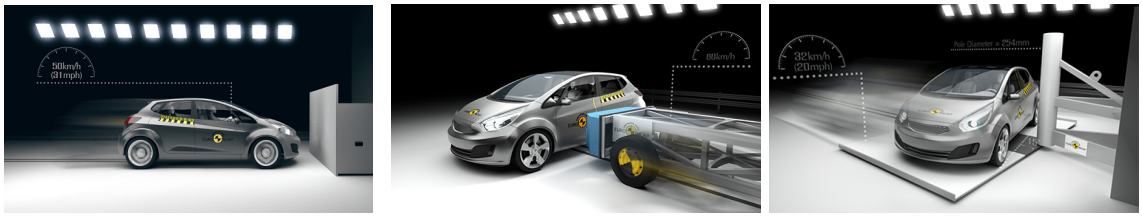
Figure 1.7: Free-shape optimization (source: Stellantis).

1.3.3 Required specifications

Before reaching the production phase, a vehicle has to respect numerous specifications. The latter first deal with the state regulations that can concern safety requirements for the passengers and the neighbourhood of the car, environmental issues and possible disturbances. Internal requirements from the company also have to be taken into account, treating for example the driving comfort of the customers.

The specifications to respect are also computed using the FE models and deal with three main performances: stiffness, crashworthiness and acoustic aspects. Indeed, there are essential mechanical constraints, reversal constraints that treat the case where the vehicle is reversed, static performance, vibro-acoustic comfort and constraints ensuring some resistance to different impact cases.

Different load cases are used to evaluate the resistance to an impact and they focus on different parts of the vehicle. The main types of crash studies concern side, lateral and frontal impacts. The latter are the most dangerous kind of impact for the passengers. The simulations consider crash cases with poles, barriers and pedestrians. A low-speed crash is also considered for the reparability of the vehicle. Some examples of crash tests are depicted in Figure 1.8.



(a) Frontal impact (b) Lateral barrier impact (c) Lateral pole impact

Figure 1.8: Crash tests on an automobile vehicle (source: Euro NCAP).

1.4 Key challenges

The optimization problems considered involve functions that are not explicitly known since, for instance, the physical constraints and the weight are computed from FE simulations. Hence, each constraint is treated as a blackbox and no derivative is available.

The optimization focuses on specific perimeters of the vehicle implying more than 20 variables related to about 10 parts of the vehicle. The problems are considered as medium- to large-dimensional BBO problems.

Moreover, querying a model is expensive, taking about 20 minutes for a stiffness computation and approximately 12 hours for a crash calculation. Numerous iterations are thus not conceivable in an industrial frame where the solution is desired in a limited time.

Furthermore, typical size optimization problems met at Stellantis are highly constrained as they can involve more than a hundred of constraints.

In order to add new optimization levers in the size optimization, different types of variables have to be taken into account. Indeed, size optimization can involve continuous shape parameters such as beam length or bend radius, the thicknesses of the parts take discrete admissible values, the consideration of optional parts involves binary parameters and integrating the choice of materials as optimization variables adds categorical inputs as there is no ordering between materials. Yet, mixed-variable solvers are rare and often computationally greedy.

This PhD aims at designing a new constrained blackbox optimization algorithm capable of handling mixed types of variables and providing a considered good solution within a restricted number of blackbox evaluations.

1.5 Organization of the thesis

In order to meet the research objectives of this PhD and considering the complexity of the problem, the strategy was to proceed gradually. Thus, solvers were first considered in an

unconstrained blackbox optimization context. To do so, optimization problems stemming from the literature were used to benchmark methods in continuous and mixed-integer contexts. Moreover, given the important times of the FE calculations, a small FE model was designed with the idea of possessing a finite element blackbox enabling quick comparisons of algorithms. Eventually, the main contribution of this PhD is the design of a surrogate-based method to solve constrained mixed-variable blackbox optimization problems within restricted evaluation budgets. The thesis manuscript is organized as follows.

Chapter 2 presents a review of the main DFO strategies for solving continuous and mixed-variable blackbox optimization problems. In particular, classical derivative-free approaches are presented and surrogate-based optimization is introduced. Moreover, methods for dealing with mixed variables are mentioned.

In Chapter 3, methods from the open-source Python library SciPy are benchmarked on continuous unconstrained optimization problems from the literature (the latter are naturally considered as blackbox functions).

Then, continuous and mixed-integer problems from the literature are used in Chapter 4 to evaluate variants of ORTHOMADS, which is a direct search algorithm. The considered best variants are compared to heuristic and non-heuristic methods.

The FE model developed is presented in Chapter 5 and used to compare three DFO methods on different constrained optimization problems.

Chapter 6 presents a new derivative-free surrogate-based algorithm called BOA, for Blackbox Optimization Algorithm, designed for solving expensive blackbox optimization problems with general inequality constraints and mixed variables. The method is used with different types of surrogate models on problems stemming from the literature and applications from the automotive industry, and is compared with two surrogate-assisted variants of ORTHOMADS.

Finally, Chapter 7 outlines the core contributions of this thesis and summarizes the principle results. Tracks for future works are also presented.

Notation

Let S be a real subspace, S_+ denotes its nonnegative values, S^* means that 0 is removed and S_+^* stands for the strictly positive values of S. Besides, $\mathcal{N}(m, C)$ denotes a multivariate normal distribution with mean vector m and covariance matrix C and $\mathbb{E}$ stands for an expected value. Let a be a real value, $\lfloor a \rfloor$ (respectively $\lceil a \rceil$) designates the highest (respectively smallest) integer b such that $b \leq a$ (respectively $b \geq a$). The cardinality of a set I is indicated as $|I|$. Furthermore, we use $\|\cdot\|$ for the Euclidean norm. Finally, $\widehat{f}$ is used to denote a surrogate model of a real-valued function f.

2

Literature review

2.1 Overview

In order to solve a continuous optimization problem, gradient-based and gradient-free methods exist. However, in a BBO problem such as Equation (1.1), the derivability is unknown as the objective and constraints involved are blackbox functions: they have no analytical formulations and can only be queried from inputs to return output values. Given the expensive computational costs of the functions we consider in this thesis, computing finite differences approximations is not conceivable either. Thus, the resolution of Equation (1.1) goes through derivative-free methods.

The increasing need to solve real-world optimization problems and problems with nonexistent or practically unusable derivatives have engendered a fast development of these methods.

There are several families of DFO techniques and in this chapter we will classify them in two subgroups that are classical DFO methods and model-based methods. The first family gathers direct search methods and other approaches such as simplex-based methods and heuristics. They originally use only true evaluations of the functions in the optimization process, which is specifically limiting in case of numerically costly functions.

Differently, model-based methods solve the optimization problem with surrogate approximations and perform parsimonious evaluations of the expensive functions. Yet, the construction of a model can be challenging with the risk of overfitting noisy functions or the surrogate could be too approximate to capture the trends of a function if it is built from few evaluations. Attention has also to be payed on the cost related to building the models, in particular when the dimension of the problem increases.

However, these two types of DFO techniques are less and less separate since surrogates are increasingly used also in classical DFO methods to assist the optimization, making them surrogate-assisted methods.

In this chapter, we will first describe optimization methods that are not necessarily designed for mixed variables or constraints. The specific handling of such features is discussed in Section 2.5.

2.2 A few words on gradient-based optimization

Before presenting DFO methods, we propose a quick reminder of optimization based on derivatives. For a differentiable function f, the gradient at $x = [x_1, \dots, x_n]^\top$, where $n \in \mathbb{N}^*$ is the dimension, is defined as the vector of partial derivatives with respect to each component:

$$\nabla f(x) := \begin{bmatrix} \frac{\partial f}{\partial x_1}(x) \\ \frac{\partial f}{\partial x_2}(x) \\ \dots \\ \frac{\partial f}{\partial x_n}(x) \end{bmatrix}. \tag{2.1}$$

As the gradient gives information about the variations of the function, it is fundamental in an optimization context and it should always be used if it is easily available. Indeed, for x_* a local minimizer of f, the first order optimality condition writes: $\nabla f(x_*) = 0$. Solving this equation gives all critical points of f including all local and global optima. If the gradient is itself differentiable, the Hessian of f at x is defined as:

$$\nabla^2 f(x) := \begin{bmatrix} \frac{\partial^2 f}{\partial x_1^2}(x) & \frac{\partial^2 f}{\partial x_1 \partial x_2}(x) & \dots & \frac{\partial^2 f}{\partial x_1 \partial x_n}(x) \\ \frac{\partial^2 f}{\partial x_2 \partial x_1}(x) & \frac{\partial^2 f}{\partial x_2^2}(x) & \dots & \frac{\partial^2 f}{\partial x_2 \partial x_n}(x) \\ & & \ddots & \\ \frac{\partial^2 f}{\partial x_n \partial x_1}(x) & \frac{\partial^2 f}{\partial x_n \partial x_2}(x) & & \frac{\partial^2 f}{\partial x_n^2}(x) \end{bmatrix}. \tag{2.2}$$

The second order necessary and sufficient optimality conditions add that, for x_* satisfying the first order necessary optimality condition, if $\nabla^2 f(x_*)$ is positive definite then $f(x_*)$ is

a local minimum, and that if x_* is a local minimizer then $\nabla^2 f(x_*)$ is positive semi-definite.

These optimality conditions are useful to investigate and recognize a local optimum, sometimes global when the function is known to be convex for instance. Newton-type methods, that are iterative linesearch methods, are commonly used in this context and give access to rates of convergence. Many publications deal with this kind of methods and, in particular, a good reference is the book of [Ortega and Rheinboldt, 2000].

When the Hessian is unavailable or computationally expensive, derivations called quasi-Newton methods use second-order approximations. Among the most known is the Broyden-Fletcher-Goldfarb-Shanno (BFGS) algorithm [Nocedal and Wright, 2006] that uses gradient information to iteratively update an approximation of the inverse Hessian that does not need any matrix inversion. In BFGS, the search direction p_k at iteration k is computed as:

$$p_k = -B_k \nabla f(x_k), \tag{2.3}$$

where x_k is the k^{th} iterate and B_k approximates the inverse Hessian at x_k with a recursive formula.

The Limited-memory BFGS (L-BFGS), described in [Nocedal, 1980], is based on BFGS recursion for the approximation of the inverse Hessian but uses only a limited number of the past updates. It is particularly suited for high-dimensional problems.

Adaptations of gradient-based methods to DFO exist. In [Powell, 1964], a method for finding the minimizer of a continuous optimization problem without use of derivatives is introduced. Powell's method may be considered as a derivative-free version of the conjugate gradient method [Hestenes and Stiefel, 1952]. Indeed, at each iteration k, it performs $N \in \mathbb{N}^*$ linesearch steps along conjugate directions $\mathcal{D} = \{d_1, d_2, \dots, d_N\}$ and the new iterate is computed as $x_k = x_{k-1} + \sum_{i=1}^{N} \alpha_i d_i$ with $\{\alpha_i, i = 1, \dots, N\}$ a real subset. The direction of best decrease is then removed from the set $\mathcal{D}$ while the direction corresponding to the linear combination $\sum_{i=1}^{N} \alpha_i d_i$ is added as new search direction. The procedure is repeated until a stopping criterion, such as the decrease of the objective function, is satisfied.

2.3 Classical derivative-free methods

Derivative-free optimization dates back to the work of [Fermi and Metropolis, 1952] introducing the coordinate search algorithm and later developed with simplex-based methods. Classical derivative-free techniques gather several kinds of optimization methods, including simplex-based methods, direct search algorithms and population-based approaches.

It can be noted that here we distinguish direct search and simplex-based methods.

The latter can be considered as part of the first type but simplex-based methods have the particularity that they use simplices and we describe them in a separate section.

2.3.1 Simplex-based methods

As their name indicates, simplex-based methods use simplices in their optimization procedures. In a space of dimension $n \in \mathbb{N}^*$, a simplex is a convex polytope which has $(n+1)$ vertices. As examples, in dimension 1, it is simply a segment defined by two points and in two dimensions, it is a triangle. Simplex-based methods use transformations of simplices to replace one or several vertices of a simplex by points with better function values.

The simplex-based method of [Spendley et al., 1962] proceeds of reflections of the worst point of the simplex, that is the point with the greatest function value, with respect to the centroid of the remaining points, or shifts all but the best vertex in its direction. An extension of this method is the Nelder-Mead (NM) simplex method which is, undoubtedly, the most popular and used simplex-based method. It is a heuristic method introduced by [Nelder and Mead, 1965] for local unconstrained optimization.

The NM algorithm is iterative and starts with a non-degenerated simplex whose vertices are evaluated. The worst point with respect to the objective value is identified and geometric transformations are applied to attempt to replace it with a better point. Indeed, reflections, expansions and contractions of the simplex occur to identify a new one for the next iteration.

The reflection of the worst point is applied with respect to the centroid of the points of the simplex except the worst one. If the new point is better than the second worst point, the worst point is replaced by its reflection. If the reflection is better than all points of the simplex, an expansion point is computed and if it is better than the reflection, then it replaces the worst point. When the reflection point is worse than the second worst point, an outside contraction is processed if the reflection point is better than the worst point, otherwise an inside contraction occurs. In case these transformations do not come up with a better point, a contraction of the simplex preserving only the best point is applied. The procedure is described in Algorithm 1 for the minimization of a real-valued function f and illustrations of a NM procedure are available in Figure 2.1.

An adaptation of the NM method that adjusts the parameters of the algorithm according to the dimension of the problem is introduced in [Gao and Han, 2012]. It is convenient for treating high-dimensional problems.

The classical NM method does not always converge to a stationary point even for relatively simple functions like twice differentiable convex functions. The problem of [McKinnon, 1998] using a continuously differentiable function is one of the examples showing

Algorithm 1: The Nelder-Mead simplex method

Initialize a non-degenerated simplex $\{x_1, x_2, \ldots, x_{n+1}\}$
Initialize parameters $\gamma_r > 0$, $\gamma_e > 1$, $\gamma_{oc} \in (0,1)$, $\gamma_{ic} \in (-1,0)$ and $\gamma_s \in (0,1)$
while *a stopping condition is not satisfied* **do**
    Rename the vertices x_1, x_2, …, x_{n+1} of the simplex such that $f(x_1) \leq f(x_2) \leq \cdots \leq f(x_{n+1})$
    Compute the centroid of all vertices but the worst point: $x_c = \frac{1}{n}\sum_{k=1}^{n} x_k$
    Compute a reflection of the worst point: $x_r = x_c + \gamma_r(x_c - x_{n+1})$
    Evaluate $f(x_c)$
    if $f(x_1) \leq f(x_r) < f(x_n)$ `// Reflection`
    then
        $x_{n+1} \leftarrow x_r$
    else if $f(x_r) < f(x_1)$ `// Expansion`
    then
        Compute the expansion point: $x_e = x_c + \gamma_e(x_c - x_{n+1})$
        Evaluate $f(x_e)$
        if $f(x_e) < f(x_r)$ **then**
            $x_{n+1} \leftarrow x_e$
        else
            $x_{n+1} \leftarrow x_r$
    else if $f(x_n) \leq f(x_r) < f(x_{n+1})$ `// Outside contraction`
    then
        Compute outside contraction point: $x_{oc} = x_c + \gamma_{oc}(x_c - x_{n+1})$
        Evaluate $f(x_{oc})$
        if $f(x_{oc}) < f(x_r)$ **then**
            $x_{n+1} \leftarrow x_{oc}$
        else
            $x_{n+1} \leftarrow x_r$
    else if $f(x_{n+1}) \leq f(x_r)$ `// Inside contraction`
    then
        Compute inside contraction point: $x_{ic} = x_c + \gamma_{ic}(x_c - x_{n+1})$
        Evaluate $f(x_{ic})$
        if $f(x_{ic}) < f(x_{n+1})$ **then**
            $x_{n+1} \leftarrow x_{ic}$
    else
        `// Shrink`
        Evaluate f on $\mathcal{S} = \{x_1 + \gamma_s(x_k - x_1)\}_{k=2,\ldots,n+1}$
        Replace vertices x_2 to x_{n+1} by the points of $\mathcal{S}$

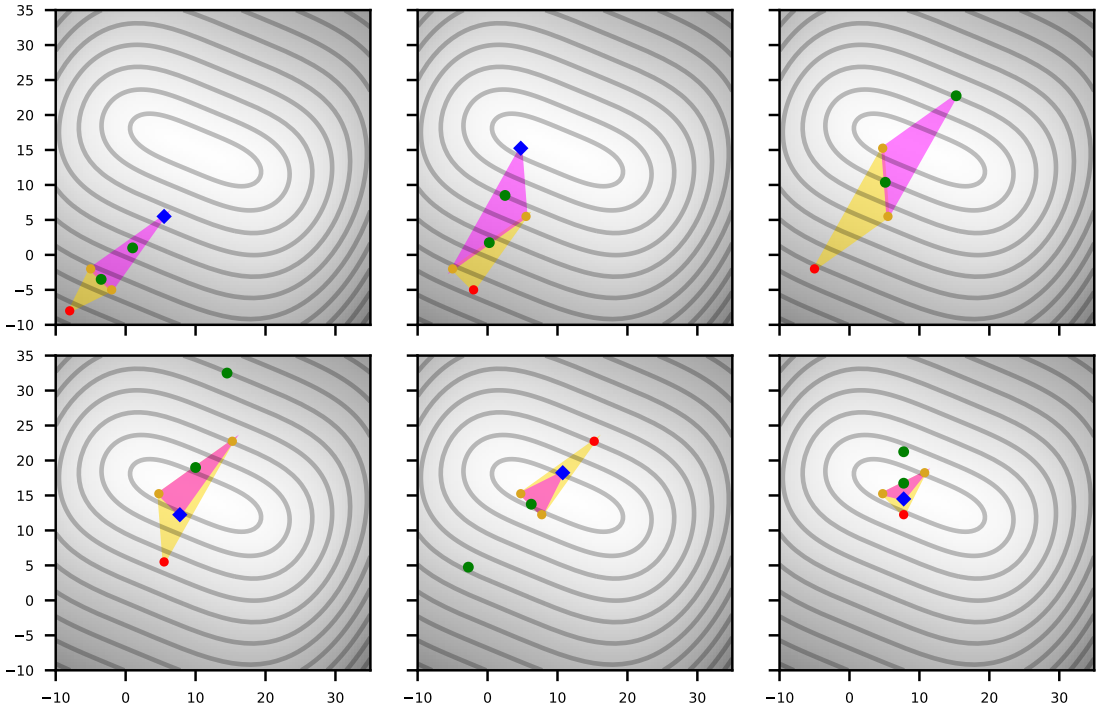


Figure 2.1: Iterations of the Nelder-Mead method where the current simplex is yellow and the chosen next one is magenta. The worst point to replace is represented by a red dot.

that the method can be stuck in a non-stationary point. However, modifications of the algorithm were proposed to deal with this issue.

Indeed, [Kelley, 1999] proposes a sufficient decrease criterion based on an approximation of the gradient, that is to be satisfied at every iteration. For smooth functions, if this condition is satisfied at each iteration, then the method is guaranteed to converge to a stationary point. For iterations that do not respect the criterion, a restart of the simplex is applied, creating a smaller one with orthogonal edges based on the sufficient decrease condition.

A simplex-based method also using reflections, contractions and expansions is introduced in [Tseng, 1999]. It differs from the NM method in several aspects as the acceptance of a new vertex in the simplex is based on some fortified-descend criteria or the fact that more than one vertices can be chosen for possible improvement. The global convergence of the method is proven for continuously differentiable functions under some assumptions.

2.3.2 Direct search methods

Direct search methods seem to have been introduced in the 1950's [Fermi and Metropolis, 1952] in the context of unconstrained optimization and later in the 1960's with [Hooke and Jeeves, 1961] from where the expression *direct search* originates. At the time, they were justified using geometric intuition more than mathematics, and convergence proofs were not available. In fact, the interest for global convergence proofs started with the Armijo-Goldstein-Wolfe conditions [Armijo, 1966, Wolfe, 1969, Goldstein, 2013] introduced later for the global convergence proof of the steepest descent method [Cauchy, 1847].

Referring to [Hooke and Jeeves, 1961], a direct search is a method that directs the optimization by sequentially evaluating points according to a certain strategy and proceeding to comparisons with a considered current best point. The comparisons use only function values and no derivative. More generally, [Wright, 1996] describes a direct search as a method using only function evaluations and that does not approximate derivatives.

Direct search methods are widely used in BBO, partly due to the simplicity of the main principle, the wide application possibilities and the fact that many direct search methods exhibit convergence proofs.

A first type of direct search techniques are pattern search methods that perform pattern moves on a lattice. They include the coordinate search (CS) [Fermi and Metropolis, 1952] on a grid defined by the coordinate directions. The algorithm starts with an initial guess that it tries to improve. To do so, if $n \in \mathbb{N}^*$ is the dimension of the problem, it iteratively evaluates the $2n$ neighbours of the incumbent on the grid along the coordinate directions and their opposites. If these evaluations come up with a better point in terms of objective function value, the incumbent is replaced by the best point. Otherwise, the grid size is reduced and the new neighbours on the grid are evaluated and compared to the incumbent. The procedure is repeated at each iteration until a stopping condition is met.

The generalized pattern search (GPS) method is introduced in [Torczon, 1997] and allows more variety in the selection of the search directions than CS. Indeed, GPS uses search directions that positively span $\mathbb{R}^n$ to evaluate new candidate points on a lattice and these directions are not the same at each iteration.

The algorithm first performs a global exploration, called the *search* phase, where points can be evaluated anywhere on the mesh. This step can use a heuristic method for instance or a model of the function. The evaluations try to replace the current incumbent by a better point. If the search does not manage to do so, a second phase corresponding to a local exploitation, and called the *poll*, is performed in which points are evaluated in the vicinity of the current best point and towards positive spanning directions, still on the mesh.

Let $\mathbb{D} = \{d_1, d_2, \ldots, d_{|\mathbb{D}|}\}$ denote a finite set of directions of $\mathbb{R}^n$. The positive span of $\mathbb{D}$ is defined as:

$$pspan(\mathbb{D}) = \left\{ \sum_{k=1}^{|\mathbb{D}|} \lambda_k d_k, \lambda_k \geq 0 \right\}. \tag{2.4}$$

By definition, $\mathbb{D}$ positively spans $pspan(\mathbb{D})$. In particular, if $pspan(\mathbb{D}) = \mathbb{R}^n$, $\mathbb{D}$ consists of positive spanning directions of $\mathbb{R}^n$. Using positive spanning directions offers the guaranty that at least one of them is a descent direction. If either the search or the poll finds a better point, the iteration is qualified as successful and the mesh size can be increased. Otherwise, the iteration is unsuccessful and the size of the mesh is reduced at the next iteration. The procedure is repeated until a stopping condition is met.

A generalization of GPS is the generating set search (GSS) [Kolda et al., 2003] that has a stronger condition to consider an evaluation as successful. Indeed, an iteration k is successful if a sufficient decrease condition $f(x_k) < f(x_{k-1}) - \rho(\delta_k)$ is satisfied, where ρ is a continuous decreasing forcing function and δ_k is the mesh size parameter.

Other methods use adaptive sets of search directions. It is the case of the well-known Mesh Adaptive Direct Search (MADS) [Abramson et al., 2009b, Audet and Dennis, Jr., 2006, Audet and Dennis, Jr., 2009]. The latter is a generalization of GPS but which introduces a frame parameter in the poll step of the algorithm and uses variable sets of search directions. MADS also proceeds to a global search and a local poll step when the search phase is unsuccessful, and evaluates candidate points on a mesh.

At iteration k, the mesh M_k is determined by the current iterate x_k, a mesh parameter size $\delta_k > 0$ and a matrix D whose columns consist of $p \in \mathbb{N}^*$ positive spanning directions such that:

$$M_k := \{x_k + \delta_k D y : y \in \mathbb{N}^p\}, \tag{2.5}$$

where the columns of D form a positive spanning set $\{D_1, D_2, \ldots, D_p\}$.

The candidate points evaluated in the poll belong to a frame F_k that has a radius of $\Delta_k > 0$ called the poll size parameter. The frame is defined as follows:

$$F_k := \{x \in M_k : \|x - x_k\|_\infty \leq \Delta_k b\}, \tag{2.6}$$

where $b = \max\{\|d\|_\infty, d \in \mathbb{D}\}$ and $\mathbb{D} \subset \{D_1, D_2, \ldots, D_p\}$ is a finite set of poll directions. The latter are updated at every iteration and are such that their union over iterations becomes asymptotically dense in $\mathbb{R}^n$.

In case of a successful iteration, both mesh and frame size parameters are increased, otherwise they are decreased. It is well noted that the mesh size is more severely decreased than the frame size and, as a result, the choice of a candidate point in the poll becomes

more flexible with unsuccessful iterations. Usually, $\delta_k = \min\{\Delta_k, \Delta_k^2\}$.

Under some assumptions, including bounded level sets of the objective function, MADS globally converges to a stationary point as proven in [Audet and Dennis, Jr., 2006] for constrained continuous optimization problems but also in the case of mixed variables, as shown in [Abramson et al., 2009a]. The convergence relies on the poll step of the algorithm.

The MADS procedure is described in Algorithm 2. This description of the algorithm is taken from [Dahito et al., 2021] and inspired from [Audet and Hare, 2017]. Instantiations of MADS are available in the Nonlinear Optimization with the MADS algorithm (NOMAD) open-source software [Audet et al., 2022b].

Algorithm 2: The mesh adaptive direct search (MADS)

Initialize $k = 0$, $x_0 \in \mathbb{R}^n$, $D \in \mathbb{R}^{n \times p}$, $\Delta_0 > 0$, $\tau \in (0,1) \cap \mathbb{Q}$, $\epsilon_{\text{stop}} > 0$
1. Update $\delta_k = \min\{\Delta_k, \Delta_k^2\}$
2. **Search**
  If $f(x) < f(x_k)$ for $x \in S_k$ then $x_{k+1} \leftarrow x$, $\Delta_{k+1} \leftarrow \tau^{-1}\Delta_k$ and go to 4
  Else go to 3
3. **Poll**
  Select $\mathbb{D}_{k,\Delta_k}$ such that $P_k := \{x_k + \delta_k d : d \in \mathbb{D}_{k,\Delta_k}\} \subset F_k$
  If $f(x) < f(x_k)$ for $x \in P_k$ then $x_{k+1} \leftarrow x$, $\Delta_{k+1} \leftarrow \tau^{-1}\Delta_k$ and go to 4
  Else $x_{k+1} \leftarrow x_k$ and $\Delta_{k+1} \leftarrow \tau\Delta_k$
4. **Termination**
  If $\Delta_{k+1} \geq \epsilon_{\text{stop}}$ then $k \leftarrow k+1$ and go to 1
  Else stop

DIRECT [Jones et al., 1993], standing for *dividing rectangles* and also referring to direct search, is a deterministic derivative-free technique used for solving expensive mixed-variable BBO problems and which exhibits global convergence properties. It is an extension of the one-dimensional Shubert's method. The search occurs in the hyperrectangle defined by the bounds of the optimization problem and scaled to be a unit hypercube. This hypercube is iteratively trisected in hyperrectangles, the centers of which are evaluated and potentially optimal hyperrectangles are chosen. It is well noticed that only the centre points are evaluated.

The modification of [Jones, 2001] performs the subdivision on a single long side instead of all long sides. In an unconstrained case, one or several hyperrectangles possessing the lowest function value, one of the biggest hyperrectangles and the lower-right part of a convex hull, is chosen for the trisection and evaluations of the centers of the new formed rectangles. The convex hull is calculated from a set of dots on the graph representing the objective values of the centers of the hyperrectangles according to the distance between

centers and vertices. The algorithm balances between local and global search approaches thanks to a weighting parameter on the center-vertex distance and acting on the choice of potentially optimal rectangles.

A pseudo-code of the procedure in the constrained case is described in Algorithm 3 and inspired from [Jones, 2001]. The distance between the centre of a hyperrectangle r and the vertices of the hyperrectangle is denoted d_r. Let f and $\{g_j, j = 1, 2, \ldots, m\}$ be the objective and constraint functions, respectively, with $m \in \mathbb{N}$ the number of inequality constraints. The rate of change is defined as

$$h_r(f^*) := \frac{\max(f_r - f^*, 0) + \sum_{j=1}^{m} c_j \max(g_{rj}, 0)}{d_r}, \tag{2.7}$$

where f_r and g_{rj} are respectively the objective value and the j^{th} constraint value in the centre of rectangle r, f^* is the global minimum of the function f and c_j is a real coefficient associated to the constraint $j \in \{1, 2, \ldots, m\}$. In practice, f^* is a certain real value satisfying $f^* \leq f_{\min} - \epsilon$ with $\epsilon > 0$ and $f_{\min}$ the minimum objective value computed so far in the algorithm. The parameters s_0 and $\{s_j, j = 1, 2, \ldots, m\}$ represent the sum of observed rates of change for respectively the objective and the m constraints, and are updated at each iteration. The number of splits along the direction i is denoted t_i with $i \in \{1, 2, \ldots, n\}$. Let x_{mid} stand for the centre of a parent hyperrectangle, x_{left} and x_{right} are respectively the centers of the left and right child hyperrectangles.

In the unconstrained case, the global convergence to a global optimum is ensured when the function is continuous in its neighbourhood. In [Finkel and Kelley, 2006], the algorithm is proven to converge to a Karush-Kuhn-Tucker (KKT) point. The convergence is also ensured, under some assumptions, for constrained optimization problems. For such problems, [Jones, 2001] considers an auxiliary function that gathers the objective and constraints (but is not a penalty function) and, if the current best point is infeasible, the hyperrectangles that minimize a rate of change are chosen for the next iteration.

2.3.3 Heuristic global search methods

In the context of a restricted number of function evaluations, heuristic techniques can be used to speed up the search of a solution. They generally provide good approximations in the resolution of an optimization problem. However, a heuristic may be specifically designed for a given problem, in which case it highly depends on it. Besides, this type of methods provide no convergence guarantee to an optimum. Meta-heuristic methods are a subclass of heuristics commonly used to tackle generic problems and that gathers several families of techniques, including evolutionary algorithms.

Algorithm 3: The dividing rectangles (DIRECT) algorithm

Initialize the scaled hypercube r and evaluate its centre x_c
Initialize $x_{\min} \leftarrow x_c$ and $f_{min} \leftarrow f(x_c)$
Initialize $s_j \leftarrow 0, j = 0, 1, \ldots, m$
$t_i \leftarrow 0, i = 1, 2, \ldots, n$
while *the maximum evaluation budget is not reached* **do**
    Compute $c_j \leftarrow \frac{s_0}{\max(s_j, 10^{-30})}, j = 1, 2, \ldots, m$
    if *no feasible point was found* **then**
        Select the hyperrectangle minimizing the rate of change $h_r(f^*)$ required to bring the weighted sum of the constraint violations to 0
    else
        identify the hyperrectangles contributing to the lower envelope of the functions $h_r(f^*)$
    $\mathcal{S}$ denotes the set of chosen hyperrectangles
    Choose a rectangle r from $\mathcal{S}$
    Identify the long side with the smallest t_i value (and smallest index if there are several possibilities)
    if *a left child exists* **then**
        Trisects r along dimension i
        Evaluate the centers of the new hyperrectangles formed
        $t_i \leftarrow t_i + 1$
    Update the best candidate $x_{\min}$ and its objective value $f_{\min}$
    if *the maximum evaluation budget is not reached* **then**
        Evaluate the midpoint of the left third
        Update $x_{\min}$ and $f_{\min}$
    if *the maximum evaluation budget is not reached* **then**
        Evaluate the midpoint of the right third
        Update $x_{\min}$ and $f_{\min}$
    $s_0 \leftarrow s_0 + \sum_{child=left}^{right} \frac{|f(x_{child}) - f(x_{mid})|}{\|x_{child} - x_{mid}\|}$
    $s_j \leftarrow s_j + \sum_{child=left}^{right} \frac{|g_j(x_{child}) - g_j(x_{mid})|}{\|x_{child} - x_{mid}\|}$
    If the problem has only integer variables, identify hyperrectangle children that are singletons and discard them.
    $\mathcal{S} \leftarrow \mathcal{S} \setminus \{r\}$

Simulated annealing

Simulated annealing (SA) is an iterative meta-heuristic inspired from metallurgy and that has been successfully used in BBO. Annealing is a technique that consists in sequentially heating and slowly cooling a material to improve its ductility.

[Metropolis et al., 1953] proposed the SA optimization method using a modified Monte Carlo algorithm to simulate the cooling process and the different thermodynamic equilibrium states of annealing. In [Kirkpatrick et al., 1983], SA was suggested for combinatorial problems before being adapted to problems with continuous variables in [Bélisle et al., 1993].

Let x_k be the incumbent at iteration $k \in \mathbb{N}$. From x_k, the Metropolis algorithm uses a Monte Carlo type algorithm to generate a new candidate point $\tilde{x}$, that can be seen as a new state of the material in the annealing process. The latter is evaluated with the objective function, representing the energy of the material, and accepted if the value is better than that of the incumbent. Otherwise, it is accepted with the following probability:

$$p(\tilde{x}|x_k) = \exp\left(-\frac{f(\tilde{x})-f(x_k)}{T_k}\right), \tag{2.8}$$

where T_k is a temperature parameter. The decreasing sequence $\{T_k\}_{k=0,1,\dots}$, that converges to 0, is called the cooling schedule and enables the algorithm to escape from local optima. The sequence commonly starts with a high temperature value to favour exploration. Indeed, this allows SA to accept worse points with a higher probability.

Evolutionary algorithms

Evolutionary algorithms (EAs) were developed in the 1950's and 1960's to solve engineering optimization problems. They are stochastic iterative methods that take their inspiration from nature and, in particular, from Darwin's evolution theories. In an EA, the components of a point are chromosomes, the points are called individuals and a set of individuals is referred to as a population. An offspring of $\lambda \in \mathbb{N}^*$ individuals is generated from a parent population with $\mu \in \mathbb{N}^*$ individuals.

The selection of the parents is said to be *elitist* if the best points from both parents and offspring are considered, in which case it is called a *plus selection* and referred to as $(\lambda + \mu)$-selection. If only the best points of the offspring are considered, it is a *comma selection* denoted (λ, μ)-selection and the algorithm is non-elitist.

EAs do not need any assumption about the function to optimize. However, many function evaluations are often needed, in particular when the dimension of the problem is high, and they provide no guarantee on the quality of the solution.

There are several types of EAs, among them are notably genetic algorithms (GAs) and evolution strategies (ESs). The main principle of an EA is to apply genetic operators to a parent population in order to create an offspring population that can be, in turn, selected at the next iteration to be among the new parents. Different types of selections occur according to the methods, they can be random or based on comparisons from a fitness function for instance. Among often used genetic operators are mutations and crossovers which we describe hereafter. The steps followed by an EA are summarized in Algorithm 4.

Algorithm 4: Main steps of an evolutionary algorithm

```
1 Initialize a first population 𝒫
2 Evaluate the individuals of 𝒫 with the objective and constraint functions
3 while a stopping condition is not satisfied do
4     From 𝒫, select a parent population 𝒫̃
5     Generate an offspring by applying genetic operators on the individuals of 𝒫̃
          // e.g. crossovers and mutations
6     Replace the individuals of 𝒫̃ contained in 𝒫 by their offspring
7     Evaluate the new individuals of 𝒫
```

Mutations are used in EAs to create diversity by perturbing some variables, which can avoid premature convergence to a local solution. There are varieties of mutation operators and here we give an overview with a few examples for binary-coded variables and real-coded ones.

On binary-coded variables, a *binary mutation* can be used. It creates a child by flipping the bits (from 0 to 1 or 1 to 0) of an individual with a certain probability defined for each variable. Similarly, a *mirror mutation* replaces the value of a variable by its mirror with respect to the middle of its feasible domain interval.

Some mutations swap variables like the *swap mutation* that swaps two chosen variables while *scramble mutation* permutes several of them. In *inversion mutation*, a subset of variables is mirrored: the first of the list takes the value of the last and so on.

For real values, *real mutation* randomly changes genes by other values in their boundary intervals.

Uniform mutation randomly chooses a variable and assigns it a value chosen uniformly as random in its bounds.

Non-uniform mutation is an adaptive operator that reduces the divergence from the original individuals as the generations increase. Thus, close to an optimum, the mutation has little change on an individual.

Boundary mutation randomly assigns a boundary value to randomly chosen variable.

Power mutation and *polynomial mutation* respectively use a power and polynomial distribution. With power mutation one can control the diversity of the population thanks to a parameter.

Finally, *Gaussian mutation* uses the Gaussian error function and either adds or removes a random normal-distributed quantity to each variable.

Crossovers are used as recombination operators. Similarly to mutations, there exist different types of crossover operators and many of them are based on binary encoding. As most known GAs use two parents to create two children, we mainly talk about this case but the generalization to more parents and more or less children is possible. [Umbarkar and Sheth, 2015] gives a summary of the existing crossover operators used in GAs and [Kora and Yadlapalli, 2017] gives the principal ones with a focus on the resolution of the famous travelling salesman problem.

Children can inherit equally, that is with same probability, from both parents: with *uniform crossover*, each variable has the same probability to be chosen for a child and a second child is the complementary of its fellow. A variant is half uniform crossover where only half of the differing variables can be swapped.

k−points crossover is another type. For single-point crossover, a point between two chromosomes is randomly selected and, from it, all the following variables are exchanged between parents. In the case of $k \in \mathbb{N}^*$ points, these ones are randomly selected and the parents are combined between these points. In other words, single-point crossover is successively applied at each crossover point.

Single-point crossover is applied in *shuffle crossover* after a random shuffle of some variables for both parents. After that, the shuffled features take their original values back.

In *reduced surrogate crossover*, a point is randomly selected among the locations of the variables where the parents differ. This operation is done only if there are at least two varying variables between the parents.

In *elitist crossover*, the selection and crossover phases are combined. Indeed, the whole population is randomly shuffled before any parent selection. Then, from a couple of parents, two vectors are computed from a crossover. The two best of the four are selected as offspring.

Finally, *simulated binary crossover* [Deb and Agrawal, 1995] was created for real-coded variables with the idea of preserving the properties of 1−point crossover. Indeed, it preserves the average values of the individuals before and after the operation. Let β be a spread factor following a certain distribution, $x^{(1)}$ and $x^{(2)}$ respectively the same variables

of two parents numbered 1 and 2 such that $x^{(2)} > x^{(1)}$, and $y^{(1)}$ and $y^{(2)}$ respectively the corresponding variables of the two children 1 and 2. The offspring is created from the parents as follows:

$$y^{(1)} = \frac{1}{2}(x^{(1)} + x^{(2)} - \beta(x^{(2)} - x^{(1)})) \quad \text{and} \quad y^{(2)} = \frac{1}{2}(x^{(1)} + x^{(2)} + \beta(x^{(2)} - x^{(1)})). \tag{2.9}$$

Genetic algorithms were originally introduced by [Bremermann, 1958] and later popularized by [Holland, 1975] and [Golberg, 1989]. GAs are a kind of EAs used in many contexts including signal processing and recurrent neural networks. They use crossover and mutation operators to generate offspring.

The nondominated sorting genetic algorithm (NSGA-II) [Deb et al., 2002] is among the most famous GAs and was initially designed for solving continuous multi-objective optimization problems. In NSGA-II, a dominance order is used to prioritize the global population in nondominated sub-populations (or fronts). Consider an unconstrained optimization problem with $\alpha \in \mathbb{N}^*$ objective functions f_1, f_2, ..., f_α. For two distinct points x and $\tilde{x}$, x *dominates* $\tilde{x}$ (denoted $x \prec \tilde{x}$) if and only if $f_i(x) \leq f_i(\tilde{x}), \forall i = 1, \ldots, \alpha$ and at least one of these inequalities is strict. If these inequalities hold but none of them is strict, that is x and $\tilde{x}$ have the same function values, they are equivalent The notation $\preceq$ includes the equivalence case.

Two rankings are performed in the algorithm. The first-order ranking is based on the *nondominated sorting* that ranks the individuals according to their dominance in the objective space. Indeed, the nondominated points are identified and given a non-domination rank of zero. They are then temporarily discounted from the population in order to identify a new set of nondominated points that are assigned a non-domination rank equal to that of the previous nondominated set plus 1. The points are discounted from the population and the process is repeated until each point of the population belongs to a relative nondominated front and is allocated a non-domination rank.

The second-order ranking is based on the *crowding distance* that is an indicator of the density of each point in each nondominated front. To compute the crowding distance, the individuals of a front are ranked for each objective ($f_k(x^{(1)}) \leq f_k(x^{(2)}) \leq \ldots \leq f_k(x^{(N)}), k = 1, \ldots, \alpha$) and the neighbours of each point in each objective space are identified. The points on the extremities ($x^{(1)}$ and $x^{(N)}$) are given infinite distance values. For each $x^{(i)}$ with $i \in \{2, \ldots, N-1\}$, the following distance value is computed:

$$\frac{f_k(x^{(i+1)}) - f_k(x^{(i-1)})}{f_k^{\max} - f_k^{\min}}, \tag{2.10}$$

where $f_k^{\max}$ and $f_k^{\min}$ are respectively the maximum and minimum of f_k. The crowding

distance of a point x in a front corresponds to the sum of its distance values for all objectives, and we denote it $c(x)$. The points on each side of a considered one form a cuboid around the sample point and the crowding distance corresponds to the mean value of the length of an edge of the cuboid. A crowding-comparison operator $\prec_n$ is used for the second-order ranking. A point x dominates another point $\tilde{x}$ with respect to the crowding-comparison operator, that is $x \prec_n \tilde{x}$, if and only if either $x \prec \tilde{x}$ or x and $\tilde{x}$ have the same domination rank and $c(x) > c(\tilde{x})$. Thus, in the same front, the less crowded points are preferred.

NSGA-II is a modification of the nondominated sorting genetic algorithm (NSGA) that performs two rankings. The first one is a ranking selection method based on the domination rank. The second one ranks the points of a same front using a niche method. This method is rarely used in practice because it is strongly dependent to a sharing parameter, that is used to know the proximity between two points, and has an expensive complexity. NSGA-II uses the crowded-comparison operator in place of the sharing function approach of NSGA.

For optimization problems with relaxable and quantifiable constraints, NSGA-II uses a binary tournament selection, or constraint dominance, that ranks the individuals according to their constraint violations and objective values. Three cases can be considered to define the constraint dominance between two points:

1. If only one of them is feasible, the latter dominates the infeasible one
2. If they are both infeasible, the solution with the lowest constraint violation is dominating
3. If both are feasible, the classical dominance applies.

Hence, the lower the constraint violation, the better the rank, and for equal constraint violations, the classical nondominated sorting is used. The procedure of the NSGA-II algorithm can be described as in Algorithm 5.

Initially binary-coded, the extension to continuous variables was introduced by [Bethke, 1980], using an approximate binary decomposition. Several modifications of the algorithm exist, such as ϵ-NSGA-II. The parameter $\epsilon > 0$ is a user-defined precision for each objective. A grid of size ϵ is generated in the objective space and the nondominated sorting is based on blocks of the grid. The evaluated points are stored in sorted archives, the firsts of which are ϵ-nondominated points.

Although NSGA-II is a multi-objective solver, it can be adapted for single-objective optimization problems. Indeed, for an unconstrained single-objective minimization problem, the dominance criterion is simply the natural ordering between the objective values. If the

Algorithm 5: The nondominated sorting genetic algorithm (NSGA-II)

Initialize a parent population size parameter $N \in \mathbb{N}^*$
Determine an initial population $\mathcal{P}_0$ of $M \geq N$ individuals
Evaluate the individuals of $\mathcal{P}$ with the α objective and β constraint functions
$\mathcal{P} \leftarrow \mathcal{P}_0$
while *a stopping condition is not satisfied* **do**
    Determine the relative constraint nondominated fronts $(\mathcal{F}_1, \mathcal{F}_2, \ldots)$ from $\mathcal{P}$
        // Fast nondominated sorting
    $\mathcal{S} \leftarrow \varnothing$ // Set of selected (parent) points
    $l \leftarrow 1$
    while $|\mathcal{S}| + |\mathcal{F}_l| \leq N$ **do**
        $\mathcal{S} \leftarrow \mathcal{S} \cup \mathcal{F}_l$
        $l \leftarrow l + 1$
    if $|\mathcal{S}| < N$ // Crowding distance
     then
        Compute the crowding distance of each individual of $\mathcal{F}_l$
        Sort $\mathcal{F}_l$ in descending order using the crowding-comparison operator $\prec_n$
         Add the best $(N - |\mathcal{S}|)$ points of the sorting to $\mathcal{S}$
    Apply crossovers and mutations to $\mathcal{S}$ to generate an offspring $\mathcal{Q}$
    Evaluate the individuals of $\mathcal{Q}$ with the objective and constraint functions
        $\mathcal{P} \leftarrow \mathcal{P} \cup \mathcal{Q}$ // Add the offspring to the global population

problem is constrained, constrained-dominance is still used and thus feasible solutions are preferred.

