Parallel surrogate-based algorithms for solving expensive optimization problems

Guillaume Briffoteaux

Grascomp
MADIS Mathématiques, sciences du numérique et de leurs interactions

THÈSE

préparée dans le cadre d'une cotutelle
pour obtenir le grade de

Docteur en Sciences de l'Ingénieur et Technologies de l'Université de Mons
Docteur en Informatique et Applications de l'Université de Lille

présentée et soutenue par

Guillaume Briffoteaux

le 21 Octobre 2022

Parallel surrogate-based algorithms for solving expensive optimization problems

---

Algorithmes parallèles et basés sur méta-modèles pour la résolution de problèmes d'optimisation coûteux

---

devant le jury composé de

	Imen Chakroun	Chargée de recherche	IMEC Leuven
	Gregory Coussement	Professeur	Université de Mons
	Patricia Stolf	Professeur	Université de Toulouse
	David Duvivier	Professeur	Université Polytechnique Hauts-de-France
Rapporteurs	Amir Nakib	Professeur	Université Paris Est
	Frédéric Saubion	Professeur	Université d'Angers
Directeurs	Daniel Tuyttens	Professeur	Université de Mons
	Nouredine Melab	Professeur	Université de Lille
Co-directeur	Mohand Mezmaz	Chargé de recherche, HDR	Université de Mons

A la mémoire de mon père et de mon grand-père.
A ma mère, ma grand-mère et mes sœurs.

Abstract

Combining machine learning, parallel computing and optimization gives rise to Parallel Surrogate-Based Optimization Algorithms (P-SBOAs). These algorithms are useful to solve black-box computationally expensive simulation-based optimization problems where the function to optimize relies on a computationally costly simulator. In addition to the search landscape topography, that may exhibit features complicating the optimization, the search is limited by a computational budget due to the objective function expensiveness.

This thesis focuses on the design of P-SBOAs to deal with various search landscape characteristics and computational budgets. The distinction between very and moderately expensive problems is introduced through the definition of the budget as a limited time on a restricted amount of computing resources. The three dimensions of the design space of P-SBOAs as considered in this work are the surrogate model, the definition of promisingness of new candidate solutions and the coupling between the optimizer and the surrogate. On the one hand, machine learning is triggered to build surrogate models that imitate the simulator in order to evaluate and/or locate new promising solutions in a fast way. On the other hand, parallel computing is leveraged to perform multiple simulations simultaneously in order to reduce the execution time. The overall challenge consists in adequately allocating the budget to the tasks of simulation, surrogate training and acquisition of new solutions.

Surrogate models should be trained fast especially to handle moderately expensive problems where the training-set size may become significant, they should also provide adequate predictive capacity to approximate rugged landscapes and predictive uncertainty information to guide the search. The Bayesian Neural Network approximated *via* Monte-Carlo Dropout (BNN_MCD) is investigated as it gathers all the desired features. Firstly, it is used to build Parallel Surrogate-Assisted Evolutionary Algorithms (P-SAEAs) by evaluating and filtering candidate solutions. Secondly, it is employed along with Gaussian Processes (GPs) to design new Parallel Surrogate-Driven Algorithms (P-SDAs) where sub-surrogates are optimized in parallel to produce multiple new promising solutions. The promisingness of solutions is defined by dynamically changing the trade-off between exploration and exploitation during the search through ensembles of evolution controls.

Systematic experiments are conducted on multiple benchmark problems as well as on a real-world application of Covid-19 control to compare an extensive range of algorithm designs. The results demonstrate that P-SAEAs are much more adapted to solve moderately expensive problems while P-SDAs are to be put forward on very expensive ones. In P-SAEAs, the definition of promisingness promoted by the numerical results consists in favoring exploration at the early stage and exploitation at the latter stage of the search while more intensification is preferred in P-SDAs. The BNN_MCD surrogate model shows to perform well on multi-modal landscapes with weakly informative global structure and GPs are promoted otherwise. Consequently, a new hybrid algorithm retaining the best of P-SAEAs and P-SDAs is proposed to offer robustness with respect to the computational budgets. The novel method demonstrates a striking parallel scalability and produces the best solutions on the Covid-19 contact reduction problem featuring multi-modality and weak global structure.

Résumé

De la combinaison de l'apprentissage automatique, du calcul parallèle et de l'optimisation résultent les algorithmes d'optimisation parallèles et basés sur des méta-modèles (Parallel Surrogate-Based Optimization Algorithms, P-SBOAs). Ces algorithmes sont utiles à la résolution de problèmes d'optimisation boîte-noire, coûteux en calculs et basés sur la simulation pour lesquels la fonction à optimiser repose sur un simulateur coûteux en calculs. En plus de la topographie du paysage de recherche, pouvant montrer des caractéristiques compliquant l'optimisation, la recherche est limitée par un budget de calculs dû au coût de la fonction objectif.

Dans cette thèse, l'attention est focalisée sur la conception des P-SBOAs pour traiter divers caractéristiques du paysage de recherche et budgets de calculs. La distinction entre problèmes très et modérément coûteux est introduite au travers de la définition du budget comme étant un temps limité sur une quantité restreinte de ressources de calculs. Les trois dimensions de l'espace de conception des P-SBOAs considéré dans ces travaux sont le méta-modèle, la définition du degré de promesse des solutions candidates et le couplage entre l'optimisateur et le méta-modèle. D'un côté, l'apprentissage automatique est utilisé pour construire un méta-modèle qui imite le simulateur afin d'évaluer et/ou localiser rapidement de nouvelles solutions prometteuses. D'un autre côté, le calcul parallèle est mis à profit pour réaliser simultanément plusieurs simulations afin de réduire le temps d'exécution. Le challenge général consiste à allouer le budget de façon adéquate aux tâches de simulation, entraînement du méta-modèle et acquisition de nouvelles solutions.

Le méta-modèle doit être entraînable rapidement surtout dans le cas des problèmes modérément coûteux où la taille de l'ensemble d'entraînement peut devenir significative. Il doit aussi offrir une capacité prédictive adéquate pour se rapprocher des paysages rugueux et fournir de l'information sur l'incertitude prédictive afin de guider la recherche. Les réseaux de neurones bayesiens (Bayesian Neural Network) approchés par Monte-Carlo Dropout (BNN_MCD) sont étudiés car ils rassemblent toutes les caractéristiques désirées. Premièrement, ils sont utilisés pour construire des algorithmes évolutionnaires parallèles assistés par méta-modèles (Parallel Surrogate-Assisted Evolutionary Algorithms, P-SAEAs) en évaluant et filtrant les solutions candidates. Deuxièmement, ils sont employés avec des processus gaussiens (Gaussian Processes, GPs) pour concevoir de nouveaux algorithmes parallèles guidés par méta-modèles (Parallel Surrogate-Driven Algorithms, P-SDAs) où des sous-méta-modèles sont optimisés en parallèle pour produire de multiple nouvelles solutions prometteuses. Le degré de promesse des solutions est défini en changeant dynamiquement le compromis entre exploration et exploitation durant la recherche *via* des ensembles de contrôles d'évolution.

Des expériences systématiques sont menées sur plusieurs problèmes artificiels ainsi que sur une application réelle de contrôle de l'épidémie de Covid-19 afin de comparer une vaste gamme d'algorithmes. Les résultats démontrent que les P-SAEAs sont beaucoup plus adaptés à la résolution de problèmes modérément coûteux alors que les P-SDAs sont à mettre en avant sur des problèmes très coûteux. Avec les P-SAEAs, la définition du degré de promesse promu par les résultats numériques consiste à favoriser d'abord l'exploration et ensuite l'exploitation au cours de la recherche alors que plus d'intensification est préférée

avec les P-SDAs. Le méta-modèle BNN_MCD démontre de bonnes performances sur des paysages multi-modales avec une structure globale peu informante et les GPs sont mis en avant dans les autres cas. Par conséquent, un nouvel algorithme hybride retenant le meilleur des P-SAEAs et P-SDAs est proposé pour offrir plus de robustesse quant aux budgets de calculs. La nouvelle méthode démontre une mise à l'échelle frappante à l'augmentation des unités de calculs et produit les meilleures solutions sur le problème lié à la Covid-19 exposant un paysage multi-modal et une structure global peu informante.

Remerciements

Je remercie vivement mes directeurs de thèse Nouredine Melab, Professeur à l'Université de Lille et directeur de l'équipe BONUS (CRIStAL et Inria Lille) et Daniel Tuyttens, Professeur et chef du service MARO à l'Université de Mons pour l'encadrement, les relectures, les discussions et les encouragements qu'ils m'ont apportés tout au long de la thèse.

Mes vifs remerciements vont également à mon co-directeur de thèse Mohand Mezmaz, Chargé de recherche à l'Université de Mons, pour ses précieux conseils, et les longs échanges scientifiques et philosophiques auxquels nous nous sommes prêtés.

Je remercie également Romain Ragonnet, Chercheur à la Monash University de Melbourne, pour sa confiance et pour la collaboration que nous avons établie. Je remercie plus largement les membres du Department of Public Health and Preventive Medicine qui était prêt à m'accueillir en Australie avant que l'épidémie de Covid n'en décide autrement.

Une partie des études publiées dans le cadre de cette thèse a été réalisée conjointement avec Maxime Gobert et Pierre Tomenko, respectivement doctorant et étudiant en Master à l'Université de Mons. Je les remercie d'avoir rendu possible ces collaborations.

J'adresse mes remerciements à tous les membres du service MARO de l'Université de Mons et de l'équipe BONUS de l'Université de Lille, et en particulier à Jan Gmys pour ses conseils concernant la formation doctorale et la rédaction.

Je souhaite remercier Gregory Coussement (Professeur, Université de Mons) d'avoir accepté de présider le jury du comité d'accompagnement et de défense de ma thèse.

Je tiens à remercier les rapporteurs Frédéric Saubion (Professeur, Université d'Angers) et Amir Nakib (Professeur, Université Paris Est) pour leur attentive évaluation de mes travaux.

Je remercie également Imen Chakroun (Chargée de recherche, IMEC Leuven) et Patricia Stolf (Professeur, Université de Toulouse) d'avoir accepté de participer au jury ainsi que David Duvivier (Professeur, Université Polytechnique Hauts-de-France) pour sa participation au jury et ses commentaires judicieux lors des comités d'accompagnement.

Toute ma gratitude va à ma famille, ma maman, ma grand-mère et mes sœurs pour le soutien que nous nous apportons. Je suis reconnaissant envers tous mes amis pour leur présence, en particulier Axel, Clément, Florian, Jordy, Julie, Justine, Justinne, Kittie, Laura, Manon, Marine, Martin, Mathilde, Norman, Pierre, Sophie, Soria, Virgile ainsi que les copains de Reims, Alisson, Fab, Jordan, Laetitia, Marion, Maxime, Noemy, Quentin, et Romain.

List of acronyms

AB-MOEA - Adaptive Bayesian Multi-Objective Evolutionary Algorithm
ANN - Artificial Neural Network
ANN_BLR - Bayesian Linear Regression model with ANN basis functions
AP - Acquisition Process
BNN - Bayesian Neural Network
BNN_HMC - Bayesian Neural Network trained through HMC
BNN_MCD - BNN approximated *via* Monte-Carlo Dropout
cl-mean - Constant Liar with mean
Covid-19 - Coronavirus Disease 2019
CPU - Computer Processing Unit
CTS - Complete Training Set
DFR - Direct Fitness Replacement
DOE - Design of Experiments
EA - Evolutionary Algorithm
EC - Evolution Control
EI - Expected Improvement
EGO - Efficient Global Optimization
GD - Gradient-Descent
GP - Gaussian Process
GP_HMC - Gaussian Process trained through HMC
GP_RBF - GP with Radial Basis Functions kernel
HCAP - Hybrid Concurrent Acquisition Processes
HMC - Hamiltonian Monte Carlo
HSAP - Hybrid Successive Acquisition Processes
IC - Infill Criterion
IFR - Indirect Fitness Replacement
iKRG - interpolation Kriging model
LCB - Lower Confidence Bound
LHS - Latin Hyper-cubes Sampling
NAVLL - Negative Average Validation Log-Likelihood
NDF - Non-Dominated Front
NDR - Non-Dominated Rank
NDS - Non-Dominated Set
NSGA-II - Non-dominated Sorting Genetic Algorithm II
MCMC - Monte-Carlo Markov Chain
MO - Multi-Objective
OOP - Object-Oriented Programming
PF - Pareto Front
PS - Pareto Set
PI - Probability of Improvement
POV - Predicted Objective Value
P-EA - Parallel Evolutionary Algorithm

P-SAEA - Parallel Surrogate-Assisted Evolutionary Algorithm
P-SBO - Parallel Surrogate-based Optimization
P-SBOA - Parallel Surrogate-based Optimization Algorithm
P-SDA - Parallel Surrogate-Driven Algorithm
pySBO - Python platform for Surrogate-based Optimization
q-EGO - parallel variant of EGO
q-Pareto - P-SDA based on Pareto dominance
q-post-HMC - P-SDA based on Hamiltonian Monte-Carlo sampling
q-subnets - P-SDA based on q sub-networks
RBF - Radial Basis Functions
rKRG - regression Kriging model
RTS - Reduced Training Set
RVEA - Reference Vector guided Evolutionary Algorithm
SaaEF - Surrogate as an Evaluator and a Filter
sb - Surrogate Believer
SAEA - Surrogate-Assisted Evolutionary Algorithm
SAEA-ME - Surrogate-Assisted Evolutionary Algorithm for Medium Scale Expensive problems
SBX - Simulated Binary Cross-over
SDA - Surrogate-Driven Algorithm
SMBO+EA - Surrogate Model Based Optimization + Evolutionary Algorithm
SO - Single-Objective
UML - Unified Modelling Language
VMSE - Validation Mean Squared Error
vR2 - Validation Correlation Coefficient

Contents

Introduction

Simulation software are widespread tools to predict the consequences of a decision. What would be the epidemic trajectory if it is decided to put the whole population on lockdown? How many deaths caused by a virus could be avoided if it is chosen to only vaccinate the elderly? These are questions simulators can help to answer. Consequently, they can be employed to suggest adequate decisions to take in a given situation. These decisions could consist in better lockdown modalities to slow down the spread of a virus, in better strategies to distribute the vaccine doses to reduce the number of deaths. These last two problems are examples of black-box simulation-based optimization problems where the simulation is exclusively used to evaluate the quality of possible decisions.

Evolutionary Algorithms (EAs) represent handy optimizers to deal with black-box problems as they only require to evaluate candidates. Nevertheless, EAs need to perform a substantial amount of simulations to provide significant results. This behavior poses severe restrictions when the simulation is computationally expensive as in the Covid-19 (Coronavirus disease 2019) related problem tackled in this thesis. The complementary ways to deal with the computational complexity is to harness machine learning and parallel computing. Parallel computers contribute to reduce the execution time of the optimization by performing multiple simulations at once. Machine Learning algorithms extract knowledge and identify patterns from data. From a set of real-valued input-output observations, regression and interpolation models learn the underlying mapping so as to predict what would happen at unobserved points. Because predicting is computationally cheap, these methods are used as surrogate models for computationally expensive simulators. Nonetheless, the gain in terms of computing time is counterbalanced by a loss of accuracy. It is primordial to attain and preserve the balance between prediction fidelity and computational cost.

The synergy between simulation, optimization, parallel computing and machine learning gives rise to the so-called Parallel Surrogate-based Optimization Algorithms (P-SBOAs), suited to solve computationally expensive simulation-based problems. The design of P-SBOAs is tedious as it relies on the association of components, and the respect of rules, stemming from diverse fields. Particularly, the No Free Lunch theorem complicates the choice of the surrogate model by stating that no one machine learning algorithm is better than all others in all cases. By carrying the responsibility to propose new promising candidates to simulate, the surrogate-optimizer coupling is also an important challenge. Defining the promisingness of solutions is another laborious piece of work when the number of simulations is restricted. It is reasonable to assume that, similarly to the surrogate selection, the determination of the coupling and the definition of the promisingness are problem-dependent. While unexpensive optimization problems are only characterized by the topography of the search landscape [Mer+11], the amount of the computational budget allocated to the search is an additional feature inherent to expensive optimization [FSK08b].

Plugging the surrogate to the EA gives rise to Parallel Surrogate-Assisted Evolutionary Algorithms (P-SAEAs) [JOS01]. Differently, in Parallel Surrogate-Driven Algorithms (P-SDAs), the surrogate is the core component and the leader of the search. A very famous P-SDA is the extension of the sequential Efficient Global Optimization (EGO) algorithm [JSW98] for parallel simulations of q new candidates (q-EGO) [GRC10]. The q-EGO algorithm consists in sequentially maximizing a specific metric of promisingness, called the Expected Improvement (EI), q times so as to propose q new candidate solutions for parallel processing. Between two consecutive EI optimizations, the Kriging surrogate model is updated to locate a new point.

A first limitation of q-EGO is to quickly converge [Ber+19]. While this behaviour is desired for very expensive problems, the spotted optimum is usually a local one, and therefore a continuous search progression is preferable for moderately expensive problems. The fast convergence is due to a high degree of exploitation as soon as the search begins so enhancing exploration may be a good idea. To do so, the surrogate can be employed as an evaluator and/or a filter in a P-SAEA. As an evaluator, the surrogate is used to evaluate new individuals instead of the simulator. As a filter, the surrogate helps to discard unpromising candidates. In any case, the diversity of individuals within the population of the EA maintains a certain level of exploration.

Another issue of q-EGO is the limited predictive capacity and the computationally expensive training of the Kriging model [Ras06]. To approximate very rough search landscapes, higher capacity models such as Bayesian Neural Networks (BNNs) may be employed [Bis06]. Similarly to Kriging, BNNs are capable of providing predictive uncertainty in the form of standard deviation, a key information to define the promisingness. Recent approximation techniques allow for a fast training of the model [Gal16]. Reducing training complexity may present benefits for moderately expensive problems as more training samples are accumulated.

A third disadvantage of q-EGO is the poor performance of the EI metric of promisingness for handling medium-to-large-scale problems regarding the number of decision variables [Reh+20]. Multiple combinations of the predicted objective value and the predictive uncertainty may be envisioned to settle this problem like adapting the definition of the promisingness during the search. Adaptivity may also extend the limits of q-EGO related to premature convergence as EI has been identified as an impacting factor in this respect [Ber+19].

Thesis scope

In this thesis, the focus is directed towards the design of P-SAEAs and P-SDAs for solving black-box computationally expensive optimization problems. The design is guided by the choice of the surrogate model, the way the surrogate is coupled with the optimizer and the definition of promisingness. Different designs are proposed and compared both conceptually and empirically *via* systematic numerical experiments on diverse problems exhibiting different search landscapes and computational budgets.

It is commonly admitted that P-SBOAs are convenient to handle very expensive problems, where the surrogate training is negligible compared to the simulation in terms of computational load [FSK08b]. For these problems, the computational budget is classically expressed as a limited number of simulations. In this thesis, the budget is expressed as a limited number of computing cores and a limited duration in order to best reflect the reality. Indeed, training a surrogate is potentially a computationally expensive task. Moreover, the budget is defined in such a way to consider moderately expensive optimization problems thus allowing to demonstrate the pertinence of P-SBOAs in such scenarios.

Benchmark functions are used in the experiments reported in this thesis to compare the different approaches. For these artificial problems, the analytical formula of the objective function is known and the evaluation time is insignificant [Han+10]. Knowing the shape of the search landscape associated to a problem allows to infer the performance of methods with respect to its topography (multi-modality, global structure, conditioning, *etc.*). Additionally, a real-world application to Covid-19 contact reduction is considered. The optimization exercise consists in finding the best contact reduction strategy to circumscribe the number of deaths while attaining herd immunity. In this case, the expensiveness of the simulation and the constraint hamper landscape analysis prior to the search.

Contributions
The addressed issues and proposed contributions are summarized in the following:

- Surrogate models should demonstrate high predictive capacity to approximate rugged landscapes and fast training to handle moderately expensive problems. Furthermore, they should provide predictive uncertainty to define the promisingness. Previously published studies on P-SBOAs do not consider BNNs and published comparisons between surrogates are limited to very expensive problems [Bre+18; Wan+16; Ton+21]. **We propose to investigate the BNNs approximated *via* Monte-Carlo Dropout (BNN_MCD) as surrogate model.** BNN_MCD shows the advantages to train fast, to provide a predictive standard deviation and to accurately approximate rough landscapes [Gal16]. It is compared with widely used surrogates as Kriging both within and outside P-SBOAs.

- Proposing a well-diversified set of new interesting candidate solutions is a major challenge in P-SBOAs to allow for parallel simulations without wasting the computational budget. The surrogate-optimizer coupling mainly determines the Acquisition Process (AP), responsible of acquiring new candidates. APs have not been studied in the contexts of moderately expensive problems and landscapes with weak global structures. **Firstly, we propose SaaEF, a new P-SAEA where the surrogate is both employed as an evaluator and a filter to handle moderately expensive problems. Secondly, we come up with q-subnets and q-post-HMC, two novel P-SDAs with an AP based on sub-surrogates sampling to manage multi-modal landscapes with weak global structure.** The approaches are experimentally compared with the state-of-the-art q-EGO [GRC10].

- Defining the promisingness amounts to set a trade-off between exploration and exploitation. Recognizing as promising the uncertain solutions boosts diversification while considering as promising the solutions with a favorable predicted objective value enhances intensification. Adequately balancing exploration and exploitation is critical in optimization [Tal09]. In P-SAEAs and P-SDAs, the promisingness is implemented by the Evolution Control (EC) and the Infill Criterion (IC) respectively. All the existing ECs and the overwhelming majority of ICs are static [LHS12]. **In this thesis, ensembles of ECs and ICs are proposed to dynamically adapt the trade-off between diversification and intensification during the search.** They are tested against widely-used criteria such as EI.

- Combining different surrogates, APs and ECs or ICs within a hybrid P-SBOA brings the hope for robust methods according to the problems characteristics. A first attempt conducting in this direction has shown a serious drawback regarding the scaling in the number of computing cores [Reh+18]. **We propose a hybrid method**

borrowing the AP from P-SAEA and q-EGO and encompassing two surrogates and two ECs. Our method proves to be reliable on the real-world application to Covid-19 contact reduction for distinct computational budgets and a high number of computing cores.

- Coding P-SBOAs appeals for a software platform that gathers re-usable components together and sets down the foundations to ensure components inter-changeability. **We come up with pySBO, a modular Python platform facilitating P-SBOAs implementation.** All the approaches investigated in the present study are implemented *via* pySBO. Besides, the extensibility of the platform is proved by adding new algorithmic components stemming from multi-objective optimization. The utility of pySBO is demonstrated by the resolution of a multi-objective problem of Covid-19 vaccine distribution. Freely available at https://github.com/GuillaumeBriffoteaux/pySBO, open-source and exemplified, pySBO is documented through dedicated web-pages accessible at https://pysbo.readthedocs.io to maximize its accessibility to the scientific community.

Thesis structure
This manuscript is organized in five chapters:

Chapter 1 provides the necessary background to understand the methods developed to solve expensive black-box simulation-based optimization problems. At the crossroads of machine learning, optimization and parallel computing, the terminology and notations are established prior to describe the EAs, the design stages of P-SBOAs, the surrogate building and the surrogate-optimizer couplings. Afterwards, a review of the literature of the field of Parallel Surrogate-based Optimization (P-SBO) is carried out. The reviewed publications are focused on the computational budget definition, the surrogate model selection, the definition of promisingness and the acquisition processes. Finally, the real-world application to Covid-19 contact reduction and the benchmark problems are outlined.

Chapter 2 focuses on the design of P-SAEAs. The BNN_MCD surrogate is introduced and confronted empirically to recognized interpolation and regression models externally to the optimization loop. It is then integrated as an evaluator and a filter, thus producing the new SaaEF. To define the promisingness, state-of-the-art ECs are dissected and ensembles of ECs are proposed to adapt it at execution time. Finally, the calibration of the methods and numerical experiments are reported to investigate the impact of the ECs, surrogates and APs with respect to the search landscape characteristics.

Chapter 3 concerns the design of P-SDAs. Firstly, the analogy between ECs and ICs is established, and hence, the proposed ensembles of ECs are used to revisit q-EGO. Secondly, fast APs, namely q-subnets, q-post-HMC and q-Pareto, are explained. The sampling of q sub-networks from a global net and the sampling of q sets of parameters from the posterior of a Bayesian model are the base principle of q-subnets and q-post-HMC respectively. The q-Pareto strategy is built around bi-criterion ECs. Thirdly, extensive numerical experiments involving multiple APs, ECs, surrogates and training-set sizes are performed to extract knowledge about the behaviour of diverse architectures on an extended range of scenarios.

Chapter 4 begins with a confrontation of P-SAEAs *versus* P-SDAs according to all the possible design choices (APs, ECs, surrogates). The accuracy and computational complexity of the methods with respect to the computational budgets and the search landscape features are particularly scrutinized. The chapter is continued with the con-

struction of parallel hybrid methods including two surrogates and two APs, each equipped with its own EC. Numerical tests show the merit of the new hybrid algorithms (HCAP and HSAP) compared with another state-of-the-art parallel hybrid method (SMBO+EA) especially regarding parallel scalability. The chapter is ended with a posterior landscape analysis of the real-world Covid-19 contact reduction problem, made possible thanks to the simulations accumulated throughout the experimental stages.

Chapter 5 is dedicated to the modular pySBO Python platform proposed in this thesis. The modular structure reflecting a scalable code architecture is exposed and the publicly available documentation is presented. The extensibility of the platform is then demonstrated through the incorporation of new algorithmic components allowing to implement multi-objective algorithms. The usefulness of pySBO is exhibited through an empirical comparison of multi-objective approaches and the resolution of a real-world Covid-19 vaccine distribution problem.

This thesis is ended with a summary of the knowledge gained and guidelines on how to choose P-SBOAs to solve black-box expensive simulation-based optimization problems. Lastly, perspectives and future works are drawn.

Chapter 1

Parallel Surrogate-based Optimization

Contents

1.1 Introduction

At the intersection of optimization, machine learning and parallel computing, the field of Parallel Surrogate-based Optimization (P-SBO) arose at the end of the 20^{th} century to solve expensive black-box optimization problems. This chapter brings the background to P-SBO and provides a literature review centered on the challenges tackled in this investigation. Finally, the optimization problems tackled in this thesis are introduced and detailed.

1.2 Solving expensive black-box simulation-based optimization problems

1.2.1 Search landscape and expensiveness

Search landscape

In this thesis, we deal with continuous single-objective minimization problems. The task is to find the decision vector $\boldsymbol{x} \in \mathcal{D}$ minimizing the objective value $y = f(\boldsymbol{x})$ where $f : \mathcal{D} \to \mathcal{M}$ is called the objective function. Focusing on minimization does not damage generalization as maximization problems are recovered by setting the objective function to $-f$. The search space $\mathcal{D}$ and the objective space $\mathcal{M} = f(\mathcal{D})$ are both real spaces of dimension $d \in \mathbb{N}^* \backslash \{1\}$ and $m = 1$ respectively.

The graph $(\mathcal{D}, f(\mathcal{D}))$ of f is called the search landscape by analogy with topography [Tal09]. To illustrate the metaphor, let's assume that $d = 2$. In this particular case, minimizing f could be thought of as finding the spot of lowest altitude in a delimited geographical region. It is relatively easy to locate a low altitude spot in the Netherlands, a country characterized by flatness. Conversely, the rugged relief map of the Alps range makes the prospecting more arduous. The diversity of the landscapes' shapes (multi-modal, weak global structure, *etc.*) entails heterogeneous degrees of complexity of the resolution. Multi-modal landscapes are sprinkled with local optima and care must be taken to avoid getting trapped in a local basin of attraction [Ker+15]. For landscapes with weak global structures, meta-data are not informative enough to properly guide the search [Mer+11].

Expensiveness

Expensive black-box simulation-based optimization problems arise from the proliferation of simulation software tools in many fields. Six-hours-long simulations are performed in [GFL08; Gla+09] to optimize helicopter rotor blades. Conversely, in [AS19], a watershed model calibration involves simulations lasting only 10 seconds. The expensiveness is not strictly defined but rather depends on the computational and time resources at hand as well as the search landscape characteristics. In any case, a strong limitation of the computational budget allocated to the search implies a harder resolution.

From the optimizer point of view, the simulator is a black box as its internal functioning is masked. This masking uncorrelates the simulation code from the optimization program thus facilitating the management and maintenance of both software tools. The black-box nature restraints the information about the problem to prior expectations transmitted by the developers of the simulator.

The expensiveness and the black-box aspect hamper to infer the shape of the search landscape, which could have helped the choice of an appropriate solver. Besides, in case of constrained problems, the severity of the constraint complicates the localization of the feasible part of the search space and hence further complicating the landscape analysis. In this troublesome situation, it is pertinent to appeal to evolutionary algorithms.

1.2.2 Evolutionary Algorithms

Global picture of Evolutionary Algorithms
An Evolutionary Algorithm (EA) is a meta-heuristic that has been employed in numerous studies [Tal09]. It is inspired from the natural evolution of species according to which a population evolves through life cycles (generations) of birth, reproduction and death of its members. Algorithm 1 depicts the steps of an EA. First, a set of n_{pop} individuals (candidate solutions) is sampled from the search space and evaluated by the objective function to create the initial population. Next, this population goes through cycles of selection, reproduction and replacement to form new generations of individuals. The stopping criterion is generally based on the computational budget.

Algorithm 1 Evolutionary Algorithm

Input
n_{pop}: population size
n_{gen}: maximum number of generations

1: $\mathcal{P} \leftarrow$ initial_sampling(n_{pop})
2: evaluation($\mathcal{P}$)
3: **for** $i = 1 : n_{gen}$ **do**
4: $\quad \mathcal{P}_{par} \leftarrow$ select_parents($\mathcal{P}$)
5: $\quad \mathcal{P}_{chld} \leftarrow$ reproduction($\mathcal{P}_{par}$)
6: $\quad$ evaluation($\mathcal{P}_{chld}$)
7: $\quad \mathcal{P} \leftarrow$ replacement($\mathcal{P}$, $\mathcal{P}_{chld}$)
8: **end for**
9: **return** best individual(s) from $\mathcal{P}$

The main challenge when conceiving an EA is to balance between exploration and exploitation of the search space. Favoring exploration tends to search unknown regions of the search space whereas favoring exploitation tends to scrutinize on regions where good candidates have previously been observed (promising regions).

The optimization problems presented in Sub-section 1.6 are all continuous thus an individual is encoded in the EA as a decision vector made of the continuous decision variables of the problem. The objective function is a black-box simulator that can be used by the EA without additional modification.

Initialization
At initialization (line 1 in Algorithm 1), a crucial care should be taken to offer a well diversified initial population so as to avoid premature convergence. A random sampling does not guarantee a uniform coverage of the search space. Latin Hyper-cubes Sampling (LHS) consists to sub-divide the search space into hyper-cubes of equal sizes and to add sample points such that escaping each occupied hyper-cube is feasible in all the directions parallel to the axes. The set of all possible Latin Hyper-cube samplings is searched to identify the sampling maximizing the uniform spread. This popular technique [FSK08d] guarantees both a uniform coverage of the space and a uniform distribution of the projections of the initial points onto the axes. An example of LHS of four points in a two-dimensional space is displayed in Figure 1.1. To complete the initialization, the population is evaluated (line 2 in Algorithm 1).

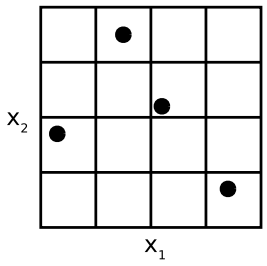

Figure 1.1: Example of Latin Hyper-cube Sampling of four points in two dimensions.

Selection
After initialization, a new generation starts with the selection of parent individuals (line 4 in Algorithm 1). Selected parents have the opportunity to transmit their characteristics to the next generation. The main principle of selection is that better is an individual in terms of objective value, better is its chance to be selected as parent. The tournament selection randomly samples n_t individuals with replacement from the population and only retains the best one. The process is repeated as many times as the required number of parents.

Reproduction
Children individuals are generated according to two reproduction operators: cross-over and mutation (line 5 in Algorithm 1). The role of the cross-over operator is to transmit parent characteristics to the children while the role of the mutation operator is to randomly disturb a child by applying a small perturbation to its decision variables. Both operators are applied according to a probability and should be chosen such that the proposed solutions are always valid and that every solution from the search space can be reached. The probability of cross-over p_c represents the probability to actually apply the cross-over operator to each couple of parents. The probability of mutation p_m represents the probability of applying the mutation operator to each decision variable of an individual.

The Simulated Binary Cross-over (SBX) lies among the most popular cross-over operators [Tal09]. As a parent-centric operator, the generated children are located in the neighborhood of one of their parents. Let's denote $\boldsymbol{x_p^{(1)}}, \boldsymbol{x_p^{(2)}}$ the parents and $\boldsymbol{x_c^{(1)}}, \boldsymbol{x_c^{(2)}}$ the children. The SBX operator is given by:

$$x_{ci}^{(1)} = 0.5\lfloor(1+\beta_i)x_{pi}^{(1)} + (1-\beta_i)x_{pi}^{(2)}\rfloor \tag{1.1}$$

$$x_{ci}^{(2)} = 0.5\lfloor(1-\beta_i)x_{pi}^{(1)} + (1+\beta_i)x_{pi}^{(2)}\rfloor \tag{1.2}$$

where:

$$\beta_i = \begin{cases} 2r_i^{\frac{1}{\eta_c+1}} & \text{if } r_i \leqslant 0.5 \\ \left(\frac{1}{2(1-r_i)}\right)^{\frac{1}{\eta_c+1}} & \text{otherwise} \end{cases} \tag{1.3}$$

The parameter η_c is the user-defined cross-over distribution index and $r_i \in [0,1]$ is a random number. When $\beta_i > 1$, the children are located out of the hyper-cube formed by the parents in the search space whereas they are inside the hyper-cube when $\beta_i < 1$. When $\beta_i = 1$, the children are the same than the parents.

As a prominent example of mutation operator is the polynomial mutation [Tal09]. Let's denote by $\boldsymbol{x_c}$ a child resulting from cross-over and $\boldsymbol{x_{mc}}$ the resulting mutated child:

$$\boldsymbol{x_{mc}} = \boldsymbol{x_c} + \boldsymbol{m} \tag{1.4}$$

where the perturbation vector $\boldsymbol{m}$ is given by:

$$m_i = (x_{ci}^U - x_{ci}^L)\delta_i \tag{1.5}$$

where x_{ci}^U and x_{ci}^L are the upper bound and the lower bound for the i-th component of $\boldsymbol{x_c}$ respectively and δ_i is given by:

$$\delta_i = \begin{cases} (2r_i)^{\frac{1}{\eta_m+1}} - 1 & \text{if } r_i < 0.5 \\ 1 - (2(1-r_i))^{\frac{1}{\eta_m+1}} & \text{otherwise} \end{cases} \tag{1.6}$$

The parameter η_m is the user-defined mutation distribution index and $r_i \in [0, 1]$ a random number.