Other ways of handling constraints in tNSGA-II are also available. An example is the approach of [Ray et al., 2001] that performs three types of nondominated sorting. The first one uses the objective functions, the second one is based on the constraint violations and the third one is a combination of objectives and constraints. No penalization parameter is necessary as the criteria are compared separately.

Evolution strategies were initially developed in the 1960's for parameter optimization of complex systems. ESs represent a class of EAs that mimics the principles of organic evolution and use only mutations as genetic operators to generate offspring. They were developed by [Rechenberg, 1973] and [Schwefel, 1981], and use the comma and plus selections.

The $(1 + 1)$-ES is probably among the simplest ESs. At each iteration, the algorithm generates a single offspring by applying a mutation operator on a parent and the best

candidate between the parent and the offspring is kept for the next generation. [Rechenberg, 1973] introduced the one fifth success rule with the main idea of keeping the success probability of the iterations to roughly $\frac{1}{5}$. This rule was applied by [Kern et al., 2004] to the $(1+1)$-ES: if the ratio of the iterations that come up with an offspring better than the parent over all the iterations is greater than $\frac{1}{5}$, the variance of the sampling distribution is increased, otherwise it is decreased.

Among the biggest advances in EAs is the development of auto-adaptive techniques to update the mutation parameters. The idea of adaptive variance is employed in the Covariance Matrix Adaptation - Evolutionary Strategy (CMA-ES) [Hansen and Auger, 2014] that is a state-of-the-art ES which deterministically adapts the covariance matrix of the sampling distribution. This meta-heuristic minimizes a continuous function f.

Let $\mathcal{N}(m, C)$ denote a normal distribution of mean vector m and covariance matrix C. It can be represented by the ellipsoid $x^\top C^{-1} x = 1$, the main axes of which are the eigenvectors of C and the square roots of their lengths are the associated eigenvalues. CMA-ES iteratively samples its populations from multivariate normal distributions whose parameters are updated with rank-based selection, recombination and cumulation.

The main idea behind the updates of the covariance matrices is to approach the contour lines of the objective function making use of the information from previously evaluated points. This is similar to an approximation of the inverse Hessian matrix in a quasi-Newton method.

At the generation g, the $(\mu/\mu_w, \lambda)$-CMA-ES proceeds to an independent sampling of $\lambda \in \mathbb{N}^*$ individuals $x_1^{(g)}, \ldots, x_\lambda^{(g)}$. Each individual of the next generation is updated such that:

$$x_k^{(g+1)} \sim m^{(g)} + \sigma^{(g)} \mathcal{N}(0, C^{(g)}), \forall k = 1, \ldots, \lambda, \tag{2.11}$$

where $m^{(g)}$, $\sigma^{(g)}$ and $C^{(g)}$ are respectively the mean, the step size and the covariance matrix at generation g.

Let $\{x_{1:\lambda}^{(g)}, x_{2:\lambda}^{(g)}, \ldots, x_{\lambda:\lambda}^{(g)}\}$ be the points of the population at generation g, sorted such that $f(x_{1:\lambda}^{(g)}) \leq f(x_{2:\lambda}^{(g)}) \leq \ldots \leq f(x_{\lambda:\lambda}^{(g)})$, the mean of the next generation is updated using the $\mu \leq \lambda$ best points as follows:

$$m^{(g+1)} = \sum_{i=1}^{\mu} w_i x_{i:\lambda}^{(g)}, \tag{2.12}$$

where w_i, $i = 1, \ldots, \mu$ are positive recombination weights such that $w_1 \geq w_2 \geq \ldots, \geq w_\mu$. If these weights sum to one, this convex combination lies in the convex hull of the μ best points.

The step size is updated following what is called a cumulative step size adaptation:

$$\sigma^{(g)} = \sigma^{(g-1)} \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|p_\sigma^{(g)}\|}{\mathbb{E}_{x\sim\mathcal{N}(0,I)}\|x\|} - 1\right)\right), \tag{2.13}$$

where c_σ and d_σ are constant parameters and $p^{(g)}$ is the evolution path. The latter is defined as a sequence of successive steps over a number of generations. If the selection is biased, making the norm of the evolution path smaller or greater than $\mathbb{E}_{x\sim\mathcal{N}(0,I)}\|x\|$, the step size is decreased or increased, respectively.

The covariance matrix is updated as follows:

$$C^{(g+1)} = \left(1 - c_1 - c_\mu \sum_{i=1}^{\mu} w_i\right) C^{(g)} + c_1 p_\sigma^{(g)} p_\sigma^{(g)\top} + c_\mu \sum_{i=1}^{\mu} w_i y_{i:\lambda}^{(g)} y_{i:\lambda}^{(g)\top}, \tag{2.14}$$

where $0 < c_1 < 1$, $0 < c_\mu < 1 - c_1$ and $y_{i:\lambda}^{(g)} = \frac{x_{i:\lambda}^{(g)} - m^{(g)}}{\sigma^{(g)}}$, $i = 1, \ldots, \mu$. The term $p_\sigma^{(g)} p_\sigma^{(g)\top}$ is called the rank-one update and uses cumulative information from the past generations whereas the rank-μ update term $\sum_{i=1}^{\mu} w_i y_{i:\lambda}^{(g)} y_{i:\lambda}^{(g)\top}$ uses only information of the current population.

With these updates based on the best points from the population and using the information of previous generations, the probability to sample points in promising areas increases with the iterations.

Other widely used EAs include Differential Evolution (DE) [Storn and Price, 1997], a meta-heuristic algorithm designed for continuous optimization but that provides no guarantee of optimality. In DE, a trial point is generated by combining the current best point with other individuals chosen randomly in the population. The trial vector is then sequentially filled with parameters from itself or the incumbent and the best individual between the incumbent and the created vector is chosen.

The ant colony optimization (ACO) [Dorigo, 1992] was originally designed for combinatorial optimization problems, such as the travelling salesman problem. The method is stochastic and inspired from the behaviour of ants looking for food. When ants find food, they go back to the nest by leaving pheromones that will be detected by other ants. The exchange of information is local since the insects need to be close to the pheromones to access the information. As the shortest paths will be more used, the concentration of pheromones will be higher on these paths, thus also attracting more ants.

Particle swarm optimization (PSO) was introduced in [Kennedy and Eberhart, 1995] and [Eberhart and Kennedy, 1995]. It is a stochastic archive-based EA also commonly used that takes its inspiration from social learning. In PSO, an individual is called a particle and a population is a swarm. Each particle is given a location and a velocity and evolves

according to its local best point, that is the best candidate it visited, and to the best solution encountered by the algorithm so far.

This section described deterministic and stochastic methods that directly query the functions involved in the optimization problem. Other methods try to limit the number of function calls by using models that predict the interesting outcomes: they are model-based methods.

2.4 Model-based methods

Most global optimization approaches use numerous function evaluations, which is not practicable when dealing with numerically expensive functions. In order to save expensive evaluations or accelerate the convergence of a method, substitutes of the functions can be constructed. The interest for approximation techniques has notably increased in the past two decades.

A surrogate is a model of a system that shares similarities with it but is simpler and cheaper to evaluate. Other names used to refer to surrogates include surrogate models, metamodels, response surfaces, or response surface models. However, the designations using the expression *response surface* are sometimes specifically employed for local polynomial approximations.

In order to build a surrogate of a function, data has to be provided: a design of experiments (DOE) is first set and consists of $N \in \mathbb{N}^*$ sampled points $\{x_1, \ldots, x_N\}$ in the design space that are evaluated with the true function. There exist numerous DOE techniques and [Giunta et al., 2003] gives a summary of the principle ones.

The use of surrogates can be classified according to two main purposes. First, they can aim at reproducing the most accurately as possible a behaviour in order to be used in place of the corresponding real system: it is called emulation. The other major use is in the framework of optimization with the goal of identifying promising areas where the optima of the optimization problem lie.

In this section, we describe some well-known surrogate models including polynomial regression models, radial basis functions, kriging models and multivariate adaptive regression splines.

We focus on the use of metamodels for optimization but also present a few comparison studies in the context of emulation as they give an insight of the quality of the most known surrogate techniques.

2.4.1 Main types of surrogate models

There exist multiple metamodelling techniques and we present some of them that are commonly used. Radial basis functions, kriging models and multivariate adaptive regression splines will also be considered in our research work to be presented in Chapter 6.

Polynomial response surface models

Polynomial response surfaces have long been favoured for function approximations. They were first introduced in [Box and Wilson, 1992] and later in [Myers et al., 2016] and in the context of neural networks [Hajela and Berke, 1992]. A polynomial response surface model (RSM) approximates a function f with a polynomial of degree $m \in \mathbb{N}$ using least squares regression. Among the advantages is the possession of a C^∞ estimate.

Let n denote the dimension of the variable space and $x^{(1)}, \dots, x^{(N)}$ be points where the function values are known, that is the DOE. If $n = 1$, an m-degree polynomial approximation of f at a new point x can be written as:

$$p(x) = \sum_{k=0}^{m} a_k x^k, \tag{2.15}$$

where $a = (a_0, \dots, a_m)^\top$ solves

$$\underset{a \in \mathbb{R}^{m+1}}{\text{minimize}} \quad \tfrac{1}{2}\|\Phi a - y\|^2. \tag{2.16}$$

In Equation (2.16), $y = (f(x^{(1)}), \dots, f(x^{(N)}))^\top$ is the vector of outputs at $x^{(1)}, \dots, x^{(N)}$, and Φ is the Vandermonde matrix:

$$\Phi = \begin{bmatrix} 1 & x^{(1)} & (x^{(1)})^2 & \dots & (x^{(1)})^m \\ 1 & x^{(2)} & (x^{(2)})^2 & \dots & (x^{(2)})^m \\ \dots & \dots & \dots & \dots & \dots \\ 1 & x^{(N)} & (x^{(N)})^2 & \dots & (x^{(N)})^m \end{bmatrix}. \tag{2.17}$$

Polynomial RSMs are well suited for low-dimensional problems, low-modality problems or when the information is cheap to get. However, for high dimensions, it can be hard to get enough data to build high-order polynomials. A quadratic polynomial is often used as, in many cases where the function is computationally expensive, it is a good approximation that is relatively cheap to build. Indeed, $\frac{(n+1)(n+2)}{2}$ points are needed to construct a quadratic approximation.

For $n \in \mathbb{N} \setminus \{0, 1\}$, if $m = 2$, a quadratic response surface of f at $x = (x_1, \dots, x_n)^\top \in \mathbb{R}^n$

is:

$$p(x) = a_0 + \sum_{i=1}^{n} a_i x_i + \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} a_{ij} x_i x_j + \sum_{i=1}^{n} a_{ii} x_i^2, \tag{2.18}$$

where $a = (a_1, \ldots, a_n, a_{12}, a_{13}, \ldots, a_{(n-1)n}, a_{11}, a_{22}, \ldots, a_{nn})^\top$ solves Equation (2.16) with

$$\Phi = \begin{bmatrix} 1 & x_1^{(1)} & \ldots & x_n^{(1)} & x_1^{(1)}x_2^{(1)} & \ldots & x_{n-1}^{(1)}x_n^{(1)} & \left(x_1^{(1)}\right)^2 & \ldots & \left(x_n^{(1)}\right)^2 \\ 1 & x_1^{(2)} & \ldots & x_n^{(2)} & x_1^{(2)}x_2^{(2)} & \ldots & x_{n-1}^{(2)}x_n^{(2)} & \left(x_1^{(2)}\right)^2 & \ldots & \left(x_n^{(2)}\right)^2 \\ \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots & \ldots \\ 1 & x_1^{(N)} & \ldots & x_n^{(N)} & x_1^{(N)}x_2^{(N)} & \ldots & x_{n-1}^{(N)}x_n^{(N)} & \left(x_1^{(N)}\right)^2 & \ldots & \left(x_n^{(N)}\right)^2 \end{bmatrix}. \tag{2.19}$$

Radial basis functions

Radial basis functions (RBF) [Hardy, 1971] are interpolating models that use radially symmetric functions. The latter are such that they depend only on the distance between the input variable and some fixed centre point. The norm commonly used to build RBF models in an optimization framework is the Euclidean norm. Let $f : \mathbb{R}^n \to \mathbb{R}$ be an expensive real function, with $n \in \mathbb{N}^*$. Let $x \in \mathbb{R}^n$ be a point where the function has not been evaluated, an RBF approximation of $f(x)$ is built as a weighted sum of basis functions:

$$s(x) = \sum_{i=1}^{N} \omega_i \phi(\|x - c^{(i)}\|), \tag{2.20}$$

where $\phi(\|x - c^{(i)}\|)$ are the basis functions evaluations, $\{c^{(i)}, i = 1, \ldots, N\}$ are the $N \in \mathbb{N}^*$ basis function centres and $\{w_i, i = 1, \ldots, N\}$ are real weights that can be easily estimated by interpolation or least squares. Additional terms such as polynomials can be added to the RBF formulation to increase its flexibility. Examples of traditional fixed and parametric basis functions types are given below:

- *linear* $\phi : r \mapsto r$
- *cubic* $\phi : r \mapsto r^3$
- *thin plate spline* $\phi : r \mapsto r^2 ln(r)$
- *Gaussian* $\phi : r \mapsto e^{-\frac{r^2}{2\sigma^2}}$
- *multi-quadric* $\phi : r \mapsto (r^2 + \sigma^2)^{\frac{1}{2}}$
- *inverse multi-quadric* $\phi : r \mapsto (r^2 + \sigma^2)^{-\frac{1}{2}}$.

Kriging models

Originally developed by [Krige, 1951], kriging models are well-known interpolation surrogates that build approximations of functions by reducing the mean squared error. In the variant of [Sacks et al., 1989], that is commonly used, the approximations lie on Gaussian processes.

Let $f : \mathbb{R}^n \to \mathbb{R}$ be an expensive real function, with $n \in \mathbb{N}^*$, and let $x \in \mathbb{R}^n$ be a point where the function has not been evaluated. In kriging, the function value at x is considered as the realization of a normally distributed random variable:

$$Y(x) = \mu(x) + Z(x), \tag{2.21}$$

where $\mu(x)$ is the deterministic approximation of the mean and Z is a Gaussian process of mean 0.

The assumption made on the deterministic mean gives three types of kriging. Indeed, the kriging model is *simple* if the mean is a known constant value μ. If the mean is constant but unknown, the kriging model is *ordinary*, in which case the mean is often assigned the mean value of the approximations. Finally, the model is *universal* if the deterministic mean is a linear combination of basis regression functions depending on x.

The covariance of this Gaussian process Z at two points x and $\tilde{x}$ can be written as follows:

$$Cov(Z(x), Z(\tilde{x})) = \sigma^2 R(\|x - \tilde{x}\|),$$

where σ^2 is the variance and R is a correlation function.

There exist different families of correlation functions and below are some well-known examples:

- *Gaussian* $R : \tau \mapsto e^{-\frac{\tau^2}{2\theta^2}}$
- *exponential* $R : \tau \mapsto e^{-\frac{|\tau|}{\theta}}$
- *Matérn 3/2* $R : \tau \mapsto (1 + \frac{\sqrt{3}|\tau|}{\theta})e^{-\frac{\sqrt{3}|\tau|}{\theta}}$
- *Matérn 5/2* $R : \tau \mapsto (1 + \frac{\sqrt{5}|\tau|}{\theta} + \frac{\sqrt{5}\tau^2}{3\theta^2})e^{-\frac{\sqrt{5}|\tau|}{\theta}}$,

where θ is a parameter of the correlation models.

The maximization of the likelihood function of the observed data, and often the log of the likelihood function, enables to determine the deterministic mean, the variance and the parameter θ. As a global optimization problem has to be solved to identify the model parameters, kriging is rarely used for high dimensions.

In an optimization context, kriging can be used with a merit function such as the *expected improvement* (EI) [Močkus, 1975, Jones et al., 1998] and an enrichment strategy. Let $\{x^{(1)}, \dots, x^{(N)}\}$ be the $N \in \mathbb{N}^*$ points of the DOE where the function values are known. The *improvement function* is defined as follows:

$$I(x) := \max_{x \in \mathbb{R}^n} (Y_{\min} - Y(x), 0),$$

where $Y_{\min} = \min(f(x^{(i)}), i = 1, \dots, N)$. The EI is the expectation of the improvement function and its maximization gives expected promising points to evaluate as it reduces both the uncertainty of the model and the objective function. Once evaluated, these points can be used to enrich the kriging model until convergence to an optimal value.

Multivariate adaptive regression spline

Introduced in [Friedman, 1991], multivariate adaptive regression spline (MARS) is a surrogate model using piecewise linear regression models to capture the non-linearities of a function. The model builds regression models on disjoint subregions of the variable space. The points on the boundaries of the subregions are called knots. A MARS model approximation of a real function at a point $x \in \mathbb{R}^n$, where $n \in \mathbb{N}^*$ is the dimension of the variable space, can be written as follows:

$$s(x) = \alpha_0 + \sum_{m=1}^{M} \alpha_m \mathcal{B}_m(x), \tag{2.22}$$

where $\{\mathcal{B}_m\}_{m=1}^{M}$ are $M \in \mathbb{N}^*$ maximum linearly independent interaction basis functions, α_0 is the mean of the responses, also called the intercept coefficient, and α_m is the coefficient associated to the m^{th} basis function. The basis functions are univariate and defined as the product of at least two truncated linear functions, also called hinge functions. The latter are equal to linear functions on bounded subregions defined by knot locations, and equal to zero elsewhere.

Consider a hinge basis function $\mathcal{H}$ and call x_i its unidimensional variable, with $i \in \{1, \dots, n\}$, $\mathcal{H}$ can be formulated as follows:

$$\mathcal{H}(x_i) = \max(0, x_i - k) \quad \text{or} \quad \mathcal{H}(x_i) = \max(0, k - x_i), \tag{2.23}$$

where k is its respective knot location. The m^{th} interaction basis function is defined as a

product of hinge functions:

$$\mathcal{B}_m(x) = \prod_{l=1}^{L_m} \max(0, s_{ml}(x_{i(m,l)} - k_{ml})), \tag{2.24}$$

where $L_m \in \mathbb{N}^*$ is the number of truncated linear functions, $s_{ml} = \pm 1$, $x_{i(m,l)}$ is the i^{th} input variable corresponding to the l^{th} hinge function in the m^{th} basis function and k_{ml} is the knot corresponding to $x_{i(m,l)}$.

A numerically more stable variant, called BMARS and introduced in [Bakin et al., 2000], replaces the truncated linear functions by second-order B-splines.

A few more surrogate techniques

Other metamodels exist such as artificial neural network (ANN) [McCulloch and Pitts, 1988, Broomhead and Lowe, 1988] where nonlinear functions, referred to as neurons, are connected by a certain architecture. ANNs are commonly represented using a diagram of multiple layers with weighted connections between nodes. The input variables are represented by the nodes of the first layer while those of the last layer constitute the predicted outputs. Although ANNs can model complex systems, training them is computationally expensive.

Support vector regression (SVR) derives from the theory of support vector machines introduced in [Vapnik, 1995]. In the context of optimization, it can be considered as an extension of RBF as it has a similar formulation. The model uses a tolerance corresponding to an allowed deviation from a target value and the complexity of the model is reduced by solving a constrained convex quadratic optimization problem. SVR is used for low-dimensional problems due to the cost of training the model.

Polynomial chaos expansions are described in [Xiu and Karniadakis, 2002] and consider the outputs of a function as random. The function values are approximated using polynomials of multivariate random variables that are orthogonal to the distribution of the random variables.

Finally, a moving least-squares (MLS) model [Lancaster and Salkauskas, 1981, Levin, 1998] is a kind of weighted least-squares that is computationally more intensive than a basic least-squares regression or a weighted least-squares. It can be seen as a compromise between regression and interpolation.

2.4.2 Surrogates in the context of emulation

Multiple scientific papers focus on emulation and perform quality comparisons of different surrogate models, mostly on continuous problems.

In [Kianifar and Campean, 2020], RBF, kriging and polynomial RSM, declined in a total of 11 models, are used to emulate 18 literature functions of dimensions between 2 and 20. The accuracy, robustness and efficiency of each model is respectively measured with the normalized root mean squared error, its variance and the computational time needed to build the model. The results exhibit a globally higher accuracy, robustness and efficiency of the kriging models, except on low-order non-linearity functions when the size of the DOE is small. In the latter case, polynomials are performing better than the other surrogates compared.

Two linear models, two splines, a kriging model, an ANN, an SVR and random forests are compared in [Villa-Vialaneix et al., 2012] on a corn cultivation application study. Tests are performed according to increasing dataset sizes: eight DOE sizes from 100 to ≈ 15000. The qualities of the metamodels are evaluated for N_2O prediction and for N leaching prediction according to four measures that are the R^2 coefficient, the mean squared error (MSE) but also the standard deviation of the squared error (SE) and the maximum value of the SE. Splines and kriging based methods have the best results with small and medium training datasets while ANN, SVR and random forests demonstrate the best performance with large DOEs.

The efficiencies of kriging and SVR surrogates are also evaluated in [Moustapha et al., 2018] using isotropic and anisotropic kernels. The latter are isotropic if they assume a single parameter for all directions whereas they are said to be anisotropic if a parameter is defined for each dimension. L_1 and L_2 penalizations are also applied to SVR. Comparisons are performed with increasing training datasets sizes on three continuous nonlinear functions of dimensions 2, 5 and 20 and on FE structure forming and crashworthiness models of dimensions 21 and 5, respectively. The quality measures used are the normalized MSE, the normalized average absolute error and the normalized maximum absolute error. The results show improved qualities of the models by introducing anisotropy. The anisotropic L_2-SVR with the Matérn kernels appears to be the most effective surrogate.

[Forsberg and Nilsson, 2005] compares Kriging and polynomial models on two design problems of dimension 2 and a 5-dimensional crashworthiness problem. With respect to the root mean squared error, kriging shows better outcomes but the linear and quadratic models better filter noise.

RBF, polynomial, kriging and MARS models are investigated in [Jin et al., 2001] using the R^2 coefficient. They consider 13 nonlinear mathematical problems of dimensions

lower than 16 and a 14-dimensional vehicle handling problem. RBF appears as the most competitive on most of the problems.

2.4.3 Surrogate-based optimization

Since recent years, the use of surrogates in optimization is a hot topic in DFO. According to the metamodels used, classical optimization methods using derivatives can be applied to the models. When surrogates are plugged in an optimization algorithm to guide the optimization, we talk about a *surrogate-assisted* method. If the main structure of the algorithm is based on the use of surrogates, we refer to it as a *surrogate-based* algorithm. A classical optimization method using true evaluations can be employed inside a surrogate-based method to solve subproblems defined by the cheap metamodels. In that case, the true evaluations of the classical method correspond in fact to surrogate evaluations for the original optimization problem.

In surrogate-based optimization (SBO), metamodels are used to predict the expensive outcomes of the objective and/or constraint functions and, thus, guide the search of an optimum using parsimonious evaluations of the expensive functions. Surrogates can be *static* (or non-adaptive), that is they do not evolve during the optimization process. However, most of the time in SBO, they are updated when new sample points are evaluated with the expensive functions, in which case they can be qualified as *dynamic* (or adaptive). An insight on SBO is given for instance in [Han and Zhang, 2012] and [Booker et al., 1999].

Multiple papers like [Wang and Shan, 2006] and [Forrester and Keane, 2009] review the principal types of surrogates used for optimization. Some focus on specific applications such as groundwater modelling appears in [Asher et al., 2015].

We present below the generic process followed by a surrogate-based algorithm using dynamic metamodels:

1. `[Initialization]` Generate an initial DOE and evaluate the expensive functions at each of its points. Use these evaluations to determine initial surrogates approximating the objective and constraint functions.

2. `[Sample point(s) generation]` Determine a restricted set of points $\mathcal{S}$ on which to evaluate the blackbox functions with a selection procedure involving the current surrogates.

3. `[Sample point(s) evaluation(s)]` Evaluate each point of $\mathcal{S}$ with the expensive functions.

4. `[Surrogate update]` Update the surrogates using all the available evaluations, and return to Step 2 until some stopping criterion is satisfied.

These different steps can be implemented in various ways and examples are presented in [Vu et al., 2017] and [Müller, 2016].

In an optimization context, the capability of a model to identify promising areas and to drive the algorithm towards the optimum of an optimization problem should prevail over the global accuracy of the surrogate. In this sens, surrogates do not have to be true approximations of the functions.

2.4.4 Some model-based methods

We present some optimization methods that use metamodels and separate them in two families that are local model-based methods and global model-based methods.

Local model-based methods

Local surrogates try to approach the function as accurately as possible in the neighbourhood of some fixed point. The closer one gets to this point and the more accurate is the approximation. Local model-based methods include algorithms based on Taylor series, often quadratic approximations. The latter are for instance used by quasi-Newton linesearch methods that build a local quadratic model at each iteration such that the surrogate gradient and the true gradient match (see [Nocedal and Wright, 2006] for more information).

The sequential least-squares programming (SLSQP) method developed in [Kraft, 1988] solves constrained optimization problems by approximating the objective function with a quadratic model and the constraints with linear models. It internally makes use of a quasi-Newton method and approximates the Hessian of the objective using BFGS update formula.

Trust-region methods also use local models: they build a local quadratic model of a function to minimize and the model is supposed to be a good approximation in a defined area of radius $\Delta > 0$, called the trust region. They iteratively minimize the quadratic model under a maximum norm constraint of the solution equal to Δ. The aim is to find a new stepsize in a descent direction of the function to update the current iterate (see [Conn et al., 2000] for more details on trust-region methods).

The trust-region-based adaptive RBF interpolation algorithm (TARBF) developed in [Liu et al., 2021] is a trust-region method using RBF models for the resolution of real-world engineering optimization problems. TARBF divides the optimization problem into a series of trust-region subproblems defined by cubic RBF approximations with linear polynomial

tails. The subproblems are solved by the sequential quadratic programming (SQP) solver of the SciPy library [Jones et al., 2001] to find the iterates of the algorithm. An adaptive online normalization is applied to objective and constraint values to improve the accuracy of the model within the trust region. The size of the trust region is adjusted at every iteration according to four factors: the size of the current trust region, the location of the design variable in the trust region and global design space, the history of the movement of each design variable, and the iteration point vector. Thus, the trust region is resized according to current and past information to balance between exploration and exploitation. The DOE consists of three parts. First, points are randomly generated from a maxmin stochastic sampling (MSS) strategy inside the trust region and with a minimum distance constraint between the points. Second, an extended-box selection (EBS) strategy selects previous points located in a neighbourhood of 1.4 time the size of the trust region of the current starting point. Half of these points are alternatives to new points required in the current iteration. Finally, points close to the current starting point but located out of the extended box are selected by the global intelligence selection (GIS) strategy.

Another example of trust-region algorithm using RBF models is the optimization by radial basis interpolation in trust regions (ORBIT) algorithm [Wild et al., 2008]. Its extension to constrained optimization, called CONORBIT, is developed in [Regis and Wild, 2017]. CONORBIT selects points in the trust region of the current best point to build RBF models of the objective and constraints. The minimization of a local subproblem defined by the surrogates and including a margin on the RBF constraints gives the next sample point to evaluate with the original expensive functions.

Another example of local-based method is the constrained optimization by linear approximation (COBYLA) algorithm developed in [Powell, 1994] and that performs linear approximations of the objective and constraint functions. The method iteratively solves linear programming optimization problems and adapts a stepsize that is, in particular, decreased if the new candidate point is not better with respect to the original problem.

Global model-based methods

Global model-based methods are used for global optimization and use surrogates that do not necessary try to be accurate models. Used in an optimization context, a global surrogate aims at avoiding expensive evaluations by identifying promising regions of the search space where evaluations with the original function should be performed.

Among global RBF-based methods is the constrained optimization by radial basis function interpolation (COBRA) algorithm developed in [Regis, 2014]. The method is designed for solving global continuous constrained BBO problems where the evaluations of some

blackbox functions are computationally expensive. To do that, the first phase of the algorithm is to seek a feasible point by iteratively solving the following optimization problem:

$$\begin{aligned} \min_{x\in\mathcal{X}} \quad & \sum_{j=1}^{m} \max(0,\widehat{g}_j(x))^2 \\ \text{s.t.} \quad & \widehat{g}_j(x) + \epsilon \leq 0, \qquad j = 1,\dots,m \\ & \rho - \min_{y\in\mathcal{P}} \|x-y\| \leq 0, \end{aligned} \tag{2.25}$$

where $\mathcal{X}$ is a closed bounded domain, $\widehat{f}$ and $\{\widehat{g}_j, j = 1,\dots,m\}$ are RBF models of the objective and constraints, respectively, $m \in \mathbb{N}$ denotes the number of constraints, ϵ and ρ are positive real values and $\mathcal{P}$ is the set of DOE points that have been evaluated with the blackbox functions.

Once a feasible solution with respect to the original optimization problem is found, COBRA enters a second phase iteratively solving the following optimization problem:

$$\begin{aligned} \min_{x\in\mathcal{X}} \quad & \widehat{f}(x) \\ \text{s.t.} \quad & \widehat{g}_j(x) + \epsilon \leq 0, \qquad j = 1,\dots,m, \\ & \rho - \min_{y\in\mathcal{P}} \|x-y\| \leq 0, \end{aligned} \tag{2.26}$$

where the notations are common to Equation (2.25).

The gradient-based `fmincon` function of MATLAB is used for the resolution of these problems. At each iteration, the solution of Equation (2.25) or Equation (2.26) (according to the phase) is evaluated with the blackbox functions and used to update the surrogates for the next iteration.

The best solution of COBRA is updated at each iteration. In the first phase, it is defined as the solution with the smallest number of violated constraints or, in case of equality, the one with the lower maximum of the constraint violations. In the second phase of the algorithm, a new evaluated point becomes the best if it is feasible and its objective value is strictly better than that of the best point so far.

The parameter ρ corresponds to a minimum distance requirement to already evaluated points and is updated at each iteration following a cycle of finite discrete values. The margin ϵ is fixed in the first phase. In the second phase, after $T_{\texttt{feas}} \in \mathbb{N}^*$ consecutive iterations where the candidate solutions were feasible, ϵ is reduced whereas it is increased after $T_{\texttt{infeas}} \in \mathbb{N}^*$ iterations that return infeasible points.

The difficulty of finding a feasible point and the sensitivity to the parameterization are respectively tackled by the variants COBRA-R [Koch et al., 2014] and SACOBRA [Bagheri et al., 2017].

Other RBF-based methods include the constrained local metric stochastic RBF (ConstrLMSRBF), introduced in [Regis, 2011], that is a heuristic RBF-based algorithm. In ConstrLMSRBF, points are randomly generated, usually from a Gaussian distribution around the incumbent. Using the RBF models of the objective and constraint functions, among the points with the minimum number of predicted constraint violations, a point is chosen according to its predicted objective value and its distance to already evaluated points. After evaluation with the expensive functions, this new point is used to update the best solution of the algorithm and the RBF models. As the method requires an initial feasible point, an extended ConstrLMSRBF is presented in the same paper describing COBRA and deals with this issue by introducing a two-phase approach similar to what is done in COBRA.

Kriging is also commonly used in global surrogate-based methods, in particular in the efficient global optimization (EGO) method that is very popular. In order to determine the next points for the expensive evaluations, one classically seeks the minimizers of the surrogates. However, unless the initial DOE consists of numerous well-spread points, the first surrogates may not yield to good predictions. As kriging provides the variance at every point, so a measure of the uncertainty of the model, the main idea of EGO is to balance the search between the minimization purpose and the reduction of the uncertainty. To do so, call f the function to minimize whose kriging model was built, the algorithm uses the EI (introduced in Section 2.4.1) defined as follows:

$$EI(x) := \sigma(x)(u\phi(u) + \Phi(u)), \tag{2.27}$$

where $\sigma(x)$ is the variance at a point x, ϕ and Φ are the standard Gaussian cumulative distribution function and probability density function, respectively. The variable u is defined as:

$$u = \frac{f_{\min} - \widehat{f}(x)}{\sigma(x)}, \tag{2.28}$$

where $f_{\min}$ is the minimum function value observed so far and $\widehat{f}(x)$ is the kriging prediction of $f(x)$. By iteratively maximizing the EI, the algorithm selects points that are likely to yield to improvements and those in regions with high uncertainties. The solution of this maximization problem is evaluated with the expensive function and used to enrich the kriging model at each iteration. An enrichment procedure of EGO is depicted in Figure 2.2.

There are extensions of EGO to handle constraints such as SuperEGO [Sasena et al., 2002] that uses a penalized EI. The methods SEGOKPLS(+K) [Bouhlel et al., 2018] are extensions of SuperEGO to high dimensions that use kriging with partial least squares.

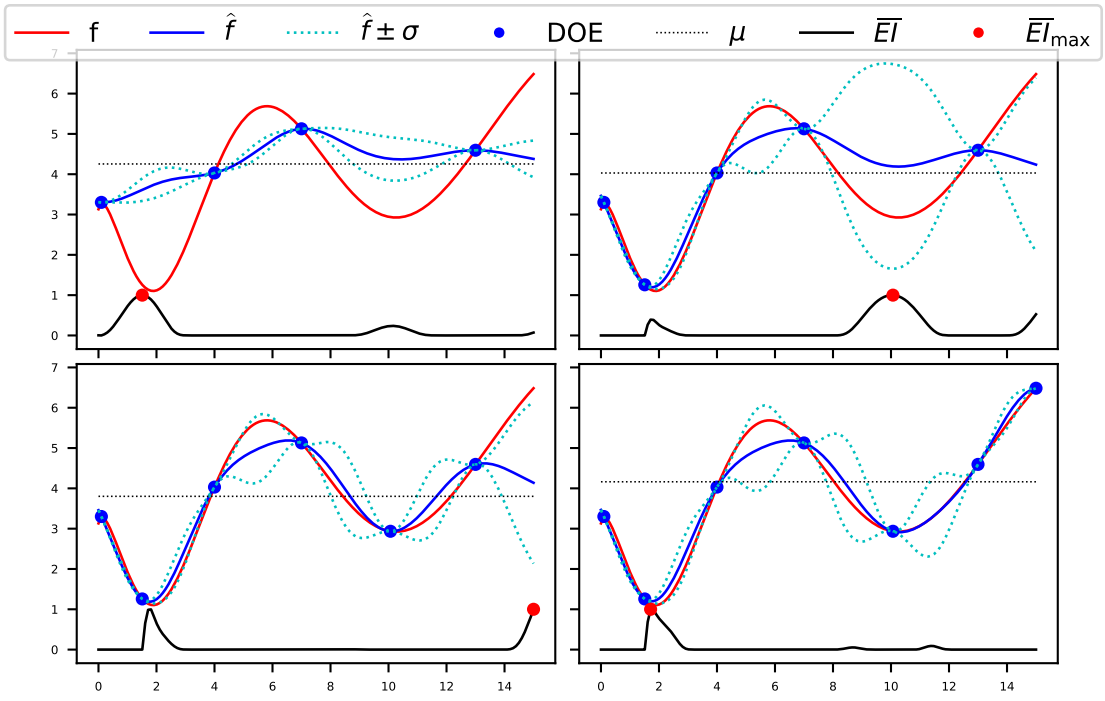


Figure 2.2: Enrichment of a kriging model $\widehat{f}$ of a function f using EGO. The curves of $\widehat{f} \pm \sigma$ are represented, where σ is the variance, as well as the DOE points, the mean of the model μ, the normalized expected improvement ($\overline{EI}$) and its maximum ($\overline{EI}_{\max}$).

2.5 Mixed-variable constrained optimization

The review of DFO methods presented in [Rios and V., 2013] highlights the fact that most DFO methods handle only continuous variables. However, in real-world optimization problems, variables can be of different natures. Moreover, not all the optimization methods presented so far in this thesis can handle the presence of general inequality constraints in the optimization problem. In this section we focus on mixed variables and constraints and give examples of methods for handling such features.

2.5.1 Dealing with mixed variables

An optimization problem can involve four main types of variables. The most used is the continuous type that stands for variables that can take any real value in a considered domain. The latter can be bounded or not. In an automotive context, examples of continuous parameters are geometric parameters such as lengths, radii or angles.

Apart from continuous variables, there are also discrete variables. The latter can take

values in a countable set of real values that can be ordered. This nature of inputs include, among others, binary numbers, integers and granular variables [Audet et al., 2019]. The latter are real values with controlled numbers of decimals and which are regularly spaced. Hence, integers are a specific case of granular variables with a granularity of 1. As an example, in an automotive vehicle context, the allowed thicknesses for the sheet metal of a part take discrete values that depend on the material and are set by the metallurgists. Other thicknesses are theoretically possible but at a way more expensive cost. At Stellantis, thicknesses are defined in granular discrete sets with a granularity of $5 \cdot 10^{-2}$ millimetres.

Categorical variables represent another type. As their name indicates, they refer to categories and are therefore not ordered. This type of inputs are usually represented using discrete values, often integers but with no ordering between them. A variable standing for the choice of a material among several ones is a good example of categorical variable. Indeed, let A and B stand for two different materials. Then, defining a continuous material model between them for the feasible material domain and which would enable to choose another material as a combination of A and B, is not possible. There is no natural ordering between them. In this case, a relative order can however be established by considering inner properties like the Young's modulus. For instance many parts of the BIW structure are steels following an elasto-plastic law. Here are some coefficients that characterize the behaviour of such steels: E_1 = 210000MPa, ν_1 = 0.3, α_1 = 270, b_1 = 440, n_1 = 0.720 and E_2 = 210000MPa, ν_2 = 0.3, α_2 = 400, b_2 = 670, n_2 = 0.870. For $i \in \{1, 2\}$, E_i denotes the Young's modulus of steel i, ν_i is the Poisson's ratio, α_i stands for the plastic yield stress, b_i is the plastic hardening parameter and n_i denotes the plastic hardening exponent.

Finally, there are meta variables [Audet et al., 2022a] that stand for variables that may affect the dimension of the problem or its number of acting constraints, that are those defining the feasible set. An example is a variable indicating the presence or the absence of an optional part of a vehicle: if the part is not considered, the related variables such as the thickness or the type of material do not intervene in the optimization problem.

Few DFO methods deal with other than continuous variables. The review of [Rios and V., 2013] comprises, in particular, the NOMAD solver that handles continuous, integer and granular variables. The solver also integrates the handling of categorical inputs thanks to an extended poll step that is only performed if the classical poll on all but categorical variables does not yield to a better point. However, the neighbourhoods of the categorical variables have to be provided by the user.

Another direct search method already mentioned and handling mixed types of variables is DIRECT. The method is, however, not effective in high dimensions: it is designed for problems with a maximum of 20 variables.

Some EAs have also been adapted for discrete variables. A mixed-integer version CMA-ES is mentioned in [Tušar et al., 2019] and consists basically in rounding the integer components of the generated points. An ACO designed for mixed-variable optimization problems, called ACO_{MV} is presented in [Liao et al., 2014], handling also categorical variables, but is not well-suited for problems with computationally expensive functions.

Discrete variables are considered in the constrained discrete optimization using response surfaces (CONDOR) algorithm introduced in [Regis, 2020]. It is a surrogate-based method for high-dimensional discrete BBO problems. CONDOR uses RBF approximations of the objective and constraint blackbox functions and performs various perturbations of the incumbent to find a better point.

Integer constrained expensive BBO problems are handled by SO-I (standing for surrogate optimization - integer) presented in [Müller et al., 2014]. It also uses RBF models and finds a feasible point by iteratively minimizing the sum of predicted constraint violations. When a feasible point is found, the method iteratively solves a penalty augmented objective function. At each iteration, a new point is evaluated with the blackbox functions to enrich the models.

The mixed-integer surrogate optimization (MISO) framework is introduced in [Müller, 2016] for the resolution of mixed-integer unconstrained expensive BBO problems. It uses RBF surrogates and implements different sampling strategies

Another RBF-based method handling mixed-integer variables but also constraints is SO-MI (standing for surrogate optimization - mixed-integer), developed in [Müller et al., 2013]. The method needs the presence of a feasible point in the initial DOE and models a penalty augmented objective function to evaluate four points at each iteration. These points are selected from four groups generated by random perturbations of the variables and uniform random points generations. The best point of each group is chosen according to two criteria relative to the surrogate predictions and the distance to already evaluated points. SO-MI is asymptotically complete, that is it converges to a global optimum with probability one.

The MARSOPT algorithm developed in [Martinez et al., 2017] aims at globally optimizing non-convex piecewise linear MARS models subject to constraints involving linear regression and piecewise linear MARS models. It uses branch-and-bound for the optimization of the model and a Pareto evaluation procedure is proposed as a measure for quality and robustness of surrogates, even in the absence of an enrichment strategy, that is no additional point than the initial DOE is evaluated with the true functions.

A final example is the global optimization with surrogate approximation of constraints

(GOSAC) method [Müller and Woodbury, 2017] for the resolution of mixed-integer problems with expensive constraints and a cheap objective. Cubic RBF models of every constraints are used to predict the feasibility of new sample points. GOSAC considers "QRSK" constraints, meaning quantifiable, relaxable, simulation and known. These assumptions involve that the simulation model will not crash. The cheap objective is also supposed to be computable separately from the constraints. The algorithm proceeds in two phases, the first of which seeks a feasible point by fitting RBF models to the constraints and optimizing an unconstrained multi-objective subproblem defined by the constraints. Points that are too close to already evaluated points are discarded and, from the remaining set of solutions, the one with the smallest constraint violation is evaluated with the expensive constraints. The process is repeated until a feasible solution is found. Then, the second phase consists in improving this point by solving an auxiliary problem with the original objective and the surrogates of the constraints. If the solution of this subproblem is too close to the incumbent, the new incumbent is chosen to maximize the minimum allowed distance to the already evaluated points.

2.5.2 Handling general inequality constraints

A lot of real-world optimization problems are subject to constraints and the latter can be classified using the taxonomy of [Le Digabel and Wild, 2015].

In many cases, the constraints of an optimization problem are *quantifiable* and, thus, enable the quantification of the degree of feasibility or violation. Let $(g_j)_{j\in J}$ be $|J|$ constraint functions of a BBO problem with J a finite set of integer indices, such that the j^{th} constraint is satisfied if $g_j(x) \leq 0$. A simple measure of quantifiable violation of this constraint is $\max(0, g_j(x))$. Note that for a feasible point, this violation is equal to 0.

Another measure is the *constraint violation function* h using the sum of squared violations and defined as follows:

$$h(x) := \begin{cases} \sum_{j\in J} \max(0, g_j(x))^2 & \text{if } x \in \mathcal{X} \\ \infty & \text{otherwise,} \end{cases} \tag{2.29}$$

where $\mathcal{X} \subseteq \mathbb{R}^n$ is the search space, $n \in \mathbb{N}^*$ being the dimension of the problem. The function h is nonnegative and $h(x) = 0$ if and only if x is a feasible point with respect to $(g_j)_{j\in J}$. For points out of the search space, h takes infinite values.

Sometimes, however, constraints do not give numerical values but only the information of whether the constraint is satisfied or not: they are *nonquantifiable*. Such outputs can however be represented as binaries.

Constraints are called *unrelaxable* if they must be satisfied to get meaningful outputs, and *relaxable* otherwise.

Others, called *hidden constraints*, may not be known in advance, unlike *known* constraints. Hidden constraints are not rare in BBO. They refer to internal requirements of the oracle that are not explicitly known to the solver. An example is the query of an FE model: although a design point is feasible with respect to the known constraints of the optimization problem, the simulation can fail (or crash) due, for instance, to numerical noise or divergence in the FE resolutions caused by different meshing.

Finally, *a priori* constraints refer to constraints whose feasibility can be determined without a simulation and are opposed to *simulation-based* constraints which require the run of a simulation to assess their feasibility.

Among the main methods used to deal with relaxable and quantifiable inequality constraints are penalty, filter and barrier methods that we describe below.

Penalty methods

Penalty methods are popular to handle constraints. They consist in integrating the constraint violations in the objective of the optimization problem using weights. The penalized problem is unconstrained, its objective is equal to the original one for feasible points but adds (or removes for a maximization problem) a penalty for infeasible points. For an objective f to minimize, an example of penalized objective is:

$$f(x) + \mu h(x), \tag{2.30}$$

where μ is a positive penalization coefficient and h is defined as in Equation (2.29). Adding penalization terms can however introduce severe slope changes in the objective function and can be misleading for the optimization solver or bring numerical instability for high values. Moreover, there is no general rule about how to choose penalization coefficients.

The augmented Lagrangian method is similar to a penalty method but also adds a term that mimics the Lagrange multipliers. It was originally designed in [Hestenes, 1969, Powell, 1969] to handle equality constraints but was generalized to inequality constraints in [Rockafellar, 1973] by transforming each inequality $g_j \leq 0$ to the equality $g_j + z_j = 0$ with $z_j \geq 0$ a slack variable.

For an optimization problem of the form of Equation (1.1), the augmented Lagrangian function is defined as follows:

$$f_{\mathcal{L}}(x) := f(x) + \sum_{j=1}^{|J|} \phi(g_j(x), \gamma_j, \mu), \tag{2.31}$$

where $(\gamma_j)_{j\in J}$ are estimates of the Lagrange multipliers and $\mu > 0$ is a penalty coefficient. The function ϕ is defined as follows:

$$\phi(g_j(x), \gamma_j, \mu) := \begin{cases} -\gamma_j g_j(x) + \frac{\mu}{2} g_j^2(x) & \text{if } g_j(x) \leq \frac{\gamma_j}{\mu} \\ -\frac{\gamma_j^2}{2\mu} & \text{otherwise.} \end{cases} \tag{2.32}$$

This method is for instance used to handle inequality constraints in EAs such as the augmented Lagrangian version of CMA-ES (AL-CMA-ES) presented in [Dufossé and Hansen, 2021].

Filter methods

Filter methods can be used for constrained optimization. The principle is to treat the constraint violation as another objective to minimize. Thus, a bi-objective minimization problem is considered where the first objective is the original objective f of the problem and the second one is h or any nonnegative function representing the constraint violation.

In order to emphasize the constraints satisfaction, the authors of [Fletcher et al., 2002a, Fletcher and Leyffer, 2002, Fletcher et al., 2002b] use feasibility restoration steps. Such a step occurs when an iterate is not accepted by the current filter. A restoration phase focuses on iteratively reducing the constraint violation until a feasible iterate of the considered subproblem is computed. Their work has inspired a filter method presented in [Audet and Dennis Jr, 2004], used in pattern search algorithms and that we describe hereafter.

The algorithm uses a maximum allowed constraint violation value $h_{\max}$. A filter $\mathcal{F}$ is defined as a finite set of infeasible points where no point strictly dominates another one with respect to the two objectives (the original one and the constraint violation function). Let $f_{\min}$ stand for the minimum feasible value of f found so far. A point x is filtered if one of these three conditions is satisfied:

1. $h(x) = 0$ and $f(x) \geq f_{\min}$
2. x is infeasible and $h(x) \geq h_{\max}$
3. $h(x) > 0$ and there exists $\bar{x} \in \mathcal{F}$ such that $\bar{x} \preceq x$.

At each iteration of the method, after the evaluations of new candidate points, the filtered points are rejected and the unfiltered infeasible points are added to $\mathcal{F}$. An iteration is considered successful if unfiltered points are found.

Barrier methods

A barrier method solves an unconstrained optimization problem minimizing a modified objective function. The latter assigns high or infinite values to points belonging to a certain infeasibility region.

Let Ω be the feasible domain of the optimization problem minimizing the objective f, the extreme barrier (EB) method [Audet and Dennis, Jr., 2006] uses the extreme barrier function:

$$f_\Omega(x) := \begin{cases} f(x) & \text{if } x \in \Omega \\ \infty & \text{otherwise.} \end{cases} \tag{2.33}$$

This function is used in place of the original objective in the optimization process such that all infeasible points are rejected.

The EB method can be used only if a feasible point is already known. To cope with this requirement, a two-phase barrier method considers two unconstrained optimization problems: it first minimizes the constraint violation function and then minimizes the EB function if the solution of the first optimization problem is feasible. Bound and hidden constraints can be handled using an EB approach.

Another kind of barrier method is the progressive barrier (PB) whose main idea is to exclude points with high constraint violations. It uses the constraint violation function of Equation (2.29) and is less severe than the EB. Similarly to the filter method presented above, a maximum allowed constraint violation $h_{\max}$ is used to reject points. This threshold is initialized at ∞, and is updated in a nonincreasing way at each iteration. The minimum possible value of $h_{\max}$ is 0 and this case corresponds to an EB approach.

Candidate points in a PB are ranked according to a nondominated sorting explained hereafter. For x and $\bar{x}$ two points of $\mathcal{X}$, call Ω the feasible domain of the optimization problem, x dominates $\bar{x}$ in the PB (we write $x \prec_h \bar{x}$) if and only if one of these conditions is satisfied:

1. $x \in \Omega$, $\bar{x} \in \Omega$ and $f(x) < f(\bar{x})$
2. $x \in \Omega$ and $\bar{x} \in \mathcal{X} \setminus \Omega$
3. $x \in \mathcal{X} \setminus \Omega$, $\bar{x} \in \mathcal{X} \setminus \Omega$ and the inequalities $f(x) \leq f(\bar{x})$ and $h(x) \leq h(\bar{x})$ hold with at least one strict inequality.

Hence, feasible points are always preferred to infeasible ones and both the objective and the constraint violation function are considered when dealing with infeasible points. The progressive barrier can be applied for relaxable constraints, namely those that do not lead to failures or aberrant outputs of the blackbox.

There are other types of barrier methods like the logarithmic barrier method. For quantifiable constraints, it uses the following barrier function:

$$f_{\log}(x) \coloneqq f(x) - \frac{1}{t} \sum_{j=1}^{|J|} \log(-g_j(x)), \tag{2.34}$$

where t is a positive parameter, $(g_j)_{j \in J}$ are the constraints and J is a finite set of integer indices. This function, that is defined only for strictly feasible points, is sequentially minimized with increasing values of t.

3

Performance evaluations of continuous algorithms in a BBO context

3.1 Motivation

In order to solve a blackbox optimization problem such as Equation (1.1), it is important to first inquire existing methods before deciding which one to use. To help with this decision, some tools have been developed to compare the performance of algorithms. In particular, data profiles [Moré and Wild, 2009] are frequently used in DFO and BBO to benchmark algorithms: they show, given some precision or target value, the fraction of problems solved by an algorithm according to the number of function evaluations. There also exist suites of academic test problems: although the latter are treated as blackbox functions, they are analytically known, which is an advantage to understand the behaviour of an algorithm. There are also available industrial problems but they are rare.

Assessing the performance of the solvers is essential as it provides information on the solutions quality when the dimension of the problem increases or on particular groups of problems for instance. In particular, testing them on well-understood problems helps in identifying advantages and drawbacks for future improvements and developments of new

methods. Benchmark studies also highlight how the performance of particular implementations evolve over time.

As real-world optimization problems are often complex, involving for instance blackbox functions, mixed variables or multiple inequality constraints, the working plan is to operate gradually in the challenges. This chapter describes the benchmark study introduced in [Varelas and Dahito, 2019] that makes use of a free benchmarking platform briefly presented in the next section. The experiments use eight multivariate local solvers and an evolutionary global algorithm that are compared on unconstrained continuous instances from the literature used as blackbox functions.

The problems addressed in this chapter can be written as:

$$\min_{x \in \mathcal{X}} f(x), \tag{3.1}$$

where f is a continuous real blackbox function and $\mathcal{X}$ is a closed bounded domain of $\mathbb{R}^n$ with $n \in \mathbb{N}^*$ the dimension of the problem.

3.2 The COCO platform

The comparing continuous optimizers (COCO) platform, described in [Hansen et al., 2021], is a framework enabling easy benchmark studies of blackbox solvers. It provides several suites of standard well-known test problems and some less regular variants. The test suites include, among others, the `bbob` suite of standard continuous functions, `bbob-noisy` consisting of functions with different types and levels of noise, `bbob-largescale` with functions in high dimensions and the `bbob-mixint` suite of mixed-integer problems. In this chapter, we focus on the `bbob` continuous test suite.

3.2.1 The continuous `bbob` testbed

The `bbob` suite [Finck et al., 2009] of COCO comprises 24 continuous functions from the literature, all available in six dimensions: $2, 3, 5, 10, 20$ and 40. Each function is declined in 15 instances that are obtained by applying transformations in variable and objective space on the raw functions. There can be for example rotations, translations or nonlinear transformations. The objective is to obtain less regular and non-trivial variants since real-world problems are expected to have irregularities. This makes a total of $24 \times 15 \times 6 = 2160$ available instances in the `bbob` benchmark suite.

The instances are treated as blackbox functions used in an unconstrained optimization context and are classified in five groups: separable, moderate, ill-conditioned, multimodal

weakly structured and multimodal with global structure. Let n stand for the dimension of the problem, all functions are known to have their global optima in $[-5, 5]^n$.

3.2.2 Performance measures

Among the various tools available for algorithm comparison, COCO notably provides empirical cumulative distribution functions (ECDF) plots for every dimension aggregated on all functions, but also for each function or by function group. They show the empirical runtime distributions that are computed as the number of function evaluations to reach given function target values, divided by the dimension.

A function target value is defined as $f_t = f^* + \Delta f$, where f^* is the minimum value of a function f and Δf is a target precision. In the `bbob` suite, COCO sets 51 target values and the latter are chosen to be evenly log-spaced between 10^{-8} and 10^2. The corresponding target precision are denoted Δf_t, $t = 1, \ldots, 51$.

The ordinate axis of an ECDF plot ranges between 0 and 1, indicating the success rate, that is the fraction of problems solved. Hence, when all targets (including the smallest one 10^{-8}) are reached by an algorithm, its ECDF curve reaches the highest ordinate axis that is 1.

The average runtime (aRT), used in the figures and tables, is computed over all relevant trials as the number of function evaluations executed during each trial while the best function value did not reach a given function target value f_t, summed over all trials and divided by the number of trials that actually reached f_t [Price, 1997, Hansen et al., 2012].

The statistical significance of the results is evaluated in COCO using the rank-sum test for a given target Δf_t. For each trial of an algorithm, it uses either the number of needed function evaluations to reach Δf_t (inverted and multiplied by -1), or, if the target was not reached, the best Δf-value achieved, measured only up to the smallest number of overall function evaluations for any unsuccessful trial under consideration.

Crosses appear on ECDF and indicate the end of the experiment, usually when the maximum evaluation budget is reached. After a cross, COCO uses simulated restarts to give an estimation of the runtimes of the unsuccessful runs, meaning those that did not yield to a function value at least equal to 10^{-8}.

The ECDF plots also hold a curve labelled *best 2009*. The latter corresponds to an artificial solver used as a reference and whose data[1] comes from the BBOB-2009 workshop comparing 31 solvers. The *best 2009* curve shows the performance as if an artificial solver knew at every iteration what a function looked like.

[1] https://coco.gforge.inria.fr/doku.php?id=bbob-2009-results

3.3 Benchmarking multivariate solvers

In this section, a variety of solvers either in a blackbox setting or in a gradient based setting with approximations of the gradients is benchmarked. In particular, multivariate optimization solvers from the Python SciPy[2] library are compared, under default or modified parameter settings.

A similar study for a previous version of the library, that benchmarked six solvers of the library under default parameters has been presented in [Baudiš, 2014], where the Basin Hopping [Wales and Doye, 1997] restart strategy was used within each independent restart. It is useful to compare, though, how particular implementations of such methods evolved and improved over time.

In this study we follow a policy where independent restarts are applied when the corresponding termination criteria are met, until a given budget of function evaluations is exhausted. Based on a preliminary investigation, we choose proper parameter settings and termination conditions for some algorithms such that their performance is not deteriorated.

The contribution in comparison to [Baudiš, 2014] is threefold: for the common benchmarked solvers, we compare the different parameter settings and restart policies. Furthermore, complete data sets for all dimensions are included (in [Baudiš, 2014] the results were restricted to dimensions 2, 5 and 20) and three additional solvers are benchmarked, as well as the adaptation of the Nelder-Mead method to high dimensions.

3.3.1 Considered algorithms

The following algorithms were benchmarked, where a star indicates those included in [Baudiš, 2014] as described above: Nelder-Mead*, adaptive Nelder-Mead, Powell*, BFGS*, L-BFGS-B*, conjugate gradient*, truncated Newton, differential evolution, COBYLA and SLSQP*

The NM method, its adaptive version for high dimensions, Powell's conjugate direction method, BFGS, DE, COBYLA and SLSQP were presented in Chapter 2. The three first methods are respectively denoted "Nelder-Mead", "adapt-Nelder-Mead", and "Powell" in the graphs.

L-BFGS-B [Liu and Nocedal, 1989] is the modification of L-BFGS, described in Section 2.2, to handle box constraints. Thus, it is also based on BFGS recursion for the approximation of the inverse Hessian.

The nonlinear conjugate gradient algorithm by Polak and Ribiere [Nocedal and Wright, 2006] is an extension of the Newton CG method to non-quadratic functions. At iteration

[2]SciPy version 1.2.1

$k \in \mathbb{N}$, it computes a step length α_k to approximate a minimizer of the objective f in a search direction p_k. A residual is computed as $r_k := \nabla f(x_k)$ and its norm is used as a stopping criterion. Only the first derivatives are used in this method that is designated by "CG" in the graphs.

Finally, the truncated Newton algorithm [Nash, 1984, Nocedal and Wright, 2006] uses the Newton CG method with box constraints. It is denoted by "TNC" in the plots.

3.3.2 Preliminary experiments

In order to identify proper settings prior to the performance comparison of all solvers, experimentation was performed separately up to some extent, concerning in most, but not all, cases the termination tolerances in search and objective space.

The tolerance in objective values, defined by the parameter `ftol`, is investigated for the quasi Newton L-BFGS-B algorithm for high dimensional optimization. In SciPy, `ftol` is set to 10^{-8} by default. Experiments show that reducing this tolerance can be of advantage, in particular for ill-conditioned functions and for the Attractive Sector function, as presented in Figure 3.1 and Figure 3.2. This performance improvement becomes more significant with increasing dimension.

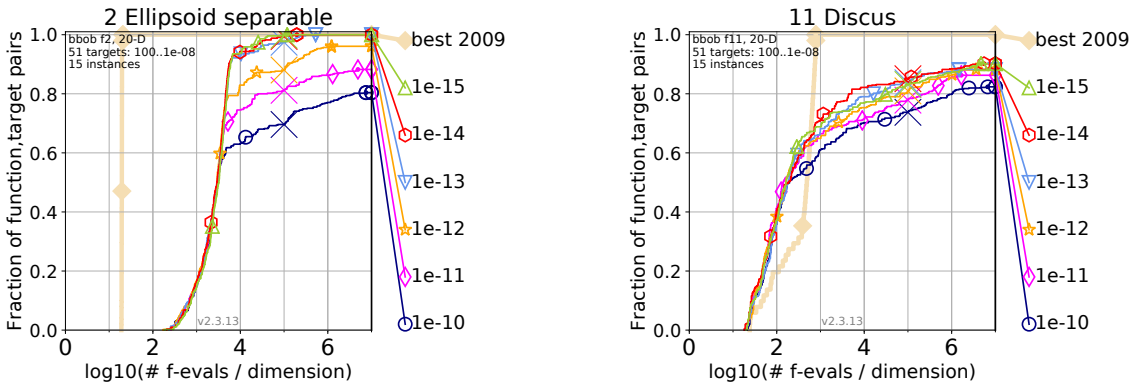


Figure 3.1: Empirical cumulative distribution of simulated (bootstrapped) runtimes, measured in number of function evaluations, divided by dimension (FEvals/DIM) for the 51 targets $10^{[-8,\ldots,2]}$ of the ill-conditioned separable Ellipsoid and Discus functions in dimension 20 for L-BFGS-B. The lines correspond to different values of f-tolerance for termination.

More importantly, the maximum number of variable metric corrections for the Hessian approximation has to be set carefully. The default value is 10 and experiments showed performance improvement for increasing values up to $2 \cdot n$, n being the problem dimension, as illustrated in Figure 3.3.

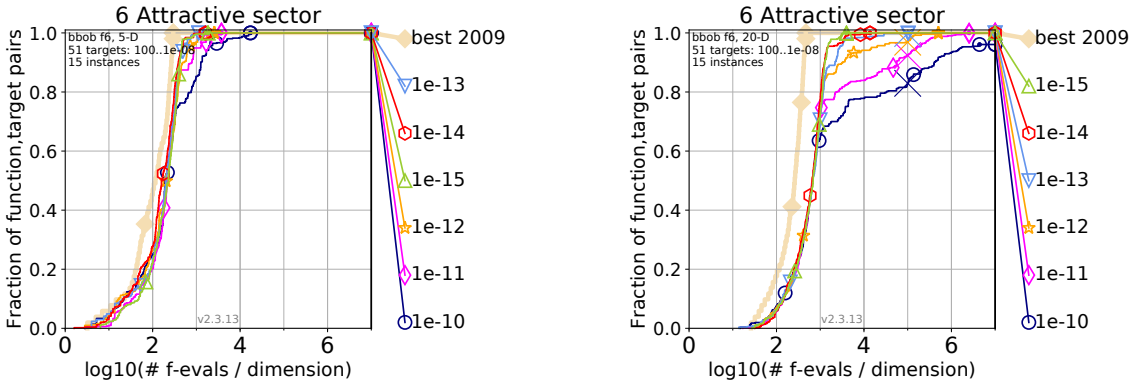


Figure 3.2: Empirical cumulative distribution of simulated (bootstrapped) runtimes, measured in number of function evaluations, divided by dimension (FEvals/DIM) for the 51 targets $10^{[-8,\ldots,2]}$ of the Attractive Sector function in dimensions 5 and 20 for L-BFGS-B. The lines correspond to different values of f-tolerance for termination.

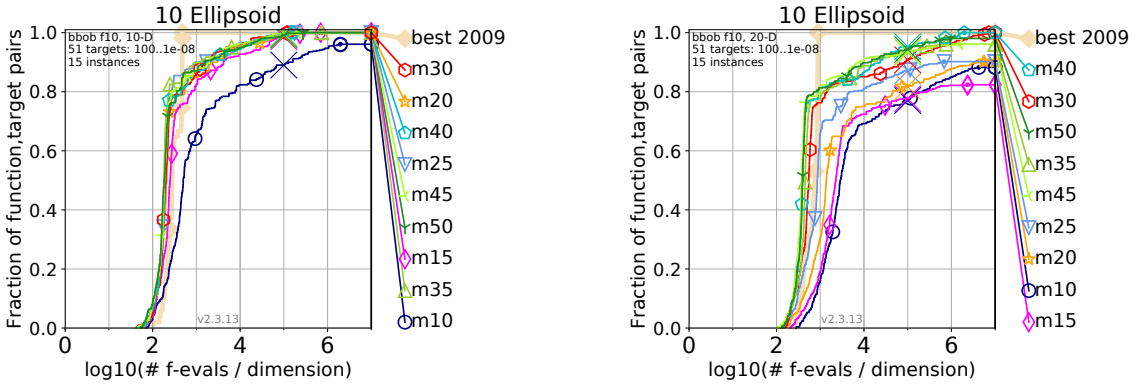


Figure 3.3: Empirical cumulative distribution of simulated (bootstrapped) runtimes, measured in number of function evaluations, divided by dimension (FEvals/DIM) for the 51 targets $10^{[-8,\ldots,2]}$ of the ill-conditioned separable Ellipsoid function in dimensions 10 and 20 for L-BFGS-B. The lines correspond to different values of maximum number of variable metric corrections for the Hessian approximation.

Furthermore, the effect of decreasing the step length for the finite difference approximation of the gradient was investigated to some extent: decreasing the default value of this parameter (10^{-8}) can improve the performance on particular functions, such as the Ellipsoid, while it shows worse success ratio on others. A more detailed study was presented in [Blelly et al., 2018]

For Powell's algorithm, decreasing `ftol` also turned out to be beneficial. Besides, different settings of the termination tolerance in the variable space `xtol` were evaluated. The values tested are in the set $\{10^{-2}, 10^{-3}, 10^{-5}, 10^{-6}\}$, the default one being 10^{-4}. Values

larger than the default one typically can make the solver faster only for the easiest targets, while smaller values can show an improved success rate for high budgets.

SLSQP has been tested for different values of `ftol` in $\{10^{-6}, 10^{-9}, 10^{-12}, 10^{-15}\}$. Same as L-BFGS-B and Powell's algorithm, it was sensitive to this parameter, with advantages when `ftol` decreases.

3.3.3 Experimental procedure

As preliminary experiments show that reducing the tolerance in function values is beneficial to L-BFGS-B, in our experimentation it was set to the float machine precision[3] for very high accuracy. Also based on these experiments, the maximum number of variable metric corrections for the Hessian approximation was set to $2 \cdot n$, n being the problem dimension. Besides, the step length for the finite difference approximations of the gradients was kept at its default value, that is 10^{-8}.

For the modified Powell's conjugate direction algorithm, the tolerance in function values `ftol` was set to 10^{-15} and that of the variable space `xtol` was kept at the defaults value 10^{-4}. The parameter `ftol` of SLSQP was set to 10^{-15} in the performance comparison. The NM simplex method is tested both in its default setting and with adaptation of parameters to the dimensionality of the problem, controlled by the `adaptive` flag.

The truncated Newton algorithm requires an estimation of the optimal f value. Since it always lies in $[-1000, 1000]$ [Finck et al., 2009] and in accordance to the blackbox setting where no prior information is available for the function, we set this value to -1000. The original BFGS, the conjugate gradient algorithm by Polak and Ribiere, DE as well as COBYLA are benchmarked in their default setting.

In cases where the solver supported constraint handling, no constraints were applied. Finally, the maximum iterations were set to values large enough (wherever applicable), in order to avoid termination before convergence.

All solvers were run on the `bbob` testbed with restarts for a maximum budget of $10^5 \cdot n$ function evaluations (at minimum, for solvers that did not support a termination callback such as COBYLA). For all runs, the initial point was chosen uniformly at random in $[-4, 4]^n$ and with the function value evaluated at this point. In the special case of DE where no initial point is given, the domain bounds were set to $[-5, 5]^n$.

[3] Equal to $2.220446049250313 \cdot 10^{-16}$

3.3.4 CPU timing

The Python code was run on several multicore machines (not exclusively) with different number of cores. The time per function evaluation, measured in 10^{-5} seconds, for different dimensions along with the corresponding processor type is presented in Table 3.1.

Algorithm	Processor Type	2-D	3-D	5-D	10-D	20-D	40-D
Nelder-Mead	Intel(R) Xeon(R) CPU E5-2667 v3 @ 3.20GHz	2.2	2.3	2.4	2.9	3.5	5.0
Adaptive Nelder-Mead	Intel(R) Xeon(R) CPU E5-2667 v3 @ 3.20GHz	2.1	2.1	2.2	2.6	3.3	4.8
Powell	Intel Core Haswell, no TSX @ 2.29GHz	2.1	2.5	2.4	2.5	2.9	4.0
BFGS	Intel(R) Xeon(R) CPU E5-2667 v3 @ 3.20GHz	2.7	2.7	2.6	2.7	3.3	5.2
L-BFGS-B	Intel(R) Xeon(R) CPU X5650 @ 2.67GHz	2.9	2.3	2.1	2.3	3.1	5.4
Conjugate Gradient	Intel(R) Xeon(R) CPU E5-2667 v3 @ 3.20GHz	2.2	1.9	1.4	1.3	1.9	3.8
Truncated Newton	Intel(R) Xeon(R) CPU E5-2683 v4 @ 2.10GHz	1.4	1.4	2.5	2.0	2.8	5.0
Differential Evolution	Intel(R) Xeon(R) CPU X5650 @ 2.67GHz	8.4	8.4	8.5	9.3	11.0	14.0
COBYLA	Intel Core Haswell, no TSX @ 2.29GHz	0.51	0.53	0.65	0.96	2.1	8.2
SLSQP	Intel Core Haswell, no TSX @ 2.29GHz	2.0	1.9	1.8	2.1	1.9	2.2

Table 3.1: CPU timing per function evaluation for all algorithms considered in dimensions 2, 3, 5, 10, 20 and 40.

3.3.5 Results

Results from experiments according to [Hansen et al., 2016b] and [Hansen et al., 2016a] on the benchmark functions are presented in Figures 3.4 and 3.5 for some functions respectively in dimensions 5 and 20. For these dimensions, the ECDF plots for the group of multimodal functions with adequate global structure are illustrated in Figure 3.6 and aggregated results over all functions are depicted in Figure 3.7. Other plots showing the average running time for all dimensions and ECDF graphs for each function, for each function group and aggregated over all functions in dimensions 5 and 20 are available in Appendix A. The experiments were performed with COCO version 2.3, the plots were produced with version 2.3. The solvers benchmarked in [Baudiš, 2014] are denoted by a prefix "B-" in the corresponding name and the data were obtained by the data archive that COCO provides.

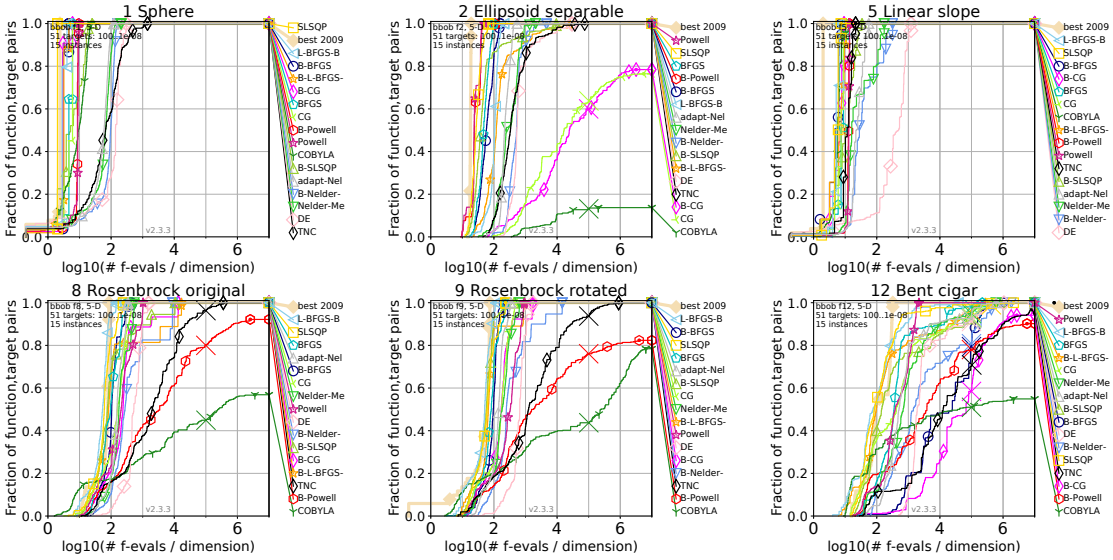


Figure 3.4: Empirical cumulative distribution of simulated (bootstrapped) runtimes, measured in number of function evaluations, divided by dimension (FEvals/DIM) for the 51 targets $10^{[-8,\ldots,2]}$ in dimension 5.

3.3.6 Observations

Aggregated results over all 24 functions of the suite are depicted in Figure 3.7 for dimensions 5 and 20 and show the effectiveness of SLSQP. In dimension 5, it has the highest success rate for a budget range $[18 \cdot n, 800 \cdot n]$ while in dimension 20, it dominates all solvers up to $\sim 1400 \cdot n$ function evaluations, after which it is outperformed by L-BFGS-B.

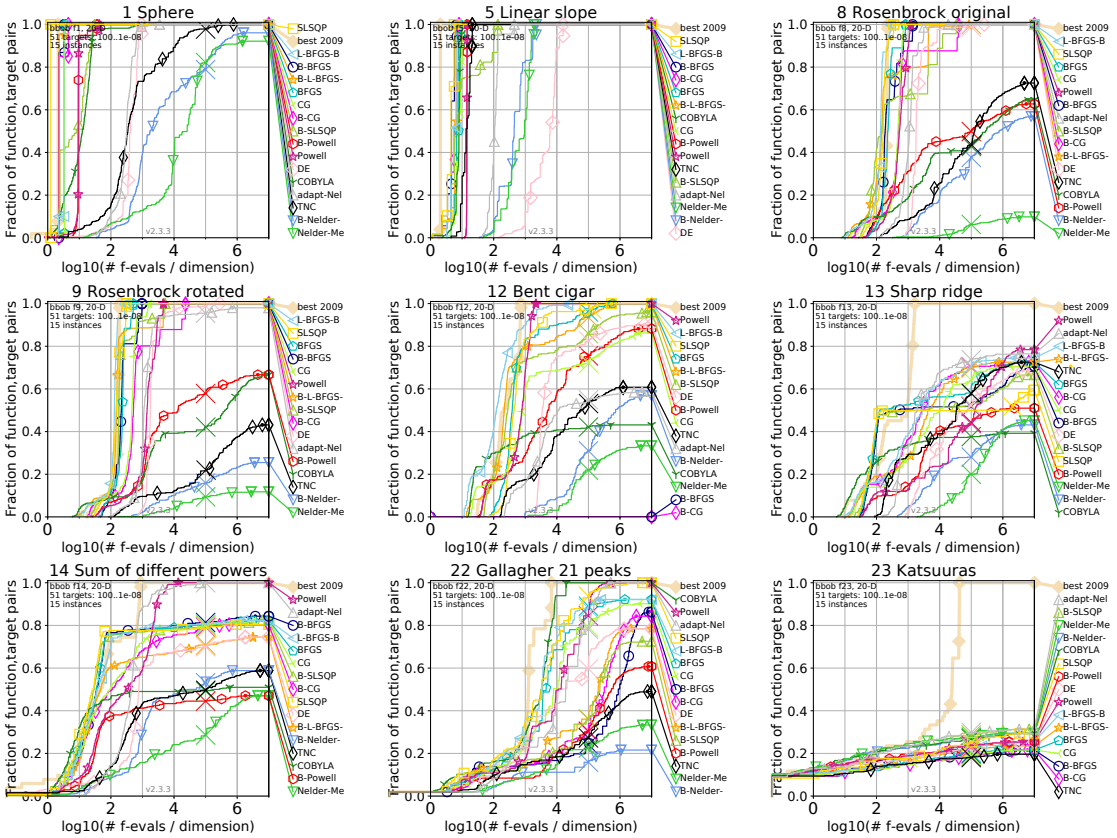
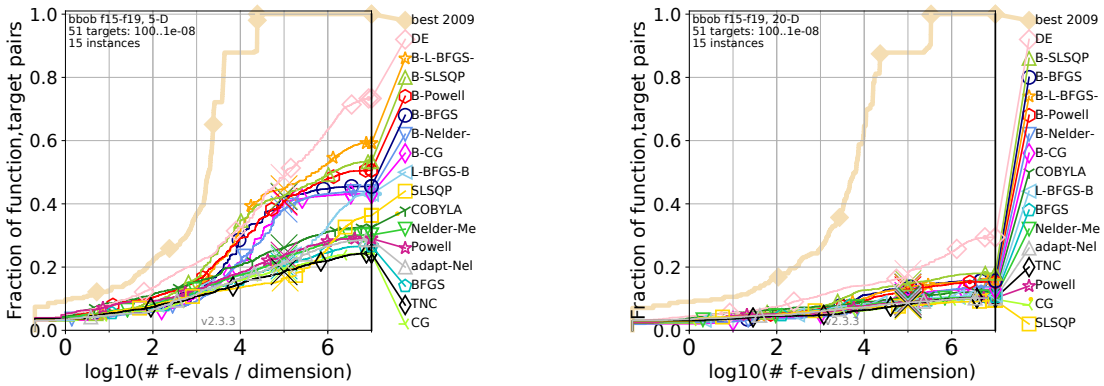


Figure 3.5: Empirical cumulative distribution of simulated (bootstrapped) runtimes, measured in number of function evaluations, divided by dimension (FEvals/DIM) for the 51 targets $10^{[-8,\ldots,2]}$ in dimension 20.

It is interesting to see the performance difference of SLSQP and B-SLSQP in unimodal functions such as the separable Ellipsoid function: in dimension 5, the runtimes are almost equal for the easiest targets and then SLSQP is faster by an increasing factor, until termination criteria start to become effective, as depicted in Figure 3.4. Performance differences between the early and recent implementation of SciPy, which are not due to the different parameter setting or restart policy of [Baudiš, 2014], are also observed for BFGS and NM, showing an improvement of the library implementation.

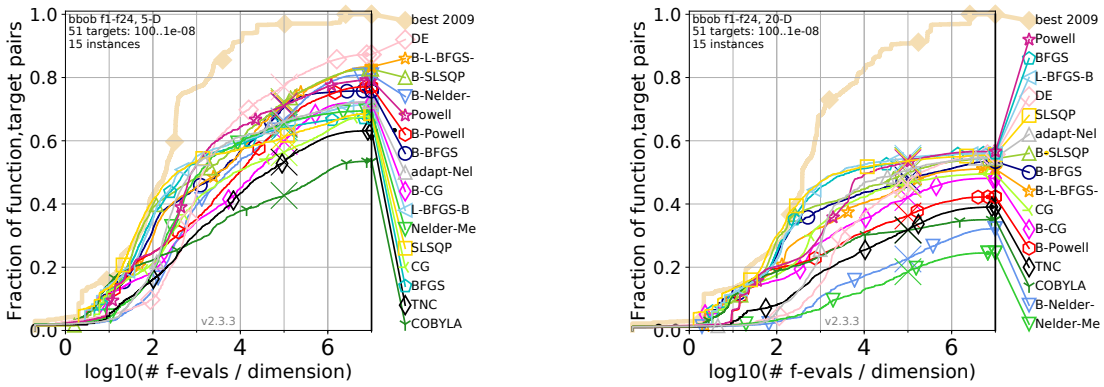
BFGS show better performance than L-BFGS-B for some functions in all dimensions, as it is the case for the Sphere, Linear Slope, original and rotated Rosenbrock and Bent Cigar functions. Overall, the picture is more diverse: BFGS has same or higher success rate in the budget ranges $[25 \cdot n, 125 \cdot n]$ and $[50 \cdot n, 400 \cdot n]$ for dimensions 5 and 20 respectively, while the runtimes always differ by less than a factor of 4.



(a) 5D, multimodal functions

(b) 20D, multimodal functions

Figure 3.6: Empirical cumulative distribution of simulated (bootstrapped) runtimes, measured in number of function evaluations, divided by dimension (FEvals/DIM) for the 51 targets $10^{[-8,\ldots,2]}$ for multimodal functions with adequate global structure in dimensions 5 and 20.



(a) 5D, all functions

(b) 20D, all functions

Figure 3.7: Empirical cumulative distribution of simulated (bootstrapped) runtimes, measured in number of function evaluations, divided by dimension (FEvals/DIM) for the 51 targets $10^{[-8,\ldots,2]}$, aggregated over all functions in dimensions 5 and 20.

For the NM method, adaptation of parameters is crucial. Without this option, the algorithm is deteriorated as the dimension increases. In dimension 20, the smallest target values for the Sphere function are not reached while in the aggregated ECDF the method is dominated by all other solvers and for all budgets.

It is interesting that COBYLA, that is based simply on linear interpolation, often achieves better performance for the fraction of easiest targets than all other solvers e.g.

for the Sharp Ridge and Sum of Different Powers functions in dimension 20, even outperforming the virtual best solver of BBOB 2009 for small budgets, as illustrates Figure 3.5. More remarkable is the performance on the multimodal Gallagher and Katsuura functions in dimension 20, where it is one of the most effective methods.

Finally, as shows Figure 3.6, DE shows the best performance among the other (local) solvers for the group of multimodal functions with adequate global structure, where also the Basin Hopping policy is of advantage. Even though the effectiveness of DE weakens with increasing dimension, it maintains the highest success rate in this function group.

4

Performance evaluations of continuous and mixed-integer algorithms

4.1 Motivation

Given the growing number of algorithms to deal with BBO problems, the choice of the most adapted method for solving a specific problem still remains complex. The MADS algorithm is commonly used in BBO and was therefore considered in this thesis. The algorithm is derived in several instantiations available in the NOMAD software that enables to choose between different numbers and types of poll directions.

In particular, the ORTHOMADS algorithm is the default MADS instantiation used in NOMAD and was introduced in [Abramson et al., 2009b]. It consists in using orthogonal directions in the poll step of MADS. It is compared to the initial LTMADS, where the poll directions are generated from a random lower triangular matrix, and to GPS algorithm on 45 problems from the literature. They show that MADS outperforms GPS and that the instantiation ORTHOMADS competes with LTMADS and has the advantage that its poll directions cover better the variable space.

The ORTHOMADS algorithm presents variants in the poll directions of the method. To our knowledge, the performance of these different variants has not been discussed in the

literature before. Our purpose is to explore this aspect by performing experiments with the ORTHOMADS variants.

This chapter presents the benchmark study of [Dahito et al., 2021] and plots for analyses are available at the following link: https://github.com/DahitoMA/ResultsOrthoMADS.

Our contributions are first the evaluations of the ORTHOMADS variants on continuous and mixed-integer optimization problems. Besides, the contribution of the search phase is studied. Two from the best variants of ORTHOMADS are identified on each of the used testbeds and their performance is compared with other algorithms including heuristic and non-heuristic techniques. The problems addressed are of the form:

$$\min_{x \in \mathcal{X}} f(x), \tag{4.1}$$

where $\mathcal{X}$ is a closed bounded domain of either $\mathbb{R}^n$ or $\mathbb{R}^c \times \mathbb{Z}^i$ with c and i respectively the number of continuous and integer variables. $n = c + i$ is the dimension of the problem and f is a blackbox function.

4.2 Related work

The performance of MADS is evaluated in several papers. As examples, a broad comparison of twenty two implementations of DFO algorithms for solving box-constrained optimization problems is performed in [Rios and V., 2013]. The methods are compared with each other according to different criteria. They use a set of 502 problems that are categorized according to their convexity (convex or nonconvex), smoothness (smooth or non-smooth) and dimensions between 1 and 300. The algorithms tested include local-search methods such as MADS through NOMAD version 3.3 and global-search methods such as the new unconstrained optimization algorithm (NEWUOA) [Powell, 2006], that uses trust regions, and CMA-ES.

NOMAD is used in [Regis, 2014] with a DACE surrogate and compared with other local and global surrogate-based approaches in the context of constrained blackbox optimization on an automotive optimization problem and twenty two test problems.

Simulation optimization deals with problems where at least some of the objective or constraints come from stochastic simulations. A review of algorithms to solve simulation optimization problems is presented in [Amaran et al., 2014], among which the NOMAD software. However, this paper does not compare them due to a lack of standard comparison tools and large-enough testbeds in this optimization branch.

Finally, in [Audet et al., 2008], the MADS algorithm is used to optimize the treatment

process of spent potliners in the production of aluminum. The problem is formalized as a 7−dimensional non-linear blackbox problem with 4 inequality constraints. In particular, three strategies are compared using absolute displacements, relative displacements and the latter with a global Latin hypercube sampling search. They show that the use of scaling is particularly beneficial on the considered chemical application.

4.3 The variants of OrthoMads

MADS, described in Section 2.3.2, is an iterative direct local search method that relies on a mesh and proceeds in two phases that are the search and the poll. It has two main instantiations called ORTHOMADS and LTMADS, the latter being the first developed. Both variants are implemented in the NOMAD software but as ORTHOMADS is to be preferred for its coverage property in the variable space, it was used for the experiments of this thesis with NOMAD version 3.9.1.

The NOMAD implementation of ORTHOMADS provides 6 variants of the algorithm according to the number of directions used in the poll or according to the way that the last poll direction is computed. They are listed below.

`ORTHO N + 1 NEG` computes $n+1$ directions among which n are orthogonal and the $(n+1)^{th}$ direction is the opposite sum of the n first ones.

`ORTHO N + 1 UNI` computes $n+1$ directions among which n are orthogonal and the $(n+1)^{th}$ direction is generated from a uniform distribution.

`ORTHO N + 1 QUAD` computes $n+1$ directions among which n are orthogonal and the $(n+1)^{th}$ direction is generated from the minimization of a local quadratic model of the objective.

`ORTHO 2N` computes $2n$ directions that are orthogonal. More precisely each direction is orthogonal to $2n-2$ directions and collinear with the remaining one.

`ORTHO 1` uses only one direction in the poll.

`ORTHO 2` uses two opposite directions in the poll.

In the plots, the variants are respectively denoted by `Neg`, `Uni`, `Quad`, `2N`, `1` and `2`.

4.4 Test of the variants of OrthoMads

In this section, we try to identify potentially better direction types of ORTHOMADS and investigate the contribution of the search phase.

4.4.1 Experimental setup

In order to test the performance of the different variants of ORTHOMADS, the COCO platform was employed and, in particular, the `bbob` and `bbob-mixint` suites of problems were used. The continuous `bbob` testbed was described in Section 3.2.

The mixed-integer suite of problems `bbob-mixint` [Tušar et al., 2019] derives the `bbob` and `bbob-largescale` [Varelas et al., 2018] problems by imposing integer constraints on some variables. It consists of the 24 functions of `bbob` available in 15 instances and in dimensions 5, 10, 20, 40, 80 and 160. The target precisions defined for this suite are identical to those of `bbob`, that are 51 values between 10^{-8} and 10^{2}.

In both suites, only problems that have a dimension lower than or equal to 20 are used. This limit in the dimensions has two main reasons: the first one is the computational cost required for the experiments and the second one is that, with the perspective of solving real-world optimization problems, 20 is already a high dimension in this expensive blackbox context. Only the first five instances of each function were used, that is a total of respectively 600 and 360 problems used from `bbob` and `bbob-mixint`, respectively. A maximal function evaluation budget of $2 \times 10^3 \times n$ was set, with n being the dimension of the considered problem.

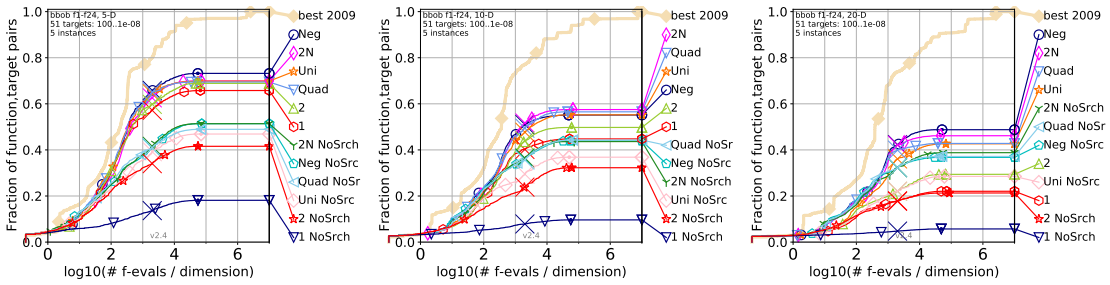
To see the contribution of the search phase, the experiments on the variants were divided in two subgroups: the first one using the default search of ORTHOMADS and the second one where the search phase is disabled. The latter is obtained by setting the four parameters `NM_SEARCH`, `VNS_SEARCH`, `SPECULATIVE_SEARCH` and `MODEL_SEARCH` of NOMAD to the value `no`. In the plots, the label *NoSrch* is used when the search is turned off. The search notably includes the use of a quadratic model and of the N-M method. The minimal mesh size was set to 10^{-11}.

Experiments were run with restarts allowed for unsolved problems when the evaluation budget is not reached. This may happen due to internal stopping criteria of the solvers. The initial points used are suggested by the method `initial_solution_proposal()` available in COCO.