Evaluation and Replacement
The next step of the life cycle aims at forming the new population by selecting individuals from a pool made of the current population and the generated children. Generally, the elitist selection strategy is considered. It consists in selecting the n_{pop} individuals presenting the best objective value.

Balancing exploration and exploitation
The balance between exploration and exploitation in the EA is realized by tuning the parameters according to the problem at hand. If no cross-over is performed, exploitation is enhanced thus low values of p_c enhances exploitation. Conversely, the mutation favors exploration by randomly disturbing the individuals thus high values of p_m boosts exploration. Larger population sizes n_{pop} favor exploration as a more extended region of the search space may be covered. Elitist replacement and tournament selection promote exploitation. The guidelines provided in [Tal09] advise to set $p_c \in [0.3, 0.9]$, $p_m = \frac{1}{d}$ where d is the number of decision variables and $n_{pop} \in [20, 200]$.

Expensiveness-related issue
In practice, it is reported that a large number of objective function evaluations is required by the EA to obtain good results [Tal09]. This may be a problem when tackling expensive optimization problems with a limited computational budget. Coupling the EA with a cheap-to-evaluate approximation model of the simulator and performing simulation-based evaluations in parallel are efficient ways to overcome this hindrance.

1.2.3 Design stages of P-SBOAs

Parallel Surrogate-based Optimization Algorithms (P-SBOAs) emerge from the synergy between parallel computing, surrogate modelling and optimization. Five stages are identified in [FSK08b] to design a P-SBOA:

1. Pre-processing;
2. Surrogate selection;
3. Optimizer specification;
4. Surrogate-optimizer coupling;
5. Parallel approach specification.

Pre-processing
The pre-processing stage consists in a landscape analysis and an initial sampling. The goal of landscape analysis is to gain information about the landscape to optimize in order to simplify the problem or to guide the choice made at the next stages. In the context of

expensive black-box problems, landscape analysis is rarely possible for the reasons aforementioned in Sub-section 1.2.1. The initial sampling provides a first set of candidate solutions that are simulated to constitute an initial database. This is the Design of Experiments (DOE). In this thesis, LHS is employed.

Surrogate selection
The surrogate modelling of f lies in determining an approximation $\hat{f}$ such that the predictions best fit the available training dataset and the unseen data at the same time. The training dataset is made of the available simulated points at a given time. Approximating well the unseen data is an important issue in machine learning called generalization [FSK08a]. To generalize well, the model type for $\hat{f}$ must be chosen such that its capacity is high enough to represent f but low enough to avoid over-fitting. Model capacity can be roughly defined as the extent of functions the model can represent (*e.g.* a first-order polynomial model has lower capacity than a second-order polynomial model). When insufficient model capacity is considered, the phenomenon of under-fitting occurs and prevent the model to accurately represent the target function. When an excessive model capacity is picked out, the model might fit the training data too precisely and capture the noise in addition to the overall behaviour we are seeking to seize. This phenomenon, called over-fitting, is to be avoided and model selection, hyper-parameters calibration and regularization techniques can help. In practice, it is tedious to choose and calibrate adequately the model for a new black-box function f.

Optimizer specification
During the optimizer specification stage, the intrinsic parameters to the EA, such as the population size and the cross-over probability, are adjusted to the problem at hand. The laboriousness of this task resides in the large number of combinations of the parameters values.

Surrogate-optimizer coupling
The surrogate-optimizer coupling is the cooperation strategy between the surrogate and the EA. The coupling defines the Acquisition Process (AP), that is the way new candidates are proposed for simulation. In the family of Parallel Surrogate-Assisted Evolutionary Algorithms (P-SAEAs), the surrogate is used as an evaluator or a filter at the evaluation step of the EA through an Evolution Control (EC). In the family of Parallel Surrogate-Driven Algorithms (P-SDAs), the surrogate drives the search through the Infill Criterion (IC). The EC and IC set the promisingness of new candidates. The balance between exploration and exploitation, crucial and definitely challenging in optimization, is significantly affected by the AP, EC and IC.

Parallel approach specification
The parallel approach specification stage is closely related to the previous stage. The idea is to efficiently use the available computing cores to improve the search outcomes. Parallelism is fine-grained when a high number of small computational tasks are executed in parallel, and coarse-grained when a small number of important computational workloads are performed simultaneously. In the current context, the fine granularity corresponds to parallelizing the internal of the simulator, which is out of the scope of this thesis as the simulator is black-box. Coarse granularity is instead envisioned by running multiple simulations in parallel. Effectively operating the computational load balancing is a dilemma when the number of simultaneous simulations is not a multiple of the available computing cores or when the simulation duration is not homogeneous.

In this study, we are particularly interested in the surrogate model selection and calibration of hyper-parameters (stage 2), the coupling between the surrogate and the EA by designing the AP, EC and IC (stage 4) and the parallel strategy (stage 5).

1.3 Surrogate building

1.3.1 Generalities

Terminology
In the machine learning literature, f is called the black-box mapping while $\boldsymbol{x}$ and y are referred to as the input vector and the target respectively. In this section, d is the number of scalar inputs and m is the number of scalar targets. The set of n available data pairs $(\boldsymbol{x^{(i)}}, y^{(i)})_{1 \leqslant i \leqslant n}$ is called the training set where $\boldsymbol{x^{(i)}}$ have been generated by a sampling method and $y^{(i)} = f(\boldsymbol{x^{(i)}})$. The set of training input vectors is denoted by a matrix X where $X_{i,j} = x_j^{(i)}$ and the set of associated training outputs is expressed by a vector $\boldsymbol{y}$ made of n components. The approximation of f is symbolized by $\hat{f}$ while the predictive standard deviation writes $\hat{s}$. Table A.1, displayed in the Appendix section, puts into perspective the terminology in use for machine learning and optimization.

Uncertainties, noise, regression and interpolation
The simulation code underlying the expensive objective function f relies on assumptions that simplify the reality and can consequently be considered as approximation of what would be actually observed in the real-world. For instance, solving ordinary or partial differential equations implies to discretize time and space *via* meshes whose discretization step size generates imprecision in the simulation outcomes. When approximating f, this imprecision is considered as irreducible as no access is granted to modify the black-box simulator and it can consequently be comparable to measurement imprecision in physical experiments [Gal16]. In all the surrogate models presented in this section, this imprecision is modelled as data-related noise and is assumed to be Gaussian:

$$\begin{aligned} y^{(i)} &= \hat{f}(\boldsymbol{x^{(i)}}; \boldsymbol{w}) + \epsilon \\ \epsilon &\sim \mathcal{N}(0, \beta^{-1}) \end{aligned} \tag{1.7}$$

where β^{-1} is a hyper-parameter standing for the variance of data-related noise and $\boldsymbol{w}$ are the parameters of the model. *A priori* knowledge about the data-related noise can serve to decide whether a regression or an interpolation surrogate model is to be preferred. High data-related noise could favor a regression model that may provide a smoother curve than interpolation and consequently only capture the overall behaviour of f. Conversely, low data-related noise may favor an interpolation model that guarantee zero prediction error on the observed training points.

In the Bayesian surrogate models presented hereafter, imprecision related to the surrogate parameters $\boldsymbol{w}$ is also taken into account by the prior distribution. The Gaussian prior is given by:

$$p(\boldsymbol{w}|\alpha) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{0}, \alpha^{-1} I) \tag{1.8}$$

where the hyper-parameter α stands for uncertainty around the parameters values. This uncertainty is qualified as reducible as it could be attenuated by gathering more observations [Gal16].

Both reducible and irreducible uncertainties can be utilized to produce the predictive standard deviation. Predictive standard deviation is a precious information in P-SBOAs because it could guide the decision of evaluating a new candidate solution with the expensive function or only predicting its objective value with the cheaper and coarser surrogate model.

Training, calibration, frequentist and Bayesian treatment
Building a surrogate consists in specializing a regression or interpolation model to mimic the target function f. Roughly speaking, this is done in a bi-level optimization approach where the inner optimization problem lies in learning the parameters $\boldsymbol{w}$ (training) while the outer optimization determines the hyper-parameters $\boldsymbol{\theta}$ (calibration). The purpose of training is to fit the training data while the aim of calibration is to generalize well. Two probability-related approaches are usually followed to formulate these optimization problems: frequentist and Bayesian.

In the frequentist view, assuming a Gaussian noise affecting the data (Equation 1.7), the parameters are chosen by maximizing the likelihood function that represents the probability of the observations given the parameters $p((X, \boldsymbol{y})|\boldsymbol{w})$. This maximization can be conducted analytically for some low capacity models, but further optimization techniques such as gradient-based methods or meta-heuristics can be required for higher capacity models [Bis06]. Maximum likelihood estimation allows to determine point estimates of the parameters and is known to induce over-fitting.

In the Bayesian view, over-fitting is avoided by accounting for both uncertainty in the estimation of the parameters and uncertainty around the data (noise) [Bis06]. In the Bayesian formalism, a prior probability distribution over the parameters is defined (Equation 1.8) and Bayes' theorem is applied to capture the knowledge provided by the observations through the likelihood:

$$p(\boldsymbol{w}|X, \boldsymbol{y}) \propto p((X, \boldsymbol{y})|\boldsymbol{w}).p(\boldsymbol{w}|\alpha) \tag{1.9}$$

or, written in words:

$$\text{posterior} \propto \text{likelihood} \times \text{prior}$$

The resulting probability distribution $p(\boldsymbol{w}|X, \boldsymbol{y})$, called the posterior, incorporates uncertainty around the parameters. Finally, the integration over the parameters space (marginalization) is carried out to produce the predictive distribution:

$$p(y^*|\boldsymbol{x^*}, X, \boldsymbol{y}) = \int p(y^*|\boldsymbol{x^*}, \boldsymbol{w}).p(\boldsymbol{w}|X, \boldsymbol{y})d\boldsymbol{w} \tag{1.10}$$

where

$$p(y^*|\boldsymbol{x^*}, \boldsymbol{w}) = \mathcal{N}(\hat{f}(\boldsymbol{x^*}; \boldsymbol{w}), \beta^{-1}) \tag{1.11}$$

according to Equation 1.7. The predictive distribution $p(y^*|\boldsymbol{x^*}, X, \boldsymbol{y})$ consequently incorporates uncertainty information about the prediction for an unseen input vector $\boldsymbol{x^*}$. In the frequentist view, this uncertainty information can only be approximated by strategies such as bootstrapping. With bootstrapping, training is realized on multiple sub-sets of the training set and the variability of the associated predictions is recorded.

Generally speaking, training and calibration are more difficult when formalized through the Bayesian point of view and are intractable in many cases due to the marginalization over the parameters space. Approximation-based methods exist but can still be computationally expensive. When tackling expensive problems, the computational budget invested into surrogate updates should not hinder the search in comparison with investments in expensive evaluations. Selecting the prior over the parameters is also a difficulty inherent to Bayesian approach as a poor choice of this distribution could cause bad predictions with high confidence.

1.3.2 Linear models

Description of the model
A linear model [FSK08a; Bis06] is defined as a linear combination of m_b non-linear basis functions ϕ_j of the input vector and is formulated as:

$$\hat{f}(\boldsymbol{x}; \boldsymbol{w}) = \sum_{j=0}^{m_b-1} w_j \phi_j(\boldsymbol{x}) = \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x}) \tag{1.12}$$

The parameters of the model are the weights $\boldsymbol{w}$. The linearity of the model with respect to the parameters confer some ease for training. A widespread choice for the basis functions is the Radial Basis Functions (RBF) that depend on the distance between the input vector and a given centre:

$$\phi_j(||\boldsymbol{x} - \boldsymbol{c^{(j)}}||) \tag{1.13}$$

where $\boldsymbol{c^{(j)}}$ is the centre associated to the j-th basis function. Different forms for ϕ have been proposed, such as:

- Polynomial $\phi_j(r) = r^j$;
- Gaussian $\phi_j(r) = \exp\left(\frac{-r^2}{2s^2}\right)$.

The polynomial basis function is an example of fixed basis function as no additional parameter is introduced. Conversely, the Gaussian basis function is an example of parametric function as the parameter s controlling the scale has been added to the model. The number m_b of basis functions, the additional parameter such as s and the centres $\boldsymbol{c^{(j)}}$ can all be treated as hyper-parameters.

Likelihood approach
For the moment, let's assume that the hyper-parameters are known. Under the Gaussian assumption over the data-related noise (1.7), let's assume that the probability distribution of the target y is given by:

$$p(y|\boldsymbol{x}, \boldsymbol{w}, \beta) = \mathcal{N}(y|\hat{f}(\boldsymbol{x}; \boldsymbol{w}), \beta^{-1}) \tag{1.14}$$

Assuming the observations $(\boldsymbol{x^{(i)}}, y^{(i)})_{1 \leqslant i \leqslant n}$ independent and identically distributed from $\mathcal{N}(\hat{f}(\boldsymbol{x}; \boldsymbol{w}), \beta^{-1})$, the likelihood function is then given by:

$$p(\boldsymbol{y}|X, \boldsymbol{w}, \beta) = \prod_{i=1}^{n} \mathcal{N}(y^{(i)}|\boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x^{(i)}}), \beta^{-1}) \tag{1.15}$$

Maximizing (1.15) with respect to $\boldsymbol{w}$ corresponds to minimizing the sum-of-square error function:

$$E(\boldsymbol{w}) = \frac{1}{2} \sum_{i=1}^{n} (y^{(i)} - \boldsymbol{w}^T \boldsymbol{\phi}(\boldsymbol{x^{(i)}}))^2 \tag{1.16}$$

Solving the maximum likelihood problem thus gives:

$$\boldsymbol{w_{ML}} = (\Phi^T \Phi)^{-1} \Phi^T \boldsymbol{y} \tag{1.17}$$

where Φ is the design matrix whose coefficients are $\Phi_{i,j} = \phi_j(\boldsymbol{x^{(i)}})$. The maximization of the likelihood with respect to β gives:

$$\beta_{ML} = \frac{1}{n} \sum_{i=1}^{n} (y^{(i)} - \boldsymbol{w_{ML}}^T \boldsymbol{\phi}(\boldsymbol{x^{(i)}}))^2 \tag{1.18}$$

and can be interpreted as the variance of the targets around the linear model.

Interpolation and regression
The interpolation RBF model is commonly obtained by setting $m_b = n$ and $\boldsymbol{c}^{(j)} = \boldsymbol{x}^{(j)}$. In this case:

$$\boldsymbol{w_{ML}} = \Phi^{-1}\boldsymbol{y} \tag{1.19}$$

where Φ is symmetric positive definite and well conditioned providing that training input vectors are spaced out enough.

A first way to obtain a regression model is to set the number of basis functions $m_b < n$. The resulting matrix in (1.17) is consequently non-square and the estimate $\boldsymbol{w_{ML}}$ is a least square estimate. A second way to obtain a regression model is to add a regularization parameter in the error function (1.16):

$$E_R(\boldsymbol{w}) = \frac{1}{2}\sum_{i=1}^{n}(y^{(i)} - \boldsymbol{w}^T\boldsymbol{\phi}(\boldsymbol{x^{(i)}}))^2 + \frac{\lambda}{2}\boldsymbol{w}^T\boldsymbol{w} \tag{1.20}$$

and in the case where $m_b = n$, (1.19) would become:

$$\boldsymbol{w_{MLR}} = (\Phi + \lambda I)^{-1}\boldsymbol{y} \tag{1.21}$$

The additional effect of introducing a regularization term is to push some weights close to zero and consequently prevent over-fitting by reducing the model complexity. The value of λ should be set to β to reflect the data-related noise but this value is often unknown. The drawback for this approach is then the tuning required to set the new hyper-parameter λ.

Bayesian approach
Even if regularization prevents over-fitting, the number and the form of the basis functions still play a role in determining the model complexity. The Bayesian approach overcomes this problem [Bis06].

The Gaussian assumption on data-related noise (1.14) and the likelihood function (1.15) still hold and the prior distribution over the parameters is assumed to be given by (1.8). Applying the Bayes' theorem provides the posterior distribution:

$$p(\boldsymbol{w}|X, \boldsymbol{y}, \alpha, \beta) = \mathcal{N}(\boldsymbol{w}|\boldsymbol{m}, S) \tag{1.22}$$

where:

$$\boldsymbol{m} = \beta S\Phi^T\boldsymbol{y} \tag{1.23}$$

$$S^{-1} = \alpha I + \beta\Phi^T\Phi \tag{1.24}$$

Finally, the predictive distribution is obtained by marginalizing over the space of parameters:

$$p(y^*|\boldsymbol{x^*}, X, \boldsymbol{y}, \alpha, \beta) = \int p(y^*|\boldsymbol{x^*}, \boldsymbol{w}, \beta).p(\boldsymbol{w}|X, \boldsymbol{y}, \alpha, \beta)d\boldsymbol{w} \tag{1.25}$$

$$= \mathcal{N}(y^*|\boldsymbol{m}^T\boldsymbol{\phi}(\boldsymbol{x^*}),\ \beta^{-1} + \boldsymbol{\phi}(\boldsymbol{x^*})^T S\boldsymbol{\phi}(\boldsymbol{x^*})) \tag{1.26}$$

It is interesting to note that the first term in the predictive variance is due to the data-related noise while the second term stands for the uncertainty around the parameters $\boldsymbol{w}$.

For the linear model, the predictive mean in Equation (1.26) can be re-written as:

$$\boldsymbol{m}^T\boldsymbol{\phi}(\boldsymbol{x^*}) = \sum_{i=1}^{n} k(\boldsymbol{x^*}, \boldsymbol{x^{(i)}})y^{(i)} \tag{1.27}$$

where $k(\boldsymbol{x}^*, \boldsymbol{x}^{(i)}) = \beta\boldsymbol{\phi}(\boldsymbol{x}^*)^T S\boldsymbol{\phi}(\boldsymbol{x}^{(i)})$. For a Gaussian basis function, the value of $k(\boldsymbol{x}^*, \boldsymbol{x}^{(i)})$ decreases when $\boldsymbol{x}^*$ and $\boldsymbol{x}^{(i)}$ move away from each other. The consequence is for the contribution of $y^{(i)}$ to be reduced when $\boldsymbol{x}^{(i)}$ is far from $\boldsymbol{x}^*$. Information close to the new input vector is given stronger influence on the prediction. This behaviour is reasonably desirable when treating real-world problems.

Hyper-parameters
A Bayesian approach can also be considered to set the values of the hyper-parameters β and α, but unlike the determination of the parameters, this approach is analytically intractable. As a workaround, it is possible to maximize the likelihood (also called the evidence function).

Advantages and disadvantages
The linearity of the model with respect to the parameters allows to derive analytically the point estimate in the likelihood approach and the predictive distribution in the Bayesian approach [Bis06]. Moreover, the non-linearity with respect to the inputs allows to represent a large range of black-box mappings. Nevertheless, there are two important drawbacks attributed to linear models. The first drawback is that high confidence can be given in the prediction far from the training input vectors. Indeed, the second term of the predictive variance in (1.26) tends to zero far from the training input vectors when a Gaussian RBF is considered. To answer this drawback, the Gaussian Process models presented below in Sub-section 1.3.3 are employed. The second drawback is that the number of basis functions tends to increase drastically when the input dimensionality d increases. Because inverting a matrix of dimension m_b^2 has a computational complexity of $O(m_b^3)$, the training cost of the model can become high. To overcome this disadvantage, the Artificial Neural Network model, presented in Sub-section 1.3.4, that relies on adaptive basis functions is used.

1.3.3 Gaussian Processes

Kernels
Instead of selecting a basis function ϕ, it is possible to select a kernel function k and derive the model from it [Bis06]. This is the base principle of Gaussian Processes (GPs). The advantage is that the feature vector $\boldsymbol{\phi}(\boldsymbol{x})$ can potentially lie in an infinite dimensional space. In other terms, some functions k are expressed as an infinite combination of features. It is actually the case for the Gaussian kernel:

$$k(\boldsymbol{x}, \boldsymbol{x'}) = \sigma \exp\left(-\frac{||\boldsymbol{x} - \boldsymbol{x'}||^2}{2s^2}\right) \tag{1.28}$$

where σ is the scale and s the length scale which determine the importance and the extent of influence of one point to another respectively. As a consequence, the model capacity allows to represent more complex functions than the linear models.

Gaussian Processes
By the GP approach, the observed outputs $\{y^{(1)}, \ldots, y^{(n)}\}$ are considered as realizations of the artificial random variables $\{Y(\boldsymbol{x}^{(1)}), \ldots, Y(\boldsymbol{x}^{(n)})\}$ respectively [Ras06]. Any subset of these random variables have a jointly Gaussian distribution determined by a mean $\boldsymbol{\mu}$ and a covariance matrix Σ which is symmetric positive semi-definite:

$$\begin{pmatrix} Y(\boldsymbol{x}^{(1)}) \\ \vdots \\ Y(\boldsymbol{x}^{(n)}) \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} \mu \\ \vdots \\ \mu \end{pmatrix}, \Sigma\right) \tag{1.29}$$

In the case of the Bayesian linear regression model (1.12) with the prior on the weights defined by Equation (1.8), the mean and covariance function of the GP are given by:

$$\mathbb{E}[\hat{f}(\boldsymbol{x};\boldsymbol{w})] = \boldsymbol{\phi}(\boldsymbol{x})\mathbb{E}[\boldsymbol{w}] = \mathbf{0} \tag{1.30}$$

$$\mathbb{E}[\hat{f}(\boldsymbol{x};\boldsymbol{w})\hat{f}(\boldsymbol{x'};\boldsymbol{w})] = \boldsymbol{\phi}(\boldsymbol{x})^T\mathbb{E}[\boldsymbol{w}^T\boldsymbol{w}]\boldsymbol{\phi}(\boldsymbol{x'}) = \alpha^{-1}\boldsymbol{\phi}(\boldsymbol{x})^T\boldsymbol{\phi}(\boldsymbol{x'}) \tag{1.31}$$

respectively. In this case, the kernel function would be defined by:

$$k(\boldsymbol{x},\boldsymbol{x'}) = \boldsymbol{\phi}(\boldsymbol{x})^T\boldsymbol{\phi}(\boldsymbol{x'}) \tag{1.32}$$

and the covariance matrix would be $\Sigma = \alpha^{-1}K$ where $K_{i,j} = k(\boldsymbol{x^{(i)}}, \boldsymbol{x^{(j)}})$. The prior over the parameters in the Bayesian linear model can be viewed as a prior over functions in the GP model.

Ordinary Kriging

Ordinary Kriging is a particular case of GP [Ras06]. It is assumed that $Y(\boldsymbol{x^{(i)}}) \sim \mathcal{N}(\mu, \alpha^{-1})$ where μ and α^{-1} are unknown. The covariance matrix of the joint Gaussian distribution is given by $\Sigma = \alpha^{-1}K$, where the correlation matrix K is defined *via* the parametrized kernel function:

$$k(\boldsymbol{x^{(i)}}, \boldsymbol{x^{(j)}}) = \exp\left(-\sum_{k=1}^{d}\eta_k|x_k^{(i)} - x_k^{(j)}|^{p_k}\right) \tag{1.33}$$

When $\boldsymbol{x^{(i)}}$ and $\boldsymbol{x^{(j)}}$ move away from each other, their difference tends to infinity and their correlation tends to 0. The hyper-parameters $\boldsymbol{p}$ define the peak of the correlation with a low value meaning an immediate decrease in the correlation when the two points begin to move away. The hyper-parameters $\boldsymbol{\eta}$ define the extent of the influence of the associated decision variable with low values indicating large influence. Building the Ordinary Kriging model consists in estimating μ, α^{-1}, $\boldsymbol{\eta}$ and $\boldsymbol{p}$.

Training and calibration

Let's first consider the interpolation Ordinary Kriging model [FSK08a; Jon01]. The first step consists in determining the parameters μ and α^{-1} by maximizing the likelihood function:

$$\mathcal{L}(X, \boldsymbol{y}|\mu, \alpha^{-1}) = \frac{1}{(2\pi\alpha^{-1})^{n/2}\det(K)^{1/2}}\exp\left(-\frac{(\boldsymbol{y}-\mathbf{1}\mu)^TK^{-1}(\boldsymbol{y}-\mathbf{1}\mu)}{2\alpha^{-1}}\right) \tag{1.34}$$

This maximization can be carried out analytically and yields:

$$\hat{\mu} = \frac{\mathbf{1}^TK^{-1}\boldsymbol{y}}{\mathbf{1}^TK^{-1}\mathbf{1}} \tag{1.35}$$

$$\hat{\alpha^{-1}} = \frac{1}{n}(\boldsymbol{y}-\mathbf{1}\mu)^TK^{-1}(\boldsymbol{y}-\mathbf{1}\mu) \tag{1.36}$$

The second step consists in determining the hyper-parameters $\boldsymbol{\eta}$ and $\boldsymbol{p}$ by maximizing the concentrated likelihood obtained by replacing μ and α^{-1} by $\hat{\mu}$ and $\hat{\alpha^{-1}}$ in (1.34). The maximization can not be performed analytically, therefore numerical optimization techniques such as meta-heuristics are used.

Prediction

Finally, for a new input vector $\boldsymbol{x^*}$, the associated prediction y^* is obtained by maximization of likelihood based on the augmented random vector:

$$\begin{pmatrix} Y(\boldsymbol{x^{(1)}}) \\ \vdots \\ Y(\boldsymbol{x^{(n)}}) \\ Y(\boldsymbol{x^*}) \end{pmatrix} \sim \mathcal{N}\left(\mathbf{1}\hat{\mu},\ \hat{\alpha^{-1}}\begin{pmatrix} K & \boldsymbol{k^*} \\ \boldsymbol{k^{*T}} & 1 \end{pmatrix}\right) \tag{1.37}$$

where $k_i^* = k(\boldsymbol{x^*}, \boldsymbol{x^{(i)}})$. The analytical resolution of the maximization yields the following predictive distribution:

$$p(y^*|\boldsymbol{x^*}, X, \boldsymbol{y}) = \mathcal{N}(y^*|\hat{\mu} + \boldsymbol{k^*}^T K^{-1}(\boldsymbol{y} - \boldsymbol{1}\hat{\mu}),\ \hat{\alpha^{-1}}(1 - \boldsymbol{k^*}^T K^{-1}\boldsymbol{k^*})) \tag{1.38}$$

where the term designating the uncertainty in estimating $\hat{\mu}$ has been neglected in the predictive variance. Unlike the linear models with Gaussian basis functions, the predictive variance given in (1.38) is high when $\boldsymbol{x^*}$ is located far from the training input vectors. The most computationally intensive part of the Kriging interpolation is the inversion of K that requires $O(n^3)$ computations [FSK08a].

Regression
The regression Ordinary Kriging model is obtained in the same way as its interpolating counterpart except that the correlation matrix K is replaced by $(K + \lambda I)$ [FSK08a]. The estimations of the parameters and hyper-parameters provide:

$$p(y^*|\boldsymbol{x^*}, X, \boldsymbol{y}) = \mathcal{N}(y^*|\hat{\mu_r} + \boldsymbol{k^*}^T (K + \lambda I)^{-1}(\boldsymbol{y} - \boldsymbol{1}\hat{\mu_r}),\ \hat{\alpha_r^{-1}}(1 + \lambda - \boldsymbol{k^*}^T (K + \lambda I)^{-1}\boldsymbol{k^*})) \tag{1.39}$$

The expressions for $\hat{\mu_r}$ and $\hat{\alpha_r^{-1}}$ are the same as (1.35) and (1.36), where K is replaced by $(K + \lambda I)$. In practice, λ is treated as a hyper-parameter and estimated by maximization of likelihood. The most computationally intensive part of the Kriging regression is the inversion of $(K + \lambda I)$ that requires $O(n^3)$ computations.

1.3.4 Artificial Neural Networks

Model, parameters and hyper-parameters
Artificial Neural Networks (ANNs) are models based on the composition of non-linear parametric basis functions [Bis06]. Contrary to linear models, ANNs are non-linear with respect to both input data and weights. The composition of basis functions provides a higher model capacity than the linear combination. Let's take as example the second-order polynomial basis functions. The linear combination of such functions allows one to represent appropriately second-order polynomials while the composition of such functions allows one to represent appropriately fourth-order polynomials.

The 2-layer ANN model is formulated as:

$$\hat{f}(\boldsymbol{x}; \boldsymbol{w}) = \sum_{j=1}^{m_u} w_j^{(2)} h\left(\sum_{i=1}^{d} w_{ji}^{(1)} x_i + w_{j0}^{(1)}\right) + w_0^{(2)} \tag{1.40}$$

where h is a non-linear activation function and m_u is the number of units composing the first layer (hidden layer). $w_{ji}^{(1)}$ and $w_{j0}^{(1)}$ are the weights and the biases of the first layer respectively, and $w_j^{(2)}$ and $w_0^{(2)}$ are the weights and the bias of the second layer (output layer) respectively. The model can be easily generalized to any number of layers.

The parameters of the model are the weights and biases while the number of layers, the number of units per layer, the connections between units, the activation functions are treated as hyper-parameters.

The universal approximation theorem states that a 2-layer ANN model with linear output can approximate any continuous function on a compact domain to arbitrary accuracy providing that the network has a sufficient number of units [Bis06].

Training
Under the Gaussian noise assumption given in Equation (1.7), let's write:

$$p(y|\boldsymbol{x}, \boldsymbol{w}) = \mathcal{N}(y|\hat{f}(\boldsymbol{x}; \boldsymbol{w}), \beta^{-1}) \tag{1.41}$$

Assuming independent and identically distributed observations, the maximization of the likelihood with respect to $\boldsymbol{w}$ corresponds to the minimization of the sum-of-square error function:

$$E(\boldsymbol{w}) = \frac{1}{2}\sum_{i=1}^{n}(\hat{f}(\boldsymbol{x^{(i)}};\boldsymbol{w}) - y^{(i)})^2 \tag{1.42}$$

The non-linearity of the network makes $E(\boldsymbol{w})$ non-convex and gives rise to a high number of local optima. Consequently, a numerical optimization method such as Gradient-Descent (GD) is used for training. As it is very unlikely to locate a global optimum such that $E(\boldsymbol{w}) = 0$, ANNs realize a regression task. GD is the approach commonly used and consists in relying on gradient information to iteratively update the parameters. Its batch version is given by:

$$\boldsymbol{w^{t+1}} = \boldsymbol{w^t} - \xi\nabla E(\boldsymbol{w^t}) \tag{1.43}$$

where $\nabla E(\boldsymbol{w^t})$ is the gradient of E evaluated on the whole training dataset. The sequential version of the GD algorithm considers to update the parameters once for each pair $(\boldsymbol{x^{(i)}}, y^{(i)})$ of training data as follows:

$$\boldsymbol{w^{t+1}} = \boldsymbol{w^t} - \xi\nabla E_i(\boldsymbol{w^t}) \tag{1.44}$$

The advantage of the sequential GD is the higher probability to escape local optima as a local optimum for the whole training dataset is not necessarily a local optimum for each of the training pairs [Bis06]. The magnitude of update is represented by ξ, called the learning rate, that is treated as a hyper-parameter. Further variants of the GD algorithm such as Adam take care of automatically adapting the learning rate during the search [GBC16]. Initialization of the parameters is also often considered as a hyper-parameter as initial values of the weights and biases have a strong impact on GD convergence.

Back-propagation

The back-propagation algorithm is used to evaluate the gradient ∇E_i for each parameter [GBC16]. It relies on the derivation chain rule to propagate the difference between the prediction and the target backwards in the network (from the output layer to the first hidden layer). In order to present the back-propagation algorithm, let's re-formulate the 2-layers ANN presented in Equation (1.40) as:

$$\hat{f}(\boldsymbol{x};\boldsymbol{w},\boldsymbol{b}) = \boldsymbol{w^{(2)}}^T\boldsymbol{h}(\boldsymbol{a^{(1)}}) + b^{(2)} \tag{1.45}$$

$$\boldsymbol{a^{(1)}} = W^{(1)}\boldsymbol{x} + \boldsymbol{b^{(1)}} \tag{1.46}$$

and let's denote $\hat{y} = \hat{f}(\boldsymbol{x};\boldsymbol{w},\boldsymbol{b})$. In the new formulation, $\boldsymbol{b^{(1)}}$ stands for the biases of the hidden layer and $b^{(2)}$ stands for the bias of the output layer. First, the weights and biases associated to the output layer are updated thanks to:

$$\nabla_{\boldsymbol{w^{(2)}}}E = \nabla_{\hat{y}}E \;.\; \nabla_{\boldsymbol{w^{(2)}}}\hat{y} = \boldsymbol{h}(\boldsymbol{a^{(1)}}) \tag{1.47}$$

$$\nabla_{b^{(2)}}E = \nabla_{\hat{y}}E \;.\; \nabla_{b^{(2)}}\hat{y} = 1 \tag{1.48}$$

Then, the weights and biases associated to the hidden layer are updated thanks to:

$$\nabla_{W^{(1)}}E = \nabla_{\hat{y}}E \;.\; \nabla_{W^{(1)}}\hat{y} = (\nabla_{W^{(1)}}\boldsymbol{h})^T \;.\; \boldsymbol{w^{(2)}} = (\nabla_{W^{(1)}}\boldsymbol{a^{(1)}})^T \;.\; \nabla_{\boldsymbol{a^{(1)}}}\boldsymbol{h} \;.\; \boldsymbol{w^{(2)}} \tag{1.49}$$

$$\nabla_{\boldsymbol{b^{(1)}}}E = \nabla_{\hat{y}}E \;.\; \nabla_{\boldsymbol{b^{(1)}}}\hat{y} = (\nabla_{\boldsymbol{b^{(1)}}}\boldsymbol{h})^T \;.\; \boldsymbol{w^{(2)}} = \nabla_{\boldsymbol{a^{(1)}}}\boldsymbol{h} \;.\; \boldsymbol{w^{(2)}} \tag{1.50}$$

Regularization

Over-fitting occurs in ANNs when the number m_u of units is excessively large to represent adequately the mapping f to be approximated. However, as the minimization of the sum-of-square error is not carried out exactly, m_u is not the only hyper-parameter affecting the generalization capacity of the model. Regularization methods have been proposed to reduce the generalization error [GBC16; Bis06].