4.4.2 Performance on continuous problems

As said previously, the contribution of the search phase was studied. The results aggregated on all functions in dimensions $5, 10$ and 20 on the `bbob` suite are depicted on Figure 4.1. They show that enabling the search step in NOMAD generally leads to an equivalent or higher performance of the variants and this improvement can be important. Besides, using one or two directions with or without search is often far from being competitive with the

other variants. In particular, `1 NoSrch` is often the worst or among the worsts, except on Discus which is an ill-conditioned quadratic function, where it competes with the variants that do not use the search. As mentioned in Section 4.1, the plots depicting the results described in this chapter are available online.



(a) 5D (b) 10D (c) 20D

Figure 4.1: ECDF plots: the variants of ORTHOMADS with and without the search step on the `bbob` problems. Results aggregated on all functions in dimensions 5, 10 and 20.

Looking at the results aggregated on all functions for `ORTHO 2N`, `ORTHO N + 1 NEG`, `ORTHO N + 1 QUAD` and `ORTHO N + 1 UNI`, the search increases the success rate from nearly 70%, 55% and 40% up to 90%, 80% and 65% respectively in dimensions 2, 3 and 5, as shown in Figure 4.1a for dimension 5. From dimension 10, the advantage of the search decreases and the performance of `ORTHO N + 1 UNI` visibly stands out from the other three variants mentioned above since it decreases with or without the search, as illustrated in Figures 4.1b and 4.1c.

Focusing on some families of functions, `Neg NoSrch` seems slightly less impacted than the other `NoSrch` variants by the increase of the dimension. On ill-conditioned problems, the variants using search are more sensitive to the increase of the dimension.

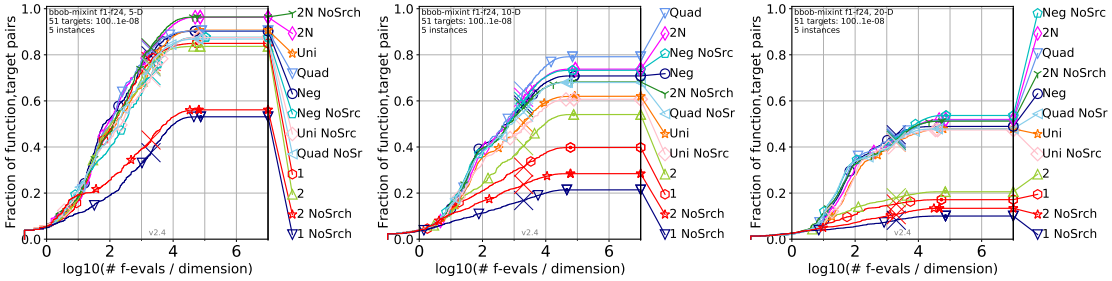
Considering multimodal functions with adequate global structure, `2N NoSrch` solves 15% more problems than the other `NoSrch` variants in 2D. In this dimension, the variants using search have a better success rate than the *best 2009* up to a budget of 200 function evaluations. From 10D, all curves are rather flat: all ORTHOMADS variants tend to a local optimum.

With increasing dimension, `Neg` is competitive or better than the others on multimodal problems without global structure, followed by `2N`. In particular, in dimension 20 both variants are competitive and outperform the remaining variants that use search on the Gallagher's Gaussian 101−me peaks function, and `Neg` outperforms them with a gap of more than 20% in their success rate on the Gallagher's Gaussian 21−hi peaks function which is also ill-conditioned.

Since `Neg` and `2N` are often among the best variants on the considered problems and have an advantage on some multimodal weakly structured functions, they are chosen for comparison with other solvers.

4.4.3 Performance on mixed-integer problems

The experiments performed on the mixed-integer problems also show a similar or improved performance of the ORTHOMADS variants when the search step is enabled in NOMAD, as illustrated in Figure 4.2 in dimensions 5, 10 and 20. Looking at Figure 4.2a for instance, in the given budget of $2 \cdot 10^3 \cdot n$, the variant denoted as `2` solves 75% of the problems in dimension 5 against 42% for `2 NoSrch`.



(a) 5D (b) 10D (c) 20D

Figure 4.2: ECDF plots: the variants of ORTHOMADS with and without the search step on the `bbob-mixint` problems. Results aggregated on all functions in dimensions 5, 10 and 20.

However, it is not always the case: the only use of the poll directions is sometimes favourable. It is notably the case on the Schwefel function in dimension 20 where the curve `Neg NoSrch` solves 43% of the problems, which is the highest success rate when the search and non-search settings are compared together.

When the search is disabled, `ORTHO 2N` seems preferable in small dimension, namely here in dimension 5 as presented in Figure 4.2a. In this dimension, it is sometimes the only variant that solves all the instances of a function in the given budget: it is the case for the step-ellipsoidal function, the two Rosenbrock functions (original and rotated), the Schaffer functions, and the Schwefel function. It also solves all the separable functions in dimension 5 and can therefore solve the different types of problems. Although the difference is less noticeable with the search step enabled, this variant is still a good choice, especially on multimodal problems with adequate global structure.

On the whole, looking at Figure 4.2, `ORTHO 1` and `ORTHO 2` solve less problems than the other variants and the gap in performance with the other direction types increases

with the dimension, whether using the search phase or not. Although the use of the search helps solving some functions in low dimension such as the sphere or linear slope functions in dimension 5, both variants perform poorly in dimension 20 on second-order separable functions, even if the search enables the solution of linear slope which is a linear function. Among these two variants, using 2 poll directions also seems better than only one, especially in dimension 10 where `ORTHO 2` solves more than 23% and 40% of problems respectively without and with use of search, against 16% and 31% for `ORTHO 1` as presented in Figure 4.2b.

Among the four remaining variants, `ORTHO N + 1 UNI` reaches equivalent or less targets than the others whether considering the setting where the search is available or when only the poll directions are used, as depicted in Figure 4.2. In particular, in dimension 5, the four variants using more than $n + 1$ poll directions solve more than 85% of the separable problems with or without search. But when the dimension increases, `ORTHO N + 1 UNI` has a disadvantage on the Rastrigin functions where the use of the search does not noticeably help the convergence of the algorithm.

Focusing on the different function types, no algorithm among the variants `ORTHO 2N`, `ORTHO N + 1 NEG` and `ORTHO N + 1 QUAD` seem to particularly outperform the others in dimensions 10 and 20. A higher success rate is however noticeable on multimodal weakly structured problems with search available for `ORTHO N + 1 NEG` in comparison with `ORTHO N + 1 QUAD` and for the latter in comparison with `ORTHO 2N`. Besides, `Neg` reaches more targets on problems with low or moderate conditioning. For these reasons, `ORTHO N + 1 NEG` was chosen for comparison with other solvers. Besides, the mentioned slight advantage of `ORTHO N + 1 QUAD` over `ORTHO 2N`, its equivalent or better performance on separable and ill-conditioned functions compared with the latter variant, makes it a good second choice to represent ORTHOMADS.

4.5 Comparison of OrthoMads with other solvers

The previous experiments showed the advantage of using the search step in ORTHOMADS to speed up convergence. They also revealed the effectiveness of some variants that are used here for comparisons with other algorithms on the continuous and mixed-integer suites.

4.5.1 Compared algorithms

Apart from ORTHOMADS, the other algorithms used for comparison on `bbob` are first, three deterministic algorithms: the quasi-Newton BFGS method, the quadratic model-based NEWUOA and the adaptive NM method. Stochastic methods are also used among which

a random search (RS) algorithm [Brooks, 1958] and three population-based algorithms: a surrogate-assisted CMA-ES, DE and PSO.

In order to perform algorithm comparisons on `bbob-mixint`, data from four stochastic methods were collected: RS, the mixed-integer variant of CMA-ES, DE and the tree-structured parzen estimator (TPE) [Bergstra et al., 2011] that is a stochastic model-based technique.

NEWUOA is the Powell's model-based algorithm for DFO. It is a trust-region method that uses sequential quadratic interpolation models to solve unconstrained derivative-free problems.

RS is a stochastic iterative method that performs a random selection of candidates: at each iteration, a random point is sampled and the best between this trial point and the incumbent is kept.

TPE is an iterative model-based method for hyperparameter optimization. It sequentially builds a probabilistic model from already evaluated hyperparameters sets in order to suggest a new set of hyperparameters to evaluate on a score function that is to be minimized.

BFGS, the adaptive NM method, CMA-ES, DE and PSO were described in Chapter 2.

4.5.2 Parameter setting

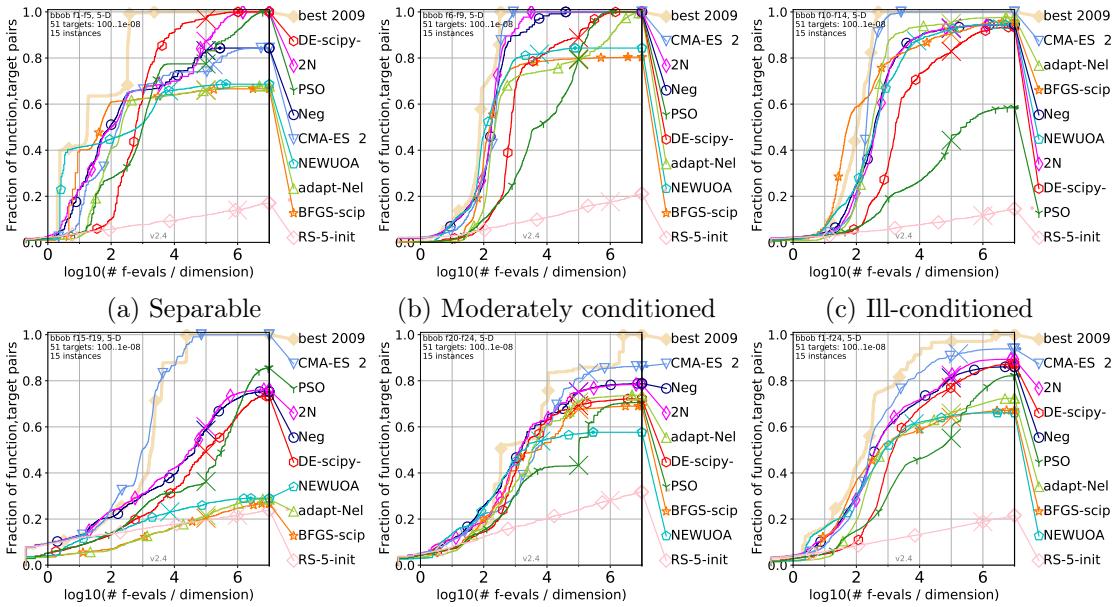
To compare the considered best variants of ORTHOMADS with other methods, the 15 instances of each function were used and the maximal function evaluation budget was increased to $10^5 \cdot n$, with n being the dimension.

For the `bbob` problems, the data used for BFGS, DE and the adaptive N-M method comes from the experiments of [Varelas and Dahito, 2019]. CMA-ES was tested in [Hansen, 2019], the data of NEWUOA is from [Ros, 2009], the one of PSO is from [El-Abd and Kamel, 2009] and RS results come from [Brockhoff and Hansen, 2019]. The comparison data of CMA-ES, DE, RS and TPE used on the `bbob-mixint` suite comes from the experiments of [Tušar et al., 2019]. All are accessible from the data archives of COCO with the `cocopp.archives.bbob` and `cocopp.archives.bbob_mixint` methods.

Performance on continuous problems

Figures 4.3 and 4.4 show the ECDF plots comparing the methods on the different function types and on all functions, respectively in dimensions 5 and 20 on the continuous suite. Compared with BFGS, CMA-ES, DE, the adaptive N-M method, NEWUOA, PSO and RS, ORTHOMADS often performs in the average for medium and high dimensions. For

small dimensions 2 and 3, it is however among the most competitive.



(a) Separable (b) Moderately conditioned (c) Ill-conditioned

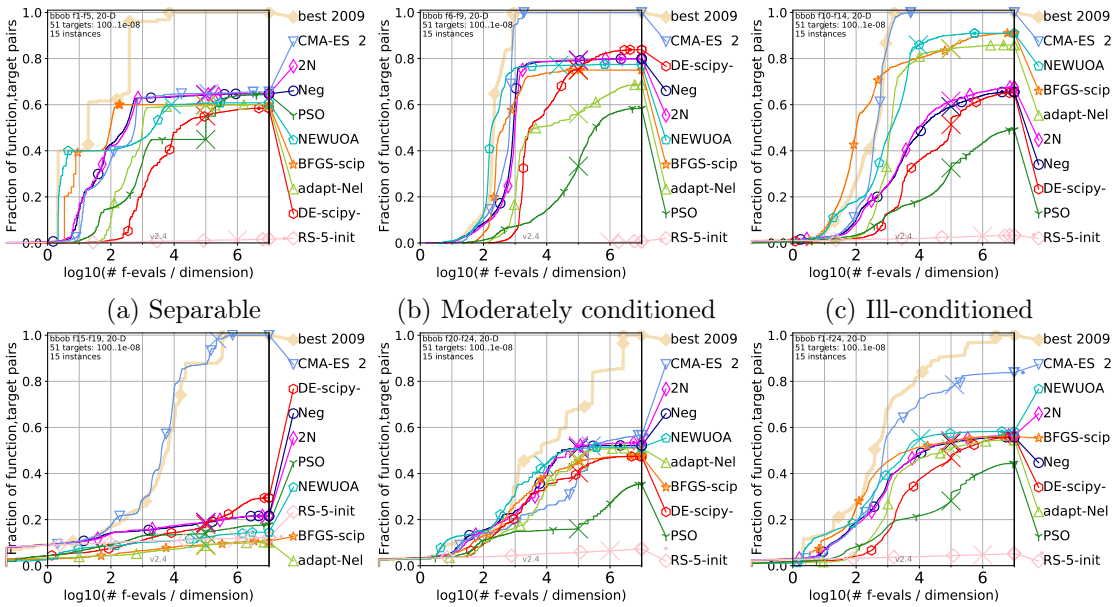
(d) Multimodal (e) Weakly structured (f) All functions

Figure 4.3: ECDF plots: comparison of the two variants `ORTHO 2N` and `ORTHO N+1 NEG` of ORTHOMADS with BFGS, NEWUOA, adaptive N-M, RS, CMA-ES, DE and PSO on the `bbob` problems. Results aggregated on the function types and on all functions in dimension 5.

Considering the results aggregated on all functions and splitting them over all targets according to the function evaluations, they can be divided in three parts. The first one consists of very limited budgets (about $20 \cdot n$) where NEWUOA competes with or outperforms the others. After that, BFGS becomes the best for an average budget and CMA-ES outperforms the latter for high evaluation budgets (above the order of $10^2 \cdot n$), as shown in Figures 4.3f and 4.4f. The obtained performance restricted to a low budget is an important feature relevant to many applications for which each function evaluation may last hours or even days.

On multimodal problems with adequate structure, there is a noticeable gap between the performance of CMA-ES, which is the best algorithm on this kind of problems, and the other algorithms as shown by Figures 4.3d and 4.4d. ORTHOMADS performs the best in the remaining methods and competes with CMA-ES for low budgets. It is even the best method up to a budget of $10^3 \cdot n$ in dimensions 2 and 3 while it competes with CMA-ES in higher dimensions for budgets lower than the order of $10^2 \cdot n$.

RS is often the worse algorithm to use on the considered problems.



(a) Separable (b) Moderately conditioned (c) Ill-conditioned

(d) Multimodal (e) Weakly structured (f) All functions

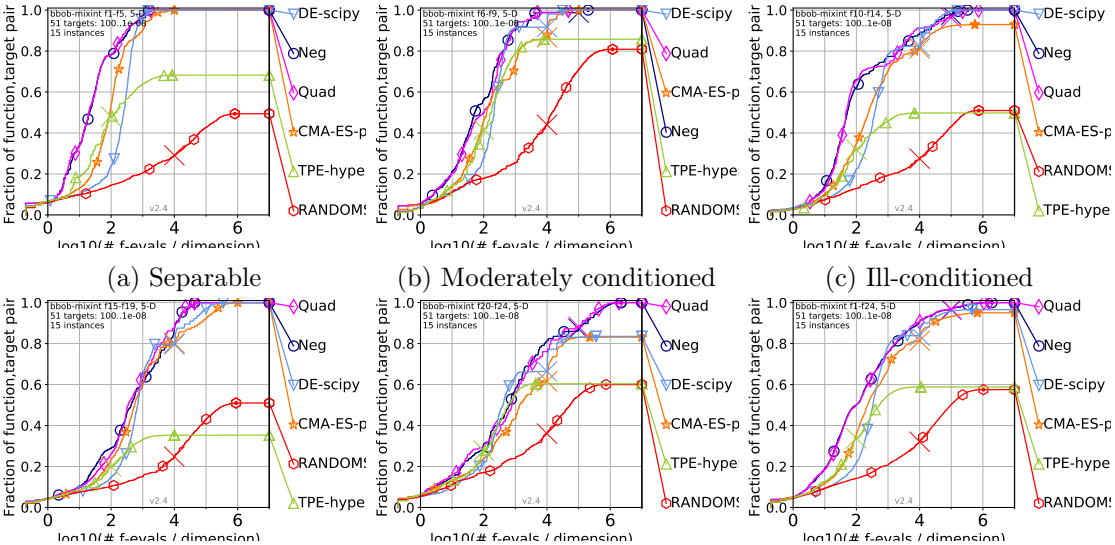
Figure 4.4: ECDF plots: comparison of the two variants `ORTHO 2N` and `ORTHO N + 1 NEG` of ORTHOMADS with BFGS, NEWUOA, adaptive N-M, RS, CMA-ES, DE and PSO on the `bbob` problems. Results aggregated on the function types and on all functions in dimension 20.

4.5.3 Performance on mixed-integer problems

Figures 4.5 and 4.6 show the ECDF plots comparing the methods on the different function types and on all functions, respectively in dimensions 5 and 20 on the mixed-integer suite. The comparisons of `NEG` and `QUAD` with CMA-ES, DE, RS and TPE show an overall advantage of these ORTHOMADS variants over the other methods. A gap is especially visible on separable and ill-conditioned problems, respectively depicted in Figures 4.5a and 4.6a and Figures 4.5c and 4.6c in dimensions 5 and 20, but also on moderately conditioned problems as shown in Figures 4.5b and 4.6b in dimensions 5 and 20.

On multimodal problems with global structure, ORTHOMADS is to prefer only in small dimensions: from dimension 10 its performance highly deteriorates and CMA-ES and DE seem to be better choices. On multimodal weakly structured functions, the advantages of ORTHOMADS compared to the others emerge when the dimension increases.

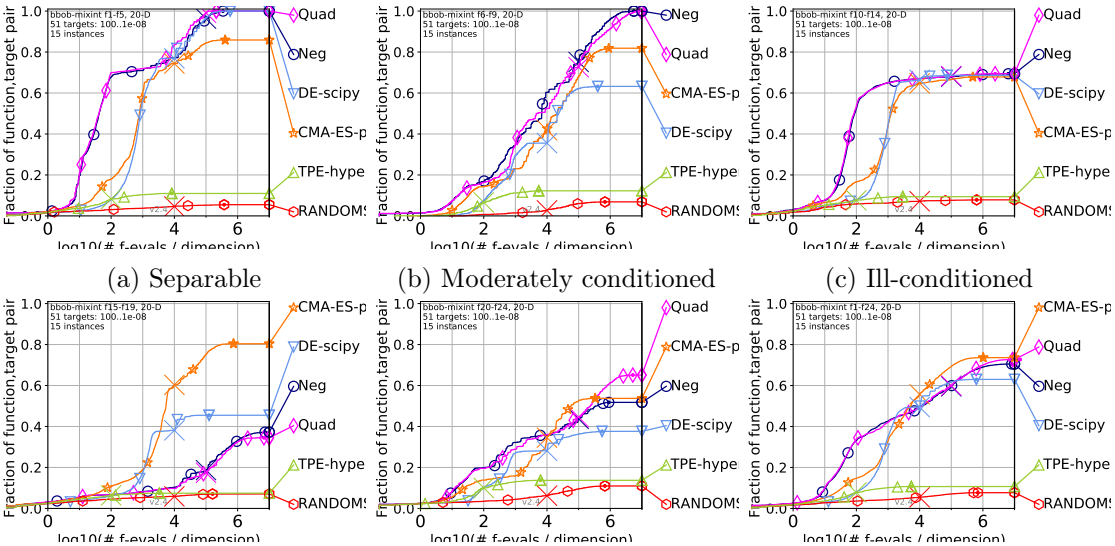
Although the performance of all methods decreases with increasing dimensions, the ORTHOMADS algorithm seems less sensitive to that. For instance, for a budget of $10^2 \cdot n$, it reaches 15% more targets than CMA-ES and TPE that are the second best algorithms for this budget. In dimension 20, this gap increases to 18% for CMA-ES and 25% for TPE.



(a) Separable (b) Moderately conditioned (c) Ill-conditioned

(d) Multimodal (e) Weakly structured (f) All functions

Figure 4.5: ECDF plots: comparison of the two variants `ORTHO N+1 NEG` and `ORTHO N+1 QUAD` of ORTHOMADS with RS, CMA-ES, DE and TPE on the `bbob-mixint` problems. Results aggregated on the function types and on all functions in dimension 5.



(a) Separable (b) Moderately conditioned (c) Ill-conditioned

(d) Multimodal (e) Weakly structured (f) All functions

Figure 4.6: ECDF plots: comparison of the two variants `ORTHO N+1 NEG` and `ORTHO N+1 QUAD` of ORTHOMADS with RS, CMA-ES, DE and TPE on the `bbob-mixint` problems. Results aggregated on the function types and on all functions in dimension 20.

On the overall picture, presented in Figures 4.5f and 4.6f, RS performs poorly. The budget allocated to TPE, which is only $10^2 \cdot n$, is way smaller than the ones allocated to the other methods. In this limited budget, TPE competes with CMA-ES in dimension 5 and is better or competitive with DE in dimensions 10 and 20. The latter competes with ORTHOMADS after a budget in the order of $10^3 \cdot n$. Thus, after $5 \cdot 10^3$ function evaluations, only DE competes with ORTHOMADS in dimension 5 where both methods reach 70% of function-target pairs. Finally, CMA-ES competes with ORTHOMADS when the budget approaches $10^4 \cdot n$ function evaluations. Hence, restricted budgets seem to favour the direct local search method while expensive budgets favour the evolutionary algorithms CMA-ES and DE.

4.6 Final remarks

This chapter investigates the performance of the different poll direction types available in ORTHOMADS on continuous and mixed-integer problems from the literature in a blackbox context. On these two types of problems, `ORTHO N + 1 NEG` competes with or outperforms the other variants of the algorithm whereas using only 1 or 2 directions is often far from being competitive.

On the continuous functions considered, the best poll direction types identified are `ORTHO N + 1 NEG` and `ORTHO 2N`, especially on multimodal weakly structured problems. ORTHOMADS is advantageous in small dimensions and achieves mean results for medium and high dimensions compared to the other algorithms. It also performs well on multimodal problems with global structure where it competes with CMA-ES for limited budgets.

For very limited budgets, the trust-region method NEWUOA is favourable on continuous problems, followed by the linesearch method BFGS for a medium budget and finally the evolutionary algorithm CMA-ES for a high budget.

The results on the mixed-integer suite show that, among the poll direction types, `ORTHO 2N` is preferable in small dimension. Otherwise, `ORTHO N + 1 NEG` and `ORTHO N + 1 QUAD` are among the best direction types. Comparing them to other methods show that ORTHOMADS often outperforms the compared algorithms and seems more resilient to the increase of the dimension. For limited budgets, ORTHOMADS seems a good choice among the other considered algorithms to solve unconstrained mixed-integer blackbox problems. This is notably interesting regarding real-world application problems and, in particular, the mixed-integer optimization problems of Stellantis, where the number of allowed blackbox evaluations is often limited to a few hundreds. In the latter case, the variables are typically the thicknesses of the sheet metals, considered as continuous, and the materials that are

categorical variables encoded as integers.

Finally, studying the contribution of the search step of ORTHOMADS shows that disabling it generally leads to a deteriorated performance of the algorithm. Indeed, the default search sequentially executes a N-M search and a quadratic model search that enable a global exploration and accelerate the convergence. However, this effect softens when the dimension increases.

5

Design of a finite element test case

5.1 Motivation

In view of the important running times of the FE models used in the automotive industry, the design of a simple test case was considered. The aim is to possess a FE blackbox implementation enabling quick benchmarking of algorithms and comparisons within reasonable times.

In order to be an interesting instance regarding real-world optimization problems targeted, the model should possess some specific features. First, the number of variables should be large enough for the automotive context. Besides, it must compute outputs that are usable for the definition of a constrained blackbox optimization problem, that are objective and constraints. Moreover, to respond to the original purpose of its design, the FE model is expected to be cheap to compute compared to the expensive real-world FE simulations. Finally, the presence of different types of variables would be an asset for the introduction of new optimization levers.

5.2 Description of the test case

5.2.1 The truss structure

The test case designed is a truss structure depicted in Figure 5.1. It consists of thirteen bars (numbered from 1 to 13) of square sections, connected by eight nodes (numbered from 1 to 8). Let x and y denote respectively the abscissa and ordinate axes of a Cartesian coordinate system, each node has two degrees of freedom corresponding to the displacements in the x and y axes. Each bar is implemented as an element of the FE model. Horizontal and vertical bars (i.e. elements 1, 2, 3, 4, 6, 7, 9, 11 and 13) have a length of 0.25m while the length of diagonal elements (5, 8, 10 and 12) is $0.25 \cdot \sqrt{2}$m. The truss is subject to a vertical force $F_{y3} = 10^4$N on node 3 while the extreme nodes 1 and 5 are clamped.

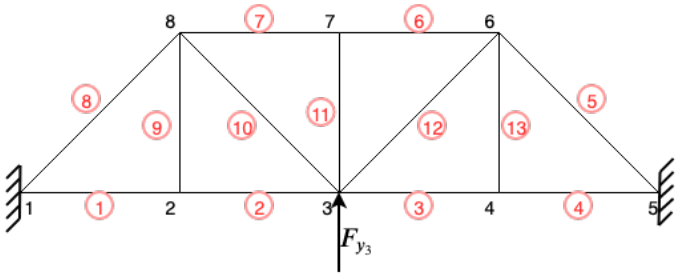


Figure 5.1: Truss structure subject to a vertical load on node 3 while nodes 1 and 5 are clamped.

5.2.2 Considered optimization problems

The section width of each element is an optimization variable, as well as the material. While the width can be chosen continuously between 0.01m and 0.05m, the available materials for each bar are chosen among aluminum, magnesium, steel and titanium. They are represented either as categorical variables represented in $\{1, .., 4\}$ or as discrete variables through the corresponding Young's modulus. The latter is a coefficient characterizing the elasticity of a material.

The ordering of the categorical variables was done so that there is no apparent relationship between them, and the Young's moduli were chosen for the discrete setting because they are the only essential material characterization that intervene in the FE code and they reflect differences in the mechanical behaviours of the materials.

Three quantities are chosen as possible objectives for the minimization problems on the structure: the weight, the cost and the compliance. Thus, the structure is expected to be as light, cheap and stiff as possible.

To compute them, in addition to the Young's moduli, the costs per weight and the densities are needed. Their values were chosen using the mineral info[1] website and Wikipedia[2]. The Young's moduli, the mass costs and the densities are described in Table 5.1.

	Young's modulus (10^6Pa)	Mass cost (€/kg)	Density (10^3 kg/m^3)
Aluminum	69	1.46	2.70
Magnesium	45	1.88	1.75
Steel	200	0.50	7.85
Titanium	105	3.60	4.50

Table 5.1: Young's moduli, mass costs and densities of the materials of the truss.

The optimization variable is denoted by z in this chapter to avoid any confusion with the abscissa axis. As the variables of each problem are the materials and the thicknesses of the elements, the dimension is then 26, that is two variables per element. The optimization variables can be depicted as $z = [m_1, m_2, \dots, m_{13}, t_1, t_2, \dots, t_{13}]$, with m_i the discrete value (including in the categorical setting) representing the material of element i, and t_i the thickness of element i with $i \in \{1, .., 13\}$. The displacement of node 3, computed from the FE code, is used as a constraint. The corresponding limit value comes from the displacements engendered when the structure is entirely made up of aluminum and when all elements are given the allowed minimum thickness of 0.01m. The optimization problems can be expressed as follows:

$$\begin{aligned} \min_{z \in \mathbb{R}^{26}} \quad & \text{weight}(z) \quad z = [m_1, \dots, m_{13}, t_1, \dots, t_{13}] \\ \textit{or} \quad & \text{cost}(z) \\ \textit{or} \quad & \text{compliance}(z) \\ \text{subject to} \quad & \begin{cases} |u_{y3}(z)| \leq u_{y3,\max} \\ z_{1:13} \in \{\text{Aluminum, Magnesium, Titanium, Steel}\} \\ z_{14:26} \in [0.01, 0.05] \text{ (m)}, \end{cases} \end{aligned} \tag{5.1}$$

where $u_{y3}(z)$ stands for the displacement of node 3 in the y axis. Let j and k be two integer values in $\{1, \dots, 13\}$ such that $j \leq k$, the notation $z_{j:k}$ refers to the component variables of z from index j to k included.

[1] https://www.mineralinfo.fr
[2] https://fr.wikipedia.org/wiki/Masse_volumique

The analytical formulas for the objectives are defined as follows:

$$\begin{aligned} \text{Cost} &= \sum_{i=1}^{13} C_i \times \rho_i \times (t_i)^2 \times L_i \\ \text{Weight} &= \sum_{i=1}^{13} \rho_i \times (t_i)^2 \times L_i \\ \text{Compliance} &= F^{\mathrm{T}} U = F_{y3} \times u_{y3}, \end{aligned}$$

where C_i, ρ_i, t_i and L_i are respectively the mass cost, the density, the thickness and the length of element $i \in \{1, \ldots, 13\}$, $F = [F_{x1}, F_{y1}, \ldots, F_{x8}, F_{y8}]^T$ is the vector of applied forces in the x and y axes at every node and consists of zeros everywhere except for the coordinate of F_{y3} and $U = [u_{x1}, u_{y1}, \ldots, u_{x8}, u_{y8}]^T$ stands for the vector of displacements along x and y at every node.

It is well noted that although the formulations of the cost and weight are known, they are treated as blackbox objectives. Moreover, F and U are computed from the FE simulation of the truss so the compliance and the displacement constraint are real blackbox functions. Computing theses quantities is computationally cheap as it takes only a fraction of a second. The code is available at: https://github.com/DahitoMA/FE_TRUSS.

5.3 Preliminary numerical experiments

5.3.1 Considered algorithms and setups

Once the simple model is created and the optimization problems defined, the next goal is to select and test some existing DFO algorithms in order to compare their performance and identify their strengths and weaknesses for the purpose of developing the desired algorithm.

To do this, NSGA-II, CMA-ES and MADS have been selected. NSGA-II was chosen as a reference benchmark tool since it is one of the best-known GAs. The choice of CMA-ES comes from the fact that it is the state-of-the-art EA. Finally, MADS was selected as a popular direct search algorithm.

The Platypus library[3] version 1.0.2 implemented in Python has been used to test NSGA-II, the version 3.0.4 of CMA-ES comes from the Python Git repository[4] and MADS was tested using the NOMAD software[5] version 3.9.1. In particular, the ORTHOMADS instantiation of MADS was used with the direction type set to `ORTHO N+1 NEG`.

[3] https://github.com/Project-Platypus/Platypus
[4] https://github.com/CMA-ES/pycma
[5] https://github.com/bbopt/nomad

While the handling of nonlinear inequality constraints is inner to NSGA-II and MADS, a penalty method has been used for CMA-ES. Let $c(z) = \max(0, g(z))$ be the constraint violation with $g(z) \leq 0$ the nonlinear inequality constraint (i.e. the vertical displacement constraint). If f denotes the objective function, the penalized function $\bar{f}$ to minimize has been chosen of the form: $\bar{f}(z) = f(z) + \alpha \cdot c(z)^2$, with $\alpha \in \mathbb{R}$. In order to choose the value of α, several numerical experiments were performed. Eventually, $\alpha = 10^9$ has been set. Experimentally noticing that the constraint violations were lower than 10^{-2}, for numerical reasons, $\bar{f}$ was coded as follows: $\bar{f}(z) = f(z) + 10^5 \cdot (c(z) \cdot 10^2)^2$.

Since NSGA-II and CMA-ES are stochastic meta-heuristics, there is a variance between two successive runs of the same algorithm even starting from the same initial populations. Consequently, for each experiment, 20 runs were launched for both methods to better evaluate their performance. Since the setting of MADS is deterministic in our experiments, with the seed set at 0, only one run of this method is performed.

In what follows we report on preliminary computational studies related to the implementation of evolutionary algorithms, summarizing the main outcomes.

5.3.2 Testing initial populations in NSGA-II and CMA-ES

Let $n_{pop} \in \mathbb{N}^*$ be the size of the population of the EA considered. In NSGA-II, the algorithm usually directly starts with n_{pop} individuals while in CMA-ES only one point, which is the mean of the initial population, and the covariance matrix are commonly given and the algorithm generates itself the initial population. The initialization setup of the population can influence the convergence of an EA.

Experiments on cost minimization were performed to see the importance of the initialization. To do this, both methods were tested in their common initialization procedures and with that of the other method. Indeed, NSGA-II and CMA-ES were tested with initial populations chosen uniformly as random in the admissible domain but also with initial populations generated by CMA-ES from a given mean chosen uniformly as random.

In both cases, the population size is set to 26, that is the number of variables, and 1000 iterations were performed for each run. Materials are treated as integer variables represented by their Young's moduli in NSGA-II and as real variables brought back to the closest feasible values at each iteration in CMA-ES.

The evolution of the best costs according to the evaluations in both initialization settings are depicted in Figure 5.2 and Figure 5.3 for NSGA-II and CMA-ES, respectively. Left figures focus on the medians of the runs while right ones highlight the quartiles of the runs. The best candidate is defined as the point minimizing the constraint. If there exist several of them, it is the one with the smallest objective value.

While giving the population generated by CMA-ES to NSGA-II disadvantages the latter, it has no or few effect to give n_{pop} random initial points to CMA-ES. Indeed, after the first phase where NSGA-II looks for a feasible solution, the decreasing phase observed on the median plots starts after around 300 function evaluations when using the population generated by CMA-ES, that is 10 times greater than when starting the algorithm with a random initial population. Besides, it increases the variance between runs. An explanation is that the population generated by CMA-ES is less diverse and this seems important in NSGA-II even with the ranking based on the crowding distance.

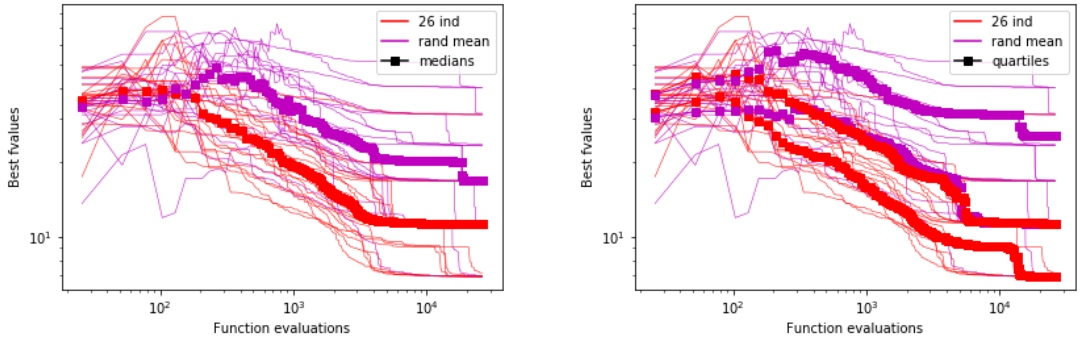


Figure 5.2: Best costs according to the evaluations for 20 runs of NSGA-II from a random initial mean (violet) and with 26 initial individuals (red). The medians of the runs (left) and the quartiles (right) are represented with thick lines.

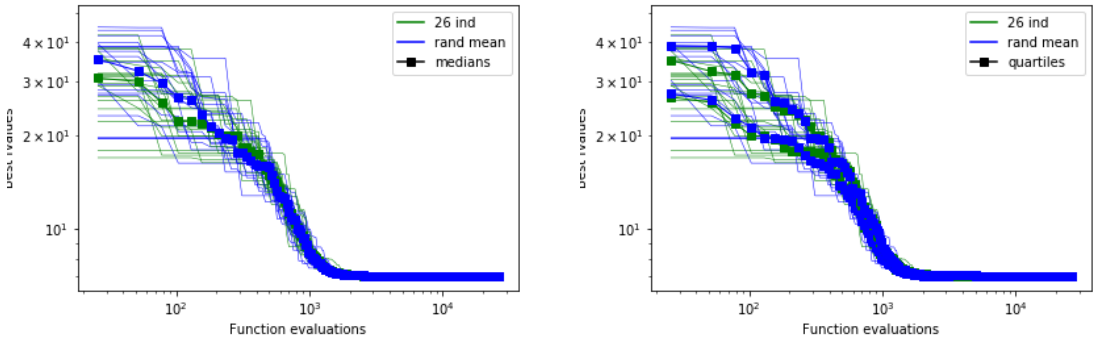


Figure 5.3: Best costs according to the evaluations for 20 runs of CMA-ES from a random initial mean (blue) and with 26 initial individuals (green). The medians of the runs (left) and the quartiles (right) are represented with thick lines.

5.3.3 Testing the encoding in NSGA-II

In NSGA-II, it can be specified whether the variables are real or integer and appropriate genetic operators are then used.

As materials can be differently represented and since their Young's modulii have big and heterogeneous gaps (in the order of 10^6Pa), it seemed interesting to look if NSGA-II behaves the same for different integer scales of this parameter. Here we test different encoding of materials to observe a potential discrepancy.

Experiments were launched with the Young's moduli either in $\{69 \cdot 10^6, 45 \cdot 10^6, 105 \cdot 10^6, 200 \cdot 10^6\}$, in $\{69, 45, 105, 200\}$ or in $\{1, 2, 3, 4\}$. The populations were initialized uniformly as random in the admissible set and with a size of 26. Besides, 1000 population generations are performed.

The evolution of the best costs according to function evaluations is depicted in Figure 5.4 for the 20 runs of each material setting tested. The medians and the quartiles of the runs are also represented.

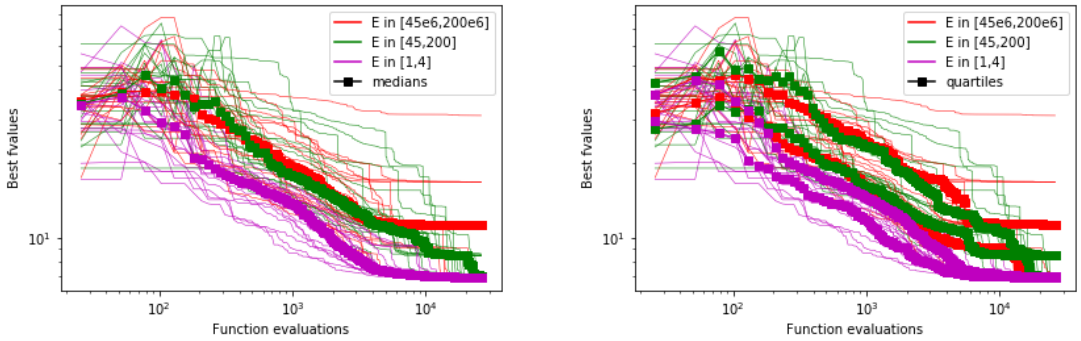


Figure 5.4: Best costs according to the evaluations for 20 runs of NSGA-II using Young's moduli as integers in $\{69 \cdot 10^6, 45 \cdot 10^6, 105 \cdot 10^6, 200 \cdot 10^6\}$ (red), in $\{69, 45, 105, 200\}$ (green) and in $\{1, 2, 3, 4\}$ (violet). The medians of the runs (left) and the quartiles (right) are represented with thick lines.

The observations are that, considering the original values of the Young's moduli or dividing them by 10^6 does not seem to have much effect.

The three variants have roughly the same convergence speed. The difference is rather in the first part of the graphs where NSGA-II is more or less fast to reach the feasible region. This induces a difference in the number of function evaluations.

Besides, using categorical variables for materials in NSGA-II decreases the variance between runs. Indeed, all the runs for this variant seem to converge to the same cost, which means that it reduces the probability to end up in different configurations from a run to another.

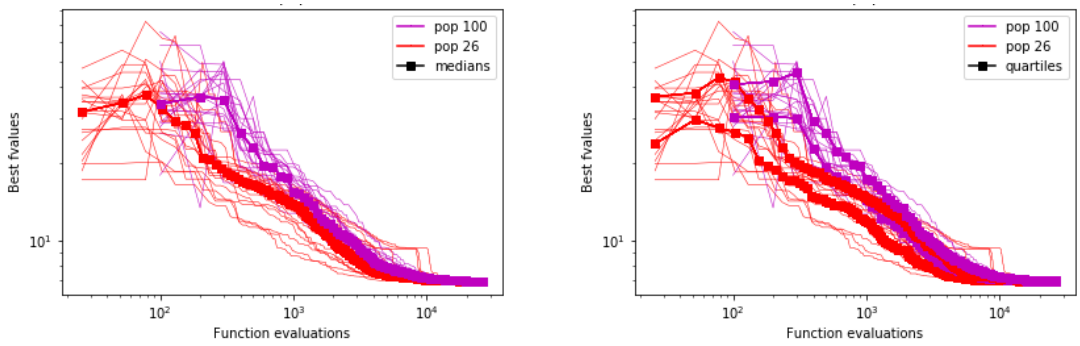


Figure 5.5: Best costs according to the evaluations for 20 runs of NSGA-II with $n_{pop} = 26$ (red) and and the default $n_{pop} = 100$ (violet). The medians of the runs (left) and the quartiles (right) are represented with thick lines.

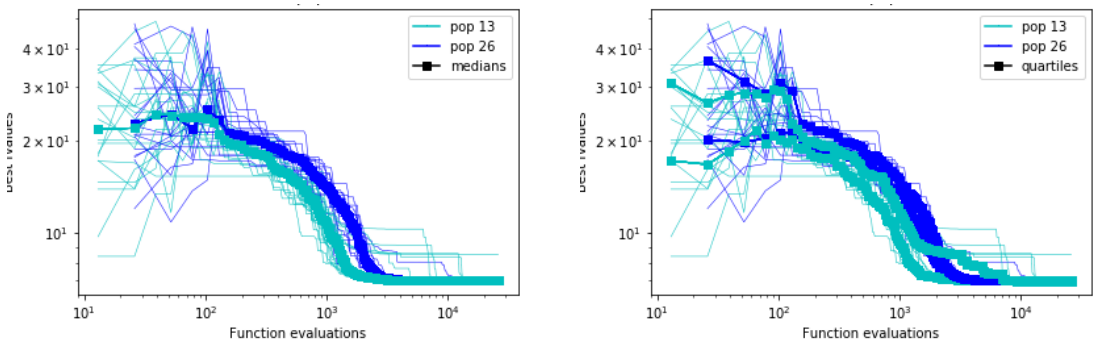


Figure 5.6: Best costs according to the evaluations for 20 runs of CMA-ES with $n_{pop} = 26$ (blue) and the default $n_{pop} = 13$ (light blue). The medians of the runs (left) and the quartiles (right) are represented with thick lines.

5.3.4 Testing the population size

Both NSGA-II and CMA-ES implementations have default population sizes. In NSGA-II, $n_{pop} = 100$ by default while it is equal to $(4 + \lfloor 3 \cdot \log(n) \rfloor)$ in CMA-ES, that is 13 for the considered problems. In these experiments, the default population size is tested for each method, as well as a size of 26 for the population, corresponding to the number of variables.

An initial mean is randomly selected for CMA-ES while a random population is given to NSGA-II. Materials are treated as categorical and 1000 population generations are performed for each of the 20 runs for each variant of an algorithm.

Figures 5.5 and 5.6 depict the evolution of the best costs according to the evaluations in both settings for NSGA-II and CMA-ES, respectively. A slight advantage is observed when NSGA-II uses populations of 26 individuals as the decrease in objective values starts with lower numbers of evaluations and the convergence occurs sooner. Regarding CMA-ES, the

medians of the runs show an advantage for the smallest population size. The discrepancy in runs is however greater when using 13 individuals and there is at least one run that does not converge to the same cost than the others. On the contrary, the quartiles are very close when the population size equals 26 and all runs seem to converge to the same value.