The limitation of the capacity can be realized by forcing some parameters of the model to be very small (weight decay) and consequently reducing artificially the number of units. This can be done by applying a norm penalty to the error function such as:

$$\tilde{E}(\boldsymbol{w}) = E(\boldsymbol{w}) + \frac{\lambda_{decay}}{2}||\boldsymbol{w}||_2^2 \tag{1.51}$$

This is a particular case of parameter norm penalty called L^2 parameter regularization. The weight decay coefficient λ_{decay} is treated as a hyper-parameter.

It can be shown that weight decay can also be realized implicitly by applying the early stopping mechanism. This mechanism determines the number of GD iterations by monitoring a validation error during training. When the validation error stops to decrease, training is stopped and the best values of the parameters found so far are restored. Early stopping is often coupled with cross-validation because a validation dataset is required.

Reducing the generalization error can also be performed by averaging the predictions of an ensemble of models. Dropout offers an efficient approximation to ensemble averaging by temporarily and randomly dropping hidden units during training [Sri+14]. Each unit is therefore forced to learn from the data independently from other units and co-adaptation is thus prevented. The probability of dropping units p_{drop} is treated as a hyper-parameter.

Hyper-parameters calibration
Calibrating the hyper-parameters of the model is an active field of research [SMS19]. Approaches from cross-validation to Bayesian optimization have been proposed.

Bayesian Neural Networks
The Bayesian treatment to Neural Network can not be addressed analytically and is intractable for models of significant sizes [Bis06]. Indeed, as the model is non-linear with respect to its parameters, the posterior distribution on the weights becomes non-Gaussian. Variational inference proposes to approximate the posterior by a Gaussian distribution. In Chapter 2, an approximation to Bayesian Neural Network based on Dropout is presented.

1.3.5 Analysis of the models

The linear models presented previously provide an acceptable model capacity but the training that consists in inverting a matrix of size m_b^2 becomes expensive when the input dimension is high. Indeed, according to the curse of dimensionality, the number of basis functions m_b needs to grow rapidly when the input dimension d increases. The GPs such as Ordinary Kriging provide a better model capacity than the linear models and the further advantage to give a high predictive standard deviation far from the training input vectors. However, the complexity of training a GP is $\mathcal{O}(n^3)$ where n is the number of training samples. Assuming that the hyper-parameters are determined appropriately, ANNs are universal approximators. Their training is more scalable than the one of linear models and GPs but they can not perform interpolation. ANNs viewed through the likelihood point of view do not provide predictive standard deviation. Moreover, treating interesting ANNs model through the Bayesian stance can only be realized through approximations and expensive computations that make the predictive standard deviation tedious to obtain.

When choosing the surrogate model to tackle an expensive optimization problem, stringent constraints arise. The training duration should stay controllable to prevent wasting the budget. Therefore, the linear models may not be a good choice when the number of decision variables is large as it is the case in the applications tackled in this thesis. Moreover, for moderately expensive problems, the training set is expected to reach a moderate size as the search proceeds thus training a GP model could be excessively costly yet. ANNs

could be preferable regarding the computational cost of training but no uncertainty information can be exploited and over-fitting could occur. ANNs are known to be accurate when a large training set is available and that is not the case for a restricted computational budget.

Whether to employ a regression or an interpolation technique depends on the prior knowledge about the landscape. A landscape soiled by noise could be better approximated and optimized by a smoothed surface stemming from a regression model. An interpolation technique applied to such a landscape could provoke over-fitting as the noise is learnt by the approximation model. Conversely, a noise-free landscape should be approximated by an interpolation technique as the prediction is guaranteed to be exact at the known points.

The likelihood approach provides generally a more computationally efficient training than the Bayesian approach but at the expense of a risk of over-fitting and a less informative predictive variance.

1.4 Coupling Surrogates with Evolutionary Algorithms

1.4.1 Surrogate as an evaluator

Direct Fitness Replacement
The most straightforward way of coupling the surrogate to the EA is at the evaluation step of the EA. This coupling is called Direct Fitness Replacement (DFR) in several literature reviews related to Surrogate-assisted Evolutionary Algorithms (SAEAs) [Día+16; SR10; Jin11; Jin05]. The main principle is to rely on the surrogate to evaluate the children assuming that the surrogate predictions are comparable to the simulator outcomes in terms of accuracy. For these methods, the population either contains exclusively predicted candidate solutions or both predicted and simulated solutions.

No Evolution Control
The naive way to DFR is to totally replace the expensive simulator by the surrogate during the search [Inn+15; SC04; LL05; Gon+06; Goe+07]. First, the surrogate is built on the initial database of simulated candidates. Then, the EA is executed with the surrogate only and the best candidate is re-evaluated by the simulator at the end of the search. The naive approach to DFR is indicated when the simulation is extremely expensive [GFL08; Gla+09]. More generally, the naive approach to DFR is successful when the surrogate is a good representation of the expensive objective function. Indeed, a poor surrogate may mislead the search and produce poor quality solutions. It is usually difficult to ensure good prediction accuracy and updating the surrogate is generally considered to overcome this difficulty [Día+16].

Evolution Control
Using the surrogate in conjunction with the expensive simulator at the EA evaluation step allows one to refine the surrogate during the search. In order to select the candidates that are to be simulated rather than only predicted, a mechanism called Evolution Control is introduced [JOS00; JOS01]. The diagram representing a P-SAEA with DFR and EC is given in Figure 1.2.

An EC has two goals:

1. To determine the number of individuals that are simulated before updating the surrogate;

2. To determine which are the most interesting individuals to simulate (promising individuals).

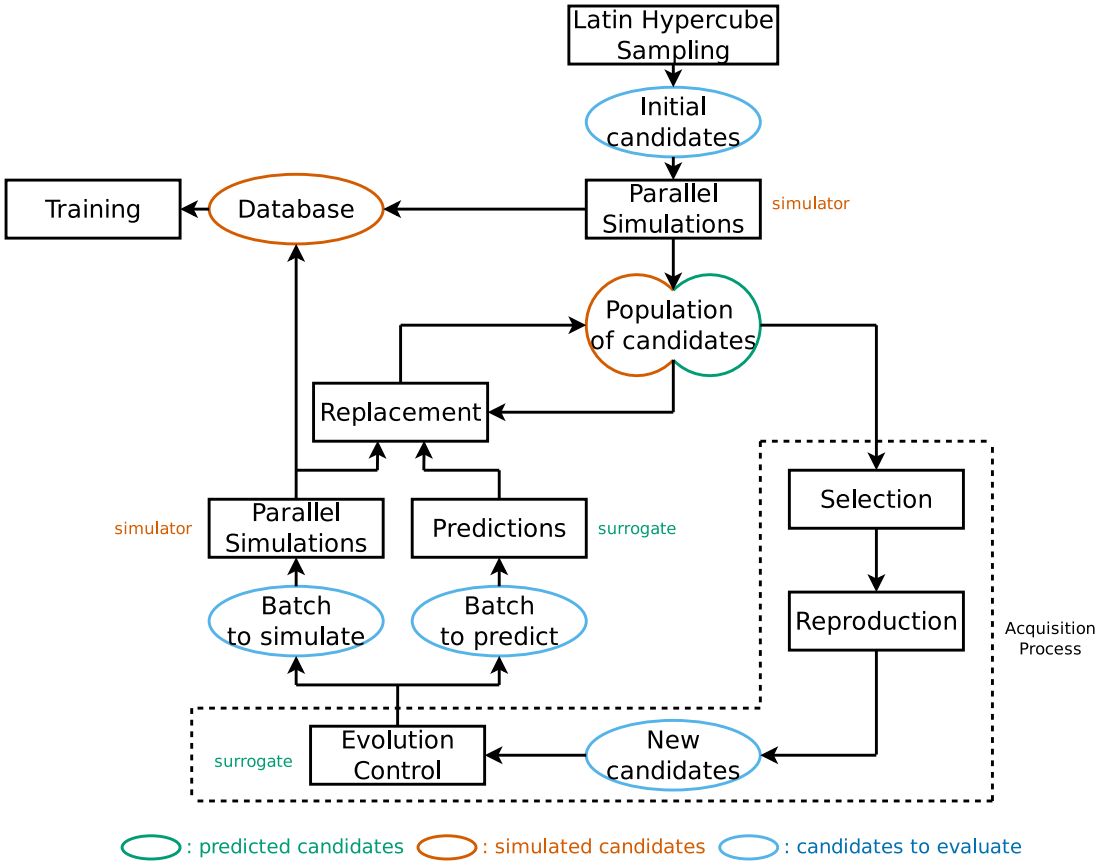

Figure 1.2: P-SAEA with Direct Fitness Replacement and Evolution Control.

Regarding the first goal and according to the established taxonomy [Día+16; Chu+19], ECs for which the number of individuals to be simulated is predetermined are called *fixed ECs* [DN07; JOS00; NKS98; Emm+02; ZWC14; NKS98]. The ECs that decide how many individuals to simulate at execution time are called *adaptive ECs* [GV05; Ros+13; JOS01; Mla+15; JS04; MM11]. Generally, when the surrogate is not accurate enough, the number of individuals to simulate is increased in order to produce more new training samples. Conversely, when the surrogate is accurate enough, the number of individuals to simulate is decreased in order to save computational budget. If the EC decides at the level of generations, it is qualified as generation-level [DN07; JOS01; MM11] while it is said to be individual-level when the number of simulations is set per generation [JOS00; NKS98; Ros+13; JS04].

Regarding the second goal, the definition of the promisingness relies on a comparison operator $>_p$ that ranks the candidates. Generally, a promising candidate will be simulated, and conversely, an unpromising candidate will be only predicted. It can be considered that a high promisingness corresponds to a low predicted objective value (POV) (assuming minimization) [JOS00; NKS98; Ros+13]. It can also be decided that a high promisingness relates to a high predictive uncertainty. In the former definition, exploitation is favored since regions of the search space exhibiting low POVs are simulated more intensively and the surrogate can thus gain precision in these areas [ZWC14; GV05; JOS01]. In the latter definition, exploration is boosted since unexplored regions are simulated more thoroughly and the surrogate can thus gain global accuracy. Only relying on exploitation may lead to premature convergence while only enhancing exploration may miss any local optima under the limited computational budget. ECs based on the surrogate are called *informed ECs*. Not all ECs are informed as it can be stated that a high promisingness coincides to a high distance to the database composed of the already simulated candidates (exploration-oriented promisingness) [NKS98].

Acquisition Process
As shown in Figure 1.2, the EC is a component of a broader process called the Acquisition Process (AP). The AP is in charge of proposing new candidates for simulation. In P-SAEAs, the AP relies on the selection and reproduction operators.

1.4.2 Surrogate as a filter

Indirect Fitness Replacement
Another way of integrating the surrogate into the EA is to use it as a filter to discard candidates that are considered as unpromising. The surrogate acts at the initialization, selection, reproduction or replacement step of the EA while the evaluation relies entirely on the simulator. This integration is called Indirect Fitness Replacement (IFR) in several literature reviews related to SAEAs [Día+16; SR10; Jin11]. The main characteristic is that the population carries only simulated individuals.

Informed operators
When incorporated at the reproduction step of the EA, the coupling is called *informed operators* as the surrogate assists the reproduction operators to discard unpromising candidates [SR10; Jin11; RH00; Syb+08; EGN06; LSS10]. The diagram representing a P-SAEA with IFR by informed operators is given in Figure 1.3. In this context, the EC plays the same role as in DFR.

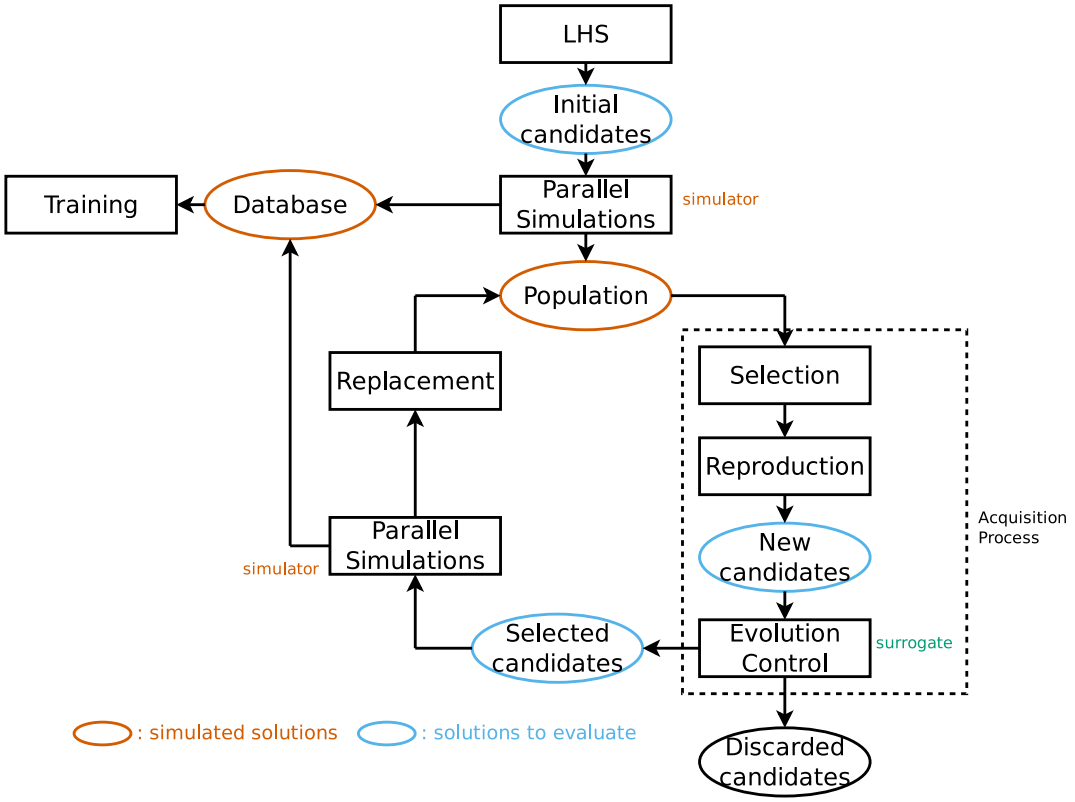

Figure 1.3: P-SAEA with Indirect Fitness Replacement by informed operators.

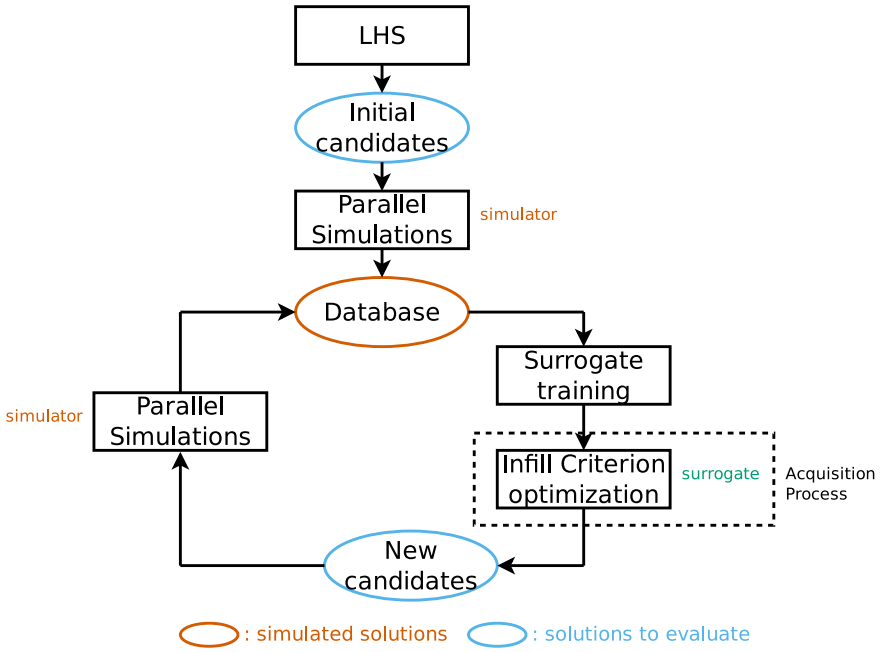

Figure 1.4: Parallel Surrogate-Driven Algorithm.

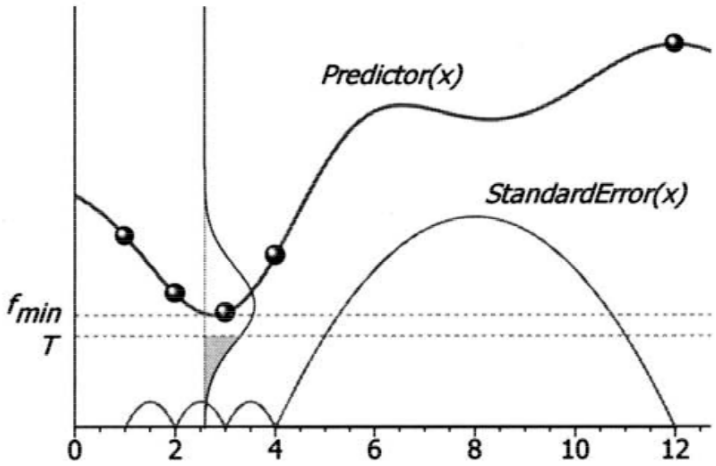

Figure 1.5: Probability of improving over the target value T (shaded area) given the surrogate predictions. Figure extracted from [Jon01].

1.4.3 Surrogate as a driver

Surrogate-Driven Algorithms
Many surrogate-based methods, referred to as Surrogate-Driven Algorithms (SDAs), do not consider to evolve a population from the beginning to the end of the search. Instead, emphasis is on carefully acquiring new promising candidates to simulate. The surrogate is at the heart of SDAs and constitute their rudder. The diagram representing a generic P-SDA is given in Figure 1.4. As shown in Figure 1.4, the AP is driven by the surrogate *via* an Infill Criterion that defines the promisingness of candidates. The IC is a real-valued scalar function or a real-valued vector function that is optimized with the aim of acquiring the most promising candidates. In the same way as the EC defines the promisingness in P-SAEAs, the IC defines it in P-SDAs. Here, a database of simulated solutions is maintained and serves as a training dataset to update the surrogate. The cycle of sampling, simulations and surrogate update is repeated until the computational budget is totally spent.

Efficient Global Optimization
Efficient Global Optimization (EGO) proposed by Jones [JSW98; Jon01] is the most popular SDA. This state-of-the-art framework relies on Kriging as surrogate and the widespread Probability of Improvement (PI) [Kus63] or Expected Improvement (EI) as IC. The PI combines the POV $\hat{f}(\boldsymbol{x})$ and the predictive standard deviation $\hat{s}(\boldsymbol{x})$ at a candidate $\boldsymbol{x}$ by:

$$PI(\boldsymbol{x}) = \begin{cases} \Phi_{\mathcal{N}}\left(\frac{T-\hat{f}(\boldsymbol{x})}{\hat{s}(\boldsymbol{x})}\right) & \text{for } \hat{s}(\boldsymbol{x}) > 0 \\ 0 & \text{for } \hat{s}(\boldsymbol{x}) = 0 \end{cases} \tag{1.52}$$

where $\Phi_{\mathcal{N}}$ is the cumulative distribution function for the normal law $\mathcal{N}(0,1)$ and T is the targeted objective. PI represents the probability of improving on T, that is the area enclosed by the GP predictive distribution below the value of T [FSK08c] as shown in Figure 1.5.

EI extends PI by integrating the amount of improvement as follows:

$$EI(\boldsymbol{x}) = \begin{cases} (y_{min} - \hat{f}(\boldsymbol{x}))\Phi_{\mathcal{N}}\left(\frac{y_{min}-\hat{f}(\boldsymbol{x})}{\hat{s}(\boldsymbol{x})}\right) + \hat{s}(\boldsymbol{x})\phi_{\mathcal{N}}\left(\frac{y_{min}-\hat{f}(\boldsymbol{x})}{\hat{s}(\boldsymbol{x})}\right) & \text{for } \hat{s}(\boldsymbol{x}) > 0 \\ 0 & \text{for } \hat{s}(\boldsymbol{x}) = 0 \end{cases} \tag{1.53}$$

where y_{min} is the best simulated objective value currently known and $\phi_{\mathcal{N}}$ is the probability density function of $\mathcal{N}(0,1)$. EI represents the expectation of the area enclosed by the Gaussian distribution below the best simulated objective value y_{min} found hitherto [FSK08c] (area below f_{min} in Figure 1.5). Over the search space, EI is high in regions of improvement thus enhancing exploitation, and high in regions of high uncertainty thus promoting exploration. EI is also high in regions that represent both a moderate degree of improvement and uncertainty. Both EI and PI prevent re-sampling by having zero value for already simulated candidates (i.e. candidates for which $\hat{s}(\boldsymbol{x}) = 0$). In [Pal+22], EGO with EI is notably employed to determine the parameters' values of a Finite Element model in order to best fit to the observations obtained by physical experiments.

q-points EGO
The EGO algorithm does not allow for parallel simulations as PI and EI are only capable of proposing one new candidate per cycle. In [GRC10], the authors introduce the extension of EI to q-points denoted *q-EI*. The analytical formulation of *q-EI* is only possible for small values of q and a heuristic called *Kriging Believer* is introduced to overcome this limitation. In *Kriging Believer*, q cycles are performed without new simulation. The diversity of the candidates is ensured by updating the Kriging model based on the POV of the last new candidates. The Kriging model is fully re-trained after the q simulations are done.

1.4.4 Analysis of the couplings

In SDAs, the surrogate crucially impacts the search guiding as no guardrail exist. In SAEAs, a deceptive surrogate would disorientate the search but the EA is in control and may recover an adequate path. In a P-SAEA with DFR, the population potentially embeds predicted individuals while in a P-SAEA with IFR, only simulated solutions constitute the population. Consequently, the risk of misleading the search due to the surrogate inaccuracy may be less pronounced for IFR than for DFR methods. In SDAs, the database is always composed of simulated solutions but, for some actual APs such as q-EGO, the surrogate is updated on a training set incorporating predicted candidates.

Selecting q candidates for parallel simulations implies the intervention of the reproduction operators, the EC and one surrogate training in P-SAEAs. In q-EGO, q maximizations of EI and $q+1$ surrogate trainings are involved. The computational complexity of the AP in q-EGO seems to be higher than the one of P-SAEAs, and hence, extending the question of budget allocation over the tasks of simulation, surrogate training and AP.

The straightforward parallel model in P-SAEAs and P-SDAs is to simulate multiple candidates simultaneously at the initialization and evaluation steps as depicted in Figure 1.2, Figure 1.3 and Figure 1.4. The number of new candidates issued per cycle is the deciding factor to control the idling of computing cores. For instance, if the EC is fixed, the number of simulations between two surrogate updates can be set as a multiple of the number of computing cores [ZWC14]. If the EC is adaptive, issues regarding computing load balancing may arise.

APs should be designed to efficiently use the computing cores but also to propose new promising and well diversified candidates. In the next section, APs proposed in the literature are reviewed.

1.5 Related works

1.5.1 Computational budget

In almost all the scientific papers surveyed during this thesis, the computational budget is set as a limited number of simulations [DN07; JOS00; LSS10; EGN06; Syb+08; MM11; ZWC14; JS04; JOS01; GV05; LHS12; Bis+14; Li+10; MP11; WJD17; RS07]. This definition is only valid if the computational cost of surrogate training and prediction is negligible compared to the computational cost of simulation. For moderately expensive problems, this assumption might not hold. In this thesis, the computational budget for the search is defined by a capped duration on a limited number of computing cores so as to take the surrogate training into account. Moderately expensive problems are characterized by a budget greater than 1000 simulations or a simulation lasting less than 5 minutes as it is the case in [DN07; JOS00; JOS01; Mla+15; Emm+02; JS04; MM11; EGN06; LSS10; GV05; Pol+00]. To the best of our knowledge, only one article [Ric+16] from the literature presents a budget defined as a capped number of computing cores during a limited duration. In [Ric+16] 180 minutes on four cores are granted to the search. In this thesis, a budget of 30 minutes on 18 computing cores is considered.

From the reviewed articles, it is tedious to derive knowledge about the suitability of the surrogate-optimizer couplings with regard to the expensiveness of the problem. In [GFL08; Gla+09], a SAEA without EC is invoked to optimize helicopter rotor blades where one simulation lasts six hours. In [JSW98], an automotive-related problem involving simulations of 20 hours is tackled by EGO. The SAEA with informed operators studied in [LSS10] is applied to artificial problems with a budget capped to 100,000 real evaluations. Another turbine blade optimization is carried out in [JOS01] where the SAEA with DFR is employed for a budget limited to 2000 simulations. The ANN-assisted EA from [JOS00] is tested on benchmark functions for a budget ranging from 150 to 1890 real evaluations. The ANN-based SDA from [Pol+00] stops after 3000 simulations to design a sailing yacht fin keel. Multiple ICs are integrated within EGO in [LHS12] and comparisons are based on the minimization of the drag of an airfoil with computational budget ranging from 80 to 200 simulations.

1.5.2 Surrogate model selection

The surrogate models employed for moderately expensive problems are either ANNs [DN07; JOS00; JOS01; JS04; MM11; GV05; Pol+00] or local models [Mla+15; Emm+02; EGN06], while global models, Kriging, RBF (global or local) are preferred when tackling more expensive problems [Chu+17; Jin11; GFL08; Gla+09; ZWC14; JSW98; LHS12; Bis+14; WJD17; GFL08; Gla+09; Li+10; WJD17; RS07]. In particular, local Kriging surrogates are used in [Emm+02] where the simulation time is one minute. A local surrogate is a model trained on a sub-set of the database, in the neighborhood of a given solution, in opposition to a global surrogate that is trained on the entire database. By reducing the number of training samples, the training time is thus alleviated. In [JS04], a problem characterized by 2000 simulations is tackled by a method relying on an ensemble of ANNs. According to the comparisons performed in [Wan+16], relying on an ensemble of surrogates to produce multiple candidates per cycle in P-SDAs may not produce better results than a single-surrogate because it is difficult to predict which surrogates are valuable. In [Bis+14], a packing profile optimization problem capped by 63 simulations is raised by a P-SDA relying on a global RBF model.

It is more laborious to elicit insights about the surrogate model selection regarding to the search landscape characteristics. Firstly, because it is hard to identify the landscape features underlying the real-world applications. Secondly, because the published investigations dedicated to surrogate comparison comprise a restricted range of surrogate model types and/or few benchmark functions. In [Bre+18], Kriging, Random Forest, Support Vector Regression and ensembles are confronted on a real-world application of an electrocardiography simulator tuning. The Kriging and the ensemble of surrogates, plugged in a SDA, provide the best performances. The conclusions drawn in [Wan+16] indicate that no one surrogate is better than the others in all cases. Support Vector Regression, Radial Basis Neural Network, Kriging and ensembles are tested within EGO that is applied on benchmarks and real-world problems.

1.5.3 Definition of promisingness

Metrics for exploration and exploitation
In [DN07], the promisingness is random as the switch between the simulator and the surrogate is fixed arbitrarily at the generation-level in a SAEA with DFR. In [NKS98], the EC suggests to simulate either the average candidate (non-informed), the candidate with highest number of neighbors in the search space (non-informed, enhancing exploitation) or the candidate with the best POV (informed, favoring exploitation). The POV is also considered in [JOS00] where solutions exhibiting a lower POV are more promising for the minimization problem, thus boosting exploitation. Most promising candidates are those with higher predictive standard deviation delivered by the surrogate in the P-SAEA with DFR highlighted in [ZWC14]. The surrogate predictive error is also used to trigger the switch between simulator and surrogate in SAEAs with DFR in [MM11; GV05]. But computing such error causes the consumption of the budget. In [Bis+14; RS05], a criterion based on the distance between candidates is used to ensure diversification.

A large part of the reviewed articles considers to balance between exploitation and exploration to define the promise of candidate solutions [Mla+15; LHS12; EGN06; RV20; Emm+02; Syb+08; Jon01; Reh+18; JSW98; RS05; MS14; RS07; ONK03; Bis+14; Li+10; MP11; WJD17]. The metrics usually used to promote exploration are the predictive standard deviation [Li+10; Reh+18; LHS12; Bis+14; EGN06; LHS12; RV20; Jon01; JSW98; Mla+15; Emm+02; ZWC14; Syb+08] and the distance from the database of already simulated solutions [RS05; Bis+14; MS14; RS07]. Ensembles of surrogates either built locally or globally are also used to favor exploration [WJD17].

Balancing exploration and exploitation
In EGO, the combination of the POV and the predictive standard deviation is realized by scalarization as demonstrated by EI in Equation (1.53) and PI in Equation (1.52). In [Reh+20], the IC consisting in minimizing the POV is compared to EI on multiple benchmark functions with different numbers of decision variables. According to the reported results, the POV performs better than EI for problems of dimension 5+. Nevertheless, it is stated that EI is to be preferred to deal with multi-modal problems for low computational budget. Another popular IC is the Lower Confidence Bound (LCB) [EGN06; LHS12; RV20]:

$$LCB(\boldsymbol{x}) = \hat{f}(\boldsymbol{x}) + \lambda_{lcb}.\hat{s}(\boldsymbol{x}) \tag{1.54}$$

where $\lambda_{lcb} \in [0, 3]$ defines the trade-off between exploration and exploitation. When $\lambda_{lcb} = 0$, the IC is only defined by the POV thus only favoring exploitation. When λ_{lcb} increases, the predictive standard deviation plays a more important role thus promoting exploration [FSK08c]. LCB shows good performances compared to EI and PI when integrated into a SAEA with informed operators and applied to multi-modal and

flat problems with six decision variables [EGN06]. In [LHS12], multiple ICs are compared within EGO to minimize the drag of an airfoil represented by 8 to 20 decision variables and for budgets ranging from 80 to 200 simulations. While EI, PI, LCB and POV prove to perform consistently well, defining the promisingness solely by maximization of the predictive uncertainty appears to be unreliable.

The combination of exploration and exploitation can also be implemented *via* Pareto dominance. In [Bis+14], the bi-objective IC consists in simultaneously minimizing the POV and maximizing the predictive standard deviation. Since EI and PI assume that the surrogate provides normally distributed predictions, the bi-objective IC provides the possibility to use different surrogates than GPs. Unfortunately, only Kriging is considered in the paper. It is reported that the bi-objective IC outperforms EI and LCB on benchmark problems with 5 or 10 decision variables.

Very few approaches consider to vary the definition of the promisingness during the search. In [RS05], the proposed IC consists in minimizing the POV while keeping the candidate at some distance from the database of already simulated solutions. At the beginning of the search, the constraint is severe so as to promote exploration and it is gradually relaxed to allow for more exploitation. Experiments realized on benchmark functions with 2 to 6 decision variables show the method to be competitive with EGO. An analogous SDA is presented in [MP11] where the IC consists in minimizing the POV over specific regions of the search space. At the beginning of the search no specific regions are defined so the search is global (exploration-oriented). Then, densely sampled regions are identified and ignored, still favoring exploration. Finally, when no more improvement is observed, only the densely-sampled regions are considered so that exploitation is favored. The approach is validated on benchmark problems of 2 to 4 decision variables.

1.5.4 Acquisition processes for parallel simulations

Idling of computing cores

A main challenge in designing the AP of a P-SBOA is to efficiently use the available computing cores. In P-SAEAs with *fixed ECs*, such as in [DN07; JOS00; Emm+02; Syb+08; EGN06; LSS10; ONK03; MS14], the number of candidates to simulate per cycle should be a multiple of the number of available computing cores to avoid idling [ZWC14]. For *adaptive ECs*, such as in [GV05; Ros+13; JOS01; Mla+15; JS04], the number of candidates to simulate can vary from one cycle to another and appeal for asynchronous parallel simulations. In the P-SAEA with informed operators proposed in [Syb+08], each computing core generates and simulates a new candidate. The steady-state replacement strategy considers one candidate at once [Tal09]. The advantage of this approach is to avoid idling of the computing cores even in case of irregular number of simulations or heterogeneous simulation duration. In the resembling method from [Bis+14], five simulations are run in parallel.

Diversification

Another important obstacle when designing the AP of a P-SBOA is to select a set of new promising candidates diversified enough. Indeed, simulating too similar candidates wastes the computational budget.

Conducting multiple optimizations is a first way to guarantee diversity. In q-EGO with *Kriging Believer*, q optimizations of EI and q updates of the Kriging models are performed to propose q new candidates. The value of q is limited to 10 in [GRC10]. In [Ric+16], q-EGO is revisited by substituting EI by LCB and applied to calibrate a classification model using four computing cores. In [SLK04], an EA with clustering and a GD with multiple restarts optimize EI to produce up to eight new candidates.

Multiple surrogates can also ensure distinctiveness of new candidates. In [ONK03], multiple local surrogates assist a memetic algorithm at the reproduction step. A memetic algorithm is an EA where the reproduction operators are substituted by local searches [Tal09; Zho+07; MC10]. A maximum of eight local RBF models are built and as many local searches are launched, resulting in as many new candidates. In the P-SAEA outlined in [AKG09], the population is divided into sub-populations, within which new candidates are selected by means of a local RBF surrogate. In [Vil+13], multiple local searches assisted by local Kriging models are carried out on multiple sub-regions of the search space. A dynamic partitioning mechanism is employed, and hence, a maximum of six candidates can be offered for parallel simulation. In [VHW13], ten optimizations of EI are realized based on ten distinct global surrogates. In [Hor+15], multiple scalarizations of the objectives is considered in a multi-objective P-SBOA. Up to four surrogates are created (one per scalarization) thus producing four candidates for parallel simulations. In [WJD17], multi-surrogates (local and global) and multiple definitions of promisingness are used to propose three new candidates.

Relying on multiple definitions of the promisingness is another path to diversification. In [BSK05], four LCB criteria with different values for λ_{lcb} are optimized to produce four new candidates per cycle. In [Li+10], EI is divided into the local EI (first part of the sum in Equation (1.53)) that favors exploitation and the global EI (second part of the sum in Equation (1.53)) oriented towards exploration. Three candidates are selected *via* the global EI and one from the local EI. In [AS16], five rules based on random sampling or maximization of EI or distances in the search and objective spaces are utilized. Firstly, a set of new candidates is generated by optimizing the surrogate POV or through local search around the least crowded candidates. Secondly, the rules are applied to sample five candidates.

The trade-off between converging to and escaping from local minima seems to be properly addressed in the aforementioned articles, but the number of computing cores seems to reach a ceiling of 10 units. More recent studies tackle this problem by adopting multiple local searches as in [Wan+20a] where 128 computing cores are efficiently used thanks to a gradient Monte-Carlo-based estimator of *q-EI*. Nonetheless, the approach is only applied to benchmark problems from 2 to 6 decision variables. In this thesis, we deal with problems of 16 decision variables and we employ up to 144 computing cores. Moreover, all the studies listed in this sub-section are only concerned with very expensive problems while we also address moderately expensive problems.

1.6 Problem instances

1.6.1 Covid-19 contact reduction

Simulating Infectious Disease Transmission and Control

The AuTuMN simulator is developed by a research group from the Department of Public Health and Preventive Medicine at Monash University in Melbourne, Australia. A collaboration with the Australian research group has been initiated in the context of this thesis. The code, publicly available at `https://github.com/monash-emu/AuTuMN/`, allows to implement deterministic models of infectious disease epidemiology. Even if the simulator is black-box, some insights are given about its internal functioning for informational purposes.

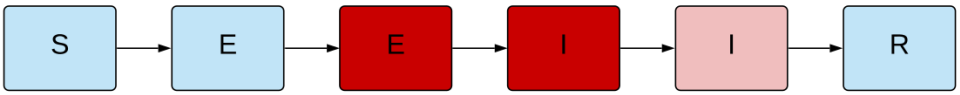

Figure 1.6: SEEIIR compartmental model for simulating Covid-19 transmission. Figure extracted from [Cal+20].

Compartments
In AuTuMN, the individuals of the population under consideration are distributed onto pre-defined compartments according to their disease state. The nature and number of compartments as well as the flows between them are determined according to the disease characteristics. For example, to simulate common cold transmission, one could choose the SIS model made of two compartments (susceptible and infectious) and two possible flows (from infectious to susceptible and from susceptible to infectious) [Het89]. Indeed, when considering the common cold, no long-term immunity exists after recovery.

SEEIIR model
To simulate Covid-19, the proposed SEEIIR model depicted in Figure 1.6 is made of six compartments: susceptible (S), non-infectious exposed (E_1), infectious exposed (E_2), early actively infectious (I_1), late actively infectious (I_2) and recovered/removed (R). The two exposed compartments (E_1 and E_2) correspond to the incubation period while the two infectious compartments (I_1 and I_2) match the active phase of the disease. All compartments are further stratified according to age with 5-years band strata from 0-4 years old to 70-74 years old and 75+ years old. The infectious compartments E_2, I_1 and I_2 are additionally further stratified according to clinical status: asymptomatic, symptomatic ambulatory not detected, detected, hospitalised and intensive care unit. Some quantities are age-dependent such as the infection fatality rate (proportion of death among the late actively infectious phase), hospitalisation risk and susceptibility to infection. Other quantities depend on the clinical status such as the rate of transmission that is reduced for the asymptomatic, detected, hospitalised and intensive care unit strata. The differential equations governing the flows of individuals from one compartment to another can be found in [Cal+20].

Some parameters of the model, such as the incubation period, the infectious period and the risk of infection per contact, are considered for calibration by an adaptive Metropolis algorithm [HST98]. The calibration targets such as the daily notification rate, intensive care unit occupancy and cumulative infection-related deaths are communicated by the local authorities of the studied country.

The outputs of the model are quantities presenting an interest for the epidemic follow-up such as incidence, hospital occupancy, intensive care unit occupancy, mortality, sero-positivity. All these quantities are computed based on the number of individuals in the compartments and the inter-compartments flows.

Covid-19 contact reduction problem

At the beginning of the Covid-19 crisis, when no vaccines were available, governments of the affected countries adopted different strategies to contain the spread of the virus. While some countries imposed lockdown and physical distancing, others, bet on reaching herd immunity by natural transmission. This approach has not proven to be effective during the first two years of the epidemic [CH21], however, at the time, studying the possible

consequences of this strategy was of importance. Very recently, the new Omicron variant of the coronavirus coupled with vaccines preventing health complications revive the debate about herd immunity [Med22].

Problem formulation
The problem consists in optimizing the contact reduction strategy to minimize the number of Covid-19-related deaths in Spain while reaching herd immunity. The Spanish population is divided into 16 age-categories and the decision variables represent the contact mitigation factors to apply to each category. For a decision vector $\boldsymbol{x} \in [0,1]^{16}$, $f_1(\boldsymbol{x})$ represents the number of deaths after the considered period and $f_2(\boldsymbol{x}) \in \{0,1\}$ is a boolean variable indicating whether herd immunity has been reached. Both quantities f_1 and f_2 are obtained *via* simulation. The optimization problem consists in finding $\boldsymbol{x}^*$ such that:

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}\in[0,1]^{16} \text{ s.t. } f_2(\boldsymbol{x})=1}{\arg\min} f_1(\boldsymbol{x}) \tag{1.55}$$

Handling constrained problems in EAs can be realized by different means [Mic+96]:

- Repair of infeasible candidates;
- Reproduction operators to ensure feasibility;
- Rejection of infeasible candidates;
- Adding feasibility as an objective;
- Penalization of infeasible candidates.

According to the guidelines given in [Mic+96], repairing is a problem-dependent method that consists in converting an infeasible candidate into a feasible one. Another problem-dependent approach is to design specific reproduction operators to guarantee the feasibility of the new candidates. For our problem, it is not known how to generate feasible candidates so designing repairing operators or specific reproduction operators is impossible. Rejecting infeasible candidates may work well only if the feasible part of the search space is convex and represents a large proportion of the whole search space. In our case, this information is not available, besides, rejecting infeasible individuals would prevent to keep knowledge about the infeasible region location. Diversely, a measure accounting for the amount of infeasibility can be considered as an additional objective to the problem. However, adding the feasibility as an objective would make the problem more complex as a boolean objective would be added. For the Covid-19 contact reduction problem, we opt for the penalization of the infeasible candidates. The penalty value is set to the approximate Spanish population size (46,000,000) as it is an upper bound for f_1. A higher value would more likely prevent the search to visit the boundary region between the feasible and infeasible search spaces.

Therefore, the problem is re-formulated as an unconstrained optimization problem by applying a penalty to the objective f_1 when herd immunity is not reached. The re-formulated problem consists thus in finding $\boldsymbol{x}^*$ such that:

$$\boldsymbol{x}^* = \underset{\boldsymbol{x}\in[0,1]^{16}}{\arg\min}\, \tilde{f}(\boldsymbol{x}) \tag{1.56}$$

where:

$$\tilde{f}(\boldsymbol{x}) = \begin{cases} f_1(\boldsymbol{x}) \text{ if } f_2(\boldsymbol{x}) = 1 \\ f_1(\boldsymbol{x}) + 46,000,000 \text{ if } f_2(\boldsymbol{x}) = 0 \end{cases} \tag{1.57}$$

Simulation
The contact reduction strategy impact is simulated in three phases. First, the past dynamic of the epidemic is analysed by calibrating uncertain parameters according to past information. Second, the contact reduction strategy is applied. Finally, restrictions are lifted and epidemic resurgence is tested.

The degrees of contact between age-categories are integrated into the model through the contact matrix C provided by [PCJ17]. $C_{i,j}$ is the average number of contacts per day that an individual of age-group j makes with individuals of age-group i. The decision variables representing the mitigation factors are applied to matrix C such that $C_{i,j}$ is replaced by $x_i.x_j.C_{i,j}$. A decision variable x_i set to zero impedes any contact to individuals from age-category i.

Prior on the landscape
According to the research group developing AuTuMN, the search landscape provided by the simulator is expected to be quite smooth as no discontinuity nor brutal variations are foreseen. Some regions with similar objective values may exist as, for instance, the contact of young children should not affect significantly the simulator outcomes. Multiple local optima with different objective values are also expected.

1.6.2 Analytical benchmark functions

Additionally to the Covid-19 contact reduction problem, three scalable artificial benchmark functions are proposed to test the P-SBOAs on landscapes presenting features known to be hard to optimize. Since the real-world application consists of 16 decision variables, it is decided to keep this number for the artificial problems.

The Schwefel function, notably used in [DTC17; Ber+19; BSK05], writes:

$$\begin{aligned} F_{sch}(x_1,\dots,x_{16}) =& 418.9828872724338 * 6 - \sum_{i=1}^{16} x_i \sin\left(\sqrt{|x_i|}\right) \\ & \text{for } x_i \in [-500, 500] \\ F_{sch}(x_1^*,\dots,x_{16}^*) =& F(420.9687,\dots,420.9687) = 0 \end{aligned} \tag{1.58}$$

It is classified as multi-modal with weak global structure in the Black-Box Optimization Benchmarking 2009 [Han+10]. Multi-modality stands for a high number of local optima implying multiple basins of attraction where the search may get stuck. The weak global structure refers to a landscape with no exploitable underlying structure. Roughly speaking, smoothing the landscape would not provide meaningful information about the localisation of the optimum. The graph of the 2-D variant of the Schwefel function displayed in Figure 1.7 illustrates this fact as smoothing would result in a flat surface.

The Rastrigin function, used in [MS14; Hor+15; DTC17; WJD17; ONK03; HGW19; Ber+19; Wan+16; Tia+19; Reh+18; BSK05], is defined by:

$$\begin{aligned} F_{ras}(x_1,\dots,x_{16}) =& 160 + \sum_{i=1}^{16} x_i^2 - 10\cos(2\pi x_i) \\ & \text{for } x_i \in [-5.12, 5.12] \\ F_{ras}(x_1^*,\dots,x_{16}^*) =& F(0,\dots,0) = 0 \end{aligned} \tag{1.59}$$

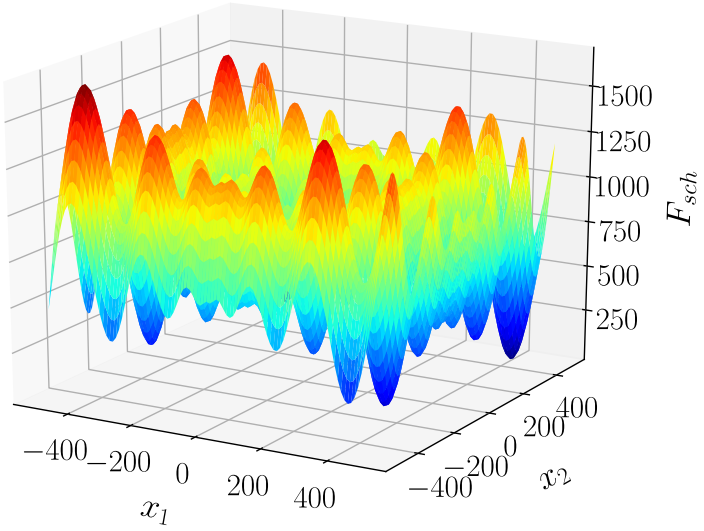

Figure 1.7: Search landscape provided by the 2-D Schwefel function.

It is qualified as multi-modal with strong global structure in [Han+10]. Removing the asperities of the Rastrigin landscape would result in an adequate landscape where the global optimum corresponds to the global optimum of the original function. For the sake of example, the graph of the 2-D Rastrigin function is displayed in Figure 1.8. By removing the asperities from the surface shown in Figure 1.8, the resulting landscape would present a unique basin of attraction easier to optimize.

The Rosenbrock function, used in [JOS01; JOS00; DTC17; WJD17; Li+10; HGW19; Ber+19; Wan+16; Tia+19; SLK04; Reh+18; BSK05], is given by:

$$\begin{aligned} F_{ros}(x_1, \ldots, x_{16}) =& \sum_{i=1}^{15} 100(x_i^2 - x_{i+1})^2 + (x_i - 1)^2 \\ & \text{for } x_i \in [-5, 10] \\ F_{ros}(x_1^*, \ldots, x_{16}^*) =& F(1, \ldots, 1) = 0 \end{aligned} \tag{1.60}$$

It is categorized as ill-conditioning in [Han+10]. Ill-conditioning refers to a condition number of the Hessian that indicates strong variation of the objective value in different directions in the search space. Ill-conditioned problems are hard to solve by GD. The Rosenbrock 2-D landscape, displayed in Figure 1.9, exhibits a curved valley characterized by low objective values and walls characterized by higher objective values. The global optimum lies in the curve of the valley. The valley-like feature makes GD oscillating on both sides while the curvature prevents the gradient to point out in the direction of the global optimum.

According to the expectations communicated by the AuTuMN research group, the landscape associated to the Covid-19 contact reduction problem may exhibit similar characteristics than those provided by the Schwefel and Rastrigin problems with respect to multi-modality. It can also present regions with similar objective values such as the Rosenbrock function.

In this thesis, no further landscape analysis is performed on the Covid-19 problem prior to the optimization. The reason is the constraint makes difficult to identify feasible candidates at the initial sampling step. Nevertheless, landscape analysis is performed *a posteriori* when a good amount of feasible simulated solutions are available.

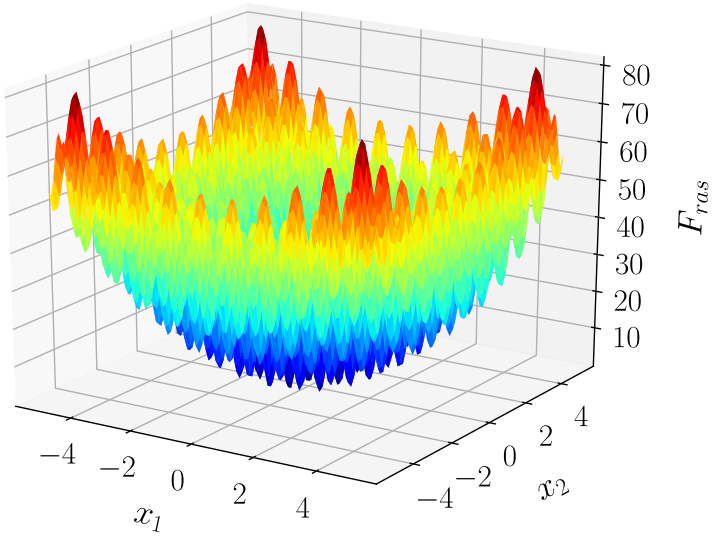

Figure 1.8: Search landscape provided by the 2-D Rastrigin function.

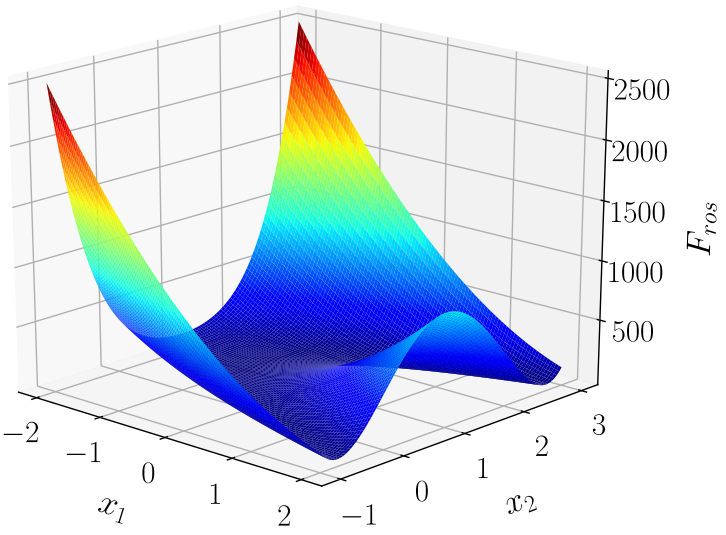

Figure 1.9: Search landscape provided by the 2-D Rosenbrock function.

Chapter 2

Parallel Surrogate-assisted Evolutionary computations

Contents

Related publications:

- Briffoteaux Guillaume, Ragonnet Romain, Mezmaz Mohand, Melab Nouredine, Tuyttens Daniel. "Evolution Control for parallel ANN-assisted simulation-based optimization application to Tuberculosis Transmission Control" in *Future Generation Computer Systems*, 2020, Vol. 113, Pages 454-467, ISSN 0167-739X. https://doi.org/10.1016/j.future.2020.07.005
- Briffoteaux Guillaume, Ragonnet Romain, Mezmaz Mohand, Melab Nouredine, Tuyttens Daniel. "Evolution Control Ensemble Models for Surrogate-Assisted Evolutionary Algorithms", in *HPCS 2020 - International Conference on High Performance Computing and Simulation*, 22-27 March 2021, Online conference. https://hal.inria.fr/hal-03332521

- Briffoteaux Guillaume, Ragonnet Romain, Mezmaz Mohand, Melab Nouredine, Tuyttens Daniel. "Towards Dynamic Selection of Evolution Controls in Parallel Bayesian Neural Network-assisted Genetic Algorithm" in *OLA'2020 - International Conference on Optimization and Learning*, Feb 2020, Cádiz, Spain. https://hal.inria.fr/hal-02867819

- Briffoteaux Guillaume, Melab Nouredine, Mezmaz Mohand, Tuyttens Daniel. "An adaptive evolution control based on confident regions for surrogate-assisted optimization" in *HPCS 2018 - International Conference on High Performance Computing and Simulation*, Jul 2018, Orléans, France. https://hal.inria.fr/hal-01922708

2.1 Introduction

The objective of this chapter is to study the design and applicability of P-SAEAs to moderately expensive problems. A coupling between a Bayesian Neural Network and an EA is highlighted and multiple definitions of the promisingness of candidate solutions are proposed to set up the balance between exploration and exploitation.

Section 2.2 presents the Bayesian Neural Network approximated *via* Monte-Carlo Dropout (BNN_MCD) that is subsequently attached to the EA. The coupling is realized by frequently substituting the simulator by the surrogate and by employing the surrogate as a filter (SaaEF). The frequency of substitution along with the filtering criterion are settled by the EC, the key component of P-SAEAs. Ensembles of ECs, enabling the dynamic definition of the promisingness during the search, are suggested and detailed in Section 2.3.

Section 2.4 focuses on surrogates from the training time and predictive capacity points of view. Conceptually bearing both the advantages of ANNs and GPs, BNN_MCD is calibrated and empirically compared to well-known models.

One primary goal of this chapter is to identify the adequate surrogate and EC to be harnessed in P-SAEA with SaaEF. In Section 2.5, intensive numerical experiments are conducted on optimization problems characterized by varying landscape features recognized tedious to manage. An expensive real-world constrained problem of Covid-19 contact reduction is notably dealt with. Finally, Section 2.6 draws the conclusions of this chapter.

2.2 BNN_MCD as an evaluator and a filter

2.2.1 Bayesian Neural Network approximated *via* Monte-Carlo Dropout (BNN_MCD)

Base principle

The BNN_MCD is an ANN trained with Dropout and that employs the Monte-Carlo Dropout technique to provide both the POV and the predictive standard deviation. As depicted in Figure 2.1, Monte-Carlo Dropout randomly samples n_{sub} sub-networks from the ANN and calculates the average $\hat{f}$ and the standard deviation $\hat{s}$ of the POV according to the following formulas:

$$\hat{f}(\boldsymbol{x}) = \frac{1}{n_{sub}} \sum_{i=1}^{n_{sub}} \hat{f}_i(\boldsymbol{x}) \tag{2.1}$$

$$\hat{s}(\boldsymbol{x}) = \sqrt{\frac{1}{n_{sub}} \sum_{i=1}^{n_{sub}} (\hat{f}_i(\boldsymbol{x}) - \hat{f}(\boldsymbol{x}))^2} \tag{2.2}$$

where $\hat{f}_i$ is the POV of the candidate $\boldsymbol{x}$ from the i-th sub-network.

Training

The training of BNN_MCD is detailed in Algorithm 2. Let:

$$\hat{f}(\boldsymbol{x}) = \boldsymbol{w^{(2)}}.\boldsymbol{h}(W^{(1)}.\boldsymbol{x}) \tag{2.3}$$

be the prediction for input $\boldsymbol{x}$ from a one-hidden-layer ANN with activation function $\boldsymbol{h}()$. $W^{(1)}$ is the matrix of weights associated to the connections from the input layer to the hidden layer and $\boldsymbol{w^{(2)}}$ is the vector of weights associated to the connections from the hidden layer to the 1-D output layer. Training with Dropout consists in minimizing the

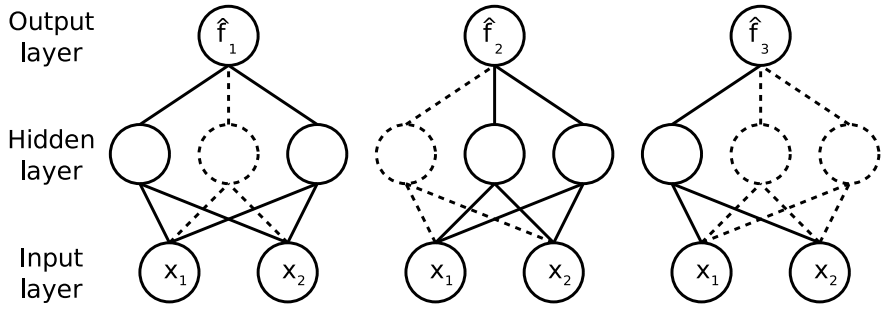

Figure 2.1: Illustration of Monte-Carlo sampling of $n_{sub} = 3$ sub-networks from a one-hidden-layer ANN. In this example, the decision vector $\boldsymbol{x}$ is 2-dimensional and $\hat{f}_i(\boldsymbol{x})$ is the POV according to sub-network i.

error between the predictions and the targets by applying GD [Rud16] on the following network:

$$\hat{f}(\boldsymbol{x}) = \boldsymbol{\epsilon^{(1)}} \odot \boldsymbol{w^{(2)}}.\boldsymbol{h}(diag(\boldsymbol{\epsilon^{(1)}}).W^{(1)}.\boldsymbol{x}) \tag{2.4}$$

Vector $\boldsymbol{\epsilon^{(1)}}$ is randomly sampled at each training iteration from a Bernouilli law with probability p_{drop}. Let's assume $\epsilon_i^{(1)} = 0$, then multiplying $diag(\boldsymbol{\epsilon^{(1)}})$ by $W^{(1)}$ zeros out the i-th row of $W^{(1)}$. In other terms, the weights connected to the i-th neuron of the hidden layer are not updated during this training iteration. The same principle applies to $\boldsymbol{\epsilon^{(1)}} \odot \boldsymbol{w^{(2)}}$ where $\odot$ symbolizes the component-wise product. Dropout shows up at lines 7 to 10 in Algorithm 2. It allows to drive each neuron to perform well independently from the remaining ones. In other terms, it prevents some neurons to strive correcting the mistake provoked by another ones (co-adaptation) [Sri+14]. At prediction, n_{sub} sub-networks are sampled from the entire ANN in the same way than Dropout [Gal16]. The algorithm for predicting through Monte-Carlo Dropout with a one-hidden-layer ANN is given in Algorithm 13 in Section B of the appendix.

The capped computational budget granted to the optimization renders impractical the composition of a validation set. A 2-fold cross-validation is adopted to overcome this difficulty [FSK08a]. At line 2 of Algorithm 2, the training set is divided in two parts $(X_0, \boldsymbol{y_0})$ and $(X_1, \boldsymbol{y_1})$. During the first iteration of the for-loop in line 3, $(X_0, \boldsymbol{y_0})$ is used as a training set and $(X_1, \boldsymbol{y_1})$ as a validation set. During the second iteration, the role of both sets are reversed.

The early stopping mechanism stops the training when the error between the predictions and the targets, computed over the current validation set, has not decreased by at least δ_{ES} during n_{ES} iterations. Early stopping helps preventing over-fitting by restoring the best parameters found so far [GBC16] and appears in lines 6, 13 and 17 in Algorithm 2.

Advantages and drawbacks

Monte-Carlo Dropout can be affiliated to [WJD17] where the predicted uncertainty arise from the diversity of surrogates in use. In [WJD17], updating the ensemble of surrogates requires as many trainings as surrogates whereas only one training is necessary for BNN_MCD.

In [Syb+08], it is stated that ANNs are appropriate for substituting rugged objective functions with limited training sets. By its top-notch similarity with ensembles of ANNs, BNN_MCD is expected to offer good predictive capacity.

Algorithm 2 Dropout training with one-hidden-layer ANN

Input

$(X_{train}, \boldsymbol{y_{train}})$: training set
p_{drop}: probability of dropping out neurons
GD: Gradient Descent algorithm
(δ_{ES}, n_{ES}): early stopping parameters
$\boldsymbol{h}()$: activation function
λ_{decay}: weight decay parameter
l: weight initialization parameter
max_epoch: maximum number of epochs

$(W^{(1)}, \boldsymbol{w^{(2)}}) \leftarrow$ initialize_weights$(\mathcal{N}(0, l))$
$(X_0, X_1, \boldsymbol{y_0}, \boldsymbol{y_1}) \leftarrow$ partition_2_fold$(X_{train}, \boldsymbol{y_{train}})$
for $i \in \{0, 1\}$ **do** ▷ 2 fold cross-validation
    $early\_stopping \leftarrow false$
    $iter \leftarrow 0$
    while (not $early\_stopping$) & ($iter < max\_epoch$) **do**
        $\boldsymbol{\epsilon^{(1)}} \leftarrow$ Bernouilli_sampling(p_{drop})
        $\boldsymbol{preds} \leftarrow \boldsymbol{\epsilon^{(1)}} \odot \boldsymbol{w^{(2)}}.\boldsymbol{h}(diag(\boldsymbol{\epsilon^{(1)}}).W^{(1)}.X_{mod(i,2)})$
        $loss \leftarrow$ compute_MSE_weight_decay$(\boldsymbol{preds}, \boldsymbol{y_{mod(i,2)}}, \lambda_{decay})$
        $(diag(\boldsymbol{\epsilon^{(1)}}).W^{(1)}, \boldsymbol{\epsilon^{(1)}} \odot \boldsymbol{w^{(2)}}) \leftarrow$ backpropagation_update$(loss, GD)$
        $\boldsymbol{preds} \leftarrow \boldsymbol{w^{(2)}} \odot \boldsymbol{h}(W^{(1)}.X_{mod(i+1,2)})$
        $loss \leftarrow$ compute_MSE_weight_decay$(\boldsymbol{preds}, \boldsymbol{y_{mod(i+1,2)}}, \lambda_{decay})$
        $early\_stopping \leftarrow$ update_early_stopping$(loss, \delta_{ES}, n_{ES})$
        $iter \leftarrow iter+1$
    end while
end for
$(W^{(1)}, \boldsymbol{w^{(2)}}) \leftarrow$ restore_best_weights()
return $(W^{(1)}, \boldsymbol{w^{(2)}})$

It is proven in [Gal16] that BNN_MCD is a computationally efficient approximation to Bayesian learning [Gra11]. The gain with respect to traditional ANNs is the accessibility of the predictive uncertainty. Nonetheless, the predictive standard deviation is derived approximately thus its informativeness should be investigated. To the best of our knowledge, BNN_MCD has only been used as a surrogate model in a pre-print article not peer-reviewed yet [Lin+18].

2.2.2 Surrogate as an evaluator and a filter (SaaEF)

The general P-SAEA framework where the surrogate is used both as an evaluator and a filter is depicted in Figure 2.2. The algorithm begins by sampling the search space through a LHS to produce a set of initial candidates that are simulated in parallel to create a database and an initial population (1). At this point, the population is exclusively composed of simulated candidates but it will also be allowed to embed predicted candidates in the following steps. Conversely, the database is intended to only accumulate the simulated candidates along with their simulated objective values. The database serves as a basis to constitute the training set used to build the surrogate (2). Next to the initial surrogate building, cycles are repeated until the computational budget allocated to the search is over. One cycle consists of the following subsequent operations: AP, simulations, surrogate update, predictions and replacement. The AP (delimited by a doted line in Figure 2.2) begins with the selection of parent candidates from the population *via* a tournament selection of size $n_t = 2$. The population of parents undergoes reproduction (3), giving birth to a set of new candidates (4). The SBX cross-over and the polynomial mutation are considered as reproduction operators. The mutation operator is systematically applied to one decision variable at least, so as to restrict the possibility of re-visiting already proposed candidates [RK17; Ron98; YC09; LYC21]. Thereafter, the EC divides the set of new candidates into three batches (5). Candidates assessed as promising by the EC constitute the batch to simulate, less promising candidates compose the batch to predict and non-promising solutions are discarded. Afterwards, the batch to simulate is processed in parallel (6) and the surrogate is updated (7). The batch to predict is then estimated with the updated surrogate (8). Elitist replacement including the simulated and predicted candidates as well as the current population is carried on to constitute the new population, probably composed of both simulated and predicted candidates (9). The general framework of EA, the LHS, the tournament selection as well as the SBX cross-over and polynomial mutation operators are described in Sub-section 1.2.2.

The SaaEF framework is a DFR method because the surrogate is used as an evaluator, and consequently, the population may embed both simulated and predicted candidates such as in [JOS00]. Our approach is also an IFR method as the surrogate informs the reproduction operators to filter out unpromising candidates as in [SR10; EGN06; Emm+02]. To the best of our knowledge, employing the surrogate both in a direct and indirect replacement mode has not been inspected.

The detailed algorithm of the proposed SaaEF is presented in Algorithm 14 in Section B of the appendix. The input arguments of the algorithm must respect some constraints to ensure the well functioning of the method. First, since the reproduction operators generate two children from two parents, the number of new candidates generated per cycle n_{chld} must be even:

$$\text{It must exist } k \in \mathbb{N}^* \text{ such that } n_{chld} = 2.k \tag{2.5}$$

Besides, it is assumed that the number of simulations q, predictions n_{pred} and discardings n_{disc} per cycle sum to the number of new candidates:

$$n_{chld} = q + n_{pred} + n_{disc} \tag{2.6}$$

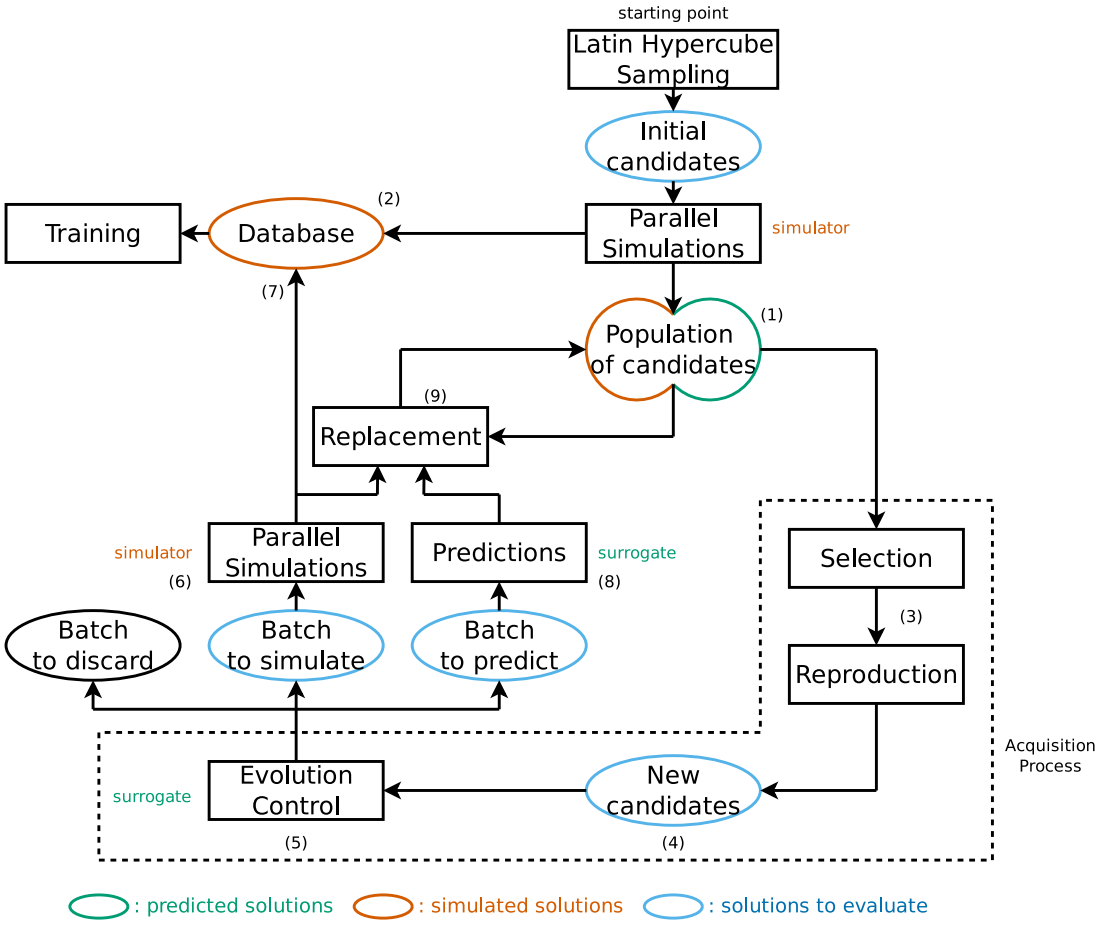

Figure 2.2: SaaEF framework. The ellipses represent the sets of candidate solutions along with their objective values or not, while the rectangles stand for the operators.

Finally, the number of simulations should be a multiple of the number of computing cores to avoid the idling of cores:

$$\exists k' \in \mathbb{N}^* \text{ such that } q = k'.n_{cores} \tag{2.7}$$

where n_{cores} is the number of available computing cores for the search. The parallel approach consisting in running multiple simulations in parallel is inspired from [JOS01].

2.3 Ensembles of Evolution Controls

Hereafter, existing ECs are highlighted and replicated for comparison with our proposals of ensembles that feature a variation of the definition of promisingness during the search.

2.3.1 Random and scalar ECs

The simplest way of defining the EC is to randomly ordering the candidates. This strategy, referred to as *rand* in this thesis, has been studied in [JOS00]. In [JOS00], it is also envisioned to rely on the POVs of the candidates. The associated EC, denoted *pov*, favors exploitation and is given by:

$$\boldsymbol{x^{(1)}} >_p \boldsymbol{x^{(2)}} \text{ if } \hat{f}(\boldsymbol{x^{(1)}}) < \hat{f}(\boldsymbol{x^{(2)}}) \tag{2.8}$$

Relying only on the surrogate predictive standard deviation provides an EC favoring exploration, denoted *stdev*:

$$\boldsymbol{x^{(1)}} >_p \boldsymbol{x^{(2)}} \text{ if } \hat{s}(\boldsymbol{x^{(1)}}) > \hat{s}(\boldsymbol{x^{(2)}}) \tag{2.9}$$

Another mean of boosting diversification is to consider the distance from the database of already simulated solutions:

$$\boldsymbol{x^{(1)}} >_p \boldsymbol{x^{(2)}} \text{ if } d_2(\boldsymbol{x^{(1)}}, database) > d_2(\boldsymbol{x^{(2)}}, database) \tag{2.10}$$

where $d_2(.,.)$ is the Euclidean distance. This EC is denoted *dist* in the following. According to the literature review presented in Sub-section 1.5.3, it is rare to employ *stdev* and *dist* alone in SBOAs.