5.4 Numerical experiments

5.4.1 Parameter setting

In this section, NSGA-II, CMA-ES and MADS are used to optimize the three BBO problems defined in Equation (5.1), that are minimizing the cost, the weight and the compliance of the truss, all subject to a displacement constraint.

For each optimization, 20 runs are performed for the stochastic methods and 1 run for MADS. The type of variable can be specified in MADS, whether it is real, integer, categorical or even periodic. For integer variables, the mesh is adapted accordingly with a minimum mesh size of 1. For these experiments, materials are treated as unordered integers for all methods but the extended poll is not used in MADS. The reason of this choice is to avoid the definition of neighbours for each material and the fact that materials would be treated only in the extended poll of the algorithm. Moreover, in that manner, no assumption is done on the ordering of the materials.

The population size is set to 26 for the EAs with an initial mean and an initial population generated uniformly as random for CMA-ES and NSGA-II, respectively. MADS is initialized with a point also chosen uniformly as random in the admissible domain. The minimum mesh size of MADS is set to 10^{-11} and the direction type is still `ORTHO N+1 NEG`.

In these experiments, although the code is written in Python, an external system call to NOMAD is employed. As this slows down the resolution of the problems, a maximum of 4000 blackbox evaluations is set for all methods. The plots of MADS are presented only from evaluation 26 to be consistent with those of the EAs used.

5.4.2 Cost optimization

The results for the optimization of the cost of the truss are depicted in Figure 5.7 for the function values and Figure 5.8 presents the evolution of the best constraint violations. Comparing the two EAs, CMA-ES globally competes with NSGA-II and performs better than the latter at the end of the optimization as the median converges slightly faster. Observing the quartiles, there are less discrepancies in its runs compared to NSGA-II. This advantage for materials defined in $\{1, 2, 3, 4\}$ is however slight. Regarding MADS,

the algorithm outperforms the other methods in the considered evaluation budget and it competes with CMA-ES in the last thousand evaluations.

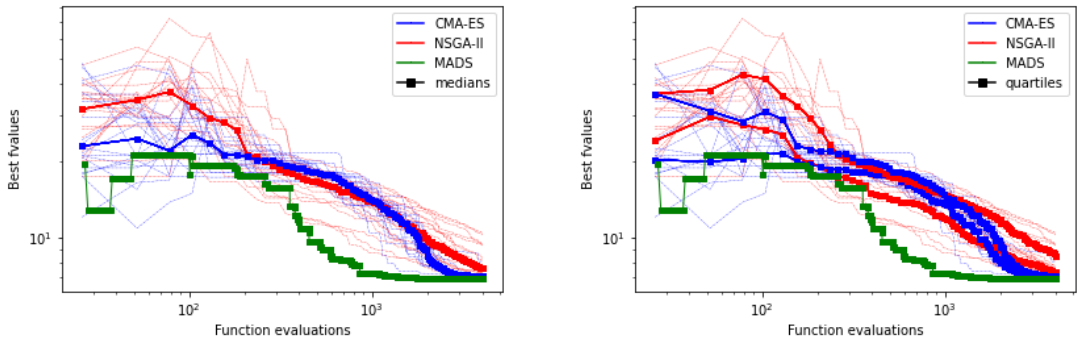


Figure 5.7: Best cost values according to the evaluations for 20 runs of NSGA-II and CMA-ES and 1 run of MADS. The medians of the runs (left) and the quartiles (right) are represented with thick lines.

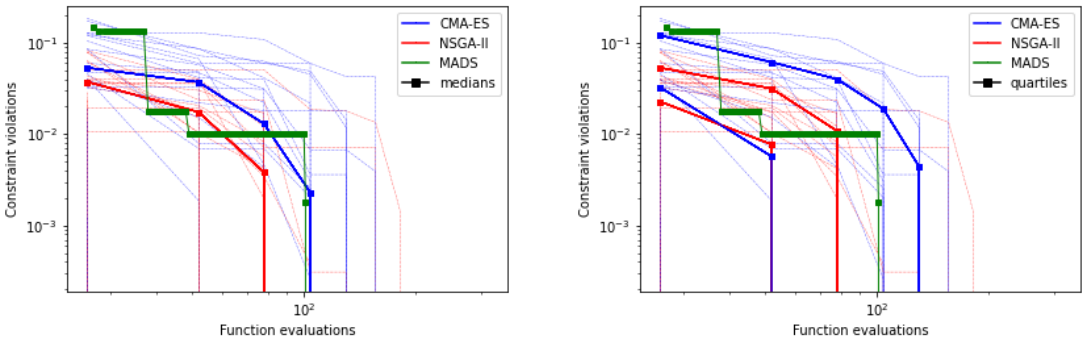


Figure 5.8: Best constraint violations in cost optimization according to the evaluations for 20 runs of NSGA-II and CMA-ES and 1 run of MADS. The medians of the runs (left) and the quartiles (right) are represented with thick lines.

5.4.3 Weight optimization

The results for weight optimization are presented in Figure 5.9 for the evolution of the best weights while Figure 5.10 shows the best constraint violations according to the evaluations.

CMA-ES and NSGA-II compete up to a budget of about 10^3 blackbox evaluations, with a slight advantage of NSGA-II. After this budget, CMA-ES globally converges to better solutions. MADS has the fastest convergence and outperforms the two EAs up to $2 \cdot 10^3$ evaluations. After this limit, CMA-ES achieves slightly better solutions.

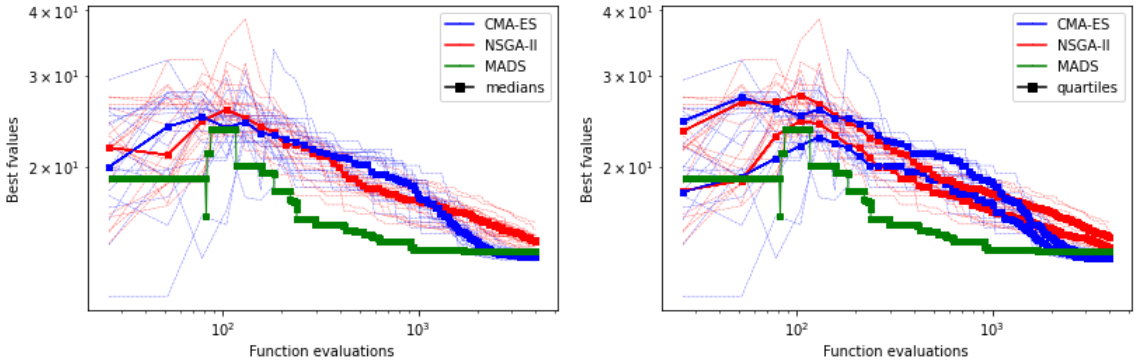


Figure 5.9: Best weight values according to the evaluations for 20 runs of NSGA-II and CMA-ES and 1 run of MADS. The medians of the runs (left) and the quartiles (right) are represented with thick lines.

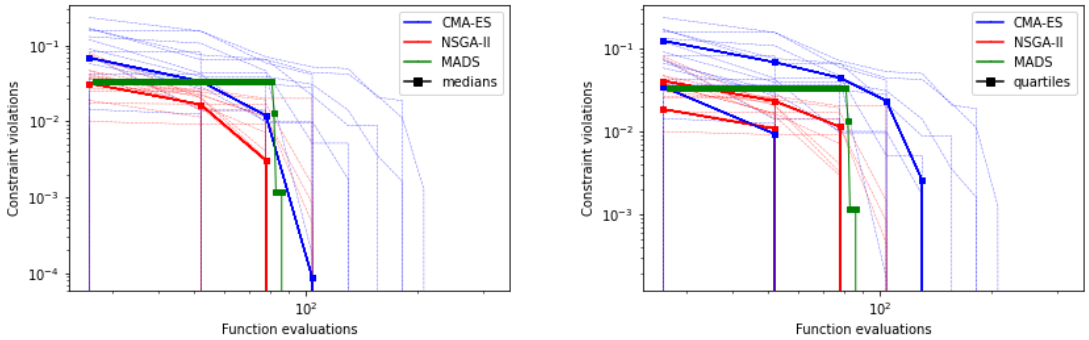


Figure 5.10: Best constraint violations in weight optimization according to the evaluations for 20 runs of NSGA-II and CMA-ES and 1 run of MADS. The medians of the runs (left) and the quartiles (right) are represented with thick lines.

5.4.4 Compliance optimization

The best compliance values according to the evaluations are presented in Figure 5.11 while Figure 5.12 plots the best constraint violations according to the evaluations. It can be noticed that the curve corresponding to MADS stops sooner than the others. This is due to the minimum mesh size stopping condition that was satisfied.

Here also, MADS performs better and has a faster convergence than CMA-ES and NSGA-II up to a budget of 10^3 blackbox evaluations and after that it competes with CMA-ES. The latter performs similarly than NSGA-II before converging to a solution. Due to the non-deterministic aspect, all runs of each EA used do not converge to the same solutions.

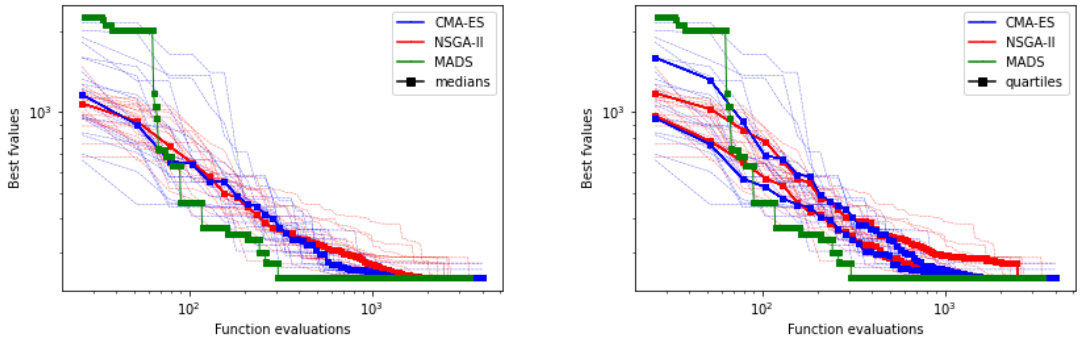


Figure 5.11: Best compliance values according to the evaluations for 20 runs of NSGA-II and CMA-ES and 1 run of MADS. The medians of the runs (left) and the quartiles (right) are represented with thick lines.

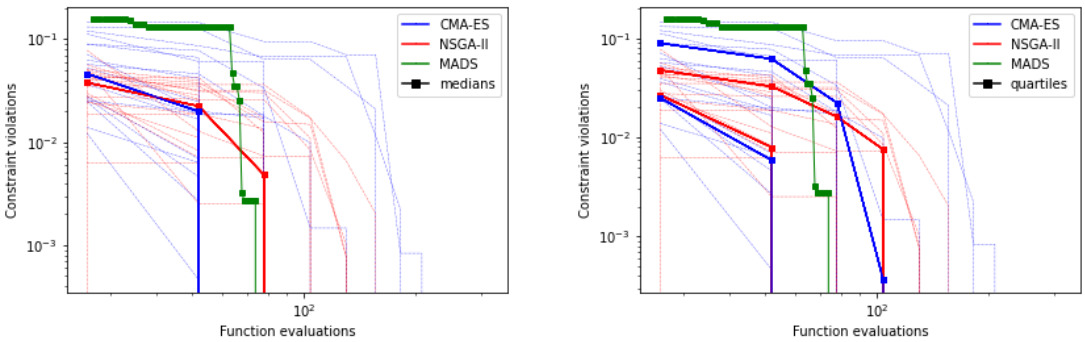


Figure 5.12: Best constraint violations in compliance optimization according to the evaluations for 20 runs of NSGA-II and CMA-ES and 1 run of MADS. The medians of the runs (left) and the quartiles (right) are represented with thick lines.

First, the minimization of the compliance may seem trivial. Unlike minimizing the global cost of the truss, which depends also on the weights, or minimizing the global weight that needs to take into account the displacement constraint, the minimum compliance should engender minimum displacement of the nodes and, thus, make the constraint inactive. For this optimization problem, we were expecting an all-steel configuration with the maximum thicknesses for all elements.

However, the optimal truss configuration found by MADS is different and depicted in Figure 5.13, where the colour of a bar indicates its material while the thickness of the line shows how thick it is. Indeed, the thickness range has been divided in four equal intervals to represent thin, medium thin, medium thick and thick bars. The minimum compliance within the evaluation budget is 241.421344N.m and corresponds to a structure

with mainly still and magnesium and an aluminum bar. The compliance value obtained is smaller than the compliance of an all-steel truss with the upper-bound thicknesses, that is 241.421356N.m.

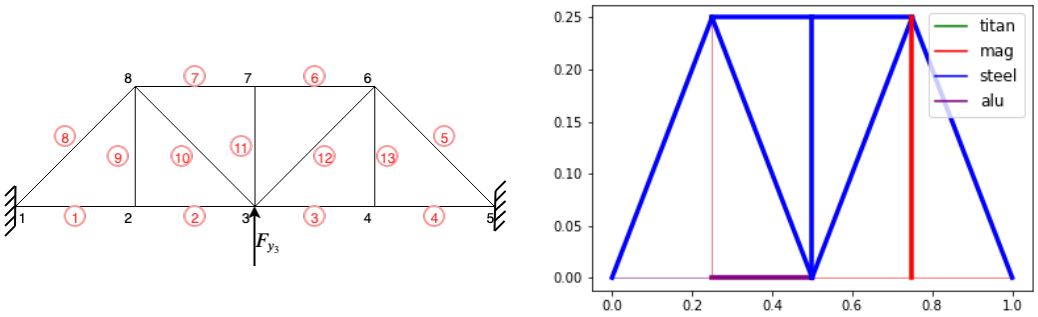


Figure 5.13: Load case (left) and solution of MADS for compliance optimization (right).

In fact, each FE bar that is not clamped can rotate freely at each node and move in the 2-dimensional plan defined by (x, y). For a clamped node, there is no degree of freedom. Besides, the elements of the truss do not receive any shearing stress so they can only be subject to normal stresses.

The formula of the compliance of the whole structure has also to be considered. As the vertical force applied on node 3 is constant, the compliance only depends on the displacement engendered at this node in the y axis. Considering these facts, when applying F_{y3}, the horizontal elements (1, 2, 3 and 4) cannot affect u_{y3}. The latter depends on how much element 11 can be retained. To do so, elements 5, 6, 7, 8, 10 , 11 and 12 have to be stiff enough, which explains why there are in steel with thick thicknesses. Actually, the materials of the other elements have little impact on the objective.

5.5 Final remarks

In this section, optimization problems minimizing the cost, the weight and the compliance of a FE truss structure were solved with NSGA-II, CMA-ES and MADS. The preliminary experiments conducted show that a diverse initial population is needed for NSGA-II whereas it seems to have very few effects on the performance of CMA-ES.

Furthermore, while the default population size leads to a slightly worse performance in NSGA-II, it enables a slightly faster convergence of CMA-ES but with a larger variance in the runs. Moreover, the categorical setting of the materials is preferable for NSGA-II.

Finally, considering a categorical setting of materials for the three methods, MADS has a faster convergence and finds the best solutions for compliance and cost optimizations.

CMA-ES competes with MADS after about $2{\cdot}10^3$ blackbox evaluations, which is huge for a BBO context. Moreover, CMA-ES globally converges sooner than NSGA-II and to better solutions.

6

Design of a surrogate-based method

6.1 Motivation

As this thesis deals with optimization problems that involve expensive computer simulations, the evaluation budgets for the resolutions are limited. Classical DFO approaches like direct search methods and EAs, in particular, require numerous blackbox calls to get considered good solutions. SBO is a promising optimization branch that enables to reduce the optimization cost by using surrogate models to choose the candidate points for the expensive evaluations.

Furthermore, most DFO methods are not designed for mixed variables and even less DFO algorithms deal with expensive BBO problems involving both mixed types of variables and general inequality constraints. Besides, the implementations of such algorithms are rarely available.

Considering these facts, the design of a new surrogate-based approach was considered to treat structural design optimization concerns raised by the multinational automotive manufacturing corporation Stellantis. They can be translated into a general form of constrained blackbox optimization problems. The work presented in this chapter is intended to be submitted to a journal and the corresponding code will be available.

6.2 Considered blackbox optimization problems

In this work, we consider mixed-variable constrained blackbox optimization problems involving both an objective and constraint functions that are computationally costly to evaluate. Such problems may be formulated as follows:

$$\begin{aligned} \min_{x \in \mathcal{X}} \quad & f(x) \\ s.t. \quad & g_j(x) \leq 0, \quad \forall j \in J, \\ & x_i \in \mathbb{R}, \quad \forall i \in C, \\ & x_i \in \mathbb{Z}, \quad \forall i \in I, \\ & x_i \in \mathcal{D}_i, \quad \forall i \in D, \end{aligned} \tag{P}$$

where

- (C, I, D) is a partition of the indices of the variables into the subsets of indices corresponding to the continuous, integer and discrete variables, respectively, such that $C \cup I \cup D \coloneqq \{1, 2, \ldots, n\}$ and n is the dimension of the problem,
- $\mathcal{D}_i$ is a finite set of ordered real values, for all $i \in D$,
- $\mathcal{X} \coloneqq \{x \in \mathbb{R}^n, l_i \leq x_i \leq u_i, \forall i \in C \cup I \cup D\}$ is the admissible subset, where l_i, u_i are the lower and upper bounds of each component x_i of x,
- $f : \mathcal{X} \to \mathbb{R}$ is the objective function to minimize,
- $g_j : \mathcal{X} \to \mathbb{R}$, $\forall j \in J$ are the constraint functions, with J a finite index set.

It is assumed that all or part of the functions f and $(g_j)_{j \in J}$ are blackbox functions and their evaluations at any given point demand important computational resources in terms of time and/or memory requirements. Besides, the constraints $(g_j)_{j \in J}$ are considered quantifiable and the presence of hidden constraints is envisaged.

Solving this kind of problems is more and more requested as it can have decisive impacts, for instance in reducing greenhouse gases emissions or improving system performance.

In what follows, by *one evaluation* at some given point x we mean the computation of the objective and all the constraints at this point. Besides, we assume that all the functions (objective and constraints) are deterministic and without noise.

6.3 Description of the proposed algorithm

This section presents a *Blackbox Optimization Algorithm*, called BOA, through its structure and main features.

6.3.1 The overall layout of BOA

The surrogate-based subproblems considered

As is usually the case in constrained BBO algorithms not requiring an initial feasible solution in the inputs, the proposed method is made up of two phases and its layout is described in Algorithm 6.

In the first phase, denoted by "Phase I", a feasible solution is sought. Differently from what is usually done, we minimize the sum of squared constraint violations added with a fraction of the original objective function. Let $\widehat{f}$ and $(\widehat{g}_j)_{j\in J}$ denote surrogate functions of the objective and constraint functions, respectively. In order to raise the chance of producing truly feasible points, the constraints $\widehat{g}_j(x) + \epsilon_j \leq 0$ are added. A minimum distance to evaluated points is also added to favour exploration. Given that the original functions in (P) are expensive, we work with their surrogates. Thus, the first phase aims at solving:

$$\begin{array}{lll} \min\limits_{x\in\mathcal{X}} & \sum_{j\in J} \max(0,\widehat{g}_j(x))^2 + \lambda\widehat{f}(x) & \\ s.t. & \widehat{g}_j(x) + \epsilon_j \leq 0, & \forall j \in J, \\ & d_{\min} - \min\limits_{y\in\mathcal{P}} \|x - y\| \leq 0, & \end{array} \qquad (OBJ^{\lambda,\epsilon}_{\texttt{PhaseI}})$$

where λ and $(\epsilon_j)_{j\in J}$ stand for some nonnegative scalar values, $d_{\min}$ is a positive value and $\mathcal{P}$ is the set of points that have been evaluated with the real blackbox functions f and $(g_j)_{j\in J}$. In our solution procedure, λ and $(\epsilon_j)_{j\in J}$ are iteratively updated inside each iteration so that the "leading" term of the objective in $(OBJ^{\lambda,\epsilon}_{\texttt{PhaseI}})$ is the first one, that is the sum of squared constraint violations. By adding a fraction of the original objective we aim at favouring the search of better feasible solutions, in terms of the original objective.

As long as the solution is predicted infeasible for the original blackbox problem and there is a considered reasonable decrease of the constraint violations predictions, this subproblem is iteratively solved with updated λ and $(\epsilon_j)_{j\in J}$. Otherwise, the solution is evaluated with the blackbox functions and used to update the surrogates. The whole process is repeated until a feasible solution is found.

The surrogate-based subproblem is solved with an external algorithm and finding a feasible point may not be guaranteed. In case the output solution does not respect the

constraints of ($OBJ^{\lambda,\epsilon}_{\texttt{PhaseI}}$), it is not evaluated and the problem is relaxed to find another solution to evaluate. The first relaxation consists in multiplying all $(\epsilon_j)_{j\in J}$ by -1. Indeed, the harder a constraint is and the bigger is the corresponding slack so, by taking the opposite, the corresponding surrogate constraint in ($OBJ^{\lambda,\epsilon}_{\texttt{PhaseI}}$) becomes easier to satisfy. Thus, the first relaxation corresponds to:

$$\begin{array}{lll} \min\limits_{x\in\mathcal{X}} & \sum_{j\in J}\max(0,\widehat{g}_j(x))^2+\lambda\widehat{f}(x) & \\ s.t. & \widehat{g}_j(x)-\epsilon_j\leq 0, & \forall j\in J, \\ & d_{\min}-\min\limits_{y\in\mathcal{P}}\|x-y\|\leq 0. & \end{array} \qquad (OBJ^{\lambda,-\epsilon}_{\texttt{PhaseI}})$$

In case the latter formulation still leads to an infeasible solution (with regards to the surrogate formulation), we focus on the distance criterion by keeping it as the only constraint. However, the constraints satisfaction is still considered through the objective of the subproblem. Hence, the second relaxation can be written as follows:

$$\begin{array}{ll} \min\limits_{x\in\mathcal{X}} & \sum_{j\in J}\max(0,\widehat{g}_j(x))^2+\lambda\widehat{f}(x) \\ s.t. & d_{\min}-\min\limits_{y\in\mathcal{P}}\|x-y\|\leq 0. \end{array} \qquad (OBJ^{\lambda})$$

As soon as a feasible solution of the original problem (P) is found, "Phase II" starts. In the latter, the goal is to improve the objective value of the best feasible solution. We aim at solving:

$$\begin{array}{lll} \min\limits_{x\in\mathcal{X}} & \widehat{f}(x) & \\ s.t. & \widehat{g}_j(x)+\epsilon_j\leq 0, & \forall j\in J, \\ & d_{\min}-\min\limits_{y\in\mathcal{P}}\|x-y\|\leq 0. & \end{array} \qquad (OBJ^{\epsilon}_{\texttt{PhaseII}})$$

Remark that the feasible region of this problem may be empty. Similarly to the first phase, ($OBJ^{\epsilon}_{\texttt{PhaseII}}$) is solved iteratively and relaxed in case its solution is infeasible. The first relaxation uses the opposite values of $(\epsilon_j)_{j\in J}$ and, thus, minimizes:

$$\begin{array}{lll} \min\limits_{x\in\mathcal{X}} & \widehat{f}(x) & \\ s.t. & \widehat{g}_j(x)-\epsilon_j\leq 0, & \forall j\in J, \\ & d_{\min}-\min\limits_{y\in\mathcal{P}}\|x-y\|\leq 0. & \end{array} \qquad (OBJ^{-\epsilon}_{\texttt{PhaseII}})$$

In case the solution is still infeasible, only the distance constraint is kept. However, in order to take into account the constraints of the original problem, the sum of squared constraint violations is added to the objective. This second relaxation of Phase II solves

(OBJ^{λ}) with λ equals 1: we denote this relaxation problem (OBJ^1). Unlike the first phase, $(\epsilon_j)_{j\in J}$ are updated only once per iteration and this is done with respect to the true constraints values (and not the surrogate predictions).

Algorithm 6 describes the general layout of BOA and calls Algorithms 7 and 8 that represent Phase I and Phase II, respectively. The latter update the slacks using Algorithm 9.

Algorithm 6: BOA

```
1  Initialize phase number: phase ← 1
2  Initialize slacks: ε_max > 0, ε_j ← 0, ∀j ∈ J
3  Initialize slack factors: σ_inc > 1, σ_dec ← 1/σ_inc, ρ ∈ (0,1), k_feas ∈ ℕ*
4  Initialize distance parameters: γ ≥ 0, Δ = {d_1, d_2, ..., d_|Δ|} ⊂ ℝ*_+, ν ∈ {1, 2, ..., |Δ|}
5  d_min ← max{γ, d_ν · min(u_i − l_i, ∀i ∈ C ∪ I ∪ D)}
6  Determine a set P_0 ⊂ X of p_0 (≤ N_max) points
7  Evaluate the functions f, (g_j)_{j∈J} at each point in P_0
8  Initialize x_best
9  Initialize h_best ← Σ_{j∈J} max(0, g_j(x_best))^2, P ← P_0
10 if max_{j∈J} g_j(x_best) ≤ 0 then
11     phase ← 2 // x_best is a feasible point in P
12     Update x_best with Algorithm 8 // BOA-Phase II
13 else
14     Update x_best, phase with Algorithm 7 // BOA-Phase I
15     if phase = 2 then
16         Update x_best with Algorithm 8 // BOA-Phase II
```

Parameters update

The slacks $(\epsilon_j)_{j\in J}$ are computed according to the predicted (in Phase I) or real constraints values (in Phase II). After $k_{\texttt{feas}} \in \mathbb{N}^*_+$ successive satisfactions of the constraint $j \in J$, the corresponding slack is decreased whereas it is increased as soon as the constraint is not satisfied, with respect to the real or predicted values depending on the phase. By waiting before decreasing a slack, we want to make sure that the satisfaction of $(\widehat{g}_j(x) + \epsilon_j)$ is "robust/reliable". The increase of ϵ_j is done such that its value is between a fraction of the violation of constraint j and an upper bound $\epsilon_{\max}$. The procedure is detailed in Algorithm 9. The construction of the surrogates is based on input and output values that are scaled in $[0, 1]$, which explains the truncation made on Line 6 of Algorithm 9. However, it should be noted that the surrogate prediction of a point that does not belong to the DOE is not guaranteed to be between 0 and 1.

Algorithm 7: BOA-Phase I

Given $phase$, $x_{\texttt{best}}$, $h_{\texttt{best}}$, $\mathcal{P}$, $N_{\max}$, Δ, ν, γ, $d_{\min}$, $\epsilon_{\max}$, $\epsilon_j, \forall j \in J$
Initialize $\kappa_j \leftarrow 0, \forall j \in J$, $threshold \in (0,1)$, $\lambda_0 \in (0,1)$
while $phase = 1$ *and* $|\mathcal{P}| < N_{\max}$ **do**
  From $\mathcal{P}$, build surrogate functions $\widehat{f}, (\widehat{g}_j)_{j\in J}$
  Initialize $dec\_lambda \leftarrow \texttt{true}$, $\lambda \leftarrow \lambda_0$
  $\widehat{x} \leftarrow$ Solve $\left(OBJ^{\lambda,\epsilon}_{\texttt{PhaseI}}\right)$ // Compute a solution $\widehat{x}$ of $(OBJ^{\lambda,\epsilon}_{\texttt{PhaseI}})$
  Evaluate $\widehat{f}(\widehat{x})$, $(\widehat{g}_j(\widehat{x}))_{j\in J}$
  $\widehat{h} \leftarrow \sum_{j\in J} \max(0, \widehat{g}_j(\widehat{x}))^2$
  /* If $\widehat{x}$ not feasible w.r.t. the surrogate constraints $\widehat{g}_j(\widehat{x}) \le 0$ for all $j \in J$, try to improve constraint satisfaction reducing λ and adjusting slacks $(\epsilon_j)_{j\in J}$ */
  while $\max_{j\in J} \widehat{g}_j(\widehat{x}) > 0$ *and* dec_lambda *is* $\texttt{true}$ **do**
    $\lambda \leftarrow \frac{1}{2}\min(\lambda, \max_{j\in J} \widehat{g}_j(\widehat{x}))$
    Update $\epsilon_j, \kappa_j, \forall j \in J$ with Algorithm 9 and the predicted constraints values $(\widehat{g}_j(\widehat{x}))_{j\in J}$
    $\widehat{x} \leftarrow$ Solve $\left(OBJ^{\lambda,\epsilon}_{\texttt{PhaseI}}\right)$
    Evaluate $\widehat{f}(\widehat{x})$, $(\widehat{g}_j(\widehat{x}))_{j\in J}$
    $\widehat{h}_{\texttt{old}} \leftarrow \widehat{h}$
    $\widehat{h} \leftarrow \sum_{j\in J} \max(0, \widehat{g}_j(\widehat{x}))^2$
    if $\widehat{h}_{\texttt{old}} - \widehat{h} < threshold$ **then**
      $dec\_lambda \leftarrow \texttt{false}$
  if $d_{\min} - \min_{y\in\mathcal{P}}(\|\widehat{x} - y\|) > 0$ *or* $\max_{j\in J}(\widehat{g}_j(x) + \epsilon_j) > 0$ **then**
    $\widehat{x} \leftarrow$ Solve $\left(OBJ^{\lambda,-\epsilon}_{\texttt{PhaseI}}\right)$ // Relax all ϵ_j in $-\epsilon_j$
    if $d_{\min} - \min_{y\in\mathcal{P}}(\|\widehat{x} - y\|) > 0$ *or* $\max_{j\in J}(\widehat{g}_j(x) + \epsilon_j) > 0$ **then**
      $\widehat{x} \leftarrow$ Solve (OBJ^{λ})
  Evaluate $f(\widehat{x})$, $(g_j(\widehat{x}))_{j\in J}$
  $h_{\texttt{old}} \leftarrow h_{\texttt{best}}$
  Update $x_{\texttt{best}}$
  $h_{\texttt{best}} \leftarrow \sum_{j\in J} \max(0, g_j(x_{\texttt{best}}))^2$, $\mathcal{P} \leftarrow \mathcal{P} \cup \{\widehat{x}\}$
  if $|\mathcal{P}| \ge 0.9 \cdot N_{\max}$ **then**
    if $h_{\texttt{best}} > 0.95 \cdot h_{\texttt{old}}$ **then**
      $d_{\min} \leftarrow \max\left(\gamma, \frac{1}{2}\cdot\min(d_1, d_{\min})\right)$
  else
    if $h_{\texttt{best}} > 0.95 \cdot h_{\texttt{old}}$ **then**
      $\nu \leftarrow \max(1, \nu - 1)$
    else
      $\nu \leftarrow \min(|\Delta|, \nu + 1)$
    $d_{\min} \leftarrow \max(\gamma, d_\nu \cdot \min(u_i - l_i, \forall i \in C \cup I \cup D))$
  if $\max_{j\in J} g_j(\widehat{x}) \le 0$ **then**
    $phase \leftarrow 2$ // $\widehat{x}$ is feasible

Algorithm 8: BOA-Phase II

Given $x_{\texttt{best}}$, $h_{\texttt{best}}$, $\mathcal{P}$, $N_{\max}$, Δ, ν, γ, $d_{\min}$, $\epsilon_{\max}$, $\epsilon_j, \forall j \in J$
Initialize $\kappa_j \leftarrow 0, \forall j \in J$
while $|\mathcal{P}| < N_{\max}$ **do**
  $f_{\texttt{old}} \leftarrow f(x_{\texttt{best}})$
  From $\mathcal{P}$, build surrogate functions $\widehat{f}, (\widehat{g}_j)_{j \in J}$
  $\widehat{x} \leftarrow$ Solve $(OBJ^{\epsilon}_{\texttt{PhaseII}})$
  if $d_{\min} - \min_{y \in \mathcal{P}} (\|\widehat{x} - y\|) > 0$ *or* $\max_{j \in J} (\widehat{g}_j(x) + \epsilon_j) > 0$ **then**
    $\widehat{x} \leftarrow$ Solve $(OBJ^{-\epsilon}_{\texttt{PhaseII}})$ // Relax all ϵ_j in $-\epsilon_j$
    if $d_{\min} - \min_{y \in \mathcal{P}} (\|\widehat{x} - y\|) > 0$ *or* $\max_{j \in J} (\widehat{g}_j(x) + \epsilon_j) > 0$ **then**
      $\widehat{x} \leftarrow$ Solve (OBJ^1) // Solve (OBJ^{λ}) with $\lambda = 1$
  Evaluate $f(\widehat{x})$, $(g_j(\widehat{x}))_{j \in J}$
  Update $x_{\texttt{best}}$, $\mathcal{P} \leftarrow \mathcal{P} \cup \{\widehat{x}\}$
  Update $\epsilon_j, \kappa_j, \forall j \in J$ with Algorithm 9 and the real constraints values $(g_j(\widehat{x}))_{j \in J}$
  if $|\mathcal{P}| \geq 0.9 \cdot N_{\max}$ **then**
    if $f(x_{\texttt{best}}) > 0.95 \cdot f_{\texttt{old}}$ **then**
      $d_{\min} \leftarrow \max\left(\gamma, \frac{1}{2} \cdot \min(d_1, d_{\min})\right)$
  else
    if $f(x_{\texttt{best}}) > 0.95 \cdot f_{\texttt{old}}$ **then**
      $\nu \leftarrow \max(1, \nu - 1)$
    else
      $\nu \leftarrow \min(|\Delta|, \nu + 1)$
    $d_{\min} \leftarrow \max\left(\gamma, d_{\nu} \cdot \min(u_i - l_i, \forall i \in C \cup I \cup D)\right)$

Algorithm 9: BOA-Update ϵ, κ

Given $\sigma_{\texttt{inc}}$, $\sigma_{\texttt{dec}}$, ρ, $k_{\texttt{feas}}$, $\epsilon_{\max}$, $\epsilon_j, \kappa_j, \forall j \in J$
Given $\tilde{g}_j(\widehat{x}), \forall j \in J$ // Predicted or real constraints values at $\widehat{x}$.
for *each* $j \in J$ **do**
  if $\tilde{g}_j(\widehat{x}) > 0$ **then**
    $\kappa_j \leftarrow 0$ // Reset feasibility counter.
    $\tilde{g}_{\texttt{trunc}} \leftarrow \min(1, \tilde{g}_j(\widehat{x}))$
    $\epsilon_j \leftarrow \min\left(\max\left(\sigma_{\texttt{inc}} \cdot \epsilon_j; \rho \cdot \tilde{g}_{\texttt{trunc}}\right), \epsilon_{max}\right)$ // Increase ϵ_j.
  else
    $\kappa_j \leftarrow \kappa_j + 1$
    if $\kappa_j \geq k_{feas}$ **then**
      $\epsilon_j \leftarrow \sigma_{\texttt{dec}} \cdot \epsilon_j$ // Decrease ϵ_j.
Return $\epsilon_j, \kappa_j, \forall j \in J$

The minimum distance parameter $d_{\min}$ is updated after every iteration in both phases according to the quality of the new evaluated point. It is set according to the minimum edge length of $\mathcal{X}$, an ordered finite set of positive values $\Delta = \{d_1, d_2, \ldots, d_{|\Delta|}\}$ and a lower bound γ. The latter depends on the nature of the variables: it is equal to 0 when there is at least one continuous variable and, otherwise, to the positive minimum gap between distinct admissible discrete or integer values.

Let $\nu \in \{1, 2, \ldots, |\Delta|\}$ be the index of the chosen value from Δ, $d_{\min}$ is initialized as $\max\{\gamma, d_\nu \cdot \min(u_i - l_i, \forall i \in C \cup I \cup D)\}$. The minimum distance is updated at each iteration: it can be increased to enforce exploration after a considered good improving in feasibility, and decreased otherwise to enable local exploitation. This is simply done by increasing or decreasing ν. In order to refine the solution of (P), lower values of $d_{\min}$ are allowed after a ratio of the evaluation budget. The maximum number of function evaluations allowed is denoted by $N_{\max}$.

The best point $x_{\texttt{best}}$ (Line 8 of Algorithm 6, Line 24 of Algorithm 7 and Line 12 of Algorithm 8) is defined in $\mathcal{P}$ as the one (or one of those) minimizing the sum of the squared constraint violations if all points are infeasible, otherwise among the feasible points in $\mathcal{P}$, it is the one (or one of those) minimizing the objective value.

6.3.2 Distinctive features of BOA

BOA shares similarities with several surrogate-based BBO solvers like COBRA. The latter also performs a two-phase optimization where the first part aims at finding a feasible candidate. However both methods differ in many aspects. We point out some of them hereafter.

COBRA is based on RBF interpolations and reported results in [Regis, 2014] only make use of this surrogate, unlike the proposed method for which we report experiments with different types of surrogates. Besides, the distance parameter is adapted in BOA according to the quality of the solution found at each iteration. The resolutions of the subproblems are also different. COBRA handles continuous variables only and uses the MATLAB function `fmincon` that employs a SQP method. BOA is designed for problems involving discrete variables and our implementation uses the direct search solver NOMAD. Furthermore, the best iterate $x_{\texttt{best}}$ of the first phase of BOA is chosen directly based on the sum of squared constraint violations instead of the number or maximum of the constraint violations. As other distinctive features, BOA updates its slacks already in Phase I and they are decreased as soon as the corresponding (predicted or original) constraints are not satisfied.

Differently from other methods such as SO-MI or CONDOR, the proposed method

does not require an initial feasible solution. We solve an auxiliary problem to determine a candidate point whereas CONDOR proceeds to different types of perturbations of the currently best solution, taking into account integrality constraints. At each iteration, SO-MI evaluates 4 candidates that are chosen from 4 groups. Each group is generated by random perturbations of the variables and uniform random points generations.

6.3.3 Adaptation of BOA to parallel evaluations

The classical BOA algorithm described conducts sequential evaluations of the blackbox. However, when the expensive optimization problem and the calculation resources enable parallel evaluations, it may be interesting to adapt the method by taking advantage of the parallelization and, hopefully, considerably reduce the total computational time. As an example, typical size optimization problems encountered at Stellantis are solved by proceeding to an order of 25 parallel evaluations of the finite element models. With this in mind, an extension of BOA to deal with parallel evaluations was designed.

Let $B \in \mathbb{N}^*$ stands for the number of allowed parallel calls to the blackbox. Apart from the initial DOE, the points evaluated in BOA come from the resolution of a surrogate subproblem. The idea now is to solve a batch of B subproblems where originally only one was solved in BOA, which leads to B points $\{\hat{x}^{(1)}, \dots, \hat{x}^{(B)}\}$ to evaluate with the expensive functions at each iteration. The subproblems can be solved in parallel or sequentially as they only involve surrogate calls which are assumed computationally negligible compared to the real blackbox evaluations. Thus, Lines 9 to 21 of Algorithm 7 and Lines 6 to 10 of Algorithm 8 are executed B times per iteration of the respective algorithm.

To do this, for each resolution b of a batch, a slack ϵ_{bj} is declined for each constraint in each resolution of the batch, with an associated decrease counter κ_{bj}, for all $(b, j) \in \{1, 2, \dots, B\} \times \{1, 2, \dots, |J|\}$, as well as there is a parameter λ_b for each subproblem of the batch in Phase I. The second phase of BOA starts as soon as one feasible point with respect to the real blackbox is found.

All candidates examined by the subproblem solver during an iteration are stored and sorted according to a lexicographic order of $\left(\sum_{j \in J} \max\left(0, \widehat{g}_j(x)\right)^2, \widehat{f}(x)\right)$. With this sorting, feasible points are preferred to infeasible ones, and lower objective values and constraint violations of the feasible and infeasible points respectively are favoured. Let $\hat{\mathcal{P}}_b$ denote the points considered during the b^{th} resolution of a batch of subproblems, with $b \in \{1, 2, \dots, B\}$, and let $V = \cup_{b=1}^{B} \hat{\mathcal{P}}_b$ be the union of the B sets of candidate points. Let us assume that V is sorted according to the above lexicographic order. The points $\{\hat{x}^{(1)}, \dots, \hat{x}^{(B)}\}$ to evaluate with the expensive functions are iteratively selected from V such that the distance to the set of evaluated points $\mathcal{P}$ and to the already selected points is greater than $d_{\min}$. This

distance is halved until we actually get B points for the parallel evaluations. This enables to not consider duplicate or very close points, and to favour exploration.

When a starting point is needed by an algorithm to solve a subproblem, as it is the case for MADS for instance, B points are chosen from the set of already evaluated points $\mathcal{P}$, considering two measures. The first one is defined by the following application:

$$\varphi : x \mapsto \begin{cases} f(x) & \text{if } g_j(x) \leq 0, \forall j \in J \\ f_{\max} + \sum_{j \in J} \max\left(0, g_j(x)\right)^2 & \text{if } \exists j \in \{1, 2, \ldots, |J|\}, g_j(x) > 0, \end{cases}$$

where $f_{\max}$ is the highest feasible objective value so far. The second criterion is the distance to the other evaluated points, that is to be maximized. Hence, the points of $\mathcal{P}$ are ranked in non-dominated sets according to the measures $(\varphi(x), -\min_{y \in \mathcal{P} \setminus \{x\}}(\|x - y\|))$, similarly to NSGA-II: the sorting is iteratively done, removing the already chosen points, until B points are selected.

The respect of the evaluation budget is controlled before the parallel evaluations: if the remaining allowed number of blackbox calls is lower than the batch size B, then the latter is updated to $(N_{\max} - |\mathcal{P}|)$ and only this number of points is chosen for the expensive evaluations.

6.3.4 Description of the components of BOA

Initial DOE

As is very often the case in surrogate based BBO methods present in the literature, the DOE method we use for our experiments is a symmetric Latin Hypercube Design (SLHD) [Ye et al., 2000]. It is a variant of the space filling Latin hypercube sampling developed by [McKay et al., 2000].

By default, the algorithm starts with a DOE of rank $(n + 1)$ and consisting of $(n + 1)$ SLHD points possibly supplemented with random points to satisfy the rank condition. The rank is also used in MISO for the starting points generation as it guarantees a unique setting for the RBF parameters.

Considered surrogate models

In this study, we consider radial basis functions, kriging models and MARS models that have shown good performance in the literature. These surrogates are described in Section 2.4.1.

As mentioned in Section 2.4.2, RBF is compared to other surrogate models in [Jin et al., 2001] and exhibit an overall best performance regarding, among others, accuracy,

robustness and efficiency. On a high-dimensional automotive benchmark problem, the authors of [Regis, 2011] conclude that the methods using RBF are performing better than other methods including a kriging-based NOMAD and a sequential quadratic programming algorithm. The work of [Müller and Shoemaker, 2014] investigates the influence of the surrogate type and the sampling strategy used in the resolution of expensive blackbox optimization problems subject to box constraints. The study notably includes cubic RBF, Gaussian kriging models and MARS models. The results on 15 continuous problems show that the ensembles including the cubic RBF often outperform those that do not use this type of surrogate. Gaussian processes are also commonly used, for instance in the work of [Kianifar and Campean, 2020] comparing also RBF and polynomial surrogates and where they show a globally better performance. In [Villa-Vialaneix et al., 2012], splines and kriging based methods exhibit the best results with small and medium training datasets on a corn cultivation application study.

Constraint handling

Slack factors are used in BOA to manage constraint satisfaction. The motivation behind this comes from [Regis, 2014] where conducted preliminary experiments exhibited solutions of the subproblems at the boundaries, where the surrogates are not accurate. As a result, in many cases these points were infeasible with respect to the true blackbox constraints. Although the subproblem formulation of the second phase is common, BOA considers the objective function already in the first phase through its parameter λ and introduces the slacks $(\epsilon_j)_{j\in J}$ for each constraint also in this phase. The update of the slacks also differs as they are updated independently for each constraint in BOA. In this way, a distinction is made between the constraints that are often or easily satisfied and the ones that are often violated. Hence, better solutions are expected by avoiding the pitfall of handling an important slack due to few constraints that are often violated and of staying far away the border of easily satisfied constraints. Besides, the increase of an ϵ_j occurs as soon as the corresponding surrogate constraint is violated.