To adjust the trade-off between exploration and exploitation, bi-criterion ECs relying on the POV $\hat{f}$ and the predictive standard deviation $\hat{s}$ are introduced. EI [JSW98], PI [Kus63] and LCB [EGN06], defined by Equations (1.53), (1.52) and (1.54) respectively, are the three most known scalarized bi-criterion ECs. The EC based on EI is referred to as *ei* and is given by:

$$\boldsymbol{x^{(1)}} >_p \boldsymbol{x^{(2)}} \text{ if } EI(\boldsymbol{x^{(1)}}) > EI(\boldsymbol{x^{(2)}}) \tag{2.11}$$

The EC relying on PI is denoted *pi* and is specified by:

$$\boldsymbol{x^{(1)}} >_p \boldsymbol{x^{(2)}} \text{ if } PI(\boldsymbol{x^{(1)}}) > PI(\boldsymbol{x^{(2)}}) \tag{2.12}$$

The EC employing LCB is called *lcb* and is set by:

$$\boldsymbol{x^{(1)}} >_p \boldsymbol{x^{(2)}} \text{ if } LCB(\boldsymbol{x^{(1)}}) < LCB(\boldsymbol{x^{(2)}}) \tag{2.13}$$

In the experiments, $\lambda_{lcb} = 1$ as it is the case in [Emm+02].

2.3.2 Pareto-based bi-criterion ECs

Alternatively to scalarization, Pareto dominance between intensification and diversification can be envisioned. Familiarity with Pareto dominance is assumed in this sub-section. If it is not the case, a background in multi-objective optimization is provided in Appendix C.

It is first proposed to simultaneously minimize the POV and maximize the predictive uncertainty and use the crowding distance $cd()$ to distinguish between solutions with the same Non-Dominated Rank (NDR):

$$\begin{aligned}
&\boldsymbol{x^{(1)}} >_p \boldsymbol{x^{(2)}} \text{ if} \\
&NDR(\boldsymbol{x^{(1)}}) < NDR(\boldsymbol{x^{(2)}}) \text{ or} \\
&[NDR(\boldsymbol{x^{(1)}}) = NDR(\boldsymbol{x^{(2)}}) \text{ and } cd(\boldsymbol{x^{(1)}}) > cd(\boldsymbol{x^{(2)}})] \text{ or} \\
&[NDR(\boldsymbol{x^{(1)}}) = NDR(\boldsymbol{x^{(2)}}) \text{ and } cd(\boldsymbol{x^{(1)}}) = cd(\boldsymbol{x^{(2)}}) = \infty \text{ and } \hat{f}(\boldsymbol{x^{(1)}}) < \hat{f}(\boldsymbol{x^{(2)}})] \\
&\text{when minimizing the POV and maximizing the predictive uncertainty.}
\end{aligned} \tag{2.14}$$

The fourth line in (2.14) allows to distinguish between the two extreme solutions of a Non-Dominated Front (NDF). When the predictive uncertainty is the predictive standard deviation, the EC is designated as *par-fs-cd* and when the predictive uncertainty is the distance to the database, the EC is denoted *par-fd-cd*. It is also suggested to use the hyper-volume contribution $hvc()$ to distinguish between solutions with the same NDR:

$$\begin{aligned}
&\boldsymbol{x^{(1)}} >_p \boldsymbol{x^{(2)}} \text{ if} \\
&NDR(\boldsymbol{x^{(1)}}) < NDR(\boldsymbol{x^{(2)}}) \text{ or} \\
&[NDR(\boldsymbol{x^{(1)}}) = NDR(\boldsymbol{x^{(2)}}) \text{ and } hvc(\boldsymbol{x^{(1)}}) > hvc(\boldsymbol{x^{(2)}})] \\
&\text{when minimizing the POV and maximizing the predictive uncertainty.}
\end{aligned} \tag{2.15}$$

Two exceptions apply to the rule (2.15): the candidate with the lowest POV is the overall most promising one and the candidate with the highest predictive uncertainty is the second most promising one. When the predictive uncertainty is the predictive standard deviation, the EC is denominated *par-fs-hvc* and when the predictive uncertainty is the distance to the database, the EC is called *par-fd-hvc*.

A similar method is suggested in [Tia+19]. Assuming n_{NDF} fronts when minimizing both the POV and the predicting standard deviation, the most promising candidates are those of NDR 1, then those of NDR n_{NDF} and at last the promisingness decreases from NDR 2 to NDR $(n_{NDF} - 1)$. Unlike our approaches, no further criterion is envisioned to distinguish between candidates with the same NDR. The IC from [Tia+19], called *partian-fs*, is illustrated in Figure B.1 in Section B of the appendix. Its conversion to EC is straightforward, and hence, it is reproduced for comparison purposes and formalized as follows:

$$\begin{aligned}
&\boldsymbol{x^{(1)}} >_p \boldsymbol{x^{(2)}} \text{ if} \\
&[NDR(\boldsymbol{x^{(1)}}) = 1 \text{ and } NDR(\boldsymbol{x^{(2)}}) \neq 1] \text{ or} \\
&[NDR(\boldsymbol{x^{(1)}}) = n_{NDF} \text{ and } NDR(\boldsymbol{x^{(2)}}) \notin \{1, n_{NDF}\}] \text{ or} \\
&[NDR(\boldsymbol{x^{(1)}}) < NDR(\boldsymbol{x^{(2)}}) \text{ and } NDR(\boldsymbol{x^{(1)}}), NDR(\boldsymbol{x^{(2)}}) \notin \{1, n_{NDF}\}] \\
&\text{when minimizing both the POV and the predictive standard deviation.}
\end{aligned} \tag{2.16}$$

We further extend the EC by replacing the predictive standard deviation by the distance to the database and referred to the new EC as *par-tian-fd*. According to [Tia+19], the advantage of using Pareto-based bi-criterion ECs compared to EI is twofold: they are better than EI for search spaces of high dimension and the surrogate must not necessarily provide a Gaussian prediction.

The Pareto-based bi-criterion and other ECs that are empirically investigated in this thesis are listed in Table 2.1.

2.3.3 Dynamic ensembles

The exploration/exploitation compromises determined by the ECs introduced previously are static all along the search. We now introduce dynamic ensembles of ECs that modify the balance according to the computational budget consumption. Such dynamism appears in SDAs, for instance, in [RS05]. Nonetheless, to the best of our knowledge no analogous strategies have been contemplated in SAEAs.

In the exclusive ensembles, only one EC is active at a time. Let $E_1, \ldots, E_{n_{EC}}$ be n_{EC} ECs either single-criterion or Pareto-based bi-criterion ECs and $0 < p_1 < \cdots < p_{n_{EC}-1} < 1$ some user-defined parameters. The dynamic exclusive ensemble strategy consists in switching from E_i to E_{i+1} as soon as a proportion p_i of the budget has been spent. At the beginning of the search, the first EC (E_1) is activated. In [RS05], a distance-based constraint is attached to the IC to gradually reducing exploration. Our proposals are an attempt to translate the IC from [RS05] into an EC with the benefit of enabling any ECs to build the ensemble. The ten dynamic exclusive ensembles participating in the further experiments are listed in Table 2.2.

In the dynamic inclusive ensemble strategy, two single-criterion ECs, denoted E_1 and E_2, work concurrently. The search is arbitrarily divided into 5 equal periods, each coinciding to a proportion of 20% of the total computational budget. To each period corresponds different participation rates for E_1 and E_2 in the building of the batches. The participation rate of E_1 decreases as the search proceeds while the one of E_2 increases. During the first period, only E_1 is activated. During the second period, E_1 participates at 75% and E_2 at 25%. During the third period, both ECs cooperate at 50% and, during the fourth period, E_1 participates at 25% and E_2 at 75%. Finally, during the last period only E_2 is activated.

The implementation for the second period is presented in Algorithm 3. In Algorithm 3, each EC generates a list of indexes corresponding to the ordering of the new candidates according to decreasing promise. These two lists are called L_1 and L_2. The first $0.75 * q$ indexes from L_1 are appended to L and removed from L_1 and from L_2. Then, the first $0.25 * q$ indexes from L_2 are appended to L and removed from L_2 and from L_1 if necessary. The same operations are repeated for the next n_{pred} candidates and, finally, for the last n_{disc} candidates so that the three batches are formed. Our inclusive ensembles can be related to [AS16] where multiple rules act simultaneously to choose new candidates. However, the participation rate of the multiple criteria is steady in [AS16]. The four dynamic inclusive ensembles implemented and tested in this study are compiled in Table 2.2.

Algorithm 3 Second period in dynamic inclusive ensemble strategy.

Input
E_1: first evolution control
E_2: second evolution control
$\mathcal{P}_c$: population of new candidates
q: number of simulations per cycle
n_{pred}: number of predictions per cycle
n_{disc}: number of discardings per cycle

$L_1 \leftarrow$ sort_indexes(E_1, $\mathcal{P}_c$)
$L_2 \leftarrow$ sort_indexes(E_2, $\mathcal{P}_c$)
$L \leftarrow \emptyset$
for $n \in \{q, n_{pred}\}$ **do**
    L.append($L_1(1 : int(0.75 * n))$
    $L_1 \leftarrow L_1 \backslash L$
    $L_2 \leftarrow L_2 \backslash L$
    L.append($L_2(1 : int(0.25 * n))$
    $L_1 \leftarrow L_1 \backslash L$
    $L_2 \leftarrow L_2 \backslash L$
end for
L.append(L_1)
$\mathcal{B}_{sim} \leftarrow \mathcal{P}_c(L(1 : q))$ ▷ Batch to simulate
$\mathcal{B}_{pred} \leftarrow \mathcal{P}_c(L(q + 1 : q + n_{pred}))$ ▷ Batch to predict
$\mathcal{B}_{disc} \leftarrow \mathcal{P}_c(L(q + n_{pred} + 1 : q + n_{pred} + n_{disc}))$
▷ Batch to discard
return $\mathcal{B}_{sim}, \mathcal{B}_{pred}, \mathcal{B}_{disc}$

2.3.4 Adaptive ensembles

The adaptive ensembles of ECs remove the need for the user to fix the parameters accounting for the switch or the participation rate in dynamic exclusive and inclusive ensembles respectively. Moreover, the transition from exploration to exploitation (or from exploitation to exploration) is not compelled to be one-way and is allowed to divert.

In our adaptive ensemble of ECs, only one EC is active at a time and the switch is regulated by a stagnation detection mechanism and a reward mechanism.

The stagnation detection mechanism is similar to early stopping used in ANN training in Sub-section 2.2.1. After 8 batches without improvement greater than 10^{-8}, the EC is swapped. The "patience" parameter and the minimum improvement value have been arbitrarily chosen.

The reward mechanism, exposed in Algorithm 4, follows the stagnation detection by electing which EC is to be activated. At each batch of candidates, all the ECs are rewarded or penalized according to the error between the simulated objective values and the POVs (line 1 of Algorithm 4). By comparison to a threshold (line 2), a small (resp. large) error produces a reward (resp. penalty) for the active EC and the ECs that would have decided to simulate, and deliver a penalty (resp. reward) for the remaining ECs. The threshold is the average squared error on the previous batch (line 11) and the amount of reward is the hyperbolic tangent of the square difference between the error and the threshold (line 2). In Algorithm 4, a component of the reward vector $\boldsymbol{r}$ can be negative and thus designates a penalty. After an EC switch, the accumulated rewards of each EC is set back to zero. The reward mechanism follows the future research direction pointed out in [Chu+19] according to which surrogate accuracy should be utilized to design ECs.

Algorithm 4 Reward mechanism in adaptive ensemble of two ECs.

Input
- E_a: currently activated evolution control
- E_d: currently deactivated evolution control
- r_a: current accumulated reward of E_a
- r_d: current accumulated reward of E_d
- q: number of simulations per cycle
- L_a: simulated candidates of the batch, as decided by E_a
- L_d: candidates of the batch that would have been simulated by E_d
- $\boldsymbol{f}$: simulated objective values of the batch
- $\hat{\boldsymbol{f}}$: predicted objective values of the batch
- t: current threshold

1: $\boldsymbol{e} \leftarrow (\boldsymbol{f} - \hat{\boldsymbol{f}})^2$
2: $\boldsymbol{r} \leftarrow \tanh(t - \boldsymbol{e})$ ▷ $r_i > 0 \Leftrightarrow e_i < t$
3: **for** $1 \leqslant i \leqslant q$ **do**
4: $\quad r_a \leftarrow r_a + r_i$
5: $\quad$ **if** $L_d[i]$ is in L_a **then**
6: $\quad\quad r_d \leftarrow r_d + r_i$
7: $\quad$ **else**
8: $\quad\quad r_d \leftarrow r_d - r_i$
9: $\quad$ **end if**
10: **end for**
11: $t \leftarrow \frac{1}{q} \sum_{i=1}^{q} e_i$
12: **return** t, r_a, r_d

The existing adaptive ICs from [Wan+20b] are translated into ECs for comparison purposes. The first EC, referred to as *ada-wang-max*, is defined as follows:

$$\begin{aligned}
&\boldsymbol{x^{(1)}} >_p \boldsymbol{x^{(2)}} \text{ if} \\
&f_{ada\text{-}wang\text{-}max}(\boldsymbol{x^{(1)}}) < f_{ada\text{-}wang\text{-}max}(\boldsymbol{x^{(2)}}) \\
&\text{where:} \\
&f_{ada\text{-}wang\text{-}max}(\boldsymbol{x}) = (1-\alpha)\frac{\hat{f}(\boldsymbol{x})}{\max_{\boldsymbol{c}\in\mathcal{B}} \hat{f}(\boldsymbol{c})} - \alpha\frac{\hat{s}(\boldsymbol{x})}{\max_{\boldsymbol{c}\in\mathcal{B}} \hat{s}(\boldsymbol{c})} \\
&\text{and} \\
&\alpha = -\frac{1}{2}\cos\left(\frac{b_s}{b}\pi\right) + \frac{1}{2}
\end{aligned} \tag{2.17}$$

where $\mathcal{B}$ is a batch of new candidates, b_s is the amount of computational budget already spent and b is the total budget. The second EC, designated as *ada-wang-min*, is obtained by minimizing the term containing the predictive standard deviation as follows:

$$f_{ada\text{-}wang\text{-}min}(\boldsymbol{x}) = (1-\alpha)\frac{\hat{f}(\boldsymbol{x})}{\max_{\boldsymbol{c}\in\mathcal{B}} \hat{f}(\boldsymbol{c})} + \alpha\frac{\hat{s}(\boldsymbol{x})}{\max_{\boldsymbol{c}\in\mathcal{B}} \hat{s}(\boldsymbol{c})} \tag{2.18}$$

At the beginning of the search $\alpha \approx 0$ and both ECs consider as promising the solutions minimizing the POV. As the search proceeds, $\alpha \to 1$ and more importance is given to the predictive standard deviation. For *ada-wang-max*, the predictive standard deviation is maximized thus exploration is favored while it is minimized in *ada-wang-min* thus enhancing exploitation of regions with low predictive uncertainty.

The four adaptive ensembles of ECs implemented for experiments are summed up in Table 2.2.

2.3.5 Voting committees

The voting committee of ECs is inspired by the field of Active Learning [ODM17]. According to a particular EC, if a candidate is among the q most promising of the batch, it receives one vote. The rule is applied with all the ECs composing the committee, consequently:

$$\begin{aligned}
&\boldsymbol{x^{(1)}} >_p \boldsymbol{x^{(2)}} \text{ if} \\
&\text{the number of ECs that recognize } \boldsymbol{x^{(1)}} \text{ as promising is higher than} \\
&\text{the number of ECs that recognize } \boldsymbol{x^{(2)}} \text{ as promising}
\end{aligned} \tag{2.19}$$

The two voting committees of ECs contemplated in the experiments are registered in Table 2.2.

2.4 Comparison of Surrogates

2.4.1 Calibration of BNN_MCD

The hyper-parameters of BNN_MCD are listed in Table 2.3 along with their values either fixed in accordance to published studies or calibrated *via* grid search. Each of the 2500 possible BNN_MCD configurations is tested on each benchmark function using training sets of 256 samples and validation sets of 1024 samples obtained through LHS. The calibration is based on minimization of the validation mean squared error (VMSE) and minimization

Table 2.1: ECs employed in the experiments. $\hat{f}$ refers to the POV, $\hat{s}$ refers to the predictive standard deviation and d_2 to the distance to the database.

Identifier	Type	Description	Reference	Used in
rand	No criterion	Random promise	Section 2.3.1	P-SAEAs, q-EGO
pov	Single-criterion	Best POV: min $\hat{f}$	Eq. (2.8)	P-SAEAs, q-EGO, q-subnets, q-GP-HMC, q-BNN-HMC, SMBO+EA
stdev		Predictive deviation: max $\hat{s}$	Eq. (2.9)	P-SAEAs, q-EGO
dist		Distance to database: max d_2	Eq. (2.10)	P-SAEAs, q-EGO
ei	Scalarized bi-criterion	Expected Improvement	Eq. (1.53)	P-SAEAs, q-EGO, SMBO+EA
pi		Probability of Improvement	Eq. (1.52)	P-SAEAs, q-EGO
lcb		Lower Confidence Bound	Eq. (1.54)	P-SAEAs, q-EGO
par-fs-cd	Pareto-based bi-criterion	(min $\hat{f}$, max $\hat{s}$), crowding distance	Eq. (2.14)	P-SAEAs, q-EGO, q-Pareto
par-fd-cd		(min $\hat{f}$, max d_2), crowding distance	Eq. (2.14)	P-SAEAs, q-EGO, q-Pareto, q-subnets, q-GP-HMC, q-BNN-HMC, HCAP, HSAP
par-fs-hvc		(min $\hat{f}$, max $\hat{s}$), hyper-volume	Eq. (2.15)	P-SAEAs, q-EGO, q-Pareto
par-fd-hvc		(min $\hat{f}$, max d_2), hyper-volume	Eq. (2.15)	P-SAEAs, q-EGO, q-Pareto
par-tian-fs		(min $\hat{f}$, min $\hat{s}$)	Eq. (2.16)	P-SAEAs, q-EGO, q-Pareto
par-tian-fd		(min $\hat{f}$, min d_2)	Eq. (2.16)	P-SAEAs, q-EGO, q-Pareto

Table 2.2: Ensembles of ECs considered in the experiments. $\hat{f}$ refers to the POV, $\hat{s}$ refers to the predictive standard deviation.

Identifier	Type	Description	Reference	Used in
dyn-df-excl	Dynamic exclusive	$(E_1, \ldots, E_{n_{EC}})$ $(p_1, \ldots, p_{n_{EC}-1})$	Section 2.3.3	P-SAEAs, q-EGO, HCAP, HSAP
dyn-df-75-excl		(*dist*, *pov*) (0.5)	Section 2.3.3	P-SAEAs, q-EGO
dyn-fd-excl		(*dist*, *pov*) (0.75)	Section 2.3.3	P-SAEAs, q-EGO
dyn-dpf-excl		(*pov*, *dist*) (0.5)	Section 2.3.3	P-SAEAs, q-EGO
dyn-fpd-excl		(*dist*, *par-fd-cd*, *pov*) (0.25, 0.75)	Section 2.3.3	P-SAEAs, q-EGO
dyn-sf-excl		(*pov*, *par-fd-cd*, *dist*) (0.25, 0.75)	Section 2.3.3	P-SAEAs, q-EGO
dyn-sf-75-excl		(*stdev*, *pov*) (0.5)	Section 2.3.3	P-SAEAs, q-EGO
dyn-fs-excl		(*stdev*, *pov*) (0.75)	Section 2.3.3	P-SAEAs, q-EGO
dyn-spf-excl		(*pov*, *stdev*) (0.5)	Section 2.3.3	P-SAEAs, q-EGO
dyn-fps-excl		(*stdev*, *par-fd-cd*, *pov*) (0.25, 0.75)	Section 2.3.3	P-SAEAs, q-EGO
		(*pov*, *par-fd-cd*, *stdev*) (0.25, 0.75)	Section 2.3.3	P-SAEAs, q-EGO
dyn-df-incl	Dynamic inclusive	(E_1, E_2)	Section 2.3.3	P-SAEAs, HCAP, HSAP
dyn-fd-incl		(*dist*, *pov*)	Section 2.3.3	P-SAEAs
dyn-sf-incl		(*pov*, *dist*)	Section 2.3.3	P-SAEAs
dyn-fs-incl		(*stdev*, *pov*)	Section 2.3.3	P-SAEAs
		(*pov*, *stdev*)	Section 2.3.3	P-SAEAs
ada-df	Adaptive	*dist*, *pov*	Section 2.3.4	P-SAEAs, q-EGO
ada-dpf		*dist*, *par-fd-cd*, *pov*	Section 2.3.4	P-SAEAs, q-EGO
ada-wang-max		min $\hat{f}$, max $\hat{s}$	Eq. (2.17)	P-SAEAs, q-EGO
ada-wang-min		min $\hat{f}$, min $\hat{s}$	Eq. (2.18)	P-SAEAs, q-EGO
com-dpf	Voting committee	*dist*, *par-fd-cd*, *pov*	Eq. (2.19)	P-SAEAs, q-EGO
com-spf		*stdev*, *par-fs-cd*, *pov*	Eq. (2.19)	P-SAEAs, q-EGO, HCAP, HSAP

Table 2.3: BNN_MCD hyper-parameters.

Symbol	Name	Value	Calibration method
n_{hl}	number of fully-connected hidden layers	1	grid search $\{1; 2; 5; 8; 10\}$
m_u	number of units per layer	1024	grid search $\{256; 512; 1024; 2048; 4096\}$
λ_{decay}	weight decay coefficient	10^{-1}	grid search $\{10^{-3}; 10^{-2}; 10^{-1}; 1\}$
l	Normal standard deviation for weights initialization	10^{-2}	grid search $\{10^{-2}; 10^{-1}; 1; 10; 100\}$
p_{drop}	dropout probability	0.1	grid search $\{0.005; 0.05; 0.1; 0.3; 0.5\}$
$h()$	activation function	Relu	[GBC16]
ξ	Adam initial learning rate	0.001	[GBC16]

of the negative average validation log-likelihood (NAVLL) both calculated after a 50-epochs training. For a validation sample $(\boldsymbol{x_{val}}, y_{val})$, the validation log-likelihood is given by:

$$\begin{aligned} &logsumexp\left(\frac{-\tau}{2}(y_{val} - \hat{f}_i(\boldsymbol{x_{val}}))^2\right) - \log(n_{sub}) - \frac{1}{2}\log(2\pi) + \frac{1}{2}\log(\tau) \\ &\text{where } \tau = \frac{1 - p_{drop}}{2.n.l.\lambda_{decay}} \end{aligned} \tag{2.20}$$

where the number of sub-networks is arbitrarily fixed to $n_{sub} = 5$ for the moment. Averaging over the validation set yields the NAVLL. The NAVLL incorporates the model uncertainty and captures how well the model fits the data with larger values indicating better accuracy [Gal16].

The BNN_MCD variants with two or three occurrences in the NDFs of the benchmarks are identified, and among them, the configuration displayed in the middle of the NDF for both Schwefel and Rosenbrock is retained. In the middle of the NDF, VMSE and NVALL are balanced, besides Schwefel and Rosenbrock are favored since the Covid-19-related problem may exhibit similar characteristics. The best NDF according to simultaneous minimization of VMSE and NAVLL and the configurations with 2 or 3 occurrences in the NDFs are provided as supplementary materials in Table B.1, Table B.2, Table B.3 and Table B.4.

2.4.2 Surrogates on the benchmark

The calibrated BNN_MCD is now confronted to a Bayesian linear regression model (ANN_BLR), a GP with Radial Basis Functions kernels (GP_RBF) and Kriging interpolation (iKRG) and regression (rKRG) models. BNN_MCD is implemented with the Keras library [Cho15] integrated within Tensorflow [Aa15], ANN_BLR, GP_RBF and the Kriging models are implemented using pybnn [Sno+15], GPyTorch [Gar+18] and pyKriging [PR15] respectively. All the software libraries used in this thesis are listed in Table F.1. The peculiarity of ANN_BLR lies in its basis functions, being the neurons of the last hidden layer of an ANN [Sno+15]. Training ANN_BLR is linear to the number of training data but cubic to the number of basis functions.

The comparison of surrogates involves the training time and the predictive capacity expressed by the validation correlation coefficient (vR2):

$$\text{vR2} = \frac{cov(\boldsymbol{y}_{val}, \hat{\boldsymbol{f}}(X_{val}))}{\sqrt{var(\boldsymbol{y}_{val})var(\hat{\boldsymbol{f}}(X_{val}))}} \tag{2.21}$$

Table 2.4: **Surrogates comparison**. Training time (TT) and validation correlation coefficient (vR2) averaged over 10 runs for each surrogate and each **benchmark problem**. Ranks according to TT and vR2 are denoted in parentheses.

	72 training samples		256 training samples	
Surrogates	TT	vR2	TT	vR2
		Schwefel		
BNN_MCD	6.56(2)	**-3.6e-02(1)**	**7.09(1)**	**-1.3e-03(1)**
ANN_BLR	**3.99(1)**	-8.3e-01(2)	1.3e+01(2)	-5.0e-01(2)
GP_RBF	4.0e+01(4)	-6.2e+01(3)	8.9e+01(3)	-6.2e+01(3)
rKRG	2.2e+01(3)	-1.0e+09(4)	2.6e+02(4)	-1.6e+09(4)
iKRG	1.2e+02(5)	-1.0e+09(5)	1.7e+03(5)	-1.6e+09(4)
		Rastrigin		
BNN_MCD	6.6(2)	-2.2e-02(2)	**8.3(1)**	-7.4e-04(2)
ANN_BLR	**4.0(1)**	-7.8e-01(3)	1.3e+01(2)	-5.6e-01(3)
GP_RBF	4.1e+01(4)	**-6.7e-04(1)**	1.0e+02(3)	**-1.8e-05(1)**
rKRG	1.5e+01(3)	-2.1e+06(4)	2.3e+02(4)	-2.7e+06(4)
iKRG	8.2e+01(5)	-2.1e+06(4)	1.5e+03(5)	-2.7e+06(4)
		Rosenbrock		
BNN_MCD	7.38(2)	**-7.0e-03(1)**	**7.21(1)**	3.4e-03(2)
ANN_BLR	**4.03(1)**	-6.4e-01(2)	1.4e+01(2)	**1.8e-01(1)**
GP_RBF	4.1e+01(4)	-5.37(3)	9.2e+01(3)	-5.37(3)
rKRG	1.1e+01(3)	-5.0e+13(4)	3.7e+02(4)	-9.4e+13(4)
iKRG	7.9e+01(5)	-5.2e+13(5)	1.8e+03(5)	-1.0e+14(5)

where $(X_{val}, \boldsymbol{y}_{val})$ is the validation set. A value of vR2 close to one points out a correlation of the landscapes, sufficient to locate optima [FSK08a]. The training set is either made of 72 or 256 samples and the validation set is composed of 1024 points. Each training and validation is repeated 10 times for each surrogate on each benchmark and the average metrics of interest are reported in Table 2.4.

About execution time, BNN_MCD is the faster model to train on 256 training samples, followed by ANN_BLR with a difference around one order of magnitude. On 72 points, BNN_MCD is slightly trailed by ANN_BLR. GP_RBF, rKRG and iKRG are much slower to train and the increase in training time is significant from 72 to 256 points. ANN_BLR training time is also affected by augmenting the number of training points but it is not the case for BNN_MCD. When the budget is expressed as a capped number of computing cores during a limited time, or when the problem is moderately expensive, fast training is desirable. BNN_MCD seems to be adequate in this respect. To accomodate GP_RBF, rKRG and iKRG in such scenarios, the size of their training sets may be curbed to control the execution time.

Regarding the predictive capacity, the results demonstrate that BNN_MCD performs best on the Schwefel problem and GP_RBF performs best on the Rastrigin problem. For the Rosenbrock problem, ANN_BLR and BNN_MCD provide the best results.

Table 2.5: Calibration of the parallel EA without surrogate.

Symbol	Name	Value	Calibration method
n_{pop}	population size	72	grid search $\{8; 18; 36; 72; 144\}$
p_c	cross-over probability	0.9	grid search $\{0.3; 0.5; 0.7; 0.9\}$
η_c	cross-over distribution index	10	set from [DN07]
p_m	mutation probability	$\frac{1}{d}$	set from [Tal09]
η_m	mutation distribution index	50	set from [DN07]
n_t	tournament size	2	set from [Deb+02]

2.5 Experiments

2.5.1 Computational budget

The experiments conducted in this section are supported by a parallel machine made of 18 computing cores provided in an Intel Xeon Gold 5220 CPU. The parallel machine is part of the Grid5000, a French infrastructure dedicated to parallel and distributed computing and enabled by several French universities [Cap+05]. The resources management system offered by Grid5000 grants to the user exclusive control of the reserved resources and ensures the uniformity of the computational environment from one experiment to another.

A slot of 30 minutes on the 18 computing cores is allocated per search. The duration of one simulation on one computing core ranges from 10 to 15 seconds for the Covid-19 contact reduction problem. To recreate analogous conditions, the simulation duration is artificially raised to 15 seconds for the benchmark problems. The upper bound for the number of simulations is 3240 thus matching the moderately expensive context.

Because the optimization algorithms are stochastic, each experiment is repeated 10 independent times to ensure statistical validity of the comparisons. Ten initial samplings are carried out independently for each problem through the pyDOE implementation of LHS [Bau+] to initialize the populations and databases of the P-SBOAs.

2.5.2 Calibration of SaaEF

To begin with, the parallel EA without surrogate is tuned as it is the baseline for comparison. The parameters are gathered in Table 2.5 along with their values either fixed in accordance to published studies or via parallel grid search on the benchmark problems.

The best configuration according to the average, median and minimum objective value found at the end of the search is $(n_{pop} = 72; p_c = 0.9)$ for the Schwefel and the Rastrigin problems and $(n_{pop} = 18; p_c = 0.9)$ for the Rosenbrock problem. Setting $n_{pop} = 8$ provides consistently bad results. Indeed, the parallel processing of eight candidates induces the idling of ten computing cores.

The calibration outcomes are provided in Appendix B. The box-plot graphs are displayed in Figure B.2, Figure B.3 and Figure B.4 and the main statistics are gathered in Table B.5, Table B.6 and Table B.7.

Afterwards, the parameters of SaaEF with BNN_MCD are calibrated. Table 2.6 sums up the parameters and their values. The EC employed for the search is *par-fs-cd*.

Increasing n_{chld} grants major opportunity for the reproduction operators to yield auspicious candidates. The number of new samples between consecutive surrogate updates is given by q for which higher values are preferred according to the calibration outcomes. In SaaEF, one surrogate update is carried out per cycle. Increasing the number of surrogate updates per cycle by means of partitioning the population of children into sub-batches has been attempted in the framework of this study but has not demonstrated any benefit.

Table 2.6: Calibration of SaaEF with BNN_MCD.

Symbol	Name	Value	Calibration method
n_{chld}	children per cycle	288	grid search $\{144; 288\}$
q	simulations per cycle	$72 = 0.25 * n_{chld}$	[JOS01; BSK05]
n_{pred}	predictions per cycle	$72 = 0.25 * n_{chld}$	[JOS01; BSK05]
n_{disc}	discardings per cycle	$144 = 0.5 * n_{chld}$	Condition (2.6)
(δ_{ES}, n_{ES})	BNN_MCD early stopping	$(10^{-8}, 32)$	grid search $\{(10^{-4}, 8), (10^{-8}, 32)\}$
	2-fold cross-validation	yes	grid search {yes, no}

The retained tuning is the one showing the best average and median objective value at the end of the search on the Schwefel problem and the fourth on the Rosenbrock problem. On Rastrigin, no configuration outperform the EA without surrogate. The main statistics are exposed in Table B.8, Table B.9 and Table B.10 as supplementary materials.

2.5.3 Experimental protocol

The calibrated P-SAEA with SaaEF is now to be applied on the benchmark and Covid-19 contact reduction problems. Three variants of the tuned BNN_MCD, referred to as BNN_MCD_5, BNN_MCD_20 and BNN_MCD_100, are plugged in the P-SAEA. The variants differ regarding the number of sub-networks $n_{sub} \in \{5, 20, 100\}$ but are all trained on the whole database of already simulated solutions. Indeed, the augmentation of training samples negligibly influences BNN_MCD training time. However, this latter influence is significant for the other surrogates as shown in Sub-section 2.4.2 and the impact is even more critical when the database grows by $q = 72$ solutions per cycle. Consequently, by the insights gained in Sub-section 2.4.2, the ANN_BLR training set is restrained to the last 256 simulated solutions, the last 72 ones for GP_RBF, the last 36 for rKRG_36 and iKRG_36 and the last 18 ones for rKRG_18 and iKRG_18. Clustering techniques are usually adopted to compose local training sets [Emm+02; LC14; EGN06; BSK05]. Here, we expect the last simulated solutions to delimit the currently promising region because they are selected by the EC.

The 33 ECs listed in Table 2.1 and Table 2.2 and the nine surrogates aforementioned are integrated within P-SAEA with SaaEF, producing 297 methods. The parallel EA without surrogate is also applied, thus rising the total number of searches to 11,920. The ECs always rely on the normalized values of the different metrics. For surrogate training, the normalization is uniform in $[0, 1]$ except for ANN_BLR for which it is normal $\mathcal{N}(0, 1)$.

2.5.4 Empirical analysis

Comparison of the surrogates

The best EC, according to the average of the ten objective values returned at the end of the searches, are displayed for each surrogate and problem in Table 2.8. It can be observed that adopting BNN_MCD as surrogate yields the best results on the Schwefel and Covid-19 problems. It is hypothesized that the landscapes of both problems exhibit similar characteristics. Among the three variants of BNN_MCD, $n_{sub} = 5$ demonstrates the best effectivity. Increasing n_{sub} likely puts the informativeness of the predictive standard deviation up, but also extends the prediction time because more sub-networks must be sampled. With BNN_MCD, saving computing time seems more beneficial. Indeed, it can be seen in Table 2.9 that the average number of simulations per search falls away when n_{sub} increases while the average training time is steady.

Table 2.7: **Normalization effect on GP_RBF training** for a training set of size 72. Training time (TT) and validation correlation coefficient (vR2) averaged over 10 runs for each **benchmark problem**.