Blackbox crash handling

In real-world application problems, a blackbox may not give outputs for all inputs. As an example, the simulation of a finite element model may crash due to divergence in solving the underlying differential equations. This often results in a "NaN" output. In order to deal with this kind of hidden constraints, the non-real outputs are set to infinity. This way, the corresponding points are added to the set of already evaluated points to avoid

duplicate evaluations and points in their close vicinity, i.e. in the ball of radius $d_{\min}$, are not considered. However, these candidates are not used in the construction of the models in order to not affect the model by making assumptions on their neighbourhoods.

Discrete variables handling

Inside NOMAD, discrete variables are treated as integers. Indeed, although integer and granular variables have a specific handling in NOMAD, there is no direct way to treat general discrete variables in the solver. However, the original variables are considered for each call of a surrogate or a blackbox. Hence, the true distances are considered to build the models.

6.4 Considered optimization problems

For the experiments, we consider instances derived from the literature and two automotive applications from Stellantis. All of them have inequality constraints (between 1 and 91) and can be considered medium to high dimensional in DFO as they have more than 10 variables.

6.4.1 Instances from the literature

The first set of instances consists of 19 constrained problems stemming from the literature, among which some are classical analytical problems and others are derived from applications. Table 6.1 gives a description of these problems in terms of numbers and types of variables, and the source papers.

We use three instances from the well-known G-problems benchmark collection and derivations of two of them. Indeed, among the three integer problems taken from [Müller et al., 2014], the problems I1 and I3 are derived from the problem G01. The problem I2 is the hmittelman problem from the MINLPLib[1] library. A derivation of G07 called MD2 with mixed variables, including discrete ones, is used and its formulation comes from [Crélot et al., 2017]. Applications about car side impact and stepped cantilever beam, respectively named MD3 and MD4 in our experiments, are also used.

Four problems that have mixed continuous and integer variables are taken from [Müller et al., 2013] and three of them derive from applications.

Among problems arisen from applications, six of them (C4, C5, C6, MI5, MI6 and MD1) come from a real-world benchmark suite introduced in [Kumar et al., 2020].

[1] https://www.minlplib.org/index.html

problem	n	$\|C\|$	$\|I\|$	$\|D\|$	$\|J\|$
C1 [Floudas and Pardalos, 1990]	13	13	0	0	9
C2 [Hock and Schittkowski, 1980]	10	10	0	0	8
C3 [Himmelblau, 1972]	15	15	0	0	1
C4 [Paul, 1987, Pant et al., 2009, Kumar et al., 2020]	14	14	0	0	15
C5 [Grandhi and Venkayya, 1988, Kumar et al., 2020]	10	10	0	0	3
C6 [Wang et al., 2018, Kumar et al., 2020]	30	30	0	0	91
I1 [Floudas and Pardalos, 1990, Müller et al., 2014]	13	0	13	0	9
I2 [Bussieck et al., 2003, Müller et al., 2014]	16	0	16	0	7
I3 [Floudas and Pardalos, 1990, Müller et al., 2014]	13	0	13	0	9
MI1 [Berman and Ashrafi, 1993, Müller et al., 2013]	11	7	4	0	7
MI2 [Yuan et al., 1988, Müller et al., 2013]	11	7	4	0	13
MI3 [Kuo et al., 2001, Müller et al., 2013]	10	5	5	0	3
MI4 [Kuo et al., 2001, Müller et al., 2013]	10	5	5	0	3
MI5 [Grossmann and Sargent, 1979, Kumar et al., 2020]	10	7	3	0	10
MI6 [Gupta et al., 2007, Kumar et al., 2020]	10	9	1	0	9
MD1 [Yokota et al., 1998, Kumar et al., 2020]	22	0	8	14	86
MD2 [Hock and Schittkowski, 1980, Crélot et al., 2017]	10	2	2	6	8
MD3 [Gu et al., 2001, Gandomi et al., 2011]	11	9	0	2	10
MD4 [Thanedar and Vanderplaats, 1995, Gandomi et al., 2011]	10	4	2	4	11

Table 6.1: Problems from the literature described with the dimension n, the number of continuous $|C|$, integer $|I|$ and discrete $|D|$ variables, respectively, and the number of inequality constraints $|J|$.

6.4.2 Applications from Stellantis

In addition to the 19 problems from the literature, we use two instances encountered at Stellantis for some of the experiments. Table 6.2 gives descriptions of these problems.

problem	n	$\|C\|$	$\|I\|$	$\|D\|$	$\|J\|$
RSMLateralCrash	34	0	10	24	24
LateralCrash	48	0	3	45	64

Table 6.2: Application problems from Stellantis described with the dimension n, the number of continuous $|C|$, integer $|I|$ and discrete $|D|$ variables, respectively, and the number of inequality constraints $|J|$.

Vehicle pole lateral crash

The first instance, called RSMLateralCrash, is a model of a pole lateral crash study, built from about 800 sample points. It is considered as representative of the expensive finite

element model and was used at Stellantis in the optimization process for faster experiments. Each call to the model takes less than a minute. The abbreviation RSM stands for response surface model. The optimization problem considered aims at minimizing the mass of a basket of parts in the battery area of the vehicle. It is a constrained mixed-variable problem with 24 inequality constraints that represent performance features to satisfy, among which deceleration, stress and displacement. There are 34 variables which correspond to the choice of the materials of 10 parts of the vehicle, treated as integers according to some ranking based on the material properties, and the thicknesses of 25 parts, that are granular variables. The materials considered are different steel grades. Details on the variable bounds and granularities are given in Table 6.3. For instance, x_{11} allows values between 1 and 4 with a granularity of 0.05, so its admissible values belong to $\{1, 1.05, 1.1, \ldots, 4\}$.

variable	lower bound	granularity	upper bound
1 to 10	1	1	10
11 to 18	1	0.05	4
19 to 34	1	0.05	2.5

Table 6.3: Variable bounds and granularities for RSMLateralCrash.

Vehicle barrier lateral crash

The second problem, denoted LateralCrash, uses an expensive FE simulation of a barrier lateral crash that takes more than 12 hours at each call. The optimization problem is similar to the first one, minimizing the mass of a bench of parts of the vehicle, and has 64 inequality constraints. There are 48 variables, among which 3 steel grades choices and 45 granular thicknesses. Table 6.4 gives details on the bounds and granularity of each variable. The 48^{th} variable allows negative values because it intervenes in a formula for the thickness computation of the corresponding part.

variable	lower bound	granularity	upper bound
1	1	1	2
2 to 3	1	1	3
4 to 5	1.2	0.1	2
6 to 9	2	0.1	6.5
10	0.6	0.05	1.1
11	0.6	0.05	1.1
12 to 16	0.6	0.05	1.15
17 to 18	0.6	0.05	1.2
19	0.6	0.05	1.3
20 to 21	0.6	0.05	1.35
22	0.6	0.05	1.4
23 to 24	0.6	0.05	1.6
25	0.65	0.05	1.65
26 to 27	0.75	0.05	1.75
28 to 37	0.8	0.05	1.8
38	0.85	0.05	1.35
39	0.95	0.05	1.45
40	1.1	0.05	2.1
41	1.3	0.05	1.8
42	1.4	0.05	2.4
43 to 44	1.5	0.05	2.5
45	1.6	0.05	2.6
46	1.9	0.05	2.9
47	2	0.05	3
48	-0.1	0.05	0.1

Table 6.4: Variable bounds and granularities for LateralCrash.

6.5 General experimental setting

The surrogate models used in BOA are built on variables and objective values that are scaled in $[0, 1]$. The constraints values are scaled in $[-1, 0]$ for violated constraints and in $[0, 1]$ for satisfied ones. Similarly as performed in SO-MI, truncations are applied to the output values in order to avoid high variations of the surrogate values. Once the number of evaluated points is greater than twice the initial DOE size, the positive and negative constraints values are truncated to the median of the positive and the median of the negative constraints values respectively. Unlike what is done in SO-MI, only the feasible objective values are truncated to their median once the number of feasible points evaluated is greater than twice the initial DOE size.

In order to lead numerical experiments, some parameters of BOA had to be set, in

particular $\epsilon_{\max}, \sigma_{\texttt{inc}}, \rho, k_{\texttt{feas}}, threshold$ and $d_{\min}$. To do this, a sensitivity analysis based on an experimental design of 27 points was performed, including 16 Plackett-Burman designs with foldover, that is with the complementary plan.

The Plackett-Burman technique [Plackett and Burman, 1946] is an experimental design used to identify the most influential parameters on some outputs and it can be seen as an economical factorial design. It is based on Hadamard matrix, that is a square matrix H consisting of 1 and -1 values and such that $HH^\top = dI$, where $d \in \mathbb{N}^*$ is the dimension of H and I is the identity matrix. The parameters are considered as independent variables and their interactions are assumed negligible. The cardinality of a Plackett-Burman design set is the smallest multiple of 4 greater than the number of inputs, and the settings to test contain points on the boundaries of the parameters ranges, that are minimum and maximum admissible values.

The 16 Plackett-Burman designs are added with 10 points from a space-filling algorithm and one point which corresponds to the middle of the considered bounds of the parameters. The modeFRONTIER[2] software was used to generate the experimental design. It also provides statistical analysis tools such as correlation matrices and t-Student analyses.

For each point, five runs of BOA with cubic RBF were performed on three literature problems from the G-problems collection. Two of them are continuous and correspond to the problems G07 and G18, that have respectively 10 and 9 variables, and the third one is the variant of G06 where the 2 variables are imposed to be integers. The maximum number of blackbox evaluations was set to 200 for the analyses. The statistical outputs observed were relative to the number of feasible runs, the number of evaluations used to leave Phase I (i.e. to find a first feasible solution) and the best feasible objective values after each phase of BOA.

The statistical significance of the results was evaluated using Student's t-tests with a p-value of 0.1 as well as surrogate analyses to set the values of the influential parameters.

In practice, we initialize the parameters as follows: $\epsilon_{\max} = 10^{-3}$, $\sigma_{\texttt{inc}} = 1.1$, $\rho = 0.5$ and $k_{\texttt{feas}} = \max(\lceil 2 \cdot \sqrt{n} \rceil, \lceil 2 \cdot \sqrt{|J|} \rceil)$, $threshold = 10^{-1}$. For the distance parameter setting, we choose $\Delta = \{5{\cdot}10^{-4}, 10^{-3}, 5{\cdot}10^{-3}, 10^{-2}, 5{\cdot}10^{-2}, 10^{-1}\}$ and $d_{\min}$ is initialized using $d_\nu = 5{\cdot}10^{-3}$.

Our implementation of BOA uses MATLAB R2020b and the subproblems of the algorithm are solved using the MATLAB version of NOMAD v3.9.1 with the option `ORTHO N+1 NEG`. The latter showed good performance in Chapter 4 compared with the other direction types of ORTHOMADS both on continuous and mixed-integer optimization problems.

Two sets of experiments are considered with different evaluation budgets. The first one aims, in the one hand, at comparing different types of surrogate models and, on the other,

[2] https://engineering.esteco.com/modefrontier/

at investigating the contribution of the parameter λ in Phase I. The second set is given a higher budget and compares BOA with NOMAD on the problems from the literature and on the instances from Stellantis.

In the presentation of the results, we denote the types of surrogates cubic RBF, MARS, and the four kinds of kriging (Gaussian, exponential, Matérn 3/2 and Matérn 5/2) as follows: R, M, KG, KE, K3 and K5, respectively.

In this study, we focus on cubic basis functions added with a polynomial tail. The construction of the RBF surrogates borrows from the dedicated part of the code of MISO that we adapted to our algorithm. MARS models are constructed using the ARESLab[3] MATLAB toolbox [Jekabsons, 2011]. Finally, Gaussian and exponential kriging models are built thanks to the DACE[4] MATLAB toolbox [Lophaven et al., 2002]. We implemented the correlation functions Matérn 3/2 and Matérn 5/2 embedded in the DACE framework. When applicable, the parameterization of the surrogates stems from preliminary experiments.

6.6 Medium-budget experiments

In the first experiments, the performance of several implementations of BOA is evaluated on 30 runs performed, starting from different DOEs that are although common to all implementations, and with a blackbox evaluation budget of 200. Each subproblem resolution inside BOA is done by NOMAD with a maximum of $25 \cdot n$ surrogate function evaluations. The performance is evaluated according to different measures starting with the number of runs out of the 30 launched that ended up with a feasible solution. The other measures used consider only the feasible runs and are the mean number of evaluations used to leave Phase I, the mean objective values after Phase I and Phase II, and the minimum feasible objective value found over all runs. Comparisons are done using the 19 problems from the literature.

6.6.1 Comparisons of surrogate models

Firstly, cubic RBF, MARS and 4 types of kriging (Gaussian, exponential, Matérn 3/2 and Matérn 5/2) are compared inside BOA. Tables 6.5, 6.6, 6.7 and 6.8 summarize the results for each family of problems, respectively on continuous, integer, mixed-integer and mixed-variable problems. We distinguish mixed-integer problems that have continuous and integer variables from mixed-variable problems that have discrete variables other than

[3] http://www.cs.rtu.lv/jekabsons/regression.html
[4] https://www.omicron.dk/dace.html

integers. In the names of the columns, "Pb" stands for the problem name and $\mathcal{S}$ for the type of surrogate model.

We first consider problems MD1 and C6 as there are not feasible results for all surrogates on these instances. Problem MD1, whose ratio of the feasible region is less than 10^{-4}, is considered difficult to solve by [Kumar et al., 2020]. Indeed, the results show that kriging models hardly find feasible solutions on this problem, only the one using Matérn 3/2 manages to find one. MARS has the best results on MD1 regarding all the considered performance measures. On C6, MARS was stopped due to long computational times. Moreover, globally longer computational times seem to be needed for the construction of MARS models. While kriging models performed badly on MD1, they have the best mean objective values on C6 and, especially, the use of Matérn 3/2 gives the best mean objective values at the end of the optimization. Cubic RBF also exhibit good results on this problem and reaches the minimum feasible objective value among the surrogates.

On the other 17 problems, MARS followed by cubic RBF finds the highest number of feasible runs. These surrogates also globally reach the best qualities in solutions. It can be noted that MARS finds the global optimum on the 30 runs on I1, and so does cubic RBF on I2. On the contrary, considering each family of problems, exponential kriging reaches the lowest numbers of feasible runs. The difference with the other kriging types is especially noticeable on the integer problems I1 and I3 where it finds only 4 and 3 feasible runs respectively, whereas most runs are feasible for the others. Moreover, this kind of kriging globally finds the worst mean feasible objective values on mixed-integer and mixed-discrete problems.

6.6.2 Investigations on λ

The parameter λ is used in the surrogate subproblem ($OBJ^{\lambda,\epsilon}_{\texttt{PhaseI}}$) to take into account the objective value in the first phase of BOA. The introduction of this parameter is a special feature of our algorithm by comparison with the other existing methods. The effects of λ are investigated by comparing the performance of the proposed method when the parameter is classically used and when it equals 0 during all the optimization. To do so, cubic RBF models are considered in BOA as this type of model outperformed most of the others in the experiments of Section 6.6.1 and it is computationally less expensive than MARS.

The same performance measures are used and presented in Table 6.9 for the 19 problems from the literature. In the surrogate column entitled $\mathcal{S}$, "$\text{R}_{\lambda=0}$" indicates the cubic RBF-based BOA that does not use λ.

Considering all problems except MD1, both variants are globally equivalent regarding

Pb	$\mathcal{S}$	$\#F$	$N^{(I)}$	$\sigma_{N^{(I)}}$	$f^{(I)}$	$f^{(II)}$	$\sigma_{f^{(II)}}$	$f^{(II)}_{\min}$
C1	R	24	85.29	66.64	-9.05	-9.94	3.29	-14.81
	M	30	78.87	43.85	-10.13	-14.01	0.76	-14.96
	KG	30	55.27	34.63	-3.81	-11.93	3.24	-15.00
	KE	16	73.56	43.17	-3.99	-13.92	1.97	-15.00
	K3	30	47.13	27.13	-2.93	-14.86	0.33	-15.00
	K5	30	47.13	29.39	-1.79	-14.49	1.07	-15.00
C2	R	30	38.07	10.88	126.93	26.09	1.01	24.94
	M	30	44.37	14.61	211.78	33.44	5.10	28.37
	KG	30	32.87	7.18	239.72	26.80	1.22	25.02
	KE	30	39.37	23.22	515.70	59.44	32.64	30.27
	K3	30	31.47	6.37	344.95	27.31	1.38	25.42
	K5	30	29.37	6.07	296.91	27.07	1.29	25.06
C3	R	30	43.87	17.12	1482.38	175.64	107.12	61.32
	M	30	49.93	19.23	4936.40	316.14	289.76	65.75
	KG	30	68.27	19.60	4783.45	463.37	421.21	101.59
	KE	30	71.73	27.78	8983.71	2312.13	2140.65	384.51
	K3	30	56.73	13.06	4068.69	690.08	1310.29	144.48
	K5	30	60.67	15.41	4507.89	436.88	240.44	66.50
C4	R	20	140.95	34.23	100220.25	99202.03	422055.30	32.71
	M	30	111.10	28.55	9722.38	4091.61	3709.92	6.76
	KG	21	115.43	33.49	1271738.01	154861.04	258460.79	15.10
	KE	10	89.30	14.13	1686509.88	123967.71	206323.43	2275.04
	K3	29	119.59	34.41	1003787.88	166552.78	289010.56	92.54
	K5	25	126.12	35.38	1312627.27	612411.72	1916379.73	32.10
C5	R	30	29.07	7.28	684.60	552.03	5.78	537.53
	M	30	23.80	6.57	883.09	569.58	17.49	548.18
	KG	30	20.40	5.04	857.74	547.44	3.58	541.68
	KE	30	20.10	4.60	871.21	562.82	11.49	548.02
	K3	30	21.67	6.63	831.20	556.14	15.98	542.45
	K5	30	20.77	5.79	831.13	554.01	10.56	541.79
C6	R	30	57.63	8.24	-4467.56	-5292.86	220.87	-5674.37
	M$^{(*)}$	-	-	-	-	-	-	-
	KG	30	52.23	7.54	-4583.62	-5307.92	231.72	-5598.43
	KE	30	52.63	5.68	-4530.97	-5298.64	143.72	-5601.75
	K3	30	52.30	4.70	-4510.78	-5371.81	171.01	-5663.18
	K5	30	53.07	5.85	-4563.47	-5325.31	171.15	-5657.87

Table 6.5: Results on continuous problems for each surrogate type: number of feasible runs $\#F$, mean number of function evaluations to reach a feasible solution N^I, its standard deviation $\sigma_{N^{(I)}}$, mean first feasible objective value $f^{(I)}$, mean best feasible objective value $f^{(II)}$, its standard deviation $\sigma_{f^{(II)}}$ and minimum feasible objective value $f^{(II)}_{\min}$ on the 30 runs. A star $^{(*)}$ indicates that the experiment was stopped due to long computational times.

Pb	$\mathcal{S}$	$\#F$	$N^{(I)}$	$\sigma_{N^{(I)}}$	$f^{(I)}$	$f^{(II)}$	$\sigma_{f^{(II)}}$	$f^{(II)}_{\min}$
I1	R	30	32.20	15.58	-11.27	-14.17	1.29	-15.00
	M	30	37.00	14.20	-12.93	-15.00	0.00	-15.00
	KG	30	34.97	16.61	-4.73	-11.00	2.44	-15.00
	KE	4	114.00	73.87	-8.25	-10.75	1.50	-12.00
	K3	26	37.65	28.83	-5.35	-11.35	2.23	-15.00
	K5	26	32.50	15.30	-5.15	-11.00	2.40	-15.00
I2	R	30	34.20	4.84	20.67	13.00	0.00	13.00
	M	30	54.57	25.39	20.30	15.97	3.67	13.00
	KG	30	49.33	18.53	33.57	18.50	10.96	13.00
	KE	30	45.47	21.83	29.07	17.07	7.07	13.00
	K3	30	47.90	31.25	26.63	16.17	7.05	13.00
	K5	30	47.87	25.91	30.67	18.57	10.99	13.00
I3	R	30	39.40	28.96	-43507.57	-49971.70	109.17	-50128.00
	M	30	83.23	35.00	-44050.30	-50143.50	67.54	-50200.00
	KG	30	39.07	14.63	-36306.83	-50186.27	15.38	-50200.00
	KE	3	161.67	7.02	-49958.00	-50064.00	84.11	-50159.00
	K3	29	45.48	8.91	-38440.90	-50183.41	20.83	-50200.00
	K5	30	38.20	8.67	-37572.90	-50180.93	23.44	-50200.00

Table 6.6: Results on integer problems for each surrogate type: number of feasible runs $\#F$, mean number of function evaluations to reach a feasible solution N^I, its standard deviation $\sigma_{N^{(I)}}$, mean first feasible objective value $f^{(I)}$, mean best feasible objective value $f^{(II)}$, its standard deviation $\sigma_{f^{(II)}}$ and minimum feasible objective value $f^{(II)}_{\min}$ on the 30 runs.

the number of feasible runs. The use of λ generally leads to more evaluations spent in Phase I but the objective values when exiting Phase I and Phase II are globally better, as well as the minimum feasible objective values among all runs. There can be a considerable difference in the quality of the solution as shown on problem C4: the mean objective value of the traditional BOA is almost 3 times better than the variant that does not use λ.

Nevertheless, on two of the three integer problems tested, although the objective values after Phase I are better when λ is used, there are slight advantages at the end of the optimization when $\lambda = 0$. The problems concerned are alterations of the same problem and only their bounds differ. This explains why the algorithm behaves similarly on them. The effect of λ on these instances may be specific to the problems. Besides, the number of evaluations spent in Phase I for I1 and I3 is higher in the traditional BOA and, therefore, there are less evaluations left in Phase II for improving the solutions.

On MD1, the variant that does not use the tested parameter was stopped because the experiment was computationally too long. The use of λ seems to help in the optimization

Pb	$\mathcal{S}$	$\#F$	$N^{(I)}$	$\sigma_{N^{(I)}}$	$f^{(I)}$	$f^{(II)}$	$\sigma_{f^{(II)}}$	$f^{(II)}_{\min}$
MI1	R	30	20.97	2.66	-0.19	-0.91	0.04	-0.94
	M	30	22.83	3.06	-0.31	-0.93	0.03	-0.94
	KG	30	29.10	12.29	-0.25	-0.92	0.05	-0.95
	KE	30	25.17	7.66	-0.33	-0.91	0.05	-0.94
	K3	30	27.27	9.05	-0.37	-0.92	0.03	-0.95
	K5	30	27.57	5.78	-0.33	-0.91	0.05	-0.95
MI2	R	30	19.90	4.37	14.81	5.90	0.31	4.73
	M	30	23.53	9.50	13.07	5.80	0.23	4.59
	KG	30	29.57	9.41	10.42	6.23	0.62	4.58
	KE	26	52.73	40.01	10.70	8.04	1.61	5.83
	K3	30	27.73	9.74	11.23	6.62	0.74	5.82
	K5	30	31.47	9.22	9.14	6.27	0.55	5.15
MI3	R	30	37.20	14.41	-0.95	-0.99968	0.00021	-0.99988
	M	30	38.63	14.09	-0.54	-0.99813	0.00188	-0.99975
	KG	30	51.03	35.79	-0.88	-0.99928	0.00133	-0.99985
	KE	23	49.35	32.23	-0.83	-0.99916	0.00151	-0.99980
	K3	27	32.89	20.27	-0.83	-0.99944	0.00051	-0.99988
	K5	24	30.08	13.43	-0.87	-0.99931	0.00146	-0.99984
MI4	R	30	41.10	10.16	-0.96	-0.99949	0.00034	-0.99998
	M	30	35.57	14.01	-0.72	-0.99928	0.00073	-0.99991
	KG	26	48.65	35.01	-0.69	-0.99734	0.01200	-0.99999
	KE	18	62.94	49.88	-0.81	-0.99592	0.01091	-0.99999
	K3	28	49.00	33.75	-0.75	-0.99812	0.00418	-0.99998
	K5	25	37.76	31.27	-0.70	-0.99889	0.00145	-0.99999
MI5	R	30	30.47	9.70	99190.66	61782.15	5174.17	58558.89
	M	30	21.93	8.19	149128.59	64093.16	4638.25	56576.26
	KG	30	35.07	12.10	112865.34	60907.19	3614.95	58652.64
	KE	30	31.47	8.84	108068.20	63041.05	7340.51	58710.03
	K3	30	29.80	9.66	100686.43	60248.08	3424.47	58632.38
	K5	30	28.17	7.94	112179.49	59898.48	2883.39	58597.03
MI6	R	30	22.83	3.71	23547.31	16989.35	29.34	16959.62
	M	30	22.13	7.77	24184.04	17094.23	674.18	16959.18
	KG	30	23.73	4.99	21710.40	16966.08	11.31	16958.33
	KE	30	25.07	6.48	24557.69	18279.88	2095.08	16958.69
	K3	30	23.10	5.09	25701.99	16972.69	19.42	16958.55
	K5	30	23.90	6.38	23463.35	16967.75	15.35	16958.31

Table 6.7: Results on mixed-integer problems for each surrogate type: number of feasible runs $\#F$, mean number of function evaluations to reach a feasible solution N^I, its standard deviation $\sigma_{N^{(I)}}$, mean first feasible objective value $f^{(I)}$, mean best feasible objective value $f^{(II)}$, its standard deviation $\sigma_{f^{(II)}}$ and minimum feasible objective value $f^{(II)}_{\min}$ on the 30 runs.

of this hard problem.

These results comfort on the utility of λ in the problem formulation of Phase I. It globally leads to better results and can lead to a faster optimization.

Pb	$\mathcal{S}$	$\#F$	$N^{(I)}$	$\sigma_{N^{(I)}}$	$f^{(I)}$	$f^{(II)}$	$\sigma_{f^{(II)}}$	$f^{(II)}_{\min}$
MD1	R	9	157.00	28.34	113.17	106.98	32.57	67.49
	M	16	142.63	25.71	73.69	73.23	19.87	52.96
	KG	0	-	-	-	-	-	-
	KE	0	-	-	-	-	-	-
	K3	1	186.00	0.00	73.10	67.28	0.00	67.28
	K5	0	-	-	-	-	-	-
MD2	R	30	31.97	7.84	210.93	33.06	4.89	31.43
	M	30	28.20	7.45	451.54	35.37	2.51	32.53
	KG	30	30.10	11.76	415.25	40.81	22.31	31.53
	KE	27	33.67	23.56	638.59	93.34	67.74	32.72
	K3	30	29.93	20.31	288.77	43.11	22.07	31.49
	K5	30	27.57	12.43	309.46	36.24	8.75	31.44
MD3	R	30	27.43	9.61	27.36	23.78	0.58	23.57
	M	30	32.53	10.33	28.43	23.73	0.16	23.59
	KG	30	28.57	11.07	29.67	23.79	0.30	23.58
	KE	30	25.30	6.44	31.31	24.41	0.71	23.58
	K3	30	25.70	5.75	30.33	23.89	0.41	23.58
	K5	30	23.73	5.45	30.56	23.90	0.45	23.57
MD4	R	30	19.17	2.68	79543.73	66962.58	1766.70	64430.08
	M	30	17.20	1.30	89018.46	65948.35	1407.97	64414.72
	KG	30	23.17	6.58	90982.70	67213.17	1819.30	64358.21
	KE	30	22.63	5.53	86514.34	68573.29	2655.00	64478.30
	K3	30	20.93	4.56	88296.62	67347.29	2218.25	64392.13
	K5	30	21.83	4.76	85685.56	67467.49	1753.36	64425.09

Table 6.8: Results on mixed-variable problems for each surrogate type: number of feasible runs $\#F$, mean number of function evaluations to reach a feasible solution N^I, its standard deviation $\sigma_{N^{(I)}}$, mean first feasible objective value $f^{(I)}$, mean best feasible objective value $f^{(II)}$, its standard deviation $\sigma_{f^{(II)}}$ and minimum feasible objective value $f^{(II)}_{\min}$ on the 30 runs.

Pb	$\mathcal{S}$	$\#F$	$N^{(I)}$	$\sigma_{N^{(I)}}$	$f^{(I)}$	$f^{(II)}$	$\sigma_{f^{(II)}}$	$f^{(II)}_{\min}$
C1	R	24	85.29	66.64	-9.05	-9.94	3.29	-14.81
	$R_{\lambda=0}$	30	21.50	1.80	-3.71	-8.87	4.08	-14.95
C2	R	30	38.07	10.88	126.93	26.09	1.01	24.94
	$R_{\lambda=0}$	30	38.40	9.57	1231.45	26.10	0.81	24.85
C3	R	30	43.87	17.12	1482.38	175.64	107.12	61.32
	$R_{\lambda=0}$	30	34.80	7.85	14444.98	198.63	130.02	57.50
C4	R	20	140.95	34.23	100220.25	99202.03	422055.30	32.71
	$R_{\lambda=0}$	19	121.95	47.99	949918.50	283360.89	658017.50	200.87
C5	R	30	29.07	7.28	684.60	552.03	5.78	537.53
	$R_{\lambda=0}$	30	17.73	1.20	897.62	552.66	5.22	542.38
C6	R	30	57.63	8.24	-4467.56	-5292.86	220.87	-5674.37
	$R_{\lambda=0}$	30	57.73	10.25	-4530.51	-5306.63	174.03	-5713.49
I1	R	30	32.20	15.58	-11.27	-14.17	1.29	-15.00
	$R_{\lambda=0}$	30	23.03	5.62	-3.63	-14.50	0.86	-15.00
I2	R	30	34.20	4.84	20.67	13.00	0.00	13.00
	$R_{\lambda=0}$	30	34.10	6.83	24.57	13.00	0.00	13.00
I3	R	30	39.40	28.96	-43507.57	-49971.70	109.17	-50128.00
	$R_{\lambda=0}$	30	35.20	18.25	-23134.57	-50018.80	120.62	-50199.00
MI1	R	30	20.97	2.66	-0.19	-0.91	0.04	-0.94
	$R_{\lambda=0}$	30	19.27	1.53	-0.10	-0.91	0.03	-0.95
MI2	R	30	19.90	4.37	14.81	5.90	0.31	4.73
	$R_{\lambda=0}$	30	22.90	5.00	14.45	6.03	0.44	4.76
MI3	R	30	37.20	14.41	-0.95	-0.99968	0.00021	-0.99988
	$R_{\lambda=0}$	30	22.17	4.78	-0.11	-0.99966	0.00022	-0.99987
MI4	R	30	41.10	10.16	-0.96	-0.99949	0.00034	-0.99998
	$R_{\lambda=0}$	30	21.97	4.91	-0.47	-0.99958	0.00046	-0.99996
MI5	R	30	30.47	9.70	99190.66	61782.15	5174.17	58558.89
	$R_{\lambda=0}$	30	19.23	2.27	166354.88	61734.22	5538.20	58545.80
MI6	R	30	22.83	3.71	23547.31	16989.35	29.34	16959.62
	$R_{\lambda=0}$	30	25.30	7.37	34421.18	16984.71	37.35	16959.44
MD1	R	9	157.00	28.34	113.17	106.98	32.57	67.49
	$R^{(*)}_{\lambda=0}$	-	-	-	-	-	-	-
MD2	R	30	31.97	7.84	210.93	33.06	4.89	31.43
	$R_{\lambda=0}$	30	33.20	9.28	1654.05	33.78	5.51	31.47
MD3	R	30	27.43	9.61	27.36	23.78	0.58	23.57
	$R_{\lambda=0}$	30	19.27	1.66	36.19	23.64	0.10	23.54
MD4	R	30	19.17	2.68	79543.73	66962.58	1766.70	64430.08
	$R_{\lambda=0}$	30	17.23	1.28	98682.65	66789.78	1800.28	64392.92

Table 6.9: Results on all problems for RBF surrogate with and without λ: number of feasible runs $\#F$, mean number of function evaluations to reach a feasible solution N^I, its standard deviation $\sigma_{N^{(I)}}$, mean first feasible objective value $f^{(I)}$, mean best feasible objective value $f^{(II)}$, its standard deviation $\sigma_{f^{(II)}}$ and minimum feasible objective value $f^{(II)}_{\min}$ on the 30 runs. A star $^{(*)}$ indicates that the experiment was stopped due to long computational times.

6.7 Large-budget experiments

In Section 6.6.1, MARS and cubic RBF exhibited the best results. However, as the construction of MARS models is computationally more expensive, cubic RBF is chosen for the larger budget experiments.

BOA used with cubic RBF surrogates is compared with two surrogate-assisted variants of NOMAD. To do so, cubic RBF and Gaussian kriging models are given to NOMAD v3.9.1 as external surrogates and are updated after each blackbox evaluation. Each run of NOMAD is given as starting point the first point of the initial DOE used for BOA, that is the first point of the SLHD.

The truncation to the median is applied to NOMAD for the surrogate construction: it is applied to the constraints after $2 \cdot (n+1)$ blackbox evaluations, and to the feasible objective values when the number of feasible points evaluated is greater than $2 \cdot (n+1)$.

It is well noticed that external surrogates are not employed in NOMAD when it is used inside BOA since the subproblem tackled are already based on models of the blackbox functions.

In the following experiments, the blackbox evaluation budget is set at 400 and each subproblem resolution of BOA uses a maximum of $100 \cdot n$ surrogate evaluations. We denote $\mathrm{B_R}$ the cubic RBF-based BOA and, respectively, $\mathrm{N_R}$ and $\mathrm{N_K}$ the NOMAD variants assisted with cubic RBF and Gaussian kriging.

6.7.1 Algorithm comparisons on benchmark problems

The first experiments with the larger evaluation budget are performed using the problems from the literature.

Table 6.10 presents the results on 18 of the problems as the two NOMAD variants did not find any feasible run on MD1 and BOA was stopped on this problem due to a long computational time (subsequent to the increase of the budget). BOA finds more feasible solutions on the continuous, integer and mixed-discrete problems and is equivalent to both variants of NOMAD on the problems with mixed continuous and discrete variables. There is, in particular, an important gap on the application problem MD4 on which all BOA runs are feasible against only 2 for NOMAD.

It is well noticed that on C1 the methods using cubic RBF models perform less than the kriging-based NOMAD regarding the number of feasible runs, suggesting that kriging captures better the complexity of this instance. Comparing the RBF-based methods on this problem, the results show on a similar number of feasible runs that BOA uses less blackbox evaluations to find the first feasible solution and the latter has on average a

better objective value, which is interesting for restricted evaluation budgets. The best solution from all runs on C1 was found by BOA.

Pb	$\mathcal{A}$	$\#F$	$N^{(I)}$	$\sigma_{N^{(I)}}$	$f^{(I)}$	$f^{(II)}$	$\sigma_{f^{(II)}}$	$f^{(II)}_{\min}$
C1	B_R	18	113.61	115.73	-9.89	-10.09	3.59	-14.99
	N_R	19	177.68	94.39	-4.27	-11.95	2.15	-14.06
	N_K	27	211.19	88.39	-3.69	-10.72	2.86	-14.53
C2	B_R	30	45.80	19.10	52.68	24.82	0.71	24.33
	N_R	30	177.77	63.63	926.49	101.51	103.95	30.72
	N_K	30	181.70	61.52	630.81	78.06	54.80	31.91
C3	B_R	30	61.70	40.51	1079.54	105.30	105.20	47.57
	N_R	30	139.03	69.43	6662.72	1748.93	2573.30	76.16
	N_K	29	146.28	70.25	7874.00	2526.50	3282.22	62.99
C4	B_R	25	186.00	71.53	5006.10	3877.70	4147.88	6.62
	N_R	7	295.43	73.70	1480.99	1031.72	2637.85	0.38
	N_K	10	245.70	108.96	42649.96	6679.82	16668.94	0.74
C5	B_R	30	50.20	29.37	613.29	530.77	3.75	525.62
	N_R	30	21.10	15.55	1040.14	619.79	50.90	557.09
	N_K	30	21.43	15.62	1042.35	609.03	38.15	550.50
C6	B_R	30	55.77	9.77	-4435.25	-5407.09	148.44	-5642.83
	N_R	30	56.83	51.49	-4587.86	-5425.54	124.31	-5770.14
	N_K	30	42.90	33.29	-4634.85	-5457.70	144.14	-5825.52
I1	B_R	30	27.97	9.28	-12.17	-14.80	0.61	-15.00
	N_R	22	220.36	110.92	-6.73	-14.18	1.74	-15.00
	N_K	23	208.09	108.30	-6.96	-14.04	1.58	-15.00
I2	B_R	30	36.93	10.94	19.90	13.00	0.00	13.00
	N_R	30	84.60	66.00	20.27	13.20	0.76	13.00
	N_K	30	75.93	57.12	20.80	13.20	0.76	13.00
I3	B_R	30	36.03	33.46	-44120.83	-50046.83	102.62	-50199.00
	N_R	23	198.96	91.48	-41986.39	-50067.00	253.23	-50200.00
	N_K	25	231.76	80.64	-42354.84	-49622.92	1992.40	-50200.00
MI1	B_R	30	22.47	4.75	-0.31	-0.90	0.05	-0.94
	N_R	30	5.87	6.01	-0.09	-0.83	0.06	-0.92
	N_K	30	5.63	5.02	-0.10	-0.85	0.06	-0.92

Continued on next page

Continued from previous page

Pb	$\mathcal{A}$	$\#F$	$N^{(I)}$	$\sigma_{N^{(I)}}$	$f^{(I)}$	$f^{(II)}$	$\sigma_{f^{(II)}}$	$f^{(II)}_{\min}$
MI2	B_R	30	29.47	49.03	14.84	6.10	0.58	5.54
	N_R	30	54.93	24.36	14.01	6.49	1.02	4.64
	N_K	30	59.47	28.51	12.71	5.74	0.66	4.63
MI3	B_R	29	60.62	32.88	-0.98	-0.99970	0.00017	-0.99988
	N_R	30	18.57	16.78	-0.06	-0.99917	0.00063	-0.99983
	N_K	30	23.30	30.99	-0.11	-0.99914	0.00058	-0.99977
MI4	B_R	30	56.33	36.63	-0.93	-0.99917	0.00065	-0.99998
	N_R	30	12.70	0.53	-0.46	-0.99996	0.00004	-0.99999
	N_K	30	12.70	0.53	-0.46	-0.99996	0.00005	-0.99999
MI5	B_R	30	55.53	24.46	73176.52	62192.92	5320.47	58505.73
	N_R	30	92.50	53.86	133934.17	76819.02	10584.92	54844.21
	N_K	30	97.77	71.98	143147.70	75466.98	10945.25	56623.22
MI6	B_R	30	28.37	23.97	19714.63	16984.42	26.82	16958.23
	N_R	30	47.43	24.64	27051.55	17042.98	144.22	16959.28
	N_K	30	52.13	35.88	30589.19	17021.32	106.00	16962.30
MD2	B_R	30	29.90	7.90	142.64	32.02	2.29	31.42
	N_R	30	147.83	58.26	1035.47	85.38	64.41	35.16
	N_K	30	157.53	55.40	872.32	90.61	107.76	35.58
MD3	B_R	30	38.97	18.51	24.86	23.62	0.09	23.53
	N_R	30	28.30	27.12	32.62	24.56	0.96	23.60
	N_K	30	26.87	22.60	33.90	24.32	0.62	23.67
MD4	B_R	30	22.80	7.13	75047.08	66812.74	2040.08	64335.74
	N_R	2	62.00	2.83	97030.00	70671.61	1229.81	69802.00
	N_K	2	126.50	16.26	75033.07	66892.43	836.45	66300.98

Table 6.10: Results on 18 problems for BOA with RBF, NOMAD with RBF and NOMAD with KG: number of feasible runs $\#F$, mean number of function evaluations to reach a feasible solution N^I, its standard deviation $\sigma_{N^{(I)}}$, mean first feasible objective value $f^{(I)}$, mean best feasible objective value $f^{(II)}$, its standard deviation $\sigma_{f^{(II)}}$ and minimum feasible objective value $f^{(II)}_{\min}$ on the 30 runs.

On more than half of the problems, BOA uses less function evaluations to find a first feasible candidate solution and the latter is on average better than the ones found by the

two variants of NOMAD.

6.7.2 Algorithm comparisons on RSMLateralCrash

The methods tested in Section 6.7.1 are used on the RSM-based lateral crash study from Stellantis with parallel and sequential evaluations.

Parallel evaluations

In order to reduce the total computational time of the optimization, the parallel version of BOA is used. For the same reason, block evaluations are allowed in both variants of NOMAD. The maximum number of parallel blackbox evaluations is set at 25 in BOA and NOMAD, and two runs are performed for each method.

It is well noticed that BOA performs the maximum number of parallel blackbox evaluations allowed at each iteration, and possibly less only at the last evaluation to not exceed the evaluation budget. Differently, NOMAD does not use the whole parallelization capacity at every iteration and can exceed the evaluation budget at its last iteration. Possible extra evaluations are not considered in the analyses.

Figures 6.1 and 6.2 present the evolution of the best feasible objective values for each method during the first and second run, respectively. The results are also summarized in Table 6.11 for each run. The number of blackbox evaluations to reach a feasible solution ($N^{(I)}$) is computed as the number of expensive evaluations performed at the end of the batch to which the feasible point belongs to.

Comparing the parallel methods, during the first run, NOMAD uses less evaluations to find a feasible solution, as shown in the lines of "run 1" of Table 6.11. This is partly due to the fact that BOA does not start with a single point but with a DOE consisting of only infeasible points. Nonetheless, a gap in the objective values is observed on Figure 6.1 when BOA finds its first feasible solution: 0.051 against 0.054 and 0.053 for NOMAD with RBF and kriging, respectively. During the last fourth of the evaluations, the RBF-assisted NOMAD performs better than BOA.

Considering now the second run presented on Figure 6.2 and in the lines corresponding to "run 2" in Table 6.11, there are clear gaps in the performance of the three solvers. The kriging-assisted NOMAD is effective in finding a feasible solution and performs better than the RBF-assisted NOMAD. The latter is outperformed by the other methods. BOA outperforms the surrogate-assisted NOMAD solvers as soon as it finds a feasible solution.

In both runs, although the plots corresponding to BOA start with a clear advantage on the objective value, the decrease seems globally slower than in NOMAD.

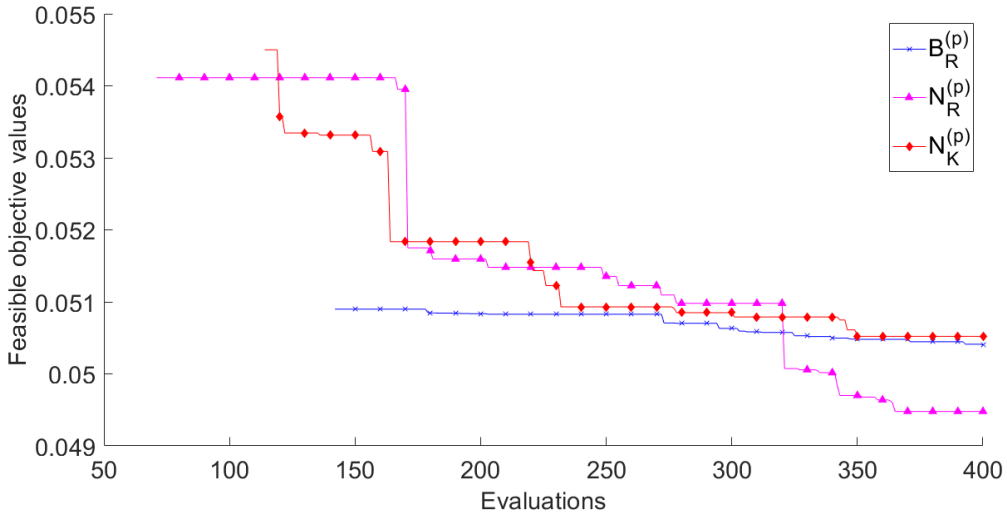


Figure 6.1: Evolution of the best feasible objective values of RSMLateralCrash according to the number of evaluations for "run 1" of parallel BOA with cubic RBF ($B_R^{(p)}$), parallel NOMAD with cubic RBF ($N_R^{(p)}$) and parallel NOMAD with Gaussian kriging ($N_K^{(p)}$).