Normalization	TT	vR2	TT	vR2	TT	vR2
	Schwefel		Rastrigin		Rosenbrock	
None	4.0e+01	-6.2e+01	4.1e+01	-6.7e-04	4.1e+01	-5.37
$[0, 1]$	4.1e-01	-5.6e-02	2.8e-01	-9.3e-02	2.8e-01	2.1e-01

On Rastrigin, GP_RBF is the most performing surrogate according to Table 2.8, as foreseen after the comparison of surrogates of Sub-section 2.4.2. Conversely, the outstanding efficiency of GP_RBF on Rosenbrock and its very low training time exposed in Table 2.9 were unexpected. The discrepancy between expectations and reality is explained by the application of normalization for training during the optimization exercises. The experiments led in Sub-section 2.4.2 are reproduced for GP_RBF trained on a normalized training set of size 72. The outcomes shown in Table 2.7 indicate an alleviation of training time by two orders of magnitude and enhancement of the vR2 on Schwefel and Rosenbrock. While the reduction in training time is due to the early stopping mechanism whose parameters are left unchanged, the improvement of vR2 is quite surprising.

Because the Schwefel function draws a graph with weak global structure, iKRG should outperform rKRG. On the contrary, the adequate underlying structure of the Rastrigin landscape should be better approximated by rKRG than iKRG. Nevertheless, the regression Kriging model always surpasses the interpolation one in Table 2.8. The justification could reside in the higher average training time and the lower average number of simulations exhibited by iKRG in Table 2.9. Restraining the training time is interesting in other cases such as Schwefel and Rosenbrock where rKRG_18 reveals better performances than rKRG_36.

All the previous observations are recovered when scrutinizing the top-5 ECs per surrogate and problem according to the average, median and minimum of the ten final objective values. These classifications are supplied in Table B.11, Table B.12, Table B.13 and Table B.14 in the appendix section.

Comparison of the ECs

The dynamic ensembles favoring exploration at the beginning and exploitation at the end of the search (*dyn-df-excl*, *dyn-df-75-excl*, *dyn-dpf-excl* and *dyn-df-incl*) are the best choice according to Table 2.8. Commended diversification at the early stages enhances surrogate accuracy and specifies the regions susceptible to bear the global optimum. At latter stages, surrogate predictions are thus more reliable and convergence is put forward. The adaptive ensembles of ECs should be able to dictate such orientation, however, a bad tuning may disrupt their expected capabilities. The Pareto-based ECs (*par-fd-hvc*, *par-fs-hvc*, *par-fd-cd*) prove their capacity on the Rosenbrock problem and seem to outperform their counterpart from the literature (*par-tian-fd* and *par-tian-fs*). The Rosenbrock landscape is certainly marked by a vast promising region (the "canyon" in Figure 1.9) that appeal never to relax exploration.

In the nomenclature drawn up to designate the ECs, the letter 'd' stands for the distance and 's' for the standard deviation. Whatever the surrogate type, the distance from the database of already simulated solutions is a more reliable metric of exploration than the predictive standard deviation according to Table 2.8. Contrary to the distance, the predictive standard deviation is built approximately by BNN_MCD and does not account for all the historical simulations for the other surrogates.

All the previous observations are confirmed by analysing the distribution of the final objective values for the best surrogate for each problem. The corresponding box plots are deferred to the appendix section in Figure B.5, Figure B.6, Figure B.7 and Figure B.8. It is worth noticing that the best ECs in terms of the averaged final objective values are also the ECs showing the lowest variance and the best median over the ten runs. By the box plots, no significant discrepancy is noted from harnessing the crowding distance instead of the hyper-volume contribution in Pareto-based ECs. Besides, the scalar ECs and the committees do not behave particularly well except *dist* on the Covid-19 application. The constraint of the real-world application may necessitate strong exploration to locate the feasible region. Indeed, among the 720 initial samples only one solution is feasible.

Comparison of the APs

In a purely DFR fashion, the P-SAEA where the surrogate is only employed as an evaluator (SaaE) is now applied on the problems at hand as well as the P-SAEA where the surrogate is only used as a filter (SaaF) in an wholly IFR style. For both approaches the number of simulations per cycle is set to $q = 72$ as in SaaEF. In SaaE, $n_{pred} = q = 72$ to follow the recommendation from [JOS01; BSK05], which implies $n_{chld} = 144$ according to Equation (2.6). It is reminded that $n_{disc} = 0$ in SaaE and $n_{pred} = 0$ in SaaF. In SaaF, the number of children is fixed to $n_{chld} = 288$ as in SaaEF and so $n_{disc} = 216$ according to Equation (2.6).

The surrogate is limited to the BNN_MCD_5 for the Schwefel and Covid-19 contact reduction problems and to GP_RBF on the Rastrigin and Rosenbrock problems as the previous results indicate their appropriateness in these situations. The best strategies according to the final objective values averaged over the ten runs are displayed in Table 2.10. According to the results, no one AP is to be privileged in all the cases. On the Rastrigin problem, the SaaE is the best choice certainly because GP_RBF is accurate enough and therefore a high degree of confidence can be assigned to the surrogate. On Rosenbrock and the Covid-19-related problem, relying on the surrogate as a filter improves over SaaEF thus marking a weaker predictive performance of the surrogate. Indeed, it is assumed that embedding incorrectly predicted individuals into the population, as in DFR, is more damaging for the evolution trajectory than incorrectly discarding solutions. The erroneously rejected individuals could be recovered in future steps in IFR while the next generations are influenced by the wrong objective values in DFR. The SaaEF strategy is the best AP on the multi-modal Schwefel problem with weak global structure. Moreover, SaaEF brings a certain robustness as it appears in the four top-5. Regarding the ECs, Table 2.10 confirms the previous results as the dynamic ensemble of ECs favoring first exploration and exploitation afterwards are the most represented in the top-5 followed by the Pareto-based ECs, notably *par-fd-cd*.

Table 2.8: **P-SAEAs with SaaEF**. Best strategies, from top to bottom, for each surrogate model according to the best objective value averaged over 10 runs.

Surrogate	Evolution Control	Average best objective value
Schwefel		
BNN_MCD_5	***dyn-df-incl***	**131.95**
BNN_MCD_100	*dyn-dpf-excl*	134.06
BNN_MCD_20	*dyn-dpf-excl*	163.49
GP_RBF	*dyn-spf-excl*	206.19
rKRG_18	*dyn-df-75-excl*	216.06
rKRG_36	*dyn-df-75-excl*	244.83
ANN_BLR	*dyn-df-incl*	267.78
iKRG_18	*dyn-df-75-excl*	448.76
no surrogate	-	607.91
iKRG_36	*dyn-df-75-excl*	663.03
Rastrigin		
GP_RBF	***dyn-dpf-excl***	**19.08**
rKRG_36	*dyn-spf-excl*	22.66
ANN_BLR	*dyn-sf-excl*	22.72
no surrogate	-	23.30
BNN_MCD_5	*dyn-df-excl*	23.40
BNN_MCD_20	*ei*	23.89
rKRG_18	*dyn-dpf-excl*	25.07
BNN_MCD_100	*pi*	25.98
iKRG_18	*dyn-fd-incl*	28.39
iKRG_36	*par-fs-hvc*	37.62
Rosenbrock		
GP_RBF	***par-fd-hvc***	**233.93**
rKRG_18	*par-fd-hvc*	414.44
rKRG_36	*par-fs-hvc*	443.60
BNN_MCD_5	*dist*	713.71
BNN_MCD_20	*dist*	677.00
BNN_MCD_100	*dist*	826.43
iKRG_18	*ei*	1086.06
no surrogate	-	1191.14
ANN_BLR	*dyn-dpf-excl*	2063.52
iKRG_36	*par-fd-cd*	2085.61
Covid-19 contact reduction		
BNN_MCD_5	***dyn-df-75-excl***	**7,455**
BNN_MCD_20	*dist*	7,532
BNN_MCD_100	*ada-dpf*	8,196
rKRG_18	*dyn-df-incl*	9,254
rKRG_36	*dyn-df-incl*	9,560
iKRG_18	*dyn-df-incl*	9,928
iKRG_36	*dyn-df-excl*	14,018
GP_RBF	*ada-df*	16,596
ANN_BLR	*dyn-df-excl*	18,450
no surrogate	-	21,483

Table 2.9: **P-SAEAs with SaaEF**. Average number of simulations per search and overall average training time (in seconds) for each surrogate. Ordering according to the average number of simulations in decreasing order from top to bottom.

Surrogate	Average number of simulations	Average training time
Schwefel		
GP_RBF	2214	0.16
rKRG_18	2084	3.52
BNN_MCD_5	2013	6.05
BNN_MCD_20	1998	5.99
BNN_MCD_100	1905	5.89
rKRG_36	1889	9.64
ANN_BLR	1797	14.63
iKRG_18	1746	16.32
iKRG_36	1281	44.61
Rastrigin		
GP_RBF	2214	0.15
rKRG_18	2075	3.82
BNN_MCD_5	2008	6.27
BNN_MCD_20	1989	6.27
BNN_MCD_100	1898	6.20
rKRG_36	1865	10.59
ANN_BLR	1798	14.62
iKRG_18	1740	16.56
iKRG_36	1283	44.60
Rosenbrock		
GP_RBF	2214	0.11
rKRG_18	2077	3.79
BNN_MCD_5	2038	5.21
BNN_MCD_20	2019	5.15
BNN_MCD_100	1927	5.05
rKRG_36	1892	9.52
ANN_BLR	1798	14.63
iKRG_18	1732	16.93
iKRG_36	1272	45.28
Covid-19 contact reduction		
GP_RBF	2714	0.16
rKRG_18	2545	2.47
rKRG_36	2462	5.98
BNN_MCD_5	2432	8.46
BNN_MCD_20	2360	8.67
BNN_MCD_100	2226	8.47
ANN_BLR	2183	14.79
iKRG_18	2112	13.76
iKRG_36	1545	39.29

Table 2.10: **SaaEF *versus* SaaE *versus* SaaF.** Top-5 strategies according to the average best objective value (10 independent runs). Ordering according to ascending value from top to bottom. BNN_MCD_5 is used on the Schwefel and Covid-19 contact reduction problems and GP_RBF on the Rastrigin and Rosenbrock benchmarks.

AP	EC	Average	AP	EC	Average
Schwefel			Rosenbrock		
SaaEF	*dyn-df-incl*	131.95	SaaF	*par-fd-cd*	137.82
SaaEF	*dyn-dpf-excl*	136.30	SaaF	*com-dpf*	156.63
SaaF	*dyn-df-incl*	153.92	SaaF	*par-fd-hvc*	203.27
SaaEF	*par-fd-cd*	167.55	SaaF	*ei*	232.37
SaaEF	*dyn-df-excl*	168.64	SaaEF	*par-fd-hvc*	233.93
Rastrigin			Covid-19 contact reduction		
SaaE	*par-fd-cd*	18.22	SaaF	*dyn-df-incl*	6854
SaaE	*dyn-df-excl*	18.69	SaaF	*dyn-df-excl*	7115
SaaEF	*dyn-dpf-excl*	19.08	SaaEF	*dyn-df-75-excl*	7455
SaaE	*com-dpf*	19.25	SaaEF	*dyn-df-excl*	7679
SaaF	*com-dpf*	19.34	SaaEF	*dist*	7837

2.6 Conclusion

In this chapter, we have examined the design of P-SAEAs for handling moderately expensive problems. We have presented a coupling where the surrogate is connected to the EA as an evaluator and a filter and we have opted for the Bayesian Neural Network built around Monte-Carlo Dropout. Predicting with BNN_MCD relies on sampling sub-networks to emulate an ensemble of ANNs with the advantage of necessitating only one training. We have also come up with innovative ensembles of Evolution Controls that balance between intensification and diversification dynamically along the search.

The budget for the exercise is expressed as a limited number of simulations in the past studies in the field of P-SBO. We formulate the budget as a capped duration on a prefixed number of computing cores to best reflect the reality because surrogate training is potentially a considerable computing load. In the literature related to P-SAEAs, the surrogate is utilized either to evaluate or to filter out candidates. SaaEF is particularly pertinent on a moderately computationally expensive problem where the landscape is multi-modal and with weak global structure. In the proposed algorithm, multiple surrogates are tested, among them the recurrently employed Kriging and Gaussian Process models. BNN_MCD appears to be the most convenient choice for hard landscapes characterized by multi-modality and weak global structure. BNN_MCD is also benefiting as it trains fast, and thus, concedes a higher number of simulations. In P-SAEAs, no modification of the exploration/exploitation trade-off during the optimization have been suggested in the past. The dynamic ensembles of ECs combining the predictive objective value and the distance from the database of already simulated solutions is the adequate design decision to manage multi-modal problems.

Chapter 3

Parallel Surrogate-driven algorithms

Contents

3.1 Introduction

This chapter focuses on the building and appropriateness of P-SDAs for moderately and very expensive problems. The sampling of sub-surrogates is investigated to design new surrogate-optimizer couplings and the influence of the training set size is notably analysed.

The definition of the promisingness is also a top-notch component of P-SDAs to balance between exploitation and exploration. In Section 3.2, the parallel between ICs and ECs is established and a specific EA is presented to optimize any EC, hence the possibility to revisit q-EGO by replacing EI. The sampling of sub-networks from a global ANN by randomly deactivating neurons (q-subnets) is a first idea reported in Section 3.3 to both alleviating the computational effort of the AP and promoting diversity. In a similar way, the Hamiltonian Monte-Carlo sampling of parameters from Bayesian models (q-post-HMC) extends the concept beyond the scope of neural networks.

Calibrations and numerical experiments are conducted and exposed in Section 3.4 to study the design of P-SDAs relatively to search landscape features, either known in the case of the benchmark functions or unknown for the real-world application to Covid-19 transmission control. The performance of the methods and the influence of the training set size are empirically assessed in both contexts of moderately and very expensive problems. Finally, Section 3.5 concludes the chapter.

3.2 From Evolution Controls to Infill Criteria

3.2.1 EC-based selection and replacement

In the P-SDAs, presented in Sub-section 1.4.3, the AP relies on the optimization of a real-valued scalar function or a real-valued vector function called the Infill Criterion (IC). The value of the IC indicates the promisingness of candidate solutions. The IC can be single-criterion such as minimization of the POV [Reh+18], scalarized bi-criterion such as EI [JSW98], Pareto-based bi-criterion such as simultaneous minimization of the POV and maximization of the predictive standard deviation [Bis+14], dynamic weighted sum of criteria involving minimization of the POV and maximization of the distance to the database [RS07], adaptive criteria such as those from [Wan+20b] already explained in Subsection 2.3.4. Depending on the type of IC, one can have recourse to different optimization algorithms such as EAs, sequential quadratic programming [Li+10] and multi-objective EAs [Bis+14].

In P-SAEAs, the EC plays the same role as the IC in P-SDAs. The EC is a comparison operator $>_p$ that orders a set of candidate solutions according to the promisingness it specifies. Converting an IC to an EC is straightforward as all ICs implicitly implement a comparison operator. For instance, higher values of EI, given in Equation (1.53), stand for higher promisingness. The conversion from IC to EC has recurrently been applied in Chapter 2.

The EA with EC-based selection and replacement we propose hereafter enables one to optimize any type of IC by basing the selection and the replacement operators on the corresponding EC. The approach is detailed in Algorithm 5 and corresponds to the AP in Figure 1.4. One iteration of the EA with EC-based selection and replacement starts by ordering the population of candidates according to decreasing promisingness (line 3 in Algorithm 5). Subsequently, the tournament selection of parents is realized according to the position of the individuals within the population where a lower index represents a better promisingness (line 4). After the reproduction step, the merged population comprising the current population and the children is ordered according to decreasing promisingness (line 7) and the best n_{pop} individuals are retained (line 8). At the end of the search, the q best

individuals according to the EC are returned for simulation. This generic implementation may incorporate any ECs, and notably the ensembles of ECs exposed in the previous chapter. It is also worth noting that when the EC is Pareto-based multi-criterion, the EA described in Algorithm 5 is a multi-objective EA. In particular, when *par-fs-cd* is chosen, the Non-dominated Sorting Genetic Algorithm (NSGA-II) [Deb+02] is recovered for the bi-objective problem of POV minimization and maximization of the predictive standard deviation.

Algorithm 5 EA with EC-based selection and replacement

Input
E: evolution control
surrogate: surrogate model
n_{pop}: population size
n_{gen}: number of generations
q: number of candidates to return

1: $\mathcal{P} \leftarrow \text{LHS}(n_{pop})$
2: **for** $i = 1 : n_{gen}$ **do**
3: $\quad \mathcal{P} \leftarrow \text{sort\_by\_EC}(\mathcal{P}, E, \textit{surrogate})$
4: $\quad \mathcal{P}_{par} \leftarrow \text{tournament\_position}(\mathcal{P}, n_{pop})$ ▷ population of parents
5: $\quad \mathcal{P}_{chld} \leftarrow \text{reproduction}(\mathcal{P}_{par})$ ▷ population of children
6: $\quad \mathcal{P} \leftarrow \mathcal{P} \cup \mathcal{P}_{chld}$
7: $\quad \mathcal{P} \leftarrow \text{sort\_by\_EC}(\mathcal{P}, E, \textit{surrogate})$
8: $\quad \mathcal{P} \leftarrow \text{elitist\_position}(\mathcal{P}, n_{pop})$
9: **end for**
10: $\mathcal{B} \leftarrow \text{get\_best}(\mathcal{P}, E, q)$
11: **return** $\mathcal{B}$

3.2.2 q-EGO revisited

The canonical q-EGO embeds the EI as IC and the Kriging model as the ersatz simulator in [GRC10]. The generic implementation of the AP in P-SDAs highlighted in the preceding sub-section allows one to revisit the original implementation of q-EGO by including alternative surrogates and definitions of promisingness.

The extension of EI for sampling multiple points is denominated q-EI in [GRC10] and writes:

$$qEI(\boldsymbol{x^{(1)}}, \ldots, \boldsymbol{x^{(q)}}) = \mathbb{E}[y_{min} - \min(Y(\boldsymbol{x^{(1)}}), \ldots, Y(\boldsymbol{x^{(q)}}))|X, \boldsymbol{y}] \tag{3.1}$$

where $q \in \mathbb{N}\backslash\{0, 1\}$ is the number of new candidates to sample, $\boldsymbol{x^{(1)}}, \ldots, \boldsymbol{x^{(q)}}$ are candidates, $(X, \boldsymbol{y})$ is the database of already simulated candidates along with their associated simulated objective values and y_{min} is the best simulated objective value found so far. The random variable $Y()$ has been defined in Sub-section 1.3.3 within the framework of GPs. An analytical formula is derived for q-EI when $q = 2$ but eliciting such a closed-form expression for $q > 2$ is intractable. Two heuristics, namely *Kriging Believer* and *Constant Liar*, have thus been devised for the AP.

The first heuristic is designated as *Kriging Believer* in [GRC10]. Our revisited version is outlined in Algorithm 6 and is renamed *Surrogate Believer* (abbreviated *sb*) as the choice for the surrogate is not narrowed down to the Kriging model any more. The AP begins with the creation of a temporary database, initially made of the previously simulated candidates along with their associated simulated objective values (line 2). Next, the EA with EC-based selection and replacement exhibited in Algorithm 5 is invoked (line

4) q times. At each iteration, one new candidate is issued and the temporary database is enriched with the new solution along with its POV (line 6). The surrogate is then updated on the temporary database (line 7). In case of GPs, the training is partial as it does not include the re-estimation of the hyper-parameters. Partial training concedes to limit the computational effort. The designation *Surrogate Believer* comes from the trust in the surrogate prediction that appears when updating the surrogate using POVs. The issue induced by an inaccurate surrogate is reflected in situations where the process gets trapped in a non-optimal region due to predictions being erroneously too low. Conversely, predictions that are erroneously too high may result in a poor screening of interesting regions. Once the multi-point AP is executed, the resulting q new candidates are simulated in parallel and the surrogate is re-trained on the updated database only made of simulated solutions as depicted in Figure 1.4.

Algorithm 6 *Surrogate Believer* AP for q-EGO

Input
database: set of already simulated solutions
E: evolution control
surrogate: surrogate model
$\hat{f}()$: surrogate's prediction function
q: number of candidates to sample
n_{pop}: population size
n_{gen}: number of generations

1: $\mathcal{B}_{sim} \leftarrow \emptyset$ ▷ batch of new candidates
2: $tmp\_database \leftarrow$ copy($database$)
3: **for** $i = 1 : q$ **do**
4: $\boldsymbol{x^{(i)}} \leftarrow$ 1-point_AP(E, *surrogate*, n_{pop}, n_{gen}, 1) ▷ Algorithm 5
5: $\mathcal{B}_{sim} \leftarrow \mathcal{B}_{sim} \cup \boldsymbol{x^{(i)}}$
6: $tmp\_database \leftarrow tmp\_database \cup \{(\boldsymbol{x^{(i)}}, \hat{f}(\boldsymbol{x^{(i)}}))\}$
7: update_surrogate(*surrogate*, *tmp_database*)
8: **end for**
9: **return** $\mathcal{B}_{sim}$

The second heuristic is called *Constant Liar* and follows the same pattern as the *Surrogate Believer* AP, only differing on the way the temporary database is updated as shown in Algorithm 7. To each new candidate is associated a constant L that plays the role of the objective value (line 7). A larger value for L favors exploration while a lower value endorses exploitation. In [GRC10], L is set either as the minimum, the mean or the maximum objective value observed in the database. In this thesis, only the variant relying on the mean, denoted *cl-mean*, is considered (line 3).

Besides extending q-EGO to new ICs and surrogates, we suggest in this thesis to evaluate the performances of the method in the context of moderately expensive problems. Indeed, the computational budget is usually very low in the investigations available in the literature. For instance, 150 to 300 simulations are granted to q-EGO for the search of an aerodynamic shape in [Liu+17]. Another example is the design of air vehicles in [Cha+15] where the evaluation relies on very expensive physical experiments. The computational effort required per AP in q-EGO may be a downside when tackling moderately expensive problems. The contributions presented in the following section follow this direction.

Algorithm 7 *Constant Liar* AP for q-EGO

Input
database: set of already simulated solutions
E: evolution control
surrogate: surrogate model
q: number of candidates to sample
n_{pop}: population size
n_{gen}: number of generations

1: $\mathcal{B}_{sim} \leftarrow \emptyset$ ▷ batch of new candidates
2: $tmp\_database \leftarrow \text{copy}(database)$
3: $L \leftarrow \text{mean}_{y \in database}(y)$
4: **for** $i = 1 : q$ **do**
5: $\boldsymbol{x}^{(i)} \leftarrow$ 1-point_AP(E, *surrogate*, n_{pop}, n_{gen}, 1) ▷ Algorithm 5
6: $\mathcal{B}_{sim} \leftarrow \mathcal{B}_{sim} \cup \boldsymbol{x}^{(i)}$
7: $tmp\_database \leftarrow tmp\_database \cup \{(\boldsymbol{x}^{(i)}, L)\}$
8: update_surrogate(*surrogate*, *tmp_database*)
9: **end for**
10: **return** $\mathcal{B}_{sim}$

3.3 Fast Acquisition Processes

3.3.1 q-subnets: sub-networks as multi-surrogate

The new P-SDA based on Artificial Neural sub-networks we introduce is inspired by the ensemble of surrogates revealed in [VHW13] and on the Monte-Carlo Dropout proposed in [Gal16] to approximate BNN training. In [VHW13], q candidates are sampled per cycle, each one derived by optimizing EI on one of the q surrogates at one's disposal. It is advised in [VHW13] to set q according to the computational capabilities that would amount to $n_{cores} = 18$ computing cores in our computational environment. Our novel strategy is called q-subnets and consists of a Monte-Carlo sampling of multiple sub-networks from a global ANN. The method encompassing a one-hidden-layer ANN is dissected in Algorithm 8 for the general scenario where q is not necessarily equal to n_{cores}. The sampling is achieved by randomly deactivating units in the global ANN to generate n_{cores} sub-networks (line 3-4) similarly to Monte-Carlo Dropout. The n_{cores} sub-networks are involved in n_{cores} parallel APs (line 5), each of them returning $\frac{q}{n_{cores}}$ candidates. (It is assumed that q is a multiple of n_{cores} and one candidate is issued per AP if $q = n_{cores}$). Exploration is guaranteed by the diversity among the n_{cores} randomly sampled sub-networks while exploitation is assured by the EC. The benefit of q-subnets over the ensemble-based mechanism from [VHW13] is the saving of $n_{cores} - 1$ trainings.

The idea behind q-subnets comes from the field of discrete space screening where a BNN is used to filter already simulated molecule shapes [Her+17]. The posterior probability distribution on the parameters of a BNN approximated *via* probabilistic back-propagation is sampled multiple times to draw various networks. In the experiments led in [Her+17], 100,000 molecules are retained by the screening strategy. Both the task and the budget in terms of simulations are not comparable to the scope of this thesis.

Algorithm 8 q-subnets AP with one-hidden-layer ANN

Input

p_{drop}: probability of dropping out neurons
$(W^{(1)}, \boldsymbol{w^{(2)}})$: weights trained with Dropout
$\boldsymbol{h}()$: activation function
E: evolution control
q: number of candidates to sample
n_{cores}: number of computing cores
n_{pop}: population size
n_{gen}: number of generations

1: $\mathcal{B}_{sim} \leftarrow \emptyset$ ▷ batch of new candidates
2: **for** $i = 1 : n_{cores}$ in parallel **do**
3: $\boldsymbol{\epsilon^{(1)}} \leftarrow$ Bernouilli_sampling(p_{drop})
4: $\hat{f}_i(\star) \leftarrow \boldsymbol{\epsilon^{(1)}} \odot \boldsymbol{w^{(2)}}.\boldsymbol{h}(diag(\boldsymbol{\epsilon^{(1)}}).W^{(1)}.\star)$ ▷ $\star$ stands for a dummy variable
5: $\boldsymbol{x^{(i)}} \leftarrow$ EC-based_AP(E, $\hat{f}_i(\star)$, n_{pop}, n_{gen}, $\frac{q}{n_{cores}}$) ▷ Algorithm 5
6: $\mathcal{B}_{sim} \leftarrow \mathcal{B}_{sim} \cup \{\boldsymbol{x^{(i)}}\}$
7: **end for**
8: **return** $\mathcal{B}_{sim}$

3.3.2 q-post-HMC: sampling of surrogates

In the same spirit than q-subnets, we come up with an approach based on the sampling of sub-surrogates from a fully Bayesian GP or a BNN. The new method, called q-post-HMC, relies on Monte-Carlo Markov Chain (MCMC) sampling and reduces the computational cost of a q-EGO cycle by only implying one surrogate training. When using the BNN, the difference with q-subnets is the preservation of the predictive capacity as no hidden unit is deactivated. However, preserving the predictive capacity could end up with a deteriorated diversity. When employing the GP as surrogate, certain landscapes are expected to be better approximated than with an ANN, thus an advantage is expected over q-subnets. The pseudo-code of q-post-HMC is shown in Algorithm 9. Hamiltonian Monte-Carlo (HMC) [Nea96] is run to sample n_{cores} configurations from the posterior distribution of the hyper-parameters of the GP or the parameters (weights and biases) of the BNN (line 3), consequently generating n_{cores} GPs or n_{cores} ANNs respectively. In the case of Ordinary Kriging, the hyper-parameters are μ, α^{-1}, $\boldsymbol{\eta}$ and $\boldsymbol{p}$, as mentioned in Section 1.3.3. The n_{cores} sub-surrogates are optimized in parallel (line 4) and the emanating q new candidates are simulated simultaneously.

The hyper-parameters of the GP and the parameters of the BNN are symbolized by $\boldsymbol{\omega}$ in Algorithm 9 and generally referred to as "parameters" in the following. Training a fully Bayesian GP or a BNN implies to marginalize over the space of parameters to produce the predictive distribution (Equation (1.10)). This marginalization is not tractable but can be approximated by the following formula:

$$p(y^*|\boldsymbol{x^*}, X, \boldsymbol{y}) \approx \frac{1}{n_s} \sum_{t=1}^{n_s} p(y^*|\boldsymbol{x^*}, \boldsymbol{w^{(t)}}) \tag{3.2}$$

where $\boldsymbol{w^{(1)}}, \ldots, \boldsymbol{w^{(n_s)}}$ are sampled from the posterior distribution $p(\boldsymbol{w}|X, \boldsymbol{y})$. A higher degree of independence among the samples provides a better approximation of the marginalization. The MCMC methods enable to accomplish the drawing without assumption over the posterior. They resort to an ergodic Markov Chain, that has $Q = p(\boldsymbol{\omega}|X, \boldsymbol{y})$ as its equilibrium distribution and that converges to it from any initial state. A Markov Chain

Algorithm 9 q-post-HMC AP

Input
E: evolution control
$surrogate(\boldsymbol{\omega})$: surrogate model (either GP or BNN)
q: number of candidates to sample
n_{cores}: number of computing cores
n_{pop}: population size
n_{gen}: number of generations

1: $\mathcal{B}_{sim} \leftarrow \emptyset$ ▷ batch of new candidates
2: **for** $i = 1 : n_{cores}$ in parallel **do**
3: $\boldsymbol{\omega^{(i)}} \leftarrow$ HMC_sampling() ▷ Sampling (hyper-)parameters
4: $\boldsymbol{x^{(i)}} \leftarrow$ EC-based_AP(E, $surrogate(\boldsymbol{\omega^{(i)}})$, n_{pop}, n_{gen}, $\frac{q}{n_{cores}}$) ▷ Algorithm 5
5: $\mathcal{B}_{sim} \leftarrow \mathcal{B}_{sim} \cup \{\boldsymbol{x^{(i)}}\}$
6: **end for**
7: **return** $\mathcal{B}_{sim}$

is defined by an initial state $\boldsymbol{\omega^{(1)}}$ and a transition distribution $T(\boldsymbol{\omega^{(t+1)}}|\boldsymbol{\omega^{(t)}})$. The equilibrium distribution is the only distribution that is invariant by the transition (*i.e.* if $\boldsymbol{\omega^{(t)}}$ stems from the distribution Q then $\boldsymbol{\omega^{(t+1)}}$ also originates from distribution Q). The challenge is to find such a Markov Chain that converges as faster as possible to Q and that outputs relatively independent samples. A number of warm-up steps n_{ws} is needed for attaining convergence.

A popular method for MCMC is the Metropolis algorithm [Met+53], where one transition consists of the three following steps:

1. A new candidate $\tilde{\boldsymbol{\omega}}$ is drawn from a proposal distribution.

2. If $Q(\tilde{\boldsymbol{\omega}}) \geqslant Q(\boldsymbol{\omega^{(t)}})$ then the candidate is accepted. Otherwise, the candidate is accepted with probability $Q(\tilde{\boldsymbol{\omega}})/Q(\boldsymbol{\omega^{(t)}})$.

3. If the candidate is accepted, then $\boldsymbol{\omega^{(t+1)}} \leftarrow \tilde{\boldsymbol{\omega}}$. In case of rejection, $\boldsymbol{\omega^{(t+1)}} \leftarrow \boldsymbol{\omega^{(t)}}$.

In order to prevent the samples to become too much dependent, the proposal distribution must both imply an acceptation rate high enough and suggest candidates different enough from the previous states. For complex and highly dimensional Q, this difficulty translates into random walks instead of an efficient covering of the distribution thus making the convergence very slow. This obstacle is circumvented by simulating a dynamic Hamiltonian system to issue new states.

In HMC [Nea96; HG14], the parameters $\boldsymbol{\omega}$ play the role of the particle's positions and auxiliary variables $\boldsymbol{r_m}$ are introduced as their momentum. The joint density is expressed in terms of a function H by:

$$P(\boldsymbol{\omega}, \boldsymbol{r_m}) \propto \exp\left(-H(\boldsymbol{\omega}, \boldsymbol{r_m})\right) \tag{3.3}$$

where H is viewed as the total energy function resulting from the sum of the potential energy E and the kinetic energy K, expressing the marginal distributions of $\boldsymbol{\omega}$ and $\boldsymbol{r_m}$ respectively:

$$P(\boldsymbol{\omega}) \propto \exp\left(-E(\boldsymbol{\omega})\right) \tag{3.4}$$
$$P(\boldsymbol{r_m}) \propto \exp\left(-K(\boldsymbol{r_m})\right) \tag{3.5}$$

The iteration involved to sample independent vectors from $P(\boldsymbol{\omega})$ first draws a value for $\boldsymbol{r_m}$ where $P(\boldsymbol{r_m})$ is set as a Gaussian. Then, the Hamiltonian system is simulated approximately by discretization through L_d leapfrog steps implicating the partial derivatives of E and a step size parameter ϵ_d. The ensuing new state $(\tilde{\boldsymbol{\omega}}, \tilde{\boldsymbol{r_m}})$ is accepted according to the Metropolis rules where lower changes in total energy increase the probability of acceptance. Indeed, the exact simulation would have preserved H. To set the parameters L_d and ϵ_d automatically, an extension to HMC, called No-U-Turn Sampler is setup in [HG14]. In addition to alleviate the burden to tweak parameters, this method further accelerates convergence by avoiding random walks and repeated samples. The HMC with No-U-Turn Sampler is the method chosen in q-post-HMC.

In [SLA12], a GP is treated in a fully Bayesian way and a MCMC sampling is carried out to compute the integrated IC that takes the uncertainty-aware hyper-parameters into account. The method from [SLA12] is not parallel but a parallel variant is proposed in [Sno+15] to deal with heterogeneous simulation duration. In [Sno+15], the integrated IC is computed by attributing fantasy values to the running simulations, similarly to *Constant Liar*. Setting the fantasy values is not an easy task and bad choices could result in missing interesting candidates. In our method, we remove this difficulty by decoupling the AP for each candidate through sub-surrogates sampling.

3.3.3 q-Pareto: a Pareto dominance-based AP

Multi-objective optimization of ICs further reduces the computational burden of the AP in P-SDAs by requiring to apply a unique optimization iteration per cycle. The P-SDA whose AP lies on a multi-objective optimization is called q-Pareto as outlined in Algorithm 10. The algorithm is identical to the EA with EC-based selection and replacement from Sub-section 3.2.1 with the exception of the EC being mandatorily Pareto-based as those from Sub-section 2.3.2. Multi-objective optimization does not issue a single candidate but rather a NDS constituted of diverse non dominated solutions from which q individuals are kept.

Algorithm 10 q-Pareto AP

Input
E: Pareto-based evolution control
surrogate: surrogate model
q: number of candidates to sample
n_{pop}: population size
n_{gen}: number of generations

1: $\mathcal{P} \leftarrow \text{LHS}(n_{pop})$
2: **for** $i = 1 : n_{gen}$ **do**
3: $\quad \mathcal{P} \leftarrow \text{sort\_by\_EC}(\mathcal{P}, E, \textit{surrogate})$
4: $\quad \mathcal{P}_{par} \leftarrow \text{tournament\_position}(\mathcal{P}, n_{pop})$ ▷ population of parents
5: $\quad \mathcal{P}_{chld} \leftarrow \text{reproduction}(\mathcal{P}_{par})$ ▷ population of children
6: $\quad \mathcal{P} \leftarrow \mathcal{P} \cup \mathcal{P}_{chld}$
7: $\quad \mathcal{P} \leftarrow \text{sort\_by\_EC}(\mathcal{P}, E, \textit{surrogate})$
8: $\quad \mathcal{P} \leftarrow \text{elitist\_position}(\mathcal{P}, n_{pop})$
9: **end for**
10: $\mathcal{B} \leftarrow \text{get\_best}(\mathcal{P}, E, q)$
11: **return** $\mathcal{B}$

The q-Pareto approach is abundant in the literature about P-SBO. In [Fen+15; HSR17], the Multi-Objective Evolutionary Algorithm based on Decomposition (MOEA/D) simultaneously maximizes the local and global EI. The infill criteria EI, PI and LCB are optimized concurrently by NSGA-II in [Lyu+18]. In [GKW17], NSGA-II is invoked to minimize the POV and maximize the predictive standard deviation. To retain q candidates from the resulting NDS, a random filtering is performed in [Lyu+18]. In [Fen+15; HSR17], maximization of the distance to the database, extreme solutions of the NDF and clustering in the objective space constitute the sampling strategy. Similar criteria are leveraged in *par-fd-cd* defined in Sub-section 2.3.2. In [GKW17], maximization of EI and the predictive standard deviation and minimization of the POV compose the screening. Using EI restricts the choice of surrogate model to GPs. In our approach, we relax the condition over the surrogate model type by discarding EI and we include the filtering within the EC definition. The experiments conducted in [Fen+15; HSR17; GKW17] feature low dimensional benchmark problems thus justifying the use of EI. A real-world application to analog circuits is considered in [Lyu+18] with 12 design variables. In all the aforementioned articles, the experimental protocol characterizes very expensive problems.

3.4 Experiments

This section is dedicated to the empirical comparison of the P-SDAs with respect to the APs, ECs and surrogates. The computational setting is the same as the one settled to investigate P-SAEAs in Sub-section 2.5.1. One search runs during 30 minutes on 18 computing cores from Grid5000 and one simulation is assumed to last around 15 seconds.

3.4.1 Calibration of GP_HMC and BNN_HMC

The surrogates employed in q-post-HMC are tuned hereafter. The Gaussian Process trained through HMC (GP_HMC) is based on the RBF kernel of Equation (1.28) with the prior distributions for the hyper-parameters set to:

$$s \sim \mathcal{U}(0.01, 10) \tag{3.6}$$

$$\sigma \sim \mathcal{U}(0, 5) \tag{3.7}$$

Uniform distributions are adopted to reflect our lack of knowledge and the bounds have been roughly tweaked by few trial-and-error tests. The Bayesian Neural Network trained through HMC (BNN_HMC) comprises two fully connected hidden layers of 50 hyperbolic tangent units each, such as in [Her+17]. The prior for the weights and biases is $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$. For both surrogates, the prior over the variance of the data-related noise of Equation (1.7) is given by:

$$\beta^{-1} \sim \mathcal{U}(0.01, 0.1) \tag{3.8}$$

The grid-search calibration of the number of warm-up steps n_{ws} for HMC, considering the values $\{5, 50, 500, 5000\}$, is reported in Table 3.1. The training and validation sets are the same than those used in Sub-section 2.4.1 to tune BNN_MCD with 256 and 1024 points respectively. The validation correlation coefficient vR2 is computed on the normalized predictions and validation targets by sampling 18 parameters through HMC and averaging the 18 predictions generated by the sub-surrogates. Increasing n_{ws} unsurprisingly raises the training time, for instance from 1 to 496 seconds for $n_{ws} = 5$ to $n_{ws} = 5000$ when training GP_HMC on Rosenbrock. The training is even more longer for BNN_HMC, reaching 20 minutes for $n_{ws} = 5000$. By analysing the vR2, 5 warm-up steps seem to be enough in GP_HMC, however, it is not so for BNN_HMC that needs 50

Table 3.1: **Calibration of GP_HMC and BNN_HMC**. Ranks according to training time (TT) and validation correlation coefficient (vR2) are denoted in parentheses.

	GP_HMC		BNN_HMC	
n_{ws}	TT	vR2	TT	vR2
	Schwefel			
5000	188(4)	0.059(2)	1203(4)	**-0.803(1)**
500	28(3)	0.057(3)	121(3)	-0.820(2)
50	5(2)	0.056(4)	23(2)	-0.927(3)
5	**2(1)**	**0.060(1)**	**10(1)**	-509(4)
	Rastrigin			
5000	167(4)	-0.0138(2)	1194(4)	-0.538(2)
500	26(3)	-0.014(3)	120(3)	**-0.422(1)**
50	5(2)	**-0.0136(1)**	22(2)	-0.745(3)
5	**1(1)**	-0.021(4)	**10(1)**	-460(4)
	Rosenbrock			
5000	496(4)	**0.628(1)**	1208(4)	**0.405(1)**
500	73(3)	0.624(2)	120(3)	0.329(2)
50	12(2)	0.585(3)	22(2)	0.298(3)
5	**1(1)**	0.448(4)	**10(1)**	-320(4)

steps. Because a trade-off between low training time and high vR2 is desired, it is decided to keep $n_{ws} = 50$ henceforth. The difference in vR2 exhibited when augmenting n_{ws} seems not to be so prominent while the difference is more meaningful for the training time.

The surrogate comparison led in Sub-section 2.4.2 is reproduced to consider GP_HMC and BNN_HMC and the training times and validation correlation coefficients are reported in Table 3.2 along with those obtained previously for BNN_MCD and GP_RBF. The GP_HMC proves to be the best predictor for Rosenbrock and performs well on Schwefel. The computational load inherent to GP_HMC comes in between those of GP_RBF and BNN_MCD. Conversely, BNN_HMC is the worst predictor in almost all cases and is the slowest to train.

3.4.2 Calibration of the optimizer

The EA with EC-based selection and replacement illustrated in Algorithm 5 is utilized in all of the P-SDAs presented hitherto. It is endowed with a SBX cross-over and a polynomial mutation with probability $p_c = 0.9$ and $p_m = \frac{1}{d} = \frac{1}{16}$ respectively and distribution index $\eta_c = 10$ and $\eta_m = 50$ respectively. The population size n_{pop} and the number of generations n_{gen} are calibrated by grid search with $n_{pop} \in \{50; 100; 150; 200\}$ and $n_{gen} \in \{50; 100; 150; 200\}$. The final results of the calibration are summed up in Table 3.3. For each (n_{pop}, n_{gen}) pair, ten runs are executed for the Schwefel and the Rosenbrock benchmark problems. For each AP, the particular choice for the (surrogate, EC) pair is noted down in Table 3.3. Although these particular configurations create a bias in the calibration procedure, the extreme computing load of a full calibration is bypassed. The box-plots of the final objective values reached at the end of the searches are deferred to the appendix section as pointed out in Table 3.3.

The configuration $(n_{pop}, n_{gen}) = (50, 100)$ provides the best averaged final objective value for q-subnets and q-EGO in the Rosenbrock problem and for q-post-HMC with GP_HMC on Schwefel. It is also the second best arrangement for q-EGO in the Schwefel function and for q-post-HMC with GP_HMC on the Rosenbrock problem. These values

Table 3.2: **Surrogates comparison**. Training time (TT) and validation correlation coefficient (vR2) averaged over 10 runs for each surrogate and each **benchmark problem**. Ranks according to TT and vR2 are denoted in parentheses.

	72 training samples		256 training samples	
Surrogates	TT	vR2	TT	vR2
		Schwefel		
BNN_MCD	6.56(3)	**-3.6e-2(1)**	7.09(3)	-1.3e-3(3)
GP_RBF	**4.1e-1(1)**	-5.6e-2(3)	**8.8e-01(1)**	5.3e-2(2)
GP_HMC	2.3(2)	-5.1e-2(2)	6.3(2)	**5.8e-2(1)**
BNN_HMC	1.5e+1(4)	-1.0(4)	2.3e+1(4)	-9.2e-1(4)
		Rastrigin		
BNN_MCD	6.6(3)	**-2.2e-2(1)**	8.3(3)	-7.4e-4(2)
GP_RBF	**2.8e-1(1)**	-9.3e-2(3)	**6.2e-1(1)**	**1.8e-2(1)**
GP_HMC	2.1(2)	-5.9e-2(2)	5.8(2)	-1.4e-2(3)
BNN_HMC	1.6e+1(4)	-1.3(4)	2.2e+1(4)	-7.4e-1(4)
		Rosenbrock		
BNN_MCD	7.38(3)	-7.0e-3(3)	7.2(2)	3.4e-3(4)
GP_RBF	**2.8e-1(1)**	2.1e-1(2)	**6.7e-1(1)**	4.8e-1(2)
GP_HMC	2.2(2)	**2.5e-1(1)**	8.4(3)	**5.0e-1(1)**
BNN_HMC	1.6e+1(4)	-3.3e-1(4)	2.2e+1(4)	2.9e-1(3)

Table 3.3: Calibration of the EA with EC-based selection and replacement (q=18).

	q-EGO *cl-mean* GP_RBF *ada-wang-min*	q-subnets BNN_MCD *pov*	q-Pareto BNN_MCD *par-fs-cd*	q-post-HMC GP_HMC / BNN_HMC *pov*
(n_{pop}, n_{gen})	(50, 100)	(50, 100)	(150, 50)	(50, 100)
Box-plot Schwefel	D.1	D.3	D.5	D.7 / D.9
Box-plot Rosenbrock	D.2	D.4	D.6	D.8 / D.10

also show acceptable performance for q-post-HMC with BNN_HMC. Setting $(n_{pop}, n_{gen}) = (150, 50)$ is the better decision according to the average for q-Pareto in the Schwefel case and also a good option in the Rosenbrock scenario. The retained values for n_{pop} and n_{gen} are kept per AP for all the following experiments.

3.4.3 Experimental protocol

The same surrogates than the ones of the previous chapter are employed for q-EGO and q-Pareto with the unique exception of ANN_BLR that is not inserted into q-Pareto on the Covid-19 problem. The number of sub-networks for BNN_MCD is fixed to 5 and all the available simulated solutions are used for training this model. For ANN_BLR, GP_RBF and the Kriging models, the training is carried out on the last 256, 72 and 18 simulations respectively. In q-post-HMC, the surrogates are trained on the last 256 simulated candidates. The Pyro [Bin+18] and NumPyro [PPJ19] libraries are triggered to implement GP_HMC and BNN_HMC. All the software libraries used in this thesis are listed in Table F.1.

In q-EGO, only the most promising candidate is retained per 1-point AP. Consequently, in the definition of the Pareto-based bi-criterion ECs *par-fd-cd* and *par-fs-cd*, the criterion imposed to distinguish between the extreme points from the first NDF is removed (fourth line in Equation (2.14)). Likewise, the special treatment of the extreme points in *par-fd-hvc* and *par-fs-hvc* is abolished (Equation (2.15) and the associated exceptions). In doing so, the aforementioned ECs will not behave similarly to *pov*. Additionally, the four dynamic inclusive ensembles of ECs are not integrated into q-EGO as they would coincide with their exclusive counterparts. The list of ECs actually added to q-EGO is displayed in Table 2.1 and Table 2.2. Both the *cl-mean* and *sb* APs are considered on the benchmarks but only *cl-mean* is applied on the Covid-19-related case. On the benchmarks, 290 q-EGO variants are executed and 145 variants are run on the real-world application. Assuming 10 independent runs, the total number of searches amounts to 10,150.

The six Pareto-based bi-criterion ECs listed in Table 2.1 are invoked in q-Pareto resulting in 30 variants (resp. 24 variants) of q-Pareto in the case of the benchmarks (resp. Covid-19 application). The number of optimization exercises adds up to 1,140.

In q-subnets and q-post-HMC, the sub-surrogates ensure exploration while the EC takes care of exploitation. Only the *pov* and *par-fd-cd* ECs are considered for these APs resulting in 240 optimizations.

Two values are tested for the number of proposals per cycle: $q \in \{18, 72\}$. The overall number of searches finally reaches 23,060.

3.4.4 Empirical analysis

Computational load of the APs
The differences between the APs in terms of computational costs are illustrated in Table 3.5 by the average number of simulations per search for each combination of AP, benchmark and surrogate and $q = n_{cores} = 18$. The average number of simulations is lower for q-EGO (between 165 and 884) than for q-Pareto (between 697 and 1907) while the average training time per surrogate is pretty much the same. This observation was expected because one q-EGO cycle consists of the sequential execution of q surrogate updates and q optimizations while only one multi-objective search and one training are needed per q-Pareto cycle.

Table 3.4: **Prediction time** (in seconds) when predicting 50 or 150 solutions. Average over the three benchmarks and the 10 runs.

Surrogate	Average prediction time (50 predictions)	Average prediction time (150 predictions)
ANN_BLR	0.001	0.001
GP_HMC	0.002	0.002
GP_RBF	0.004	0.005
BNN_MCD	0.09	0.09
rKRG	0.1	0.2
iKRG	0.1	0.2
BNN_HMC	0.4	0.6

q-subnets and q-post-HMC further mitigate the computational burden. For an average training time of around 10 seconds, q-subnets and q-post-HMC with GP_HMC allow one to simulate 1230 candidates per search in average while q-Pareto would simulate less than 1100 solutions. For an average training time of 19 seconds, around 930 simulations are performed for q-post-HMC with BNN_HMC while only 700 simulations are enabled by q-Pareto for a lower training time of 16 seconds. In q-subnets and q-post-HMC, one surrogate update and n_{cores} parallel optimizations are carried out per cycle. It can be concluded that the multi-objective optimization in a q-Pareto cycle is much slower than the n_{cores} parallel single-objective optimizations involved in q-subnets and q-post-HMC with *pov* as EC. Indeed, for Schwefel with BNN_MCD and $q = n_{cores} = 18$, we get:

- 1382 simulations in average for q-subnets with *pov*
- 1065 simulations in average for q-subnets with *par-fd-cd*
- 1108 simulations in average for q-Pareto with *par-fd-cd*

Therefore, relying on a Pareto-based bi-criterion EC is more computationally demanding due to the non-dominated sorting borrowed from NSGA-II [Deb+02].

It can be noticed in Table 3.5 that despite having a relatively low average training time, BNN_MCD shows low average numbers of simulations in q-EGO. Two reasons explain this observation. First, GP_RBF, iKRG and rKRG are only partially re-trained during one cycle. The partial training does not re-compute the hyper-parameters, therefore reducing the computational cost. The average training time reported in Table 3.5 is only calculated based on the full updates occurring at the junction of two cycles. This is an advantage over ANN_BLR and BNN_MCD for which only full trainings are realized. The second reason is to be attributed to the computational workload induced by predictions. For batches of 50 and 150 solutions to predict, the average prediction time per surrogate are displayed in Table 3.4. In this respect, the difference between ANN_BLR and BNN_MCD gets substantial for q-EGO where $q.n_{pop}.n_{gen} = 18 * 50 * 100 = 90,000$ predictions are demanded per cycle. For q-Pareto, $n_{pop}.n_{gen} = 150 * 50 = 7,500$ predictions are made per cycle, making the effect less impacting. It is important to stress the influence of the implementation over the predictive time. Indeed, BNN_MCD is coded by diverting the *Keras* library regardless of the computational efficiency. The Kriging models are constructed through *pybnn* that is suspected to lack of computational efficiency. The high prediction time reported on the batch of 150 decision vectors for rKRG justifies the low number of simulations for q-Pareto in Table 3.5 in spite of the low training time.

Table 3.5: **P-SDAs with q=18** applied to the **benchmark problems**. Average number of simulations per search and overall average training time (in seconds) for each surrogate and acquisition process. Ordering according to the average number of simulations in decreasing order from top to bottom.

Surrogate	Average number of simulations	Average training time	Surrogate	Average number of simulations	Average training time
q-EGO			q-Pareto		
Schwefel					
GP_RBF	879	0.2	GP_RBF	1893	0.17
rKRG	252	1.52	BNN_MCD	1084	7.03
iKRG	239	7.12	ANN_BLR	958	14.31
ANN_BLR	197	8.44	rKRG	904	3.50
BNN_MCD	165	6.85	iKRG	697	16.05
Rastrigin					
GP_RBF	882	0.14	GP_RBF	1902	0.13
rKRG	249	1.59	BNN_MCD	1072	7.57
iKRG	239	7.18	ANN_BLR	960	14.30
BNN_MCD	182	5.05	rKRG	891	3.74
ANN_BLR	197	8.45	iKRG	702	16.11
Rosenbrock					
GP_RBF	884	0.11	GP_RBF	1907	0.11
rKRG	247	1.63	BNN_MCD	1099	6.50
iKRG	238	7.32	ANN_BLR	959	14.34
ANN_BLR	197	8.46	rKRG	920	3.16
BNN_MCD	191	4.20	iKRG	698	16.50

Benchmark	Average number of simulations	Average training time
q-subnets		
Schwefel	1224	7.82
Rastrigin	1240	7.26
Rosenbrock	1289	6.11
q-post-HMC with GP_HMC		
Schwefel	1435	5.63
Rastrigin	1230	10.04
Rosenbrock	1168	11.67
q-post-HMC with BNN_HMC		
Schwefel	934	18.57
Rastrigin	937	18.47
Rosenbrock	938	18.21

Effectiveness of the methods
It is now proposed to focus on the performance of the methods with respect to the averaged objective value obtained at the end of the search. For each value of q and for each AP, the top configurations are ranked in Table D.1, Table D.2, Table D.3 and Table D.4 for the three benchmarks and the Covid-19 transmission control respectively. According to these tables, it is generally a better choice to set q=18 than q=72. This observation stresses the hurdle to ensure diversification in large sets of candidates. However, it is worth reminding that q=18 is set during the calibrations, which creates a bias in the comparison. The analyses led thenceforth only concentrate on q=18.

Regarding the APs, q-subnets with *pov* provides the best averaged objective value in the multi-modal Schwefel problem with weak global structure. The predictive accuracy of BNN_MCD coupled with the random sampling of sub-networks adequately identify the multiple basins of attraction. In the real-world problem, the committee-based EC *com-spf* newly introduced in the previous chapter demonstrates the best results in q-EGO with *Constant Liar* and the GP_RBF surrogate. On the Rastrigin problem, q-Pareto with rKRG and *par-tian-fd* overcomes all the alternatives while q-EGO with *Surrogate Believer*, GP_RBF and *ada-wang-min* occupies the first place of the podium on the Rosenbrock problem. No one AP is better than all the others in all the scenarios and the (surrogate, EC) pair shows to have an important influence. Consequently, the design of P-SDAs should be guided by the characteristics of the search landscape.

To distinguish between the ECs' capabilities, the box plots of the 10 final objective values are displayed in Figure D.11, Figure D.12, Figure D.13 and Figure D.14 for the three benchmarks and the real-world application respectively. For the sake of visualization, only the most interesting configurations are shown.

In q-EGO, *ada-wang-min* and *par-tian-fs* return consistently good outcomes on the benchmarks but not on the Covid-19 case where *com-spf* is better. Both *ada-wang-min* and *par-tian-fs* favor exploitation by minimizing the POV and minimizing the predictive standard deviation therefore achieving a quick convergence to the optimum spotted at the onset of the search. On Rastrigin, *pov* in *cl-mean* also provides good results, reflecting the good accuracy of GP_RBF on this landscape.

In q-Pareto, the ECs relying on the predictive standard deviation break away on the Schwefel function thus proving the informativeness of the predictive uncertainty descending from BNN_MCD. Nevertheless, *par-tian-fd* is to be preferred on Rastrigin and Rosenbrock where GP_RBF is employed.

Convergence profiles
In order to best relate the number of simulations with the averaged objective value, the convergence profiles for the best strategies for each problem are drawn and exposed in Figure 3.1 for the Schwefel problem and in the appendix section in Figure D.15, Figure D.16 and Figure D.17 for the remaining instances. The retained approaches are those showing the best average objective value for one particular number of simulations at least. The top strategies (according to averaged final objective value) are those allowing a moderate number of simulations and proposing a trade-off between fast improvement during the first bursts of simulations and continuous improvement afterwards.

Let's focus on the Schwefel scenario where the convergence profiles are gathered in Figure 3.1. The convergence demonstrated by q-EGO *cl-mean* with BNN_MCD is very fast in terms of number of simulations but the computational cost of the AP prevents to process more than 100 simulations approximately. The q-post-HMC AP based on the GP_HMC sub-surrogate also achieves a very fast improvement during the first batches of simulations

but with the additional benefit of allowing one to perform much more simulations than q-EGO, reaching around 1,400 function evaluations. Although accomplishing the highest number of simulations by reducing the update frequency (q=72), q-Pareto with *par-fs-cd* and BNN_MCD is not the most adequate strategy because it does not improve sufficiently at the beginning of the search. Instead, q-subnets with *pov* implements a convenient trade-off between fast improvement at the start-up and continuous enhancement afterwards.

On the Rosenbrock and the Covid-19 problems, whose profiles are displayed in Figure D.16 and Figure D.17 respectively, q-EGO and q-Pareto attain convergence state in few simulations. No further significant improvement is achieved subsequently and the best method according to the final average objective value is the one with the steepest improvement during the first bursts of simulations. The similarity of the curves in Figure D.16 and Figure D.17 suggests a similarity of the landscapes of the Rosenbrock and the Covid-19 problems.

For Rastrigin, in Figure D.15, an almost uninterrupted gain is granted notably for q-Pareto with rKRG and *par-tian-fd* where a relevant reduction is observed at the end of the execution. It may be easier to steadily enhance the objective value in Rastrigin in comparison to Rosenbrock due to the multi-modality of the landscape.

3.4.5 Complete training set

Reducing the training set size for the majority of the surrogates may convey the impression of unfairness in comparison to BNN_MCD which is trained on the whole database of simulated candidates. Indeed, in P-SDAs the surrogate plays the most crucial role by granting a fast improvement with respect to the number of simulations. To elucidate this impression, the previous experiments (except those with q=72 or *Surrogate Believer* or BNN_MCD) are replayed with a full training set for all the surrogates.

The budget being maintained at 30 minutes on 18 computing cores, the Kriging models are inappropriate as their training is too computationally costly. In the majority of the cases, the first Kriging training is not finished after 30 minutes. For this reason, the methods based on iKRG and rKRG are not analyzed hereafter.

The average numbers of simulations and average training times per AP for q=18 are reported in Table 3.6. By comparing with Table 3.5, training on the whole database raises the training time and reduces the number of simulations in almost all the situations. Consequently, among the methods considering the complete training set, q-subnets is the AP demonstrating the overall highest number of simulations. The case of ANN_BLR in *cl-mean* is an exception as the increase in the number of training samples seems to have no impact. This is explained by two facts. Firstly, the computational requirements of training ANN_BLR is linear to the number of training points. Secondly, because of the high computational load of a q-EGO cycle, the database does not grow meaningfully.

The top configurations per AP are reported in Table D.5, Table D.6, Table D.7 and Table D.8 for the three benchmarks and the Covid-19 problem respectively. The best strategy for each problem is not overthrown after the new executions. Using the whole database yields some improvements in punctual circumstances. For instance, the best q-post-HMC approach tagged so far in the Rastrigin problem yielded a final average objective value of 121.85 (Table D.2) while an average of 98.4 is obtained by the best q-post-HMC algorithm for GP_HMC trained on the whole database (Table D.6).

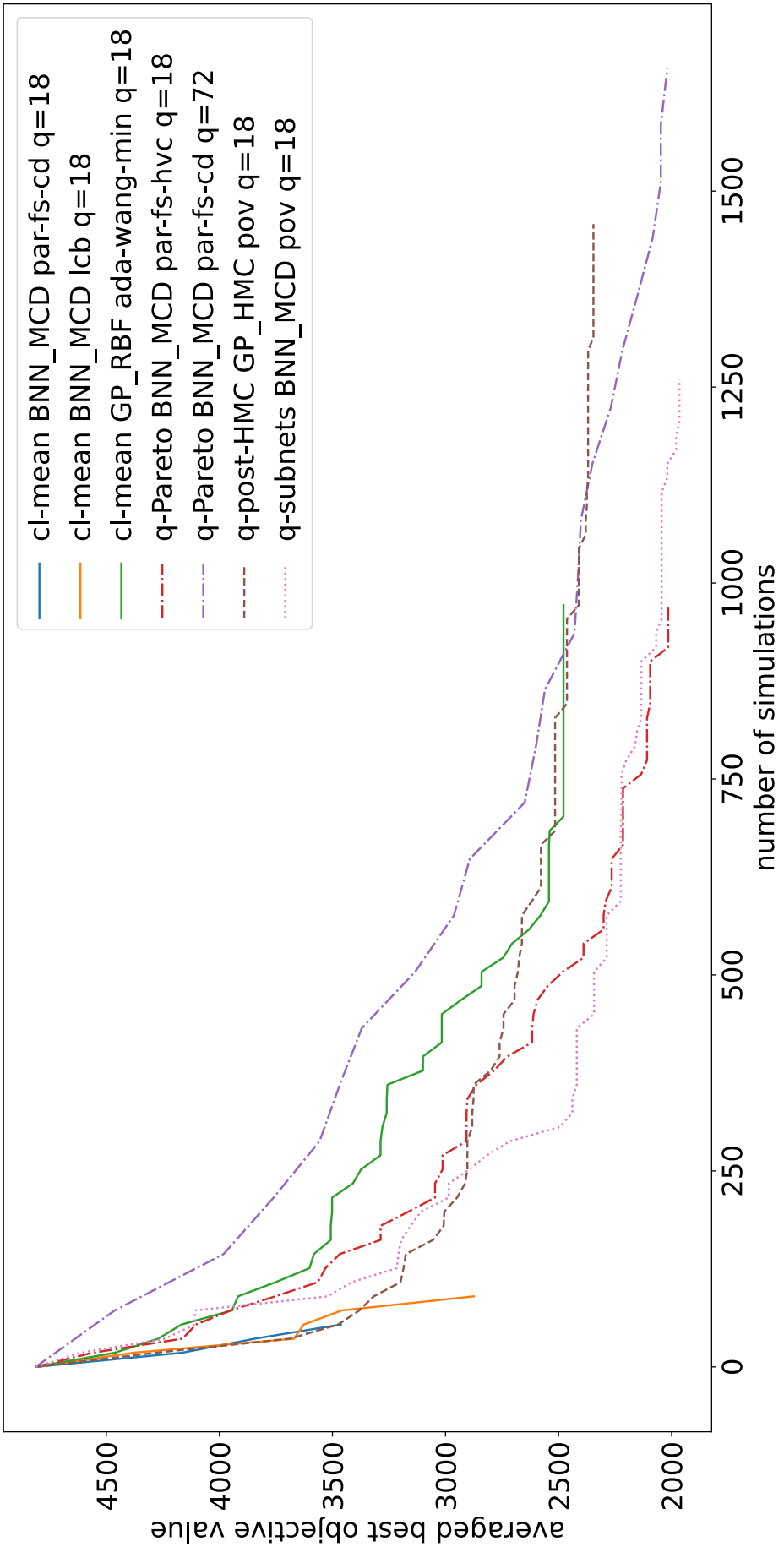

Figure 3.1: **Best P-SDAs** applied to the **Schwefel** problem. Convergence profile in terms of best objective values averaged over the 10 runs of the experiment.

Table 3.6: **P-SDAs with q=18** applied to the **benchmark problems**. The surrogates are **trained on the complete database of simulated candidates**. Average number of simulations per search and overall average training time (in seconds) for each surrogate and acquisition process. Ordering according to the average number of simulations in decreasing order from top to bottom.

Surrogate	Average number of simulations	Average training time	Surrogate	Average number of simulations	Average training time
q-EGO *cl-mean*			q-Pareto		
Schwefel					
GP_RBF	698	1.08	GP_RBF	1085	1.38
ANN_BLR	203	7.56	ANN_BLR	777	22.63
Rastrigin					
GP_RBF	702	0.78	GP_RBF	1051	1.34
ANN_BLR	204	7.56	ANN_BLR	777	22.57
Rosenbrock					
GP_RBF	703	0.74	GP_RBF	1104	2.05
ANN_BLR	204	7.56	ANN_BLR	775	22.64

Benchmark	Average number of simulations	Average training time
q-post-HMC with GP_HMC		
Schwefel	827	25.64
Rastrigin	680	36.44
Rosenbrock	703	34.79
q-post-HMC with BNN_HMC		
Schwefel	832	23.37
Rastrigin	831	23.33
Rosenbrock	840	22.97

The convergence profiles restricted to a low number of simulations offer the opportunity to compare the P-SDAs for a budget describing a very expensive problem. The convergence curves for the strategies showing the fastest improvement for less than 100 simulations are plotted in Figure 3.2 for the Schwefel function and in the appendix in Figure D.18, Figure D.19 and Figure D.20 for the remaining instances. Rastrigin apart, the best procedures consider to train the surrogate on the complete database. On Schwefel, q-post-HMC with *pov* and GP_HMC provides remarkable results for less than 30 simulations. The curves displayed for Rosenbrock in Figure D.19 look like the ones plotted in Figure D.20 for the Covid-19 problem, therefore insinuating a probable similarity of both landscapes. The best strategy on the Covid-19 problem for less than 25 simulations employs EI. In [Reh+20], it is stated that EI is to be preferred to deal with multi-modal problems and low computational budget, even for a medium or high number of decision variables.

3.5 Conclusion

In this chapter, we have inspected the construction of P-SDAs to address moderately and very expensive problems. We have come up with surrogate-optimizer couplings based on the parallel optimizations of sub-surrogates to sample q new solutions. The benefit regarding computational efficiency lies in the capping of one surrogate training per cycle. The sub-surrogates can be generated by randomly deactivating units in an Artificial Neural Network (q-subnets) or by sampling the posterior distribution over the parameters of a Bayesian model (q-post-HMC). The multiplicity of the sub-surrogates advantageously brings diversity in the set of candidates.

The new Acquisition Processes are appropriate to optimize multi-modal landscapes with weak global structure. On the one hand, the q-subnets method is the most convenient to handle a moderately expensive problem where the surrogate training is taken into account in the budget definition. The reasons stem from both the sufficient number of affordable simulations and the continuous improvement of the objective value along the search. On the other hand, for a budget characterizing a very expensive problem where the number of simulations is limited and the surrogate update is assumed to be negligible, the q-post-HMC technique is the most adequate as it rapidly converges to a local optimum. For the former budget, the numerical experiments show the pertinence of trading predictive accuracy for simulations by reducing the size of the training set. Conversely, for the latter budget setting, the effort should be invested into the surrogate building. This general guidance should be taken carefully as exceptions may arise. Finally, the integration of a committee-based Infill Criterion within q-EGO reveals interesting performances on the Covid-19 transmission control problem.

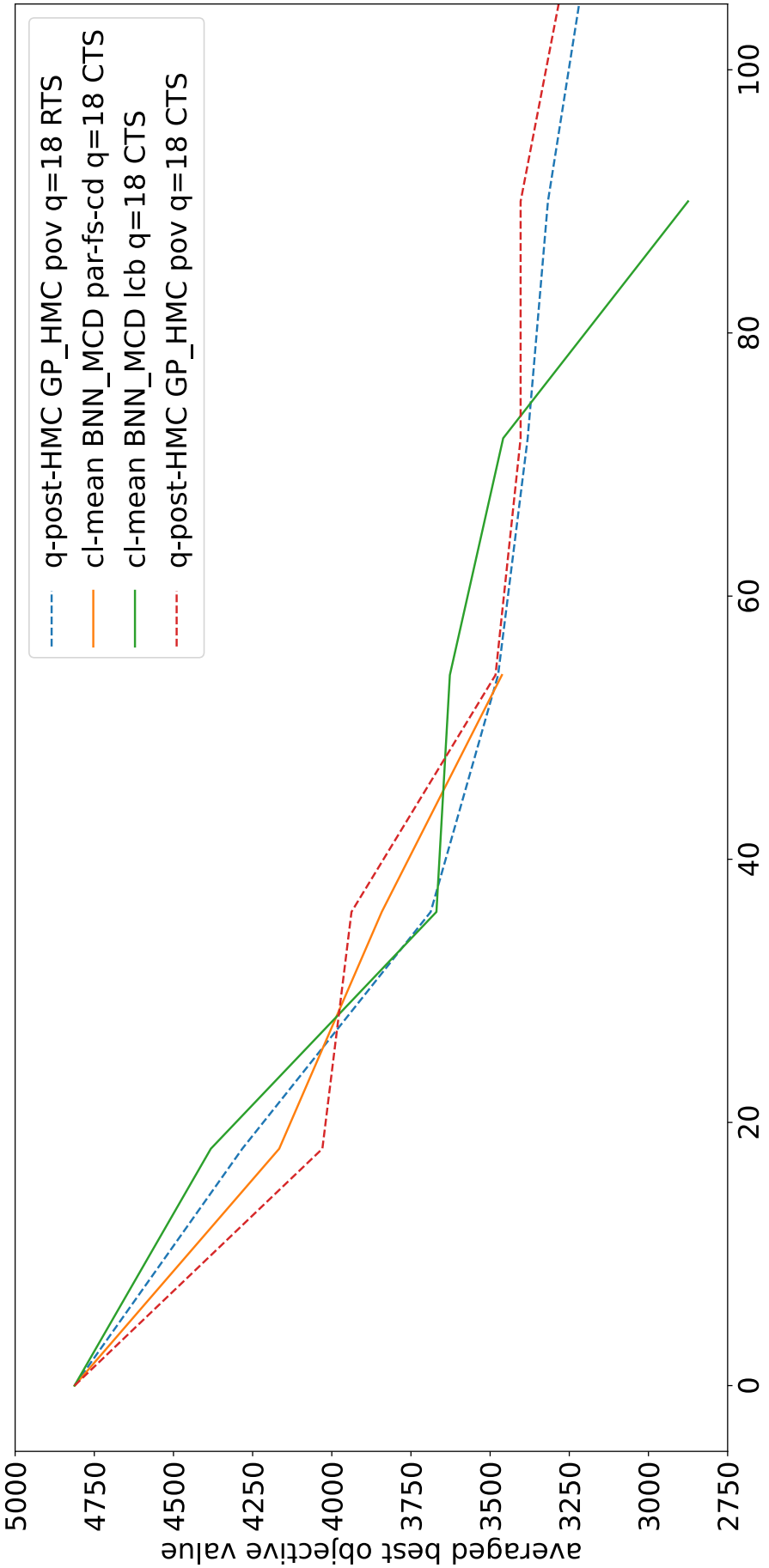

Figure 3.2: **Best P-SDAs** applied to the **Schwefel** problem. Convergence profile in terms of best objective values averaged over the 10 runs of the experiment. RTS: reduced training set. CTS: complete training set.