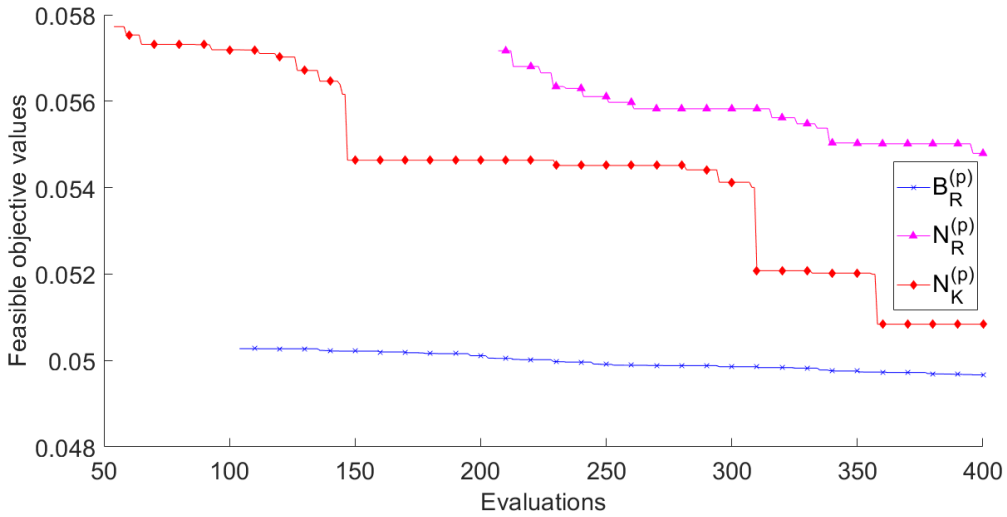


Figure 6.2: Evolution of the best feasible objective values of RSMLateralCrash according to the number of evaluations for "run 2" of parallel BOA with cubic RBF ($B_R^{(p)}$), parallel NOMAD with cubic RBF ($N_R^{(p)}$) and parallel NOMAD with Gaussian kriging ($N_K^{(p)}$).

Run	$\mathcal{S}$	$N^{(I)}$	$f^{(I)}$	$f^{(II)}$
1	$\mathrm{B}_{\mathrm{R}}^{(\mathrm{p})}$	152	0.051	0.050
	$\mathrm{N}_{\mathrm{R}}^{(\mathrm{p})}$	71	0.054	0.049
	$\mathrm{N}_{\mathrm{K}}^{(\mathrm{p})}$	114	0.054	0.051
2	$\mathrm{B}_{\mathrm{R}}^{(\mathrm{p})}$	127	0.050	0.050
	$\mathrm{N}_{\mathrm{R}}^{(\mathrm{p})}$	207	0.057	0.055
	$\mathrm{N}_{\mathrm{K}}^{(\mathrm{p})}$	54	0.058	0.051

Table 6.11: Results for each run on RSMLateralCrash for parallel BOA with RBF ($\mathrm{B}_{\mathrm{R}}^{(\mathrm{p})}$), parallel NOMAD with cubic RBF ($\mathrm{N}_{\mathrm{R}}^{(\mathrm{p})}$) and parallel NOMAD with Gaussian kriging ($\mathrm{N}_{\mathrm{K}}^{(\mathrm{p})}$): number of function evaluations to reach a feasible solution N^I, first feasible objective value $f^{(I)}$ and best feasible objective value $f^{(II)}$.

Sequential evaluations

In order to see how parallelization affects the performance of the algorithms, one run with the classical sequential versions was also performed for each of them. The comparison of the three methods is depicted in Figure 6.3 for the evolution of the best feasible objective values and summarized in Table 6.12. The results of NOMAD are identical to those of the case using the block evaluation option. Thus, the parallelization does not seem to affect the performance of NOMAD on this problem. The situation is different for BOA whose internal strategy is modified to deal with simultaneous blackbox calls. In this setting, BOA outperforms NOMAD and both the first and best feasible solutions of BOA have better objective values than those found by the two variants of NOMAD.

$\mathcal{S}$	$N^{(I)}$	$f^{(I)}$	$f^{(II)}$
B_{R}	97	0.047	0.047
N_{R}	71	0.054	0.049
N_{K}	114	0.054	0.051

Table 6.12: Results for one run on RSMLateralCrash for BOA with cubic RBF (B_{R}), NOMAD with cubic RBF (N_{R}) and NOMAD with Gaussian kriging (N_{K}): number of function evaluations to reach a feasible solution N^I, first feasible objective value $f^{(I)}$ and best feasible objective value $f^{(II)}$.

Focusing on BOA, Figure 6.4 depicts the results for one run of the classical sequential version of BOA used with cubic RBF compared with its parallel version. The sequential version needs less evaluations to leave Phase I and the feasible objective value of the best candidate is better than in the parallel version. Thus, the parallel resolution of the subproblems reduces the performance of the algorithm. This is partly due to a higher quality

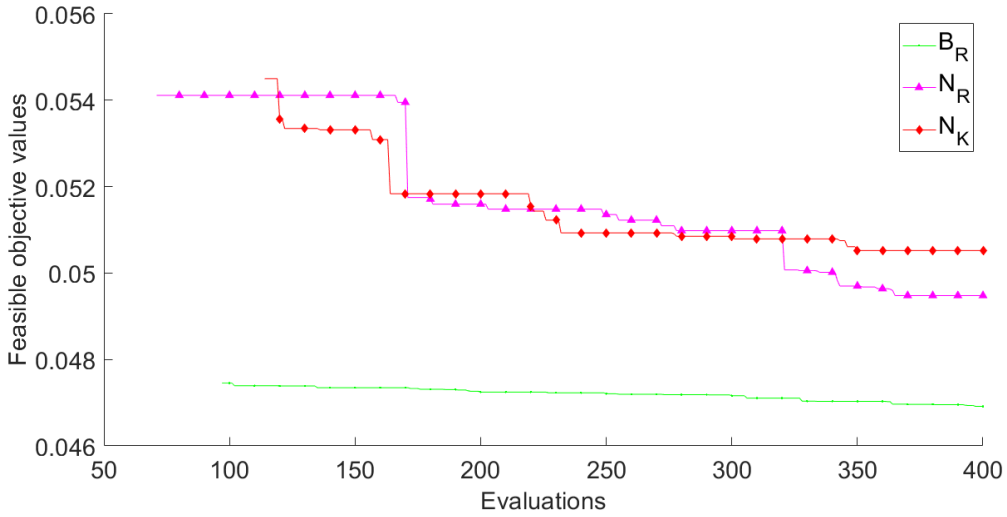


Figure 6.3: Evolution of the best feasible objective values of RSMLateralCrash according to the number of evaluations for one run of BOA with cubic RBF (B_R), NOMAD with cubic RBF (N_R) and NOMAD with Gaussian kriging (N_K).

of the surrogate in the sequential strategy where it is updated after each expensive evaluation. On this optimization problem where the costs of the blackbox calls are significantly shortened as it uses surrogates instead of the expensive finite element simulations, the total computational time was however reduced by two days thanks to the use of parallel evaluations (from approximately 9 to 7 calculation days).

6.7.3 Algorithm comparisons on LateralCrash

The parallel version of BOA using cubic RBF is tested on the lateral crash design problem from Stellantis. This problem is computationally very expensive as it makes use of finite element simulations of the vehicle and each call takes more than 12 hours. As a solution is commonly desired in a time window of two weeks, assuming that the time needed by a solver to generate a new candidate point is negligible, only a maximum of 28 evaluations are possible in a sequential mode, hence the need to evaluate candidate solutions in parallel. Furthermore, failures are frequently observed in vehicle simulations, which constitutes an additional challenge.

One run of parallel BOA is performed starting from an initial DOE that consists of 72 infeasible points generated with the same strategy used in the previous experiments. To cope with the long computational times, the maximum number of surrogate evaluations for each subproblem resolution is set to $50 \cdot n$ and block surrogate evaluations of maximum

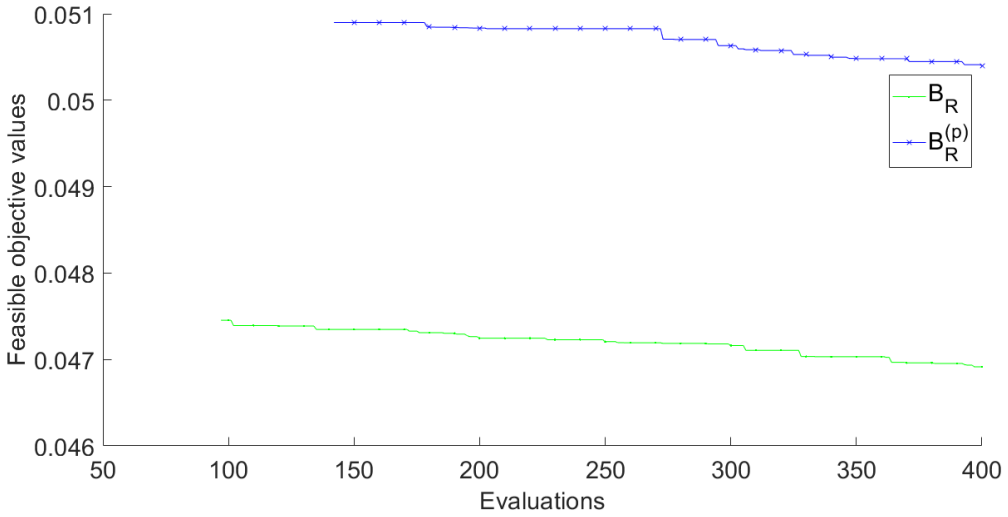


Figure 6.4: Evolution of the best feasible objective values of RSMLateralCrash according to the number of evaluations for one run of BOA with cubic RBF (B_R) and the parallel version ($B_R^{(p)}$).

50 points are allowed. The capacity of parallel blackbox evaluations is kept at 25.

The evolution of the best feasible objective values is presented in Figure 6.5 and Table 6.13 summarizes the results. The best solution of the initial DOE corresponds to an infeasible mass of 90.017kg and a best sum of squared constraint violations of $1.553 \cdot 10^5$. A feasible solution is found after 222 blackbox evaluations and the best feasible mass obtained at the end of the evaluation budget is 75.719kg. Hence, despite the difficulty of this mixed-discrete problem with 48 variables and 64 inequality constraints, parallel BOA reached a feasible solution after only 6 iterations. Indeed, the algorithm first evaluates the initial DOE before entering any phase, so $222 - 72 = 150$ expensive evaluations are actually performed during Phase I, which corresponds to 6 batches of blackbox evaluations. Moreover, it is well noticed that 31 failures of the blackbox occurred during the optimization but they did not stop BOA as its design comprises the handling of such hidden constraints.

$\mathcal{S}$	$N^{(I)}$	$f^{(I)}$	$f^{(II)}$
$B_R^{(p)}$	222	75.998	75.719

Table 6.13: Results for one run on LateralCrash for parallel BOA with RBF ($B_R^{(p)}$): number of function evaluations to reach a feasible solution N^I, first feasible objective value $f^{(I)}$ and best feasible objective value $f^{(II)}$.

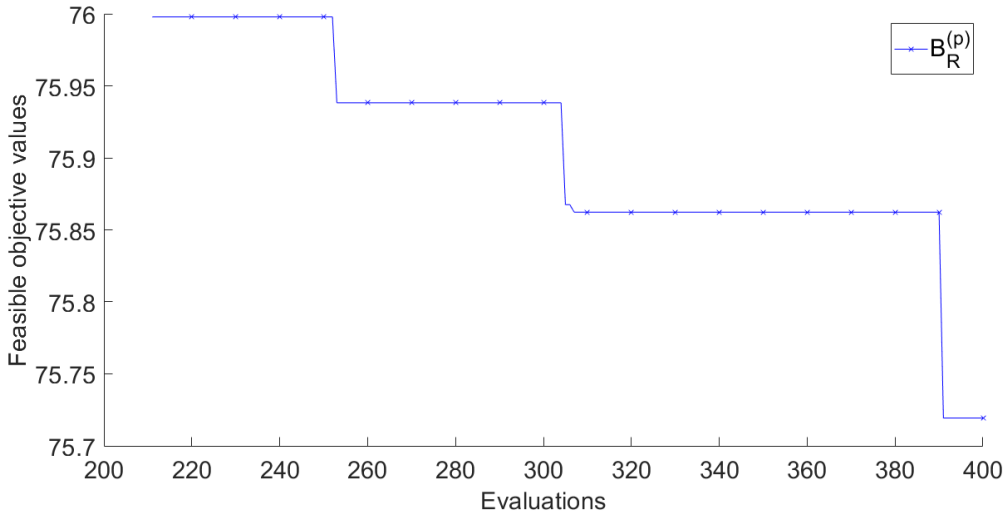


Figure 6.5: Evolution of the best feasible objective values of LateralCrash according to the number of evaluations for one run of parallel BOA with RBF ($B_R^{(p)}$).

6.8 Final remarks

This chapter presents a new surrogate-based generic method for solving expensive constrained blackbox optimization problems with mixed variables. The developed algorithm, entitled BOA, is flexible in terms of the type of models used or the solver used on its subproblems, and does not need feasible initial points thanks to its two-phase structure.

An extension of BOA for parallel evaluations is also described for real-world applications where launching batches of evaluations is essential to expect a reasonably good solution in a restricted time.

The results for different numerical experiments have been presented, using constrained optimization problems with more than 10 variables and up to 48. First comparisons of six kinds of surrogate models used for the subproblems of BOA are performed on 19 problems from the literature, including applications. They exhibit globally better performance of MARS and cubic RBF in terms of mean number of evaluations to reach a feasible solution and quality of the solution. In practice, the construction of MARS models is however computationally longer. Among the kriging types tested, the use of exponential correlation functions seems unfavourable in the presence of discrete variables.

Other experiments investigate the parameter λ that takes into account the objective value in the search of a feasible candidate solution. Comparing cubic RBF-based BOA with and without λ shows generally lower objective values at the end of Phase I when the

parameter is used but more evaluations are performed in this phase of the algorithm. In general, however, the solution obtained at the end of the optimization is still better with λ.

Then, BOA using cubic RBF is compared with two surrogate-assisted NOMAD variants and using a higher evaluation budget. The results on the literature problems show an equivalent ability of the three methods to find feasible solutions on mixed-discrete problems and a higher performance of BOA on the other types of problems. Regarding the quality of the solution, BOA globally finds lower objective values than the RBF- and kriging-assisted NOMAD. These methods are also tested on two automotive problems encountered at Stellantis.

On the response surface-based pole lateral crash problem, two runs are performed using parallel versions of each solver. They show that the first feasible solutions found by BOA have better objective values than the current best solutions of NOMAD variants at the corresponding number of evaluations. However, the number of evaluations used to find a feasible solution is better for at least one of the two NOMAD variants. On the final solutions identified, BOA is competitive or better than NOMAD. Comparing sequential and parallel versions of the three solvers on one run, the performance of NOMAD is unchanged whereas BOA outperforms its parallel extension both on the number of evaluations used in Phase I and on the best solution found.

Furthermore, BOA using parallel evaluations and cubic RBF was successfully applied to a real-world high-dimensional optimization problem from the automotive group Stellantis. Starting from an infeasible DOE, the optimization reached a feasible solution in 6 iterations only and successfully managed the failures of the blackbox.

As a summary, this study exhibits an efficiency of the first phase of BOA in finding good feasible solutions and shows advantages of the method in a context of restricted evaluation budgets, which is often the case in industry. It can be considered as a relevant method for solving real-world expensive blackbox optimization problems with mixed variables and inequality constraints.

7

Conclusion and future challenges

7.1 Conclusion

This thesis explores some deterministic and stochastic optimization methods in a blackbox context for purposes of solving real-world blackbox optimization problems with constraints and mixed variables and where the functions involved are computationally expensive. Such complex problems are notably encountered in the automotive industry.

The first contributions are the benchmark of derivative-free and gradient-based optimization methods of the SciPy library of Python, published in [Varelas and Dahito, 2019]. The experiments on the continuous `bbob` suite of COCO show effectiveness of SLSQP on the whole testbed and of the differential evolution algorithm on multimodal functions with adequate global structure. Moreover, adaptations of the parameters of the Nelder-Mead method seem necessary in high dimensions. Improvements are also observed for some methods compared to their previous implementations in SciPy.

As second contribution, the performance of the ORTHOMADS instantiation of the well-known MADS algorithm is evaluated in [Dahito et al., 2021]. The experiments comparing different poll direction types of the method exhibit the effectiveness of the setting `ORTHO N+1 NEG` on both continuous and mixed-integer optimization problems from the literature. The variants `ORTHO 2N` and `ORTHO N+1 QUAD` also show good performance respectively on

the continuous and mixed-integer instances.

Compared with other deterministic and stochastic methods, while ORTHOMADS shows advantages mainly in small dimensions on continuous problems and on continuous multimodal problems with global structure, the algorithm is globally effective on mixed-integer instances and is particularly competitive for limited budgets.

Other experiments evaluating the contribution of the search phase of the algorithm show that its use is advantageous for the performance of the algorithm on both testbeds.

A third contribution is the design and implementation of a finite element test case representing a clamped truss subject to a force applied on one of its nodes. Optimization problems minimizing the weight, the cost and the compliance of the structure were solved using ORTHOMADS, CMA-ES and NSGA-II. Mixed optimization problems are considered with, in particular, continuous thicknesses and categorical materials. The results show the effectiveness of ORTHOMADS especially for cost and compliance optimizations. Globally, CMA-ES needs less blackbox evaluations than NSGA-II to approach a good solution and competes with ORTHOMADS after large evaluation budgets.

A final contribution of this thesis is the design and implementation of a surrogate-based algorithm, called BOA, that enables the use of different types of metamodels. BOA is a two-phase algorithm designed for expensive blackbox optimization problems with mixed variables and inequality constraints. It first seeks a feasible solution of the problem by iteratively solving subproblems defined by surrogates of the blackbox functions while the second phase aims at improving the objective value of the best solution so far. The algorithm takes into account the possibility of failures of the blackbox evaluations as it commonly occurs in real-world optimization problems requiring numerical simulations.

Experiments were performed on considered medium- to large-dimensional instances from the literature but also engineering applications, all with inequality constraints. The comparisons of BOA with RBF, kriging with different correlation functions and MARS show globally better performance of RBF and MARS, although the latter is computationally more intensive. Compared with a RBF- and a kriging-assisted ORTHOMADS, the performance of BOA used with RBF is similar to that of ORTHOMADS in the search of a feasible solution on mixed-discrete instances. BOA is globally better to find a feasible solution with a good objective value and is especially competitive for restricted budgets.

Finally, in order to take advantage of possible parallelization of the blackbox calls, an extension of the algorithm to parallelization is also presented. It is used on two application problems from the automotive industry and also shows the effectiveness of BOA for restricted evaluation budgets.

7.2 Future challenges

The research performed in this thesis mainly assumes that all objective and constraint functions are expensive to evaluate. However, for some real-world optimization problems, only some of the functions are expensive to query. In the automotive industry in particular, the objective functions are often cheap to get: computing the weight or the cost of some parts on a body in white usually takes a few minutes, which is negligible in comparison to the common 12 hours needed to evaluate crash outputs. A separate handling of cheap functions, by using only their true evaluations for instance, may improve the resolutions of such optimization problems. The evaluations of the objective and constraint functions are however often performed simultaneously so this has to be taken into account.

This thesis also presents the derivative-free algorithm BOA, whose main parameters were set based on statistical analyses. Further experiments are considered to optimize some parameters such as the type and size of the DOE used, the choice of initial points considered for the resolutions of the subproblems, the solver used for their resolutions or the stopping condition of phase I.

Furthermore, a better decrease of the objective value should be possible with improvements of the second phase of the algorithm, for instance by considering alternative formulations of the corresponding subproblem.

Moreover, different types of surrogate models were considered in BOA and works on ensembles of surrogates exist in the literature. As each kind of surrogate can be preferable in some contexts, exploring ensembles of surrogates in BOA would be beneficial. One could consider for instance, to switch from a surrogate to another based on the predicted accuracy of the model or the predicted improvement of the solution.

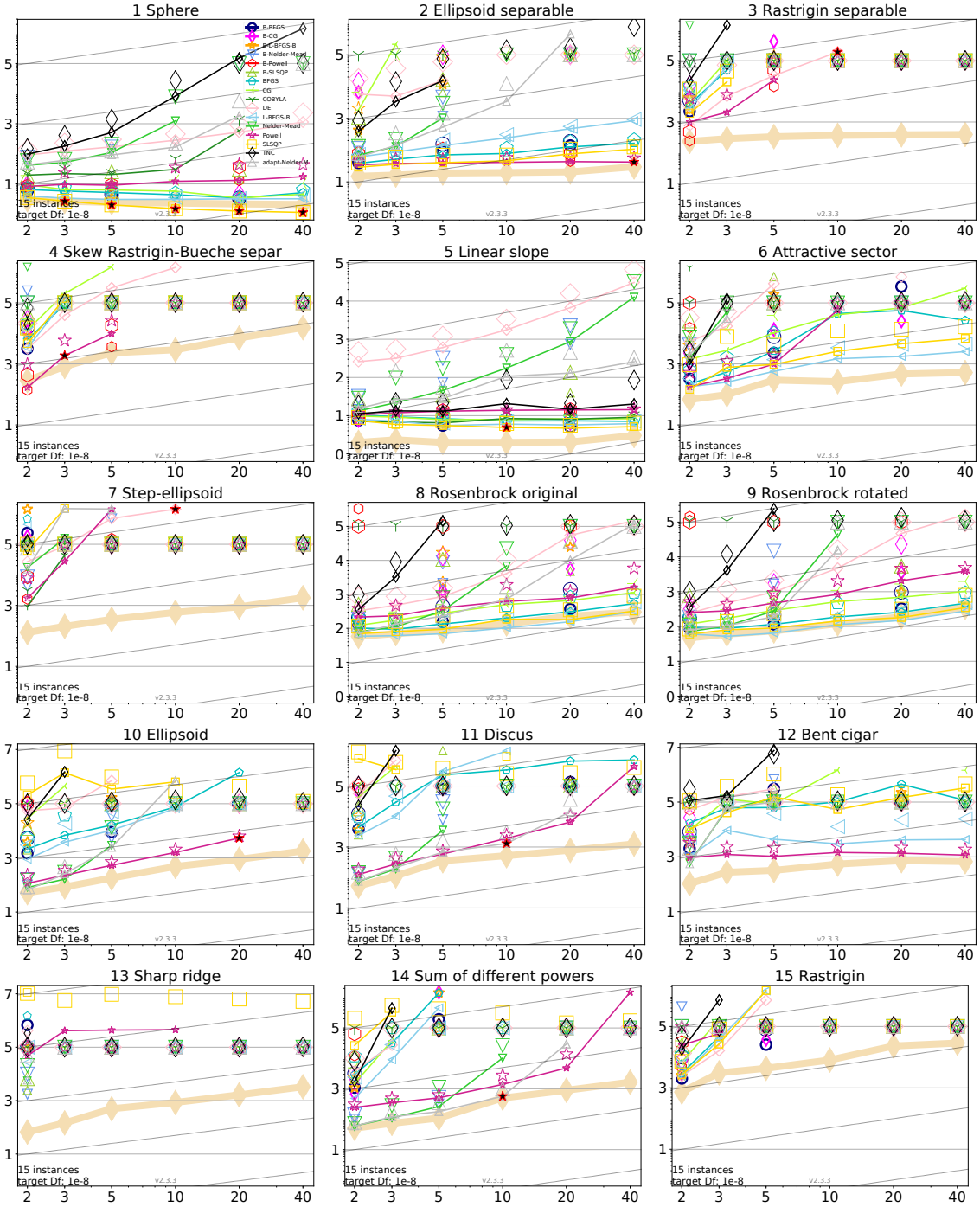
Equality constraints can be treated in the algorithm by turning them into two inequality constraints with a small margin but this may not be optimal. Thus, a specific handling of equality constraints would be interesting.

A specific handling of categorical variables is also envisaged, notably through the type of norm employed in the algorithm.

Finally, the presence of noise is a matter for future work, as well as the identification of convex areas, thanks to convex underestimators for instance, to locate the minima of the functions.

A

Performance evaluations of continuous algorithms in a BBO context



Continued on next page

Continued from previous page

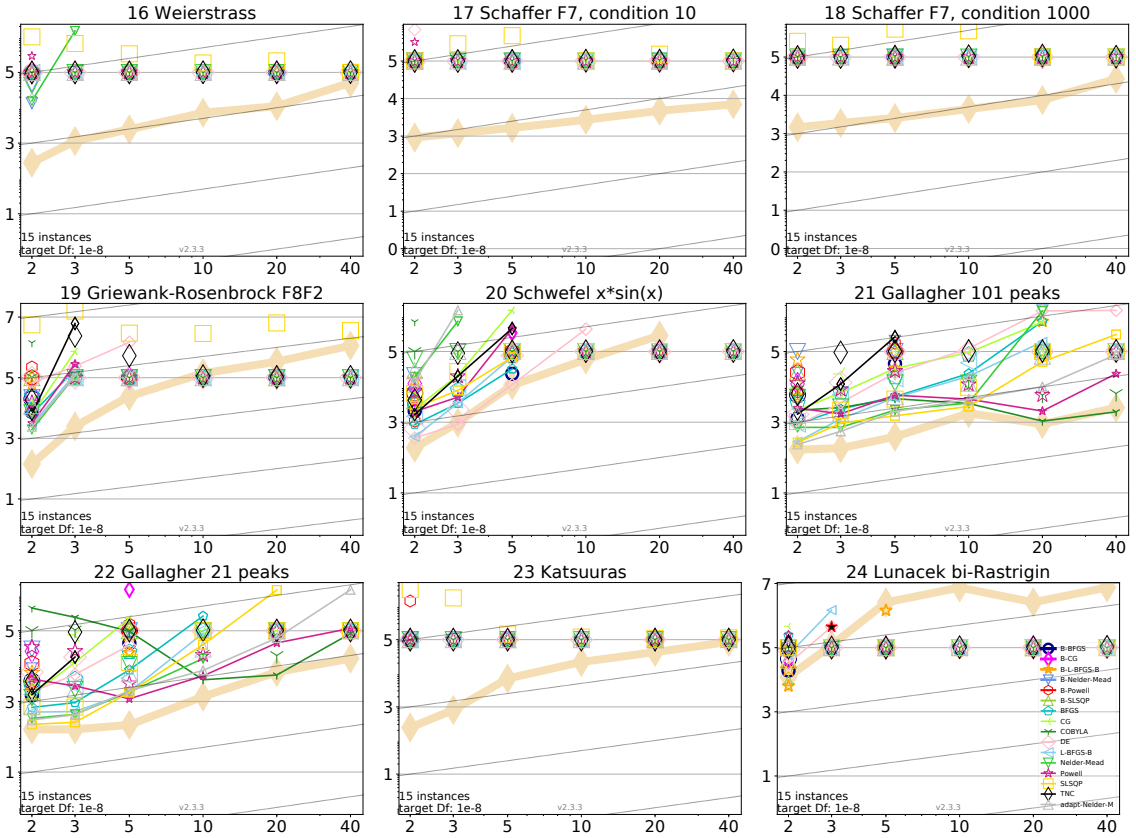
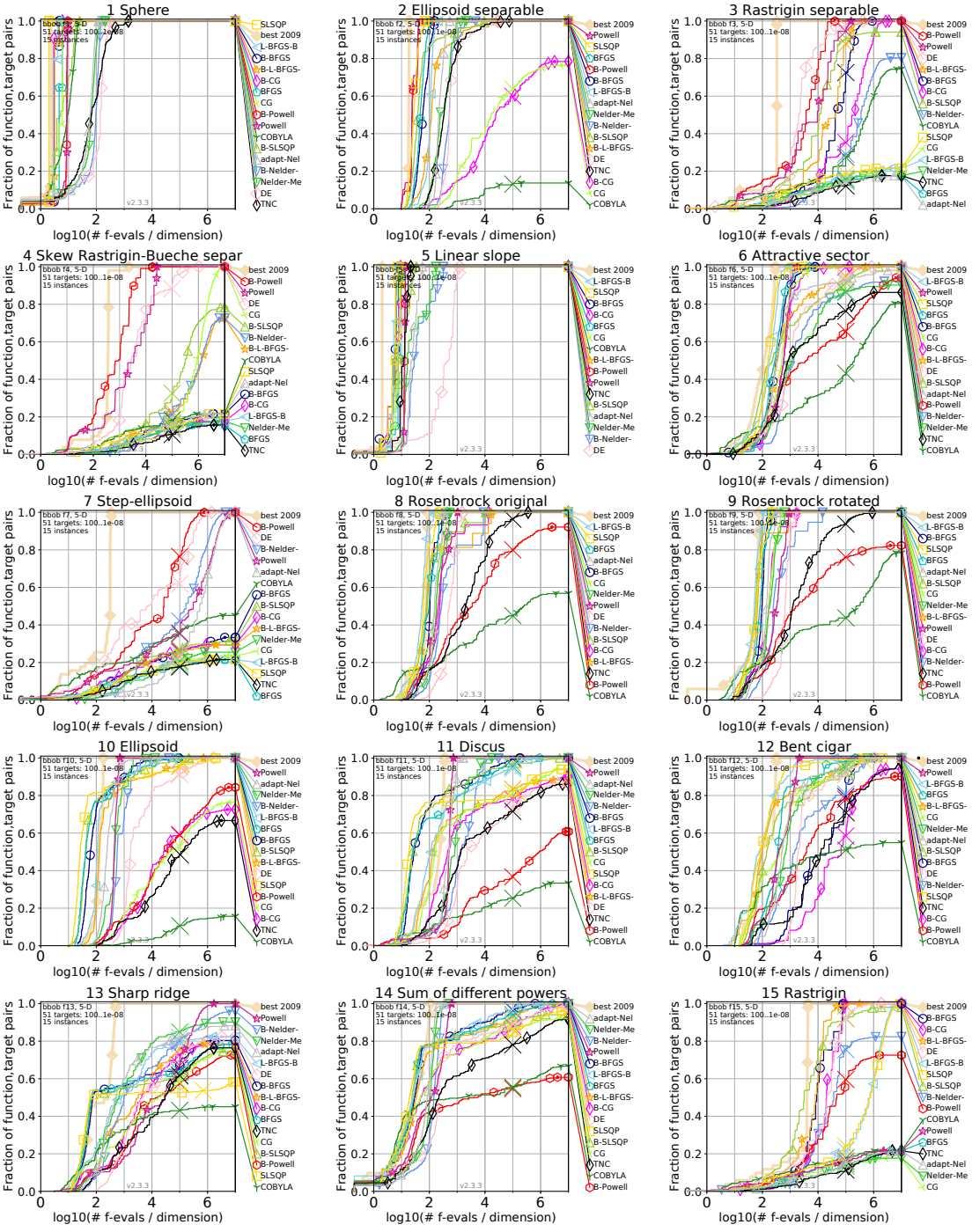


Figure A1: Average running time (aRT in number of function evaluations as $\log_{10}$ value), divided by dimension for target function value 10^{-8} versus dimension. Slanted grid lines indicate quadratic scaling with the dimension. Different symbols correspond to different algorithms given in the legend of f_1 and f_{24}. Light symbols give the maximum number of function evaluations from the longest trial divided by dimension. Black stars indicate a statistically better result compared to all other algorithms with $p < 0.01$ and Bonferroni correction number of dimensions (six). Legend: ○: B-BFGS, ◇: B-CG, ⋆: B-L-BFGS-B, ▽: B-Nelder-Mead, ⬡: B-Powell, △: B-SLSQP, ⬠: BFGS, ⋋: CG, ⋎: COBYLA, ◇: DE, ◁: L-BFGS-B, ▽: Nelder-Mead, ⋆: Powell, □: SLSQP, ◇: TNC, △: adapt-Nelder-Mead



Continued on next page

Continued from previous page

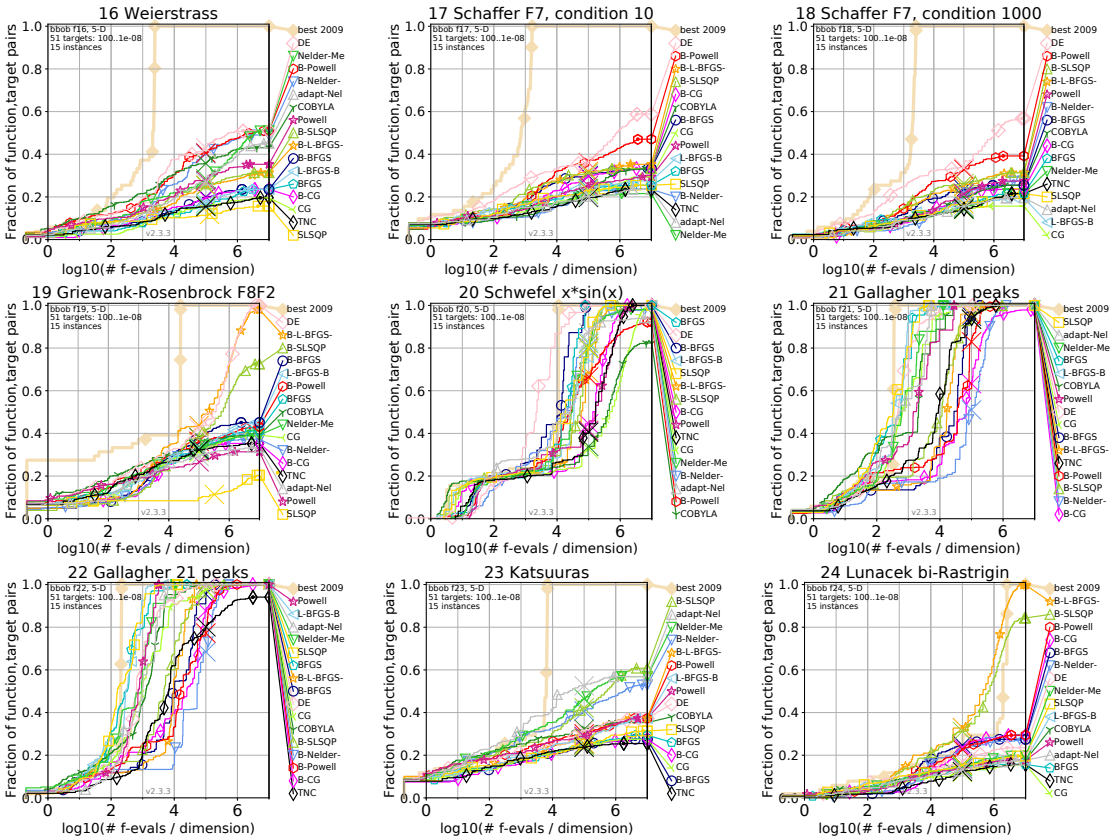
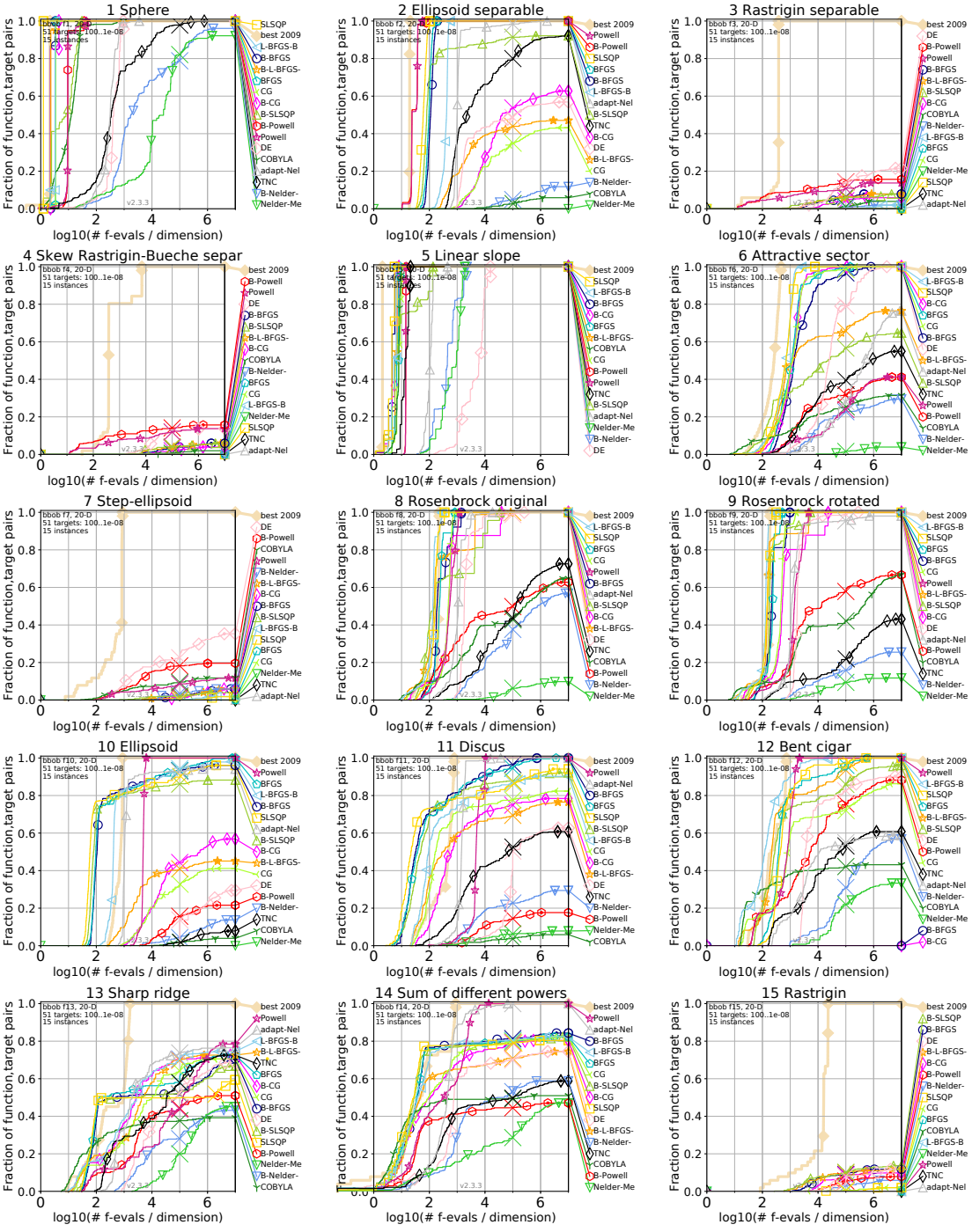


Figure A2: Empirical cumulative distribution of simulated (bootstrapped) runtimes, measured in number of objective function evaluations, divided by dimension (FEvals/DIM) for the 51 targets $10^{[-8,\ldots,2]}$ in dimension 5.



Continued on next page

Continued from previous page

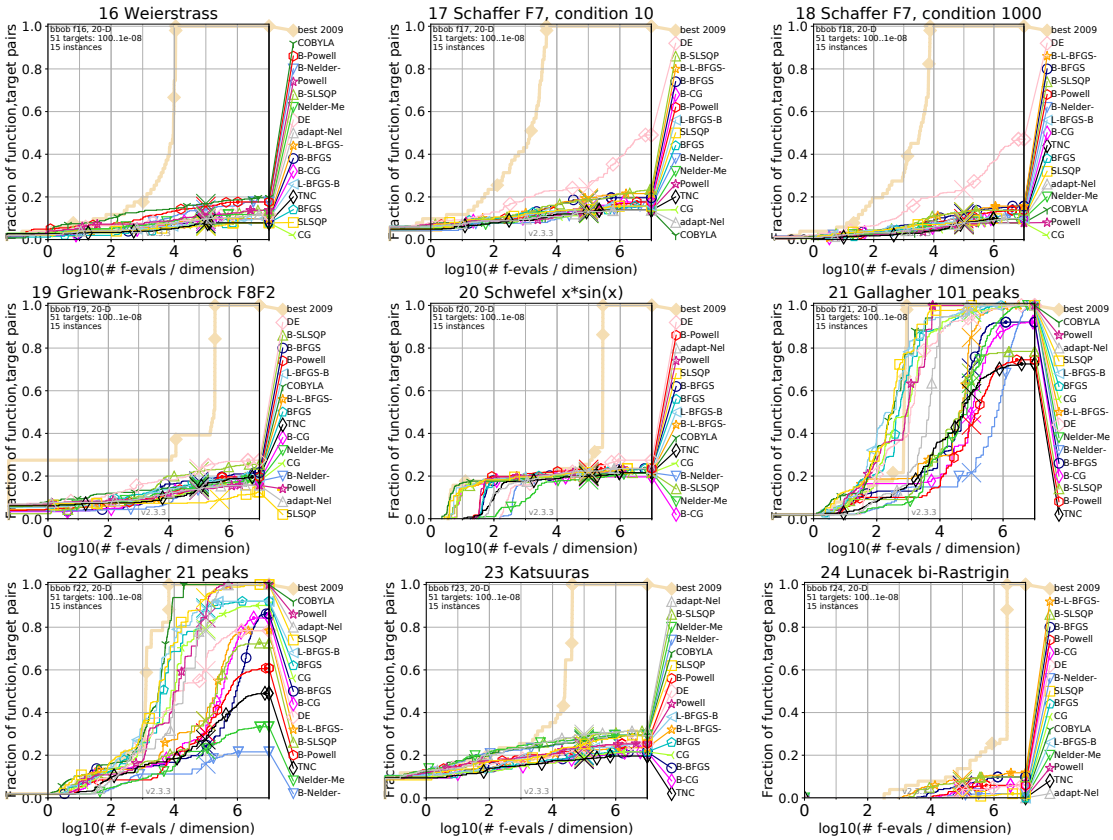


Figure A3: Empirical cumulative distribution of simulated (bootstrapped) runtimes, measured in number of objective function evaluations, divided by dimension (FEvals/DIM) for the 51 targets $10^{[-8,\ldots,2]}$ in dimension 20.

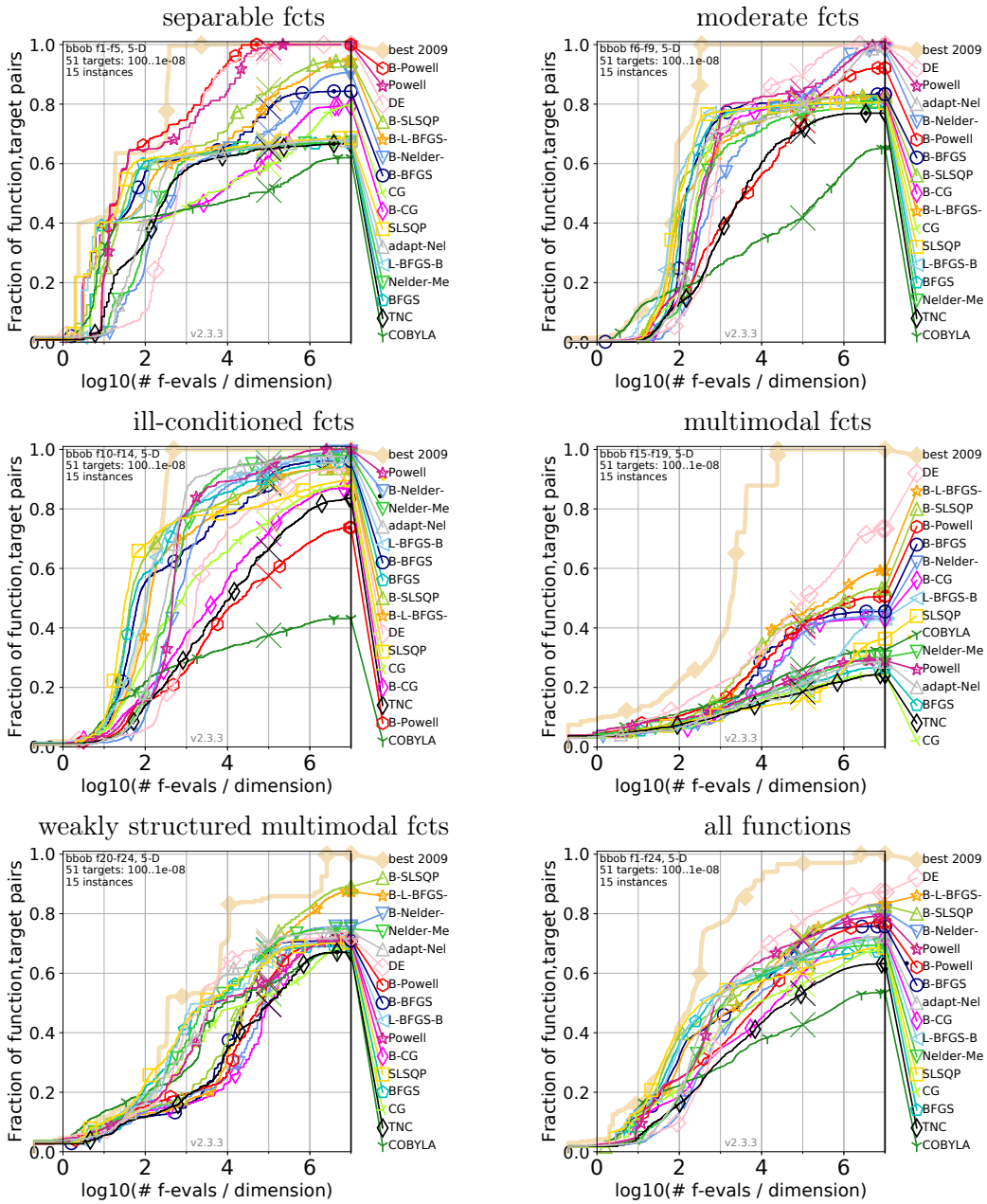


Figure A4: Empirical cumulative distribution of simulated (bootstrapped) runtimes, measured in number of function evaluations, divided by dimension (FEvals/DIM) for the 51 targets $10^{[-8,\ldots,2]}$ for all functions and subgroups in dimension 5.

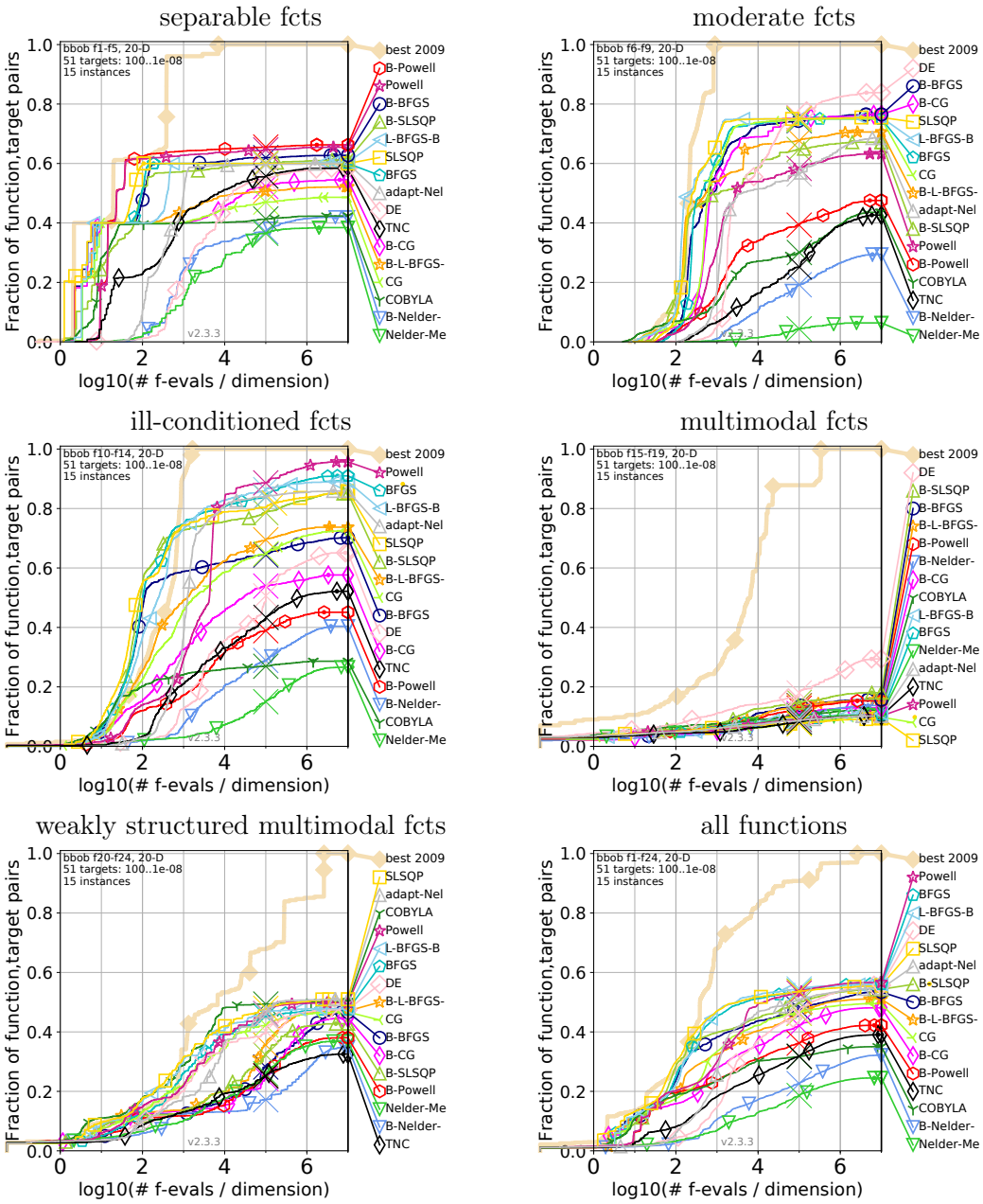


Figure A5: Empirical cumulative distribution of simulated (bootstrapped) runtimes, measured in number of function evaluations, divided by dimension (FEvals/DIM) for the 51 targets $10^{[-8,\ldots,2]}$ for all functions and subgroups in dimension 20.

Bibliography

[Abramson et al., 2009a] Abramson, M. A., Audet, C., Chrissis, J. W., and Walston, J. G. (2009a). Mesh adaptive direct search algorithms for mixed variable optimization. *Optimization Letters*, 3(1):35–47. 19

[Abramson et al., 2009b] Abramson, M. A., Audet, C., Dennis, Jr., J. E., and Le Digabel, S. (2009b). OrthoMADS: A Deterministic MADS Instance with Orthogonal Directions. *SIAM Journal on Optimization*, 20(2):948–966. 18, 63

[Allaire et al., 2014] Allaire, G., Dapogny, C., Delgado, G., and Michailidis, G. (2014). Multi-phase structural optimization via a level set method. *ESAIM: control, optimisation and calculus of variations*, 20(2):576–611. 5

[Amaran et al., 2014] Amaran, S., Sahinidis, N., Sharda, B., and Bury, S. (2014). Simulation optimization: a review of algorithms and applications. *4OR*, 12(4):301–333. 64

[Armijo, 1966] Armijo, L. (1966). Minimization of functions having Lipschitz continuous first partial derivatives. *Pacific Journal of mathematics*, 16(1):1–3. 17

[Asher et al., 2015] Asher, M. J., Croke, B. F. W., Jakeman, A. J., and Peeters, L. J. M. (2015). A review of surrogate models and their application to groundwater modeling. *Water Resources Research*, 51(8):5957–5973. 37

[Audet et al., 2008] Audet, C., Béchard, V., and Chaouki, J. (2008). Spent potliner treatment process optimization using a MADS algorithm. *Optimization and Engineering*, 9(2):143–160. 64

[Audet and Dennis, Jr., 2006] Audet, C. and Dennis, Jr., J. (2006). Mesh Adaptive Direct Search Algorithms for Constrained Optimization. *SIAM Journal on Optimization*, 17(1):188–217. 18, 19, 48

[Audet and Dennis, Jr., 2009] Audet, C. and Dennis, Jr., J. (2009). A Progressive Barrier for Derivative-Free Nonlinear Programming. *SIAM Journal on Optimization*, 20(1):445–472. 18

[Audet and Dennis Jr, 2004] Audet, C. and Dennis Jr, J. E. (2004). A pattern search filter method for nonlinear programming without derivatives. *SIAM Journal on Optimization*, 14(4):980–1010. 47

[Audet et al., 2022a] Audet, C., Hallé-Hannan, E., and Le Digabel, S. (2022a). A general mathematical framework for constrained mixed-variable blackbox optimization problems with meta and categorical variables. Technical Report G-2022-11, Les cahiers du GERAD. 43

[Audet and Hare, 2017] Audet, C. and Hare, W. (2017). *Derivative-Free and Blackbox Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, Cham, Switzerland. 19

[Audet et al., 2022b] Audet, C., Le Digabel, S., Rochon Montplaisir, V., and Tribes, C. (2022b). Algorithm 1027: NOMAD version 4: Nonlinear Optimization with the MADS algorithm. *ACM Transactions on Mathematical Software*, 48(3):35:1–35:22. 19

[Audet et al., 2019] Audet, C., Le Digabel, S., and Tribes, C. (2019). The Mesh Adaptive Direct Search Algorithm for Granular and Discrete Variables. *SIAM Journal on Optimization*, 29(2):1164–1189. 43

[Bagheri et al., 2017] Bagheri, S., Konen, W., Emmerich, M., and Bäck, T. (2017). Self-adjusting parameter control for surrogate-assisted constrained optimization under limited budgets. *Applied Soft Computing*, 61:377–393. 40

[Bakin et al., 2000] Bakin, S., Hegland, M., and Osborne, M. R. (2000). Parallel MARS algorithm based on B-splines. *Computational Statistics*, 15(4):463–484. 35

[Baudiš, 2014] Baudiš, P. (2014). COCOpf: An algorithm portfolio framework. *arXiv preprint arXiv:1405.3487*. 54, 59, 60

[Bélisle et al., 1993] Bélisle, C. J., Romeijn, H. E., and Smith, R. L. (1993). Hit-and-run algorithms for generating multivariate distributions. *Mathematics of Operations Research*, 18(2):255–266. 22

[Bergstra et al., 2011] Bergstra, J., Bardenet, R., Bengio, Y., and Kégl, B. (2011). Algorithms for hyper-parameter optimization. *Advances in neural information processing systems*, 24. 70

[Berman and Ashrafi, 1993] Berman, O. and Ashrafi, N. (1993). Optimization models for reliability of modular software systems. *IEEE Transactions on Software Engineering*, 19(11):1119–1123. 103

[Bethke, 1980] Bethke, A. D. (1980). *Genetic algorithms as function optimizers*. PhD thesis, University of Michigan. 26

[Binois, 2015] Binois, M. (2015). *Uncertainty quantification on Pareto fronts and high-dimensional strategies in Bayesian optimization, with applications in multi-objective automotive design*. PhD thesis, Ecole Nationale Supérieure des Mines de Saint-Etienne. 4

[Blelly et al., 2018] Blelly, A., Felipe-Gomes, M., Auger, A., and Brockhoff, D. (2018). Stopping criteria, initialization, and implementations of BFGS and their effect on the BBOB test suite. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '18, pages 1513–1517. ACM, Association for Computing Machinery. 56

[Booker et al., 1999] Booker, A. J., Dennis Jr, J. E., Frank, P. D., Serafini, D. B., Torczon, V., and Trosset, M. W. (1999). A rigorous framework for optimization of expensive functions by surrogates. *Structural optimization*, 17(1):1–13. 37

[Bouhlel et al., 2018] Bouhlel, M. A., Bartoli, N., Regis, R. G., Otsmane, A., and Morlier, J. (2018). Efficient global optimization for high-dimensional constrained problems by using the kriging models combined with the partial least squares method. *Engineering Optimization*, 50(12):2038–2053. 41

[Box and Wilson, 1992] Box, G. E. and Wilson, K. B. (1992). On the experimental attainment of optimum conditions. In *Breakthroughs in statistics*, pages 270–310. Springer. 31

[Bremermann, 1958] Bremermann, H. J. (1958). The evolution of intelligence. The nervous system as a model of its environment. Technical report, no.1. contract no. 477(17), Dept. of Mathematics, Univ. of Washington, Seattle. 25

[Brockhoff and Hansen, 2019] Brockhoff, D. and Hansen, N. (2019). The impact of sample volume in random search on the bbob test suite. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '19, page 1912–1919, New York, NY, USA. Association for Computing Machinery. 70

[Brooks, 1958] Brooks, S. H. (1958). A discussion of random methods for seeking maxima. *Operations research*, 6(2):244–251. 70

[Broomhead and Lowe, 1988] Broomhead, D. S. and Lowe, D. (1988). Multivariable functional interpolation and adaptive networks. *Complex Systems*, 2. 35

[Bussieck et al., 2003] Bussieck, M. R., Drud, A. S., and Meeraus, A. (2003). MINLPLib—a collection of test models for mixed-integer nonlinear programming. *INFORMS Journal on Computing*, 15(1):114–119. 103

[Cauchy, 1847] Cauchy, A. (1847). Méthode générale pour la résolution des systemes d'équations simultanées. *Comp. Rend. Sci. Paris*, 25(1847):536–538. 17

[Conn et al., 2000] Conn, A. R., Gould, N. I. M., and Toint, P. L. (2000). *Trust-Region Methods*, volume 1 of *MPS/SIAM Series on Optimization*. SIAM, Philadelphia, USA. 38

[Crélot et al., 2017] Crélot, A.-S., Beauthier, C., Orban, D., Sainvitu, C., and Sartenaer, A. (2017). Combining surrogate strategies with MADS for mixed-variable derivative-free optimization. Technical Report G-2017-70, Les cahiers du GERAD. 102, 103

[Dahito et al., 2021] Dahito, M.-A., Genest, L., Maddaloni, A., and Neto, J. (2021). On the performance of the orthomads algorithm on continuous and mixed-integer optimization problems. In Pereira, A. I., Fernandes, F. P., Coelho, J. P., Teixeira, J. P., Pacheco, M. F., Alves, P., and Lopes, R. P., editors, *Optimization, Learning Algorithms and Applications*, pages 31–47, Cham. Springer International Publishing. 19, 64, 125

[Deb and Agrawal, 1995] Deb, K. and Agrawal, R. B. (1995). Simulated binary crossover for continuous search space. *Complex systems*, 9(2):115–148. 24

[Deb et al., 2002] Deb, K., Pratap, A., Agarwal, S., and Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197. 25

[Dorigo, 1992] Dorigo, M. (1992). *Ottimizzazione, apprendimento automatico, ed algoritmi basati su metafora naturale*. PhD thesis, PhD thesis, Dipartimento di Elettronica, Politecnico di Milano, Milan, Italy. 29

[Dufossé and Hansen, 2021] Dufossé, P. and Hansen, N. (2021). Augmented Lagrangian, penalty techniques and surrogate modeling for constrained optimization with CMA-ES. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '21, pages 519–527, New York, NY, USA. Association for Computing Machinery. 47

[Eberhart and Kennedy, 1995] Eberhart, R. and Kennedy, J. (1995). A new optimizer using particle swarm theory. In *MHS'95. Proceedings of the Sixth International Symposium on Micro Machine and Human Science*, pages 39–43. IEEE. 29

[El-Abd and Kamel, 2009] El-Abd, M. and Kamel, M. S. (2009). Black-box optimization benchmarking for noiseless function testbed using particle swarm optimization. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, GECCO '09, page 2269–2274, New York, NY, USA. Association for Computing Machinery. 70

[Fermi and Metropolis, 1952] Fermi, E. and Metropolis, N. (1952). Numerical solution of a minimum problem. Los Alamos Unclassified Report LA–1492, Los Alamos National Laboratory, Los Alamos, USA. 13, 17

[Finck et al., 2009] Finck, S., Hansen, N., Ros, R., and Auger, A. (2009). Real-parameter black-box optimization benchmarking 2009: Presentation of the noiseless functions. Technical Report 2009/20, Research Center PPE. 52, 57

[Finkel and Kelley, 2006] Finkel, D. E. and Kelley, C. (2006). Additive scaling and the DIRECT algorithm. *Journal of Global Optimization*, 36(4):597–608. 20

[Fletcher et al., 2002a] Fletcher, R., Gould, N. I. M., Leyffer, S., Toint, P. L., and Wächter, A. (2002a). Global convergence of a trust-region SQP–filter algorithm for general nonlinear programming. *SIAM Journal on Optimization*, 13(3):635–659. 47

[Fletcher and Leyffer, 2002] Fletcher, R. and Leyffer, S. (2002). Nonlinear programming without a penalty function. *Mathematical programming*, 91(2):239–269. 47

[Fletcher et al., 2002b] Fletcher, R., Leyffer, S., and Toint, P. L. (2002b). On the global convergence of a filter–SQP algorithm. *SIAM Journal on Optimization*, 13(1):44–59. 47

[Floudas and Pardalos, 1990] Floudas, C. A. and Pardalos, P. M. (1990). *A Collection of Test Problems for Constrained Global Optimization Algorithms*, volume 455. Springer, Berlin, Heidelberg. 103

[Forrester and Keane, 2009] Forrester, A. I. and Keane, A. J. (2009). Recent advances in surrogate-based optimization. *Progress in Aerospace Sciences*, 45(1):50–79. 37

[Forsberg and Nilsson, 2005] Forsberg, J. and Nilsson, L. (2005). On polynomial response surfaces and kriging for use in structural optimization of crashworthiness. *Structural and multidisciplinary optimization*, 29(3):232–243. 36

[Friedman, 1991] Friedman, J. H. (1991). Multivariate adaptive regression splines. *The annals of statistics*, pages 1–67. 34

[Gandomi et al., 2011] Gandomi, A. H., Yang, X.-S., and Alavi, A. H. (2011). Mixed variable structural optimization using firefly algorithm. *Computers & Structures*, 89(23):2325–2336. 103

[Gao and Han, 2012] Gao, F. and Han, L. (2012). Implementing the Nelder-Mead simplex algorithm with adaptive parameters. *Comp. Opt. and Appl.*, 51:259–277. 14

[Gebisa and Lemu, 2017] Gebisa, A. W. and Lemu, H. G. (2017). A case study on topology optimized design for additive manufacturing. In *IOP conference series: materials science and engineering*, volume 276, page 012026. IOP Publishing. xiii, 5

[Genest, 2016] Genest, L. (2016). *Optimisation de forme par gradient en dynamique rapide*. PhD thesis, École Centrale de Lyon. 5

[Giunta et al., 2003] Giunta, A., Wojtkiewicz, S., and Eldred, M. (2003). Overview of modern design of experiments methods for computational simulations. In *41st Aerospace Sciences Meeting and Exhibit*, page 649. 30

[Golberg, 1989] Golberg, D. E. (1989). Genetic algorithms in search, optimization, and machine learning. *Addion wesley*, 1989(102):36. 25

[Goldstein, 2013] Goldstein, A. A. (2013). *Constructive real analysis*. Courier Corporation. 17

[Grandhi and Venkayya, 1988] Grandhi, R. and Venkayya, V. (1988). Structural optimization with frequency constraints. *AIAA journal*, 26(7):858–866. 103

[Grossmann and Sargent, 1979] Grossmann, I. E. and Sargent, R. W. H. (1979). Optimum design of multipurpose chemical plants. *Industrial & Engineering Chemistry Process Design and Development*, 18(2):343–348. 103

[Gu et al., 2001] Gu, L., Yang, R., Tho, C.-H., Makowskit, M., Faruquet, O., and Y. Li, Y. L. (2001). Optimisation and robustness for crashworthiness of side impact. *International Journal of Vehicle Design*, 26(4):348–360. 103

[Gupta et al., 2007] Gupta, S., Tiwari, R., and Nair, S. B. (2007). Multi-objective design optimisation of rolling bearings using genetic algorithms. *Mechanism and Machine Theory*, 42(10):1418–1443. 103

[Hajela and Berke, 1992] Hajela, P. and Berke, L. (1992). Neural networks in structural analysis and design: an overview. *Computing Systems in Engineering*, 3(1-4):525–538. 31

[Han and Zhang, 2012] Han, Z.-H. and Zhang, K.-S. (2012). Surrogate-based optimization. In Roeva, O., editor, *Real-World Applications of Genetic Algorithms*, chapter 17. IntechOpen, Rijeka. 37

[Hansen, 2019] Hansen, N. (2019). A global surrogate assisted CMA-ES. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '19, page 664–672, New York, NY, USA. Association for Computing Machinery. 70

[Hansen and Auger, 2014] Hansen, N. and Auger, A. (2014). Principled design of continuous stochastic search: From theory to practice. In *Theory and principled methods for the design of metaheuristics*, pages 145–180. Springer. 28

[Hansen et al., 2016a] Hansen, N., Auger, A., Brockhoff, D., Tušar, D., and Tušar, T. (2016a). COCO: Performance assessment. *ArXiv e-prints*, arXiv:1605.03560. 59

[Hansen et al., 2012] Hansen, N., Auger, A., Finck, S., and Ros, R. (2012). Real-parameter black-box optimization benchmarking 2012: Experimental setup. Technical report, INRIA. 53

[Hansen et al., 2021] Hansen, N., Auger, A., Ros, R., Mersmann, O., Tušar, T., and Brockhoff, D. (2021). COCO: A platform for comparing continuous optimizers in a black-box setting. *Optimization Methods and Software*, 36(1):114–144. 52

[Hansen et al., 2016b] Hansen, N., Tušar, T., Mersmann, O., Auger, A., and Brockhoff, D. (2016b). COCO: The experimental procedure. *ArXiv e-prints*, arXiv:1603.08776. 59

[Hardy, 1971] Hardy, R. L. (1971). Multiquadric equations of topography and other irregular surfaces. *Journal of geophysical research*, 76(8):1905–1915. 32

[Hestenes, 1969] Hestenes, M. R. (1969). Multiplier and gradient methods. *Journal of optimization theory and applications*, 4(5):303–320. 46

[Hestenes and Stiefel, 1952] Hestenes, M. R. and Stiefel, E. (1952). Methods of conjugate gradients for solving. *Journal of research of the National Bureau of Standards*, 49(6):409–436. 13

[Himmelblau, 1972] Himmelblau, D. M. (1972). *Applied nonlinear programming*. McGraw-Hill Book Company, New York. 103

[Hock and Schittkowski, 1980] Hock, W. and Schittkowski, K. (1980). Test examples for nonlinear programming codes. *Journal of Optimization Theory and Applications*, 30(1):127–129. 103

[Holland, 1975] Holland, J. H. (1975). Adaptation in natural and artificial systems. *The University of Michigan Press, Ann Arbor*. 25

[Hooke and Jeeves, 1961] Hooke, R. and Jeeves, T. A. (1961). "Direct search" solution of numerical and statistical problems. *Journal of the ACM (JACM)*, 8(2):212–229. 17

[Jekabsons, 2011] Jekabsons, G. (2011). ARESLab: Adaptive regression splines toolbox for matlab/octave, ver. 1.13.0. 107

[Jin et al., 2001] Jin, R., Chen, W., and Simpson, T. W. (2001). Comparative studies of metamodelling techniques under multiple modelling criteria. *Structural and multidisciplinary optimization*, 23(1):1–13. 36, 100

[Jones, 2001] Jones, D. R. (2001). Direct global optimization algorithm. In Floudas, C. A. and Pardalos, P. M., editors, *Encyclopedia of Optimization*, pages 431–440. Springer US, Boston, MA. 19, 20

[Jones et al., 1993] Jones, D. R., Perttunen, C. D., and Stuckman, B. E. (1993). Lipschitzian optimization without the lipschitz constant. *Journal of optimization Theory and Applications*, 79(1):157–181. 19

[Jones et al., 1998] Jones, D. R., Schonlau, M., and Welch, W. J. (1998). Efficient global optimization of expensive black-box functions. *Journal of Global optimization*, 13(4):455–492. 34

[Jones et al., 2001] Jones, E. D., Oliphant, T. E., and Peterson, P. (2001). SciPy: Open source scientific tools for python. 39

[Kelley, 1999] Kelley, C. T. (1999). Detection and remediation of stagnation in the Nelder–Mead algorithm using a sufficient decrease condition. *SIAM Journal on Optimization*, 10(1):43–55. 16

[Kennedy and Eberhart, 1995] Kennedy, J. and Eberhart, R. (1995). Particle swarm optimization. In *Proceedings of the IEEE international conference on neural networks*, volume 4, pages 1942–1948, Perth, Australia. IEEE Service Center, Piscataway. 29

[Kern et al., 2004] Kern, S., Müller, S. D., Hansen, N., Büche, D., Ocenasek, J., and Koumoutsakos, P. (2004). Learning probability distributions in continuous evolutionary algorithms–a comparative review. *Natural Computing*, 3(1):77–112. 28

[Kianifar and Campean, 2020] Kianifar, M. R. and Campean, F. (2020). Performance evaluation of metamodelling methods for engineering problems: towards a practitioner guide. *Structural and Multidisciplinary Optimization*, 61(1):159–186. 36, 101

[Kirkpatrick et al., 1983] Kirkpatrick, S., Gelatt, C. D., and Vecchi, M. P. (1983). Optimization by simulated annealing. *Science*, 220(4598):671–680. 22

[Koch et al., 2014] Koch, P. N., Bagheri, S., Foussette, C., Krause, P., Bäck, T., and Konen, W. (2014). Constrained optimization with a limited number of function evaluations. In Hoffmann, F. and Hüllermeier, E., editors, *Proc. 24. Workshop Computational Intelligence*, pages 119–134, Universitätsverlag Karlsruhe. 40

[Kolda et al., 2003] Kolda, T. G., Lewis, R. M., and Torczon, V. (2003). Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Review*, 45(3):385–482. 18

[Kora and Yadlapalli, 2017] Kora, P. and Yadlapalli, P. (2017). Crossover operators in genetic algorithms: A review. *International Journal of Computer Applications*, 162(10). 24

[Kraft, 1988] Kraft, D. (1988). A software package for sequential quadratic programming. Tech. rep. dfvlr-fb 88-28, DFVLR Institut für Dynamik der Flugsysteme. 38

[Krige, 1951] Krige, D. G. (1951). A statistical approach to some basic mine valuation problems on the witwatersrand. *Journal of the Southern African Institute of Mining and Metallurgy*, 52(6):119–139. 33

[Kumar et al., 2020] Kumar, A., Wu, G., Ali, M. Z., Mallipeddi, R., Suganthan, P. N., and Das, S. (2020). A test-suite of non-convex constrained optimization problems from the real-world and some baseline results. *Swarm and Evolutionary Computation*, 56:100693. 102, 103, 108

[Kuo et al., 2001] Kuo, W., Prasad, V. R., Tillman, F. A., and Hwang, C.-L. (2001). *Optimal Reliability Design: Fundamentals and Applications*. Cambridge University Press, United Kingdom. 103

[Lancaster and Salkauskas, 1981] Lancaster, P. and Salkauskas, K. (1981). Surfaces generated by moving least squares methods. *Mathematics of computation*, 37(155):141–158. 35

[Le Digabel and Wild, 2015] Le Digabel, S. and Wild, S. (2015). A taxonomy of constraints in simulation-based optimization. Technical Report G-2015-57, Les cahiers du GERAD. 45

[Levin, 1998] Levin, D. (1998). The approximation power of moving least-squares. *Mathematics of computation*, 67(224):1517–1531. 35

[Liao et al., 2014] Liao, T., Socha, K., de Oca, M. A. M., Stützle, T., and Dorigo, M. (2014). Ant colony optimization for mixed-variable optimization problems. *IEEE Transactions on Evolutionary Computation*, 18(4):503–518. 44

[Liu et al., 2021] Liu, C., Wan, Z., Liu, Y., Li, X., and Liu, D. (2021). Trust-region based adaptive radial basis function algorithm for global optimization of expensive constrained black-box problems. *Applied Soft Computing*, 105:107233. 38

[Liu and Nocedal, 1989] Liu, D. C. and Nocedal, J. (1989). On the limited memory BFGS method for large scale optimization. *Mathematical Programming*, 45:503–528. 54

[Lophaven et al., 2002] Lophaven, S. N., Nielsen, H. B., and Søndergaard, J. (2002). DACE–a matlab kriging toolbox, version 2.0. 107

[Maliki, 2016] Maliki, M. (2016). *Adaptive surrogate models for the reliable lightweight design of automotive body structures*. PhD thesis, Université Blaise Pascal – Clermont II. 5

[Martinez et al., 2017] Martinez, N., Anahideh, H., Rosenberger, J. M., Martinez, D., Chen, V. C., and Wang, B. P. (2017). Global optimization of non-convex piecewise linear regression splines. *Journal of Global Optimization*, 68(3):563–586. 44

[McCulloch and Pitts, 1988] McCulloch, W. S. and Pitts, W. (1988). Neurocomputing: foundations of research. Chapter A logical calculus of the ideas immanent in nervous activity. *Volume*, 1:15–27. 35

[McKay et al., 2000] McKay, M. D., Beckman, R. J., and Conover, W. J. (2000). A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 42(1):55–61. 100

[McKinnon, 1998] McKinnon, K. I. M. (1998). Convergence of the Nelder–Mead simplex method to a nonstationary point. *SIAM Journal on Optimization*, 9(1):148–158. 14

[Metropolis et al., 1953] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. (1953). Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092. 22

[Močkus, 1975] Močkus, J. (1975). On bayesian methods for seeking the extremum. In Marchuk, G. I., editor, *Optimization techniques IFIP Technical Conference*, pages 400–404, Berlin, Heidelberg. Springer. 34

[Moré and Wild, 2009] Moré, J. J. and Wild, S. M. (2009). Benchmarking derivative-free optimization algorithms. *SIAM Journal on Optimization*, 20(1):172–191. 51

[Moustapha et al., 2018] Moustapha, M., Bourinet, J.-M., Guillaume, B., and Sudret, B. (2018). Comparative study of kriging and support vector regression for structural engineering applications. *ASCE-ASME Journal of Risk and Uncertainty in Engineering Systems, Part A: Civil Engineering*, 4(2):04018005. 36

[Müller, 2016] Müller, J. (2016). MISO: mixed-integer surrogate optimization framework. *Optimization and Engineering*, 17(1):177–203. 38, 44

[Müller and Shoemaker, 2014] Müller, J. and Shoemaker, C. A. (2014). Influence of ensemble surrogate models and sampling strategy on the solution quality of algorithms for computationally expensive black-box global optimization problems. *Journal of Global Optimization*, 60(2):123–144. 101

[Müller et al., 2013] Müller, J., Shoemaker, C. A., and Piché, R. (2013). SO-MI: A surrogate model algorithm for computationally expensive nonlinear mixed-integer black-box global optimization problems. *Computers & Operations Research*, 40(5):1383–1400. 44, 102, 103

[Müller et al., 2014] Müller, J., Shoemaker, C. A., and Piché, R. (2014). SO-I: a surrogate model algorithm for expensive nonlinear integer programming problems including global optimization applications. *Journal of Global Optimization*, 59(4):865–889. 44, 102, 103

[Müller and Woodbury, 2017] Müller, J. and Woodbury, J. D. (2017). GOSAC: global optimization with surrogate approximation of constraints. *Journal of Global Optimization*, 69(1):117–136. 45

[Myers et al., 2016] Myers, R. H., Montgomery, D. C., and Anderson-Cook, C. M. (2016). *Response surface methodology: process and product optimization using designed experiments*. John Wiley & Sons. 31

[Nash, 1984] Nash, S. G. (1984). Newton-type minimization via the Lanczos method. *SIAM Journal on Numerical Analysis*, 21(4):770–788. 55

[Nelder and Mead, 1965] Nelder, J. A. and Mead, R. (1965). A simplex method for function minimization. *The computer journal*, 7(4):308–313. 14

[Nocedal, 1980] Nocedal, J. (1980). Updating quasi-Newton matrices with limited storage. *Mathematics of computation*, 35(151):773–782. 13

[Nocedal and Wright, 2006] Nocedal, J. and Wright, S. J. (2006). *Numerical optimization*. Springer New York, NY. 13, 38, 54, 55

[Ortega and Rheinboldt, 2000] Ortega, J. M. and Rheinboldt, W. C. (2000). *Iterative solution of nonlinear equations in several variables*. SIAM. 13

[Pant et al., 2009] Pant, M., Thangaraj, R., and Singh, V. P. (2009). Optimization of mechanical design problems using improved differential evolution algorithm. *International Journal of Recent Trends in Engineering*, 1(5):21–25. 103

[Paul, 1987] Paul, H., T. (1987). Optimal design of an industrial refrigeration system. In *Proceedings of International Conference on Optimization Techniques and Applications*, pages 427–435, Singapore, National University of Singapore. 103

[Plackett and Burman, 1946] Plackett, R. L. and Burman, J. P. (1946). The design of optimum multifactorial experiments. *Biometrika*, 33(4):305–325. 106

[Powell, 1964] Powell, M. J. D. (1964). An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The Computer Journal*, 7(2):155–162. 13

[Powell, 1969] Powell, M. J. D. (1969). A method for nonlinear constraints in minimization problems. In Fletcher, R., editor, *Optimization*, pages 283–298. Academic Press, New York. 46

[Powell, 1994] Powell, M. J. D. (1994). A direct search optimization method that models the objective and constraint functions by linear interpolation. In Gomez, S. and Hennart, J.-P., editors, *Advances in Optimization and Numerical Analysis*, pages 51–67. Springer, Dordrecht. 39

[Powell, 2006] Powell, M. J. D. (2006). The NEWUOA software for unconstrained optimization without derivatives. In *Large-scale nonlinear optimization*, pages 255–297. Springer. 64

[Price, 1997] Price, K. V. (1997). Differential evolution vs. the functions of the second ICEO. In *Proceedings of the IEEE International Congress on Evolutionary Computation*, pages 153–157, Piscataway, NJ, USA. IEEE. 53

[Ray et al., 2001] Ray, T., Tai, K., and Seow, C. (2001). An evolutionary algorithm for multiobjective optimization. *Engineering Optimization*, 33(3):399–424. 27

[Rechenberg, 1973] Rechenberg, I. (1973). Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologischen Evolution. *Frommann–Holzboog*. 27, 28

[Regis, 2011] Regis, R. G. (2011). Stochastic radial basis function algorithms for large-scale optimization involving expensive black-box objective and constraint functions. *Computers & Operations Research*, 38(5):837–853. 41, 101

[Regis, 2014] Regis, R. G. (2014). Constrained optimization by radial basis function interpolation for high-dimensional expensive black-box problems with infeasible initial points. *Engineering Optimization*, 46(2):218–243. 39, 64, 98, 101

[Regis, 2020] Regis, R. G. (2020). Large-scale discrete constrained black-box optimization using radial basis functions. In *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 2924–2931. IEEE. 44

[Regis and Wild, 2017] Regis, R. G. and Wild, S. M. (2017). CONORBIT: constrained optimization by radial basis function interpolation in trust regions. *Optimization Methods and Software*, 32(3):552–580. 39

[Rios and V., 2013] Rios, L. M. and V., S. N. (2013). Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56(3):1247–1293. 42, 43, 64

[Rockafellar, 1973] Rockafellar, R. T. (1973). A dual approach to solving nonlinear programming problems by unconstrained optimization. *Mathematical programming*, 5(1):354–373. 46

[Ros, 2009] Ros, R. (2009). Benchmarking the NEWUOA on the bbob-2009 function testbed. In *Proceedings of the 11th Annual Conference Companion on Genetic and Evolutionary Computation Conference: Late Breaking Papers*, GECCO '09, page 2421–2428, New York, NY, USA. Association for Computing Machinery. 70

[Sacks et al., 1989] Sacks, J., Welch, W. J., Mitchell, T. J., and Wynn, H. P. (1989). Design and analysis of computer experiments. *Statistical science*, pages 409–423. 33

[Sasena et al., 2002] Sasena, M. J., Papalambros, P., and Goovaerts, P. (2002). Exploration of metamodeling sampling criteria for constrained global optimization. *Engineering Optimization*, 34(3):263–278. 41

[Schwefel, 1981] Schwefel, H.-P. (1981). *Numerical optimization of computer models*. John Wiley & Sons, New-York, NY, USA. 27

[Spendley et al., 1962] Spendley, W., Hext, G. R., and Himsworth, F. R. (1962). Sequential application of simplex designs in optimisation and evolutionary operation. *Technometrics*, 4(4):441–461. 14

[Storn and Price, 1997] Storn, R. and Price, K. (1997). Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces. *Journal of global optimization*, 11(4):341–359. 29

[Thanedar and Vanderplaats, 1995] Thanedar, P. and Vanderplaats, G. (1995). Survey of discrete variable optimization for structural design. *Journal of Structural Engineering*, 121(2):301–306. 103

[Torczon, 1997] Torczon, V. (1997). On the convergence of pattern search algorithms. *SIAM Journal on Optimization*, 7(1):1–25. 17

[Tseng, 1999] Tseng, P. (1999). Fortified-descent simplicial search method: A general approach. *SIAM Journal on Optimization*, 10(1):269–288. 16

[Tušar et al., 2019] Tušar, T., Brockhoff, D., and Hansen, N. (2019). Mixed-integer benchmark problems for single- and bi-objective optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*, GECCO '19, pages 718—726, New York, NY, USA. Association for Computing Machinery. 44, 66, 70

[Umbarkar and Sheth, 2015] Umbarkar, A. J. and Sheth, P. D. (2015). Crossover operators in genetic algorithms: a review. *ICTACT journal on soft computing*, 6(1). 24

[Vapnik, 1995] Vapnik, V. N. (1995). *The Nature of Statistical Learning Theory*. Springer New York, NY. 35

[Varelas et al., 2018] Varelas, K., Auger, A., Brockhoff, D., Hansen, N., ElHara, O. A., Semet, Y., Kassab, R., and Barbaresco, F. (2018). A comparative study of large-scale variants of CMA-ES. In Auger, A., Fonseca, C. M., Lourenço, N., Machado, P., Paquete, L., and Whitley, D., editors, *International Conference on Parallel Problem Solving from Nature*, pages 3–15, Cham. Springer, Springer International Publishing. 66

[Varelas and Dahito, 2019] Varelas, K. and Dahito, M.-A. (2019). Benchmarking multivariate solvers of scipy on the noiseless testbed. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '19, page 1946–1954, New York, NY, USA. Association for Computing Machinery. 52, 70, 125

[Villa-Vialaneix et al., 2012] Villa-Vialaneix, N., Follador, M., Ratto, M., and Leip, A. (2012). A comparison of eight metamodeling techniques for the simulation of N2O fluxes and N leaching from corn crops. *Environmental Modelling & Software*, 34:51–66. 36, 101

[Vu et al., 2017] Vu, K. K., d'Ambrosio, C., Hamadi, Y., and Liberti, L. (2017). Surrogate-based methods for black-box optimization. *International Transactions in Operational Research*, 24(3):393–424. 38

[Wales and Doye, 1997] Wales, D. J. and Doye, J. P. (1997). Global optimization by basin-hopping and the lowest energy structures of lennard-jones clusters containing up to 110 atoms. *The Journal of Physical Chemistry A*, 101(28):5111–5116. 54

[Wang and Shan, 2006] Wang, G. G. and Shan, S. (2006). Review of metamodeling techniques in support of engineering design optimization. *Journal of Mechanical Design*, 129(4):370–380. 37

[Wang et al., 2018] Wang, Y., Liu, H., Long, H., Zhang, Z., and Yang, S. (2018). Differential evolution with a new encoding mechanism for optimizing wind farm layout. *IEEE Transactions on Industrial Informatics*, 14(3):1040–1054. 103

[Wild et al., 2008] Wild, S. M., Regis, R. G., and Shoemaker, C. A. (2008). ORBIT: Optimization by radial basis function interpolation in trust-regions. *SIAM Journal on Scientific Computing*, 30(6):3197–3219. 39

[Wolfe, 1969] Wolfe, P. (1969). Convergence conditions for ascent methods. *SIAM review*, 11(2):226–235. 17

[Wright, 1996] Wright, M. H. (1996). Direct search methods: Once scorned, now respectable. *Pitman Research Notes in Mathematics Series*, pages 191–208. 17

[Xiu and Karniadakis, 2002] Xiu, D. and Karniadakis, G. E. (2002). The Wiener–Askey polynomial chaos for stochastic differential equations. *SIAM Journal on Scientific Computing*, 24(2):619–644. 35

[Ye et al., 2000] Ye, K. Q., Li, W., and Sudjianto, A. (2000). Algorithmic construction of optimal symmetric latin hypercube designs. *Journal of Statistical Planning and Inference*, 90(1):145–159. 100

[Yokota et al., 1998] Yokota, T., Taguchi, T., and Gen, M. (1998). A solution method for optimal weight design problem of the gear using genetic algorithms. *Computers & Industrial Engineering*, 35(3):523–526. Selected Papers from the 22nd ICC and IE Conference. 103

[Yuan et al., 1988] Yuan, X., Zhang, S., Pibouleau, L., and Domenech, S. (1988). Une méthode d'optimisation non linéaire en variables mixtes pour la conception de procédés. *RAIRO-Operations Research*, 22(4):331–346. 103

ECOLE
DOCTORALE

Titre : Optimisation boîte noire sous contraintes et en variables mixtes avec des applications dans l'industrie automobile

Mots clés : optimisation boîte noire, optimisation sans dérivées, variables mixtes, optimisation sous contraintes

Résumé : Bon nombre de problèmes d'optimisation rencontrés dans l'industrie font appel à des systèmes complexes et n'ont pas de formulation analytique explicite: ce sont des problèmes d'optimisation de type boîte noire (ou *blackbox* en anglais). Ils peuvent être dits "mixtes", auquel cas ils impliquent des variables de différentes natures (continues et discrètes), et avoir de nombreuses contraintes à satisfaire. De plus, les évaluations de l'objectif et des contraintes peuvent être numériquement coûteuses.
Dans cette thèse, nous étudions des méthodes de résolution de tels problèmes complexes, à savoir des problèmes d'optimisation boîte noire avec contraintes et variables mixtes, pour lesquels les évaluations des fonctions sont très coûteuses en temps de calcul.
Puisque l'utilisation de dérivées n'est pas envisageable, ce type de problèmes est généralement abordé par des approches sans dérivées comme les algorithmes évolutionnaires, les méthodes de recherche directe et les approches basées sur des métamodèles.
Nous étudions les performances de telles méthodes déterministes et stochastiques dans le cadre de l'optimisation boîte noire, y compris sur un cas test en éléments finis que nous avons conçu. En particulier, nous évaluons les performances de la variante ORTHOMADS de l'algorithme de recherche directe MADS sur des problèmes d'optimisation continus et à variables mixtes issus de la littérature.
Nous proposons également une nouvelle méthode d'optimisation boîte noire, nommée BOA, basée sur des approximations par métamodèles. Elle comporte deux phases dont la première vise à trouver un point réalisable tandis que la seconde améliore itérativement la valeur de l'objectif de la meilleure solution réalisable trouvée. Nous décrivons des expériences utilisant des instances de la littérature et des applications de l'industrie automobile. Elles incluent des tests de notre algorithme avec différents types de métamodèles, ainsi que des comparaisons avec ORTHOMADS.

Title : Constrained mixed-variable blackbox optimization with applications in the automotive industry

Keywords : blackbox optimization, derivative-free optimization, mixed variables, constrained optimization

Abstract : Numerous industrial optimization problems are concerned with complex systems and have no explicit analytical formulation, that is they are blackbox optimization problems. They may be mixed, namely involve different types of variables (continuous and discrete), and comprise many constraints that must be satisfied. In addition, the objective and constraint blackbox functions may be computationally expensive to evaluate.
In this thesis, we investigate solution methods for such challenging problems, i.e constrained mixed-variable blackbox optimization problems involving computationally expensive functions.
As the use of derivatives is impractical, problems of this form are commonly tackled using derivative-free approaches such as evolutionary algorithms, direct search and surrogate-based methods.
We investigate the performance of such deterministic and stochastic methods in the context of blackbox optimization, including a finite element test case designed for our research purposes. In particular, the performance of the ORTHOMADS instantiation of the direct search MADS algorithm is analyzed on continuous and mixed-integer optimization problems from the literature.
We also propose a new blackbox optimization algorithm, called BOA, based on surrogate approximations. It proceeds in two phases, the first of which focuses on finding a feasible solution, while the second one iteratively improves the objective value of the best feasible solution found. Experiments on instances stemming from the literature and applications from the automotive industry are reported. They namely include results of our algorithm considering different types of surrogates and comparisons with ORTHOMADS.

Institut Polytechnique de Paris
91120 Palaiseau, France