Chapter 4

Parallel Hybrid methods

Contents

Related publications:

- Briffoteaux Guillaume, Gobert Maxime, Ragonnet Romain, Gmys Jan, Mezmaz Mohand, Melab Nouredine, Tuyttens Daniel. "Parallel surrogate-assisted optimization: Batched Bayesian Neural Network-assisted GA versus q-EGO" in *Swarm and Evolutionary Computation*, 2020, Vol. 57, 100717, ISSN 2210-6502. https://doi.org/10.1016/j.swevo.2020.100717
- Ragonnet Romain, Briffoteaux Guillaume, Williams Bridget M., Savulescu Julian, Segal Matthew, Abayawardana Milinda, Eggo Rosalind M., Tuyttens Daniel, Melab Nouredine, Marais Ben J., McBryde Emma S., Trauer James M. "Optimising social mixing strategies to mitigate the impact of COVID-19 in six European countries: a mathematical modelling study" in medRxiv, 2020. https://doi.org/10.1101/2020.08.25.20182162

4.1 Introduction

Merging the knowledge previously gained on P-SAEAs and P-SDAs empowers one to design new hybrid methods where the acquisition process comprises both IC optimization and informed reproduction operators. The comprehension of the Covid-19 contact reduction problem is further extended in this chapter by the application of the hybrid methods and *a posteriori* landscape analysis.

A confrontation between P-SAEAs and P-SDAs is organized in Section 4.2. The computational costs of the APs and ECs are scrutinized as well as the quality of the resolution in both contexts of moderately and very expensive problems and for varying search landscapes. The comprehension acquired on the two frameworks guides the construction of hybrid acquisition processes in Section 4.3. In this chapter, we introduce two hybrid methods: Hybrid Concurrent Acquisition Processes (HCAP) and Hybrid Successive Acquisition Processes (HSAP). On the one hand, HCAP utilizes IC optimization and informed operators concurrently at each cycle. On the other hand, HSAP bets on a successive use of both APs during the search. Numerical experiments are conducted on the Covid-19-related problem including an analysis of parallel scalability and comparisons with a reference method.

Finally, in Section 4.4 the overall best solution, found by HSAP, on the Covid-19 problem is revealed and detailed. Enabled by the large amount of simulations resulting from the numerous numerical experiments, a landscape analysis is performed. The severity of the constraint and the features of the search landscape are highlighted and explain the significance of the hybrid methods on this problem.

4.2 P-SAEAs *versus* P-SDAs

The combination of multiple factors impacts the allocation of the computational budget among the tasks dedicated to simulation, training and acquisition process. This allocation and the actual algorithmic components selected to form the surrogate-based method are the key regulators of the optimization quality. The following sub-sections are intended to summarize the observations stemming from all the previous numerical experiments and to place P-SAEAs and P-SDAs face to face.

4.2.1 Computational costs

In order to describe the allocation of the computational budget, the average number of simulations per method and the average training time of the surrogates have been monitored. In Table 4.1, these indicators are gathered for all the strategies employed up to now on the Schwefel benchmark function. It has already been observed in Chapter 3 that q-EGO is the most computationally demanding AP, followed by q-Pareto, while q-subnets and q-post-HMC are less costly in this respect. We have also deduced from both the conceptual and empirical analyses that training on the complete available database increases the training time and so decreases the part of the budget dedicated to simulation.

The overview on Table 4.1 reveals that the association of a rapidly trainable surrogate and a light AP gives to SaaEF with GP_RBF the opportunity to perform the same number of simulations than the parallel EA (P-EA) without surrogate (2214). By comparing all the algorithms relying on BNN_MCD for $q = 72$ plus q-post-HMC with GP_HMC, it can be deduced that, for a similar averaged training time, SaaEF, SaaF and SaaE are the fastest AP (from 2003 to 2012 simulations) followed by q-post-HMC (1850 simulations) and q-subnets (1821 simulations). The slight difference between q-subnets

and q-post-HMC with GP_HMC is to be attributed to the predictive computational inefficiency of BNN_MCD. Reviewing the implementation of BNN_MCD with a focus on prediction efficiency is expected to rise q-subnets at the same level of q-post-HMC in terms of computational performance.

The insights extracted from Table 4.1 for $q = 72$ are recovered for $q = 18$. Increasing q only slightly subsides the number of simulations in q-EGO as each new candidate triggers one intra-cycle (potentially partial) surrogate update. This fact does not hold for the other P-SDAs where one training is executed per cycle. Reducing q in P-SAEAs would also decrease the number of simulations as the surrogate is updated once per cycle.

The various categories of ECs carry diverse degrees of computing requirements. No disparity is noted when observing the performance of P-SAEAs as the ECs are not substantially used. Conversely, they are more considerably demanded in q-EGO. The average number of affordable simulations per ECs are reported in Table 4.2 when applying q-EGO *cl-mean* on the Schwefel problem with GP_RBF and $q = 18$. The ECs based on the distance to the database are more computationally greedy than the ones utilizing the predictive standard deviation as measure of uncertainty. The calculation of the distance is realized *via* the Scipy library [Va20] and further ways should be examined to mitigate the computing time such as C implementation with Cython [Beh+11] or just-in-time compilation through the Numba library [LPS15]. The rapidity of eliciting the predictive standard deviation depends on the surrogate implementation.

As expected by their sophistication, the voting committees, the Pareto-based bi-criterion ECs and the adaptive ensembles *ada-dpf* and *ada-df* are among the most computationally expensive ECs. The random and the scalarized bi-criterion ECs are the cheapest in terms of computations just like the adaptive *ada-wang-min* and *ada-wang-max*. The dynamic ensembles are moderately expensive except for *dyn-fs-excl*, *dyn-sf-excl* and *dyn-sf-75-excl* that entail a higher number of simulations.

4.2.2 Context of moderately expensive problem

For each framework, the top-5 strategies according to the final objective values averaged over the ten runs are reported in Table 4.3. The P-SAEAs provide the best methods in the four problems, strikingly outperforming the P-EA without surrogate (right column of Table 4.3). On the real-world application to Covid-19 transmission control and on the Rosenbrock benchmark, the P-SDA q-EGO also demonstrates its reliability by meaningfully improving over P-EA. Nonetheless, the pertinence of P-SDAs is called into question on the multi-modal Schwefel and Rastrigin functions for a budget denoting a moderately expensive problem. For such a budget, it is therefore more beneficial to invest into simulations than into surrogate accuracy as the less computationally expensive APs from the P-SAEAs are the most successful.

In P-SAEAs the ECs to promote are those favoring exploration at the early stages and exploitation thereafter such as the dynamic ensembles *dyn-df-incl*, *dyn-dpf-excl* and *dyn-df-excl*. Proceeding this way lets the surrogate learn the global landscape at the beginning of the search to exploit the acquired knowledge afterwards. The Pareto-based bi-criterion *par-fd-cd* is also a pertinent option. No such a clear conclusion can be drawn in the case of P-SDAs. On Rosenbrock, *ada-wang-min* in q-EGO and the Pareto-based ECs from [Tia+19] in q-Pareto are purely exploitation-oriented. On the Covid-19 problem, *com-spf* in q-EGO and the Pareto-based *par-fs-hvc* in q-Pareto and *par-fd-cd* in q-post-HMC balance exploration and exploitation.

Table 4.1: **P-SAEAs and P-SDAs** applied to the **Schwefel** problem. Average number of simulations and average training time (TT) per search for each surrogate and acquisition process. Ordering according to the average number of simulations in decreasing order from top to bottom. RTS: reduced training set. CTS: complete training set.

Surrogate	Average number of simulations	TT	Surrogate	Average number of simulations	TT
			P-EA q=72		
			no surrogate	2214	0.0
			SaaEF q=72		
			GP_RBF (RTS)	2214	0.16
			rKRG (RTS)	2084	3.52
			BNN_MCD (CTS)	2012	6.05
			ANN_BLR (RTS)	1797	14.63
			iKRG (RTS)	1746	16.32
			SaaF q=72		
			BNN_MCD (CTS)	2005	6.42
			SaaE q=72		
			BNN_MCD (CTS)	2003	6.48
q-Pareto q=18			q-Pareto q=72		
GP_RBF (RTS)	1893	0.17	GP_RBF (RTS)	2121	0.11
GP_RBF (CTS)	1085	1.38	BNN_MCD (CTS)	1718	6.87
BNN_MCD (CTS)	1084	7.03	ANN_BLR (RTS)	1650	14.74
ANN_BLR (RTS)	958	14.31	rKRG (RTS)	1585	3.08
rKRG (RTS)	904	3.50	iKRG (RTS)	1395	16.07
ANN_BLR (CTS)	777	22.63			
iKRG (RTS)	697	16.05			
q-post-HMC q=18			q-post-HMC q=72		
GP_HMC (RTS)	1435	5.63	GP_HMC (RTS)	1850	9.93
BNN_HMC (RTS)	934	18.57	BNN_HMC (RTS)	1436	31.36
BNN_HMC (CTS)	832	23.37			
GP_HMC (CTS)	827	25.64			
q-subnets q=18			q-subnets q=72		
BNN_MCD (CTS)	1224	7.82	BNN_MCD (CTS)	1821	7.68
q-EGO q=18			q-EGO q=72		
GP_RBF (RTS)	879	0.2	GP_RBF (RTS)	862	0.13
GP_RBF (CTS)	698	1.08	rKRG (RTS)	234	1.18
rKRG (RTS)	252	1.52	ANN_BLR (RTS)	165	8.49
iKRG (RTS)	239	7.12	iKRG (RTS)	153	4.02
ANN_BLR (CTS)	203	7.56	BNN_MCD (CTS)	138	8.49
ANN_BLR (RTS)	197	8.44			
BNN_MCD (CTS)	165	6.85			

Table 4.2: **q-EGO *cl-mean*** applied to the **Schwefel** problem with GP_RBF (RTS) and $q = 18$. Average number of simulations per Evolution Control. Ordering according to the average number of simulations in decreasing order from left to right and from top to bottom.

Evolution Control	Average number of simulations	Evolution Control	Average number of simulations
rand	1234	*par-fs-cd*	909
pov	1053	*par-fs-hvc*	889
lcb	1053	*dyn-df-75-excl*	835
dyn-fs-excl	1049	*dyn-fd-excl*	810
ada-wang-max	1049	*dyn-dpf-excl*	802
dyn-sf-excl	1047	*dist*	725
stdev	1045	*dyn-fpd-excl*	720
pi	1045	*com-spf*	718
ada-wang-min	1044	*ada-df*	669
ei	1042	*par-fd-cd*	669
dyn-sf-75-excl	1042	*par-tian-fd*	667
dyn-fps-excl	977	*par-fd-hvc*	666
dyn-spf-excl	977	*ada-dpf*	522
dyn-df-excl	941	*com-dpf*	520
par-tian-fs	912		

The distance from the database of already simulated solutions is the most informative indicator of exploration in P-SAEAs. It has been hypothesized in Chapter 2 that the predictive standard deviation fails to represent the predictive uncertainty because BNN_MCD builds it approximately and the remaining surrogates (trained on a reduced training set) lose historical information. The search outcomes from the best P-SDAs on the Rosenbrock and the Covid-19 instances displayed in Table 4.3 reveal that invoking the predictive standard deviation is actually interesting. Indeed, the predictive standard deviation favors exploration either by discovering unknown regions (such as the distance does) or by scrutinizing regions possibly characterized by important fluctuations according to the surrogate. For surrogates that possess adequate predictive capacity for the landscape at hand, the predictive standard deviation is a pertinent information. This subtlety appeals to derive ECs or APs that embed both measures.

Astonishingly, training GP_RBF on a size-restricted set seems to yield better results in P-SDAs according to Table 4.3 whereas the training time is not drastically reduced (from around 0.2 second to 1 second as exposed in Table 4.1). Restraining the training set to the last simulations may boost exploitation in the area currently screened. In P-SAEAs, BNN_MCD has only been built on the entire database and GP_RBF on the batch of new simulations. The contrast of performance of both surrogates is to be put down to the predictive capacity rather than the composition of the training set as pointed out in the comparison led in Sub-section 2.4.2.

Table 4.3: **P-SAEAs *versus* P-SDAs**. Top-5 strategies for each framework according to the final objective value averaged over 10 runs. Ordering according to ascending average final objective values from top to bottom.

Surrogate	AP EC	Average	Surrogate	AP EC q	Average	Average
P-SAEA			P-SDA			P-EA
Schwefel						
BNN_MCD (CTS)	**SaaEF *dyn-df-incl***	**131.95**	BNN_MCD (CTS)	q-subnets *pov* 18	1965.45	607.91
BNN_MCD (CTS)	SaaEF *dyn-dpf-excl*	136.3	BNN_MCD (CTS)	q-Pareto *par-fs-hvc* 18	2015.06	
BNN_MCD (CTS)	SaaF *dyn-df-incl*	153.92	BNN_MCD (CTS)	q-Pareto *par-fs-cd* 72	2019.45	
BNN_MCD (CTS)	SaaEF *par-fd-cd*	167.55	BNN_MCD (CTS)	q-Pareto *par-fs-hvc* 72	2105.19	
BNN_MCD (CTS)	SaaEF *dyn-df-excl*	168.64	BNN_MCD (CTS)	q-Pareto *par-fs-cd* 18	2192.29	
Rastrigin						
GP_RBF (RTS)	**SaaE *par-fd-cd***	**18.22**	rKRG_18 (RTS)	q-Pareto *par-tian-fd* 18	77.83	23.30
GP_RBF (RTS)	SaaE *dyn-df-excl*	18.69	GP_RBF (CTS)	*cl-mean lcb* 18	84.66	
GP_RBF (RTS)	SaaEF *dyn-dpf-excl*	19.08	GP_RBF (RTS)	*cl-mean ada-wang-min* 18	90.59	
GP_RBF (RTS)	SaaE *com-dpf*	19.25	GP_RBF (CTS)	*cl-mean bp* 18	90.82	
GP_RBF (RTS)	SaaF *com-dpf*	19.34	GP_RBF (RTS)	*cl-mean par-tian-fs* 18	92.92	
Rosenbrock						
GP_RBF (RTS)	**SaaF *par-fd-cd***	**137.82**	GP_RBF (RTS)	*sb ada-wang-min* 18	472.02	1191.14
GP_RBF (RTS)	SaaF *com-dpf*	156.63	rKRG_18 (RTS)	q-Pareto *par-tian-fd* 18	572.57	
GP_RBF (RTS)	SaaF *par-fd-hvc*	203.27	GP_RBF (RTS)	q-Pareto *par-tian-fs* 72	639.77	
GP_RBF (RTS)	SaaF *ei*	232.37	GP_RBF (RTS)	q-Pareto *par-tian-fs* 18	713.99	
GP_RBF (RTS)	SaaEF *par-fd-hvc*	233.93	GP_RBF (RTS)	q-Pareto *par-tian-fd* 72	940.72	
Covid-19						
BNN_MCD (CTS)	**SaaF *dyn-df-incl***	**6854**	GP_RBF (RTS)	*cl-mean com-spf* 18	7824	21483
BNN_MCD (CTS)	SaaF *dyn-df-excl*	7115	GP_RBF (CTS)	q-Pareto *par-fs-hvc* 18	8298	
BNN_MCD (CTS)	SaaEF *dyn-df-75-excl*	7455	GP_HMC (RTS)	q-post-HMC *par-fd-cd* 72	8648	
BNN_MCD (CTS)	SaaEF *dyn-df-excl*	7679	GP_HMC (RTS)	q-post-HMC *par-fd-cd* 18	8778	
BNN_MCD (CTS)	SaaEF *dist*	7837	GP_RBF (CTS)	q-Pareto *com-spf* 18	8897	

4.2.3 Convergence profiles

The convergence profiles of the best approaches are outlined in Figure 4.1 for Schwefel and in Appendix E for the other instances. The appropriateness of the P-SDAs is demonstrated for all the problem instances when the number of simulations is more severely capped than it has been in our budget definition. More precisely, for a budget only defined by a limited number of simulations, P-SDAs are the most rewarding methods. After a threshold specific to each search landscape, the P-SAEAs overcome the P-SDAs. It is particularly flagrant on the multi-modal Schwefel and Rastrigin landscapes in Figure 4.1 and Figure E.1 respectively. For Schwefel, the threshold lies in $[300; 500]$ while a more extended range of $[300; 700]$ is observed for Rastrigin. The phenomena is less apparent on the Rosenbrock and Covid-19 problems in Figure E.2 and Figure E.3 respectively. The threshold appears to be around 1500 simulations but considering a larger budget would allow to be more rigorous as most of the P-SDAs curves stopped before the intersection point.

The convergence plots confirm that among P-SDAs no one AP is superior than all the others in all the scenarios. Even if q-EGO shows superiority over its counterparts in three cases out of four for few simulations, q-post-HMC and q-subnets are to be put forward on the Schwefel problem. Analogously, the problem dependency hinders the choice for a unique convenient EC in P-SDAs even though *ada-wang-min* and the Pareto-based bi-criterion ECs based on the predictive standard deviation are recurrently among the most successful components.

4.3 Hybrid Acquisition Processes

The two categories of P-SBOAs, namely P-SAEAs and P-SDAs, are attractive for different budgets or landscapes. In this section, we attempt to retain the best of both classes by investigating the design of hybrid APs. The generation of new candidates is envisioned *via* both IC optimization and reproduction operators.

4.3.1 Hybrid Informed Operator and Infill Criterion-based Acquisition Processes

Two APs are combined into two novel optimization algorithms. The first AP is a *Constant Liar* with the voting committee EC *com-spf* and the GP_RBF surrogate model. The second AP is inspired by P-SAEA, where a BNN_MCD surrogate is only used as a filter to discard unpromising candidates (SaaF). Both APs are the most adequate for each framework on the Covid-19 problem as identified in the preceding chapters.

The first new hybrid method is named HCAP for "Hybrid Concurrent Acquisition Process" and is presented in Algorithm 11. The two aforementioned APs are executed concurrently at each cycle to propose new candidates that are subsequently simulated in parallel. The algorithm starts by a search space sampling *via* LHS and the evaluation of the initial candidates (line 1). The surrogates are created and the population is initialized (lines 2 to 4). At the beginning of a cycle, the first AP generates $q_1 = 9$ new promising candidates (line 6). The *Constant Liar* AP is described thoroughly in Algorithm 7. Thence, parents are selected from the population and reproduced to create a batch $\mathcal{P}_c$ of $n_{chld} = 288$ children (lines 7 and 8). From $\mathcal{P}_c$, the $q_2 = 63$ more promising candidates are retained and the remaining $n_{disc} = 225$ candidates are discarded (line 9). A total of $q_1 + q_2 = 72$ new candidates are simulated in parallel at each cycle (lines 10 and 11). Thereafter, the surrogates are updated (lines 13 and 14) and a new population is formed by elitist replacement (line 15).

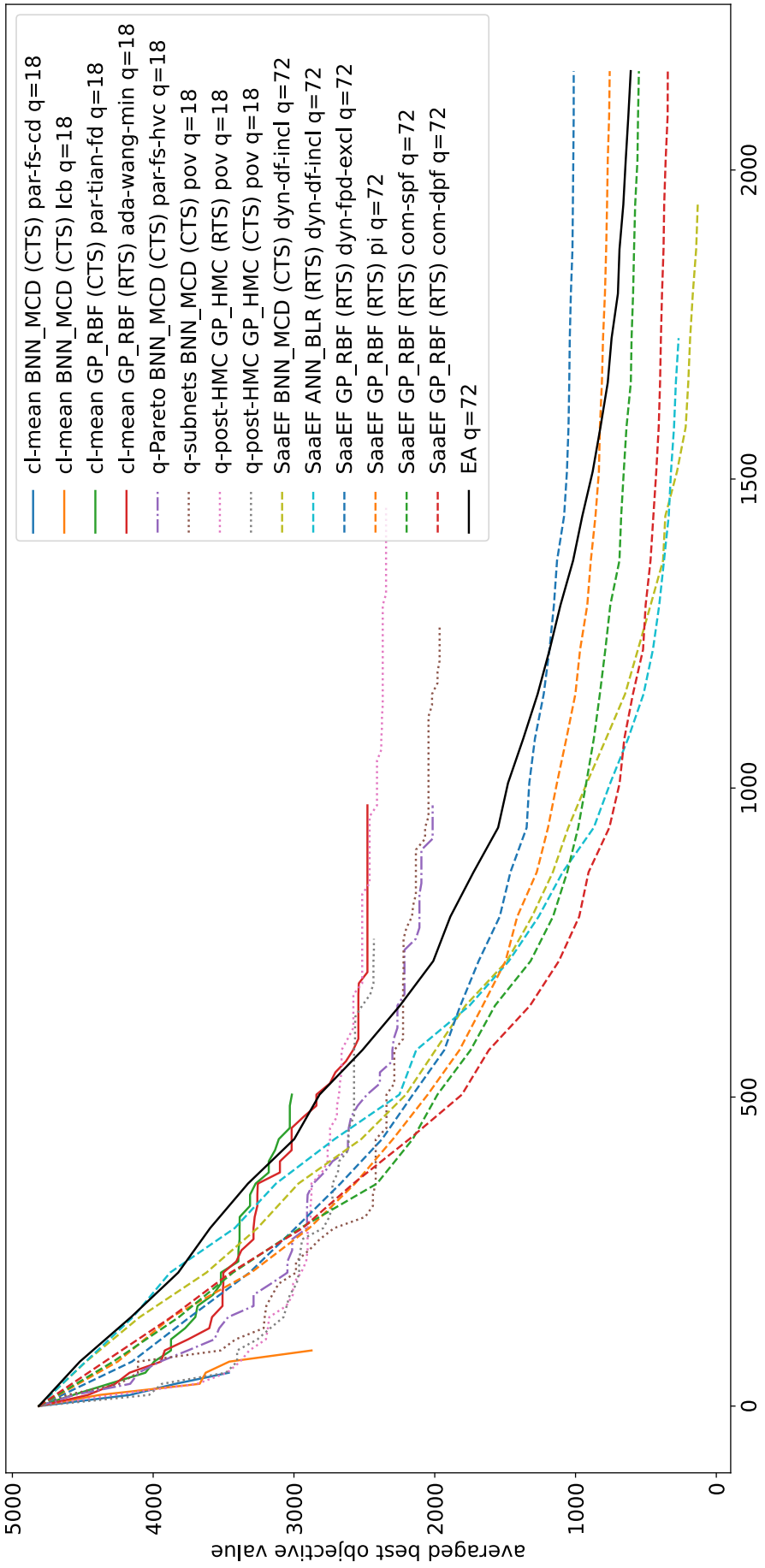

Figure 4.1: **P-SAEAs *versus* P-SDAs** application to the **Schwefel** problem. Convergence profile in terms of best objective values averaged over the 10 runs of the experiment.

Algorithm 11 Framework of HCAP.

Input

simulator: real objective function
budget: computational budget for the search
GP_RBF: surrogate model for AP1
com-spf: evolution control for AP1
$q_1 = 9$: number of candidates to simulate per cycle for AP1
$n_{pop1} = 50$: population size for AP1
$n_{gen} = 100$: number of generations for AP1
BNN_MCD: surrogate model for AP2
EC: evolution control for AP2
$n_{pop2} = 72$: population size for AP2
$n_{chld} = 288$: number of new candidates issued per cycle for AP2
$q_2 = 63$: number of candidates to simulate per cycle for AP2
$n_{disc} = 225$: number of discarding per cycle for AP2

1: *database* ← LHS+parallel_simulations(*simulator*, n_{pop2})
2: GP_RBF ← training(*database*)
3: BNN_MCD ← training(*database*)
4: $\mathcal{P}$ ← *database* ▷ initial population
5: **while** *budget*$\neq 0$ **do**
6: $\mathcal{B}_{sim1}$ ← Constant_Liar_AP(*database*, *com-spf*, GP_RBF, q_1, n_{pop1}, n_{gen}) ▷ Algorithm 7
7: $\mathcal{P}_p$ ← selection($\mathcal{P}$, n_{chld}) ▷ population of parents
8: $\mathcal{P}_c$ ← reproduction($\mathcal{P}_p$, n_{chld}) ▷ population of children
9: $\mathcal{B}_{sim2}$ ← filtering($\mathcal{P}_c$, *dyn-df-75-excl*, BNN_MCD, q_2, n_{disc})
10: $\mathcal{B}_{sim} \leftarrow \mathcal{B}_{sim1} \cup \mathcal{B}_{sim2}$
11: parallel_simulation(*simulator*, $\mathcal{B}_{sim}$)
12: *database* ← *database* $\cup$ $\mathcal{B}_{sim}$
13: GP_RBF ← training(*database*, 72)
14: BNN_MCD ← training(*database*, all)
15: $\mathcal{P}$ ← elitist_replacement($\mathcal{P}$, $\mathcal{B}_{sim}$, n_{pop2})
16: *budget* ← get_remaining_budget(*budget*, elapsed_time)
17: **end while**
18: ($\boldsymbol{x}_{min}$, y_{min}) ← get_best_cost(*database*)
19: **return** $\boldsymbol{x}_{min}$, y_{min}

The analysis led in Section 4.2 indicates that P-SDAs are relevant for few objective function evaluations and P-SAEAs to deal with moderately expensive problems. This conclusion appeals to design another hybrid method that would execute successively an AP based on IC optimization and an AP relying on evolutionary computations. The novel method is referred to as HSAP for "Hybrid Successive Acquisition Processes" and is detailed in Algorithm 12. The first stage consists of running 6 cycles of q-EGO *cl-mean* with GP_RBF and *com-spf* for $q = 18$ thus corresponding to 108 simulations (lines 2 to 11). Afterwards, P-SAEA is run with reproduction operators informed by BNN_MCD through an EC until the budget is totally consumed (lines 12 to 24). The population is initialized by taking a special care of balancing between exploration and exploitation. To foster exploitation, the 10 best candidates identified so far are included in the initial population (line 12). To boost exploration, a K-Means algorithm [Scu10; AV07] partitions the set of decision vectors from the database into 62 groups and one randomly-selected solution per cluster is added to the initial population (line 13).

K-Means is a clustering algorithm whose purpose is to minimize the "inertia" defined by:

$$\sum_{i=1}^{n_{db}} \min_{\boldsymbol{c}_{\boldsymbol{km}}^{(j)} \in \mathcal{C}} ||\boldsymbol{x}^{(i)} - \boldsymbol{c}_{\boldsymbol{km}}^{(j)}||^2 \tag{4.1}$$

where n_{db} is the number of decision vectors $\boldsymbol{x}^{(i)}$ in the database and $\mathcal{C}$ is the set of the centroids $\boldsymbol{c}_{\boldsymbol{km}}^{(j)}$ of the clusters. The number of clusters is fixed by the user and the associated centroid of each cluster is the average of the cluster's members. At the beginning, the centroids are chosen among the database and each decision vector is attached to its nearest centroid. Afterwards, an iterative process updates the centroids by computing the per-cluster average and re-affect the decision vectors. The algorithm stops when the centroids do not move significantly. K-Means is recognized as an efficient clustering tool as it scales well to large datasets and always converges providing enough time [Scu10; AV07].

The AP emphasized in [Liu+17] outputs $q = 4$ candidates by optimizing $q = 4$ ICs independently. The unique surrogate update per AP reduces the computing effort induced by q-EGO. However, the low value attributed to q points the difficulty of conserving a relevant degree of diversity when numerous new candidates are sampled at once. In [Reh+18], the authors devise a hybrid strategy, namely Surrogate Model Based Optimization + Evolutionary Algorithm (SMBO+EA), whose AP relies on both IC optimizations and reproduction operators. Given n_{cores} available computing cores, the maximization of EI and the minimization of the POV yield one new candidate each and the reproduction operators provide the remaining $n_{cores} - 2$ solutions. Consequently, n_{cores} parallel simulations are run per cycle. Numerical experiments are conducted on the Rastrigin benchmark problem with 15 decision variables and on a discrete real-world engineering problem with 49 design variables and a simulation time of several seconds. The budget for the search varies between 150 and 750 real evaluations and a Kriging model is employed as surrogate. The resulting empirical analysis proves the superiority of the hybrid method compared to some state-of-the-art P-SDAs.

SMBO+EA is reproduced in this thesis to compete with HCAP and HSAP on the Covid-19 application. Its implementation is dissected in Algorithm 21 in Appendix E for $n_{cores} = 18$. The GP_RBF surrogate model replaces the Kriging model as this latter has not been previously relevant in the problem at hand. The algorithm starts by initializing the database and building the surrogate (lines 1 to 3). A cycle consists in running the three APs in parallel. The first AP, executed on one computing core, maximizes EI to produce a new candidate that is simulated (lines 7 and 8). The second AP minimizes the POV (lines 10 and 11). The third AP generates $q = 16$ new candidates *via* reproduction of 16 parents extracted from the current population (lines 13 and 14). The 16 new candidates

Algorithm 12 Framework of HSAP.

Input

simulator: real objective function
budget: computational budget for the search
GP_RBF: surrogate model for AP1
com-spf: evolution control for AP1
$q_1 = 18$: number of candidates to simulate per cycle for AP1
$n_{pop1} = 50$: population size for AP1
$n_{gen} = 100$: number of generations for AP1
BNN_MCD: surrogate model for AP2
EC: evolution control for AP2
$n_{pop2} = 72$: population size for AP2
$n_{chld} = 288$: number of new candidates issued per cycle for AP2
$q_2 = 72$: number of candidates to simulate per cycle for AP2
$n_{disc} = 216$: number of discarding per cycle for AP2

database ← LHS+parallel_simulations(*simulator*, n_{pop2})
GP_RBF ← training(*database*)
counter=0
while *counter*< 6 AND *budget*≠ 0 **do**
    $\mathcal{B}_{sim}$ ← Constant_Liar_AP(*database*, *com-spf*, GP_RBF, q_1, n_{pop1}, n_{gen}) ▷ Algorithm 7
    parallel_simulation(*simulator*, $\mathcal{B}_{sim}$)
    database ← *database* ∪ $\mathcal{B}_{sim}$
    GP_RBF ← training(*database*)
    budget ← get_remaining_budget(*budget*, elapsed_time)
    counter=*counter*+1
end while
$\mathcal{P}$ ← get_best(*database*, 10) ▷ initial population
$\mathcal{P}$ ← $\mathcal{P}$∪ K-Means_sampling(*database*, 62)
BNN_MCD ← training(*database*)
while *budget*≠ 0 **do**
    $\mathcal{P}_p$ ← selection($\mathcal{P}$, n_{chld}) ▷ population of parents
    $\mathcal{P}_c$ ← reproduction($\mathcal{P}_p$, n_{chld}) ▷ population of children
    $\mathcal{B}_{sim}$ ← filtering($\mathcal{P}_c$, *EC*, BNN_MCD, q_2, n_{disc})
    parallel_simulation(*simulator*, $\mathcal{B}_{sim}$)
    database ← *database* ∪ $\mathcal{B}_{sim}$
    BNN_MCD ← training(*database*, all)
    $\mathcal{P}$ ← elitist_replacement($\mathcal{P}$, $\mathcal{B}_{sim}$, n_{pop2})
    budget ← get_remaining_budget(*budget*, elapsed_time)
end while
($\boldsymbol{x}_{min}$, y_{min}) ← get_best(*database*, 1)
return $\boldsymbol{x}_{min}$, y_{min}

are simulated in parallel on 16 cores. Once the simulations completed, the master computing core retrieves all the information newly acquired to update the database (lines 17 and 18). The surrogate and the population are finally updated (lines 19 and 20) and the cycle is repeated until the computational budget is wasted.

In SMBO+EA, no EC is used in the AP based on the reproduction operators whereas a dynamic ensemble of ECs or a Pareto-based EC helps to discard unpromising candidates in HCAP and HSAP. Relying on an EC at this step gives more opportunity to the reproduction operators to generate good candidates. The objective pointed out in [Reh+18] for future works is to improve the performance of the method when n_{cores} increases. Indeed, in the experiments reported in [Reh+18], SMBO+EA performs similarly to P-EA (without surrogate) for $n_{cores} = 15$. In HCAP and HSAP, the use of two surrogates from different types aims at enhancing diversification in the batch of new samples and improving the overall performance of the hybrid methods. In SMBO+EA, the three APs are performed in parallel while the two APs from HCAP are performed sequentially thus giving a slight advantage to SMBO+EA regarding idleness of computing cores.

4.3.2 Experiments on Covid-19 contact reduction

The experimental protocol is the same as the one from Sub-section 2.5.3 and Sub-section 3.4.3 to allow ones to compare the new hybrid methods with the best established P-SAEAs and P-SDAs.

The GP_RBF is trained on a controlled-size set in HCAP whereas the whole database is used in HSAP and SMBO+EA. In both HCAP and HSAP, the EC is either *dyn-df-excl*, *dyn-df-incl* or *par-fd-cd*. The BNN_MCD is always updated thanks to all the simulations performed so far. A total of seven parallel hybrid methods are applied 10 independent times, resulting in a total of 70 new searches.

Figure 4.2 shows the distribution of the 10 best objective values obtained at the end of the search for the hybrid strategies, the best P-SAEAs and P-SDAs revealed previously and the P-EA. The corresponding ranking according to the average final objective value is displayed in Table 4.4. It can be observed that the new hybrid method HSAP with *dyn-df-incl* significantly outperforms the best method known so far (SaaF BNN_MCD *dyn-df-incl*) by decreasing the average objective from 6,854 to 4,178. The average, median and variance of the results are all improved when employing HSAP as shown in Figure 4.2. During the second phase in HSAP, when candidates are generated by informed operators, it is better to use both the exploitation-oriented measure *pov* and the exploration metric *dist* at each cycle. Indeed, the outcomes are slightly less interesting when using the dynamic exclusive ensemble of ECs. More generally, harnessing the *dyn-df-incl* EC is the best choice in HSAP, HCAP and SaaF according to Table 4.4. The concurrent combination of APs proposed by HCAP is also a reliable strategy as, it outperforms all the non-hybrid methods and SMBO+EA as displayed in Figure 4.2 and Table 4.4. It can be noticed that SMBO+EA behaves as expected as it produces results similar to the P-EA without surrogate.

The convergence profile for the parallel hybrid strategies, the best P-SAEAs and P-SDAs as well as the P-EA are displayed in Figure E.4. Expectedly, HSAP and the P-SDAs exhibit a similar very steep curve for less than 108 simulations. After the AP switch in HSAP, the improvement is slowed down but a continuous progress is noted until around 600 simulations where the convergence is almost reached. Figure 4.3 displays a zoom that highlights the benefit from using HSAP with *dyn-df-incl* over *cl-mean* with GP_RBF trained on a reduced training set (RTS) from 300 simulations. Firstly, HSAP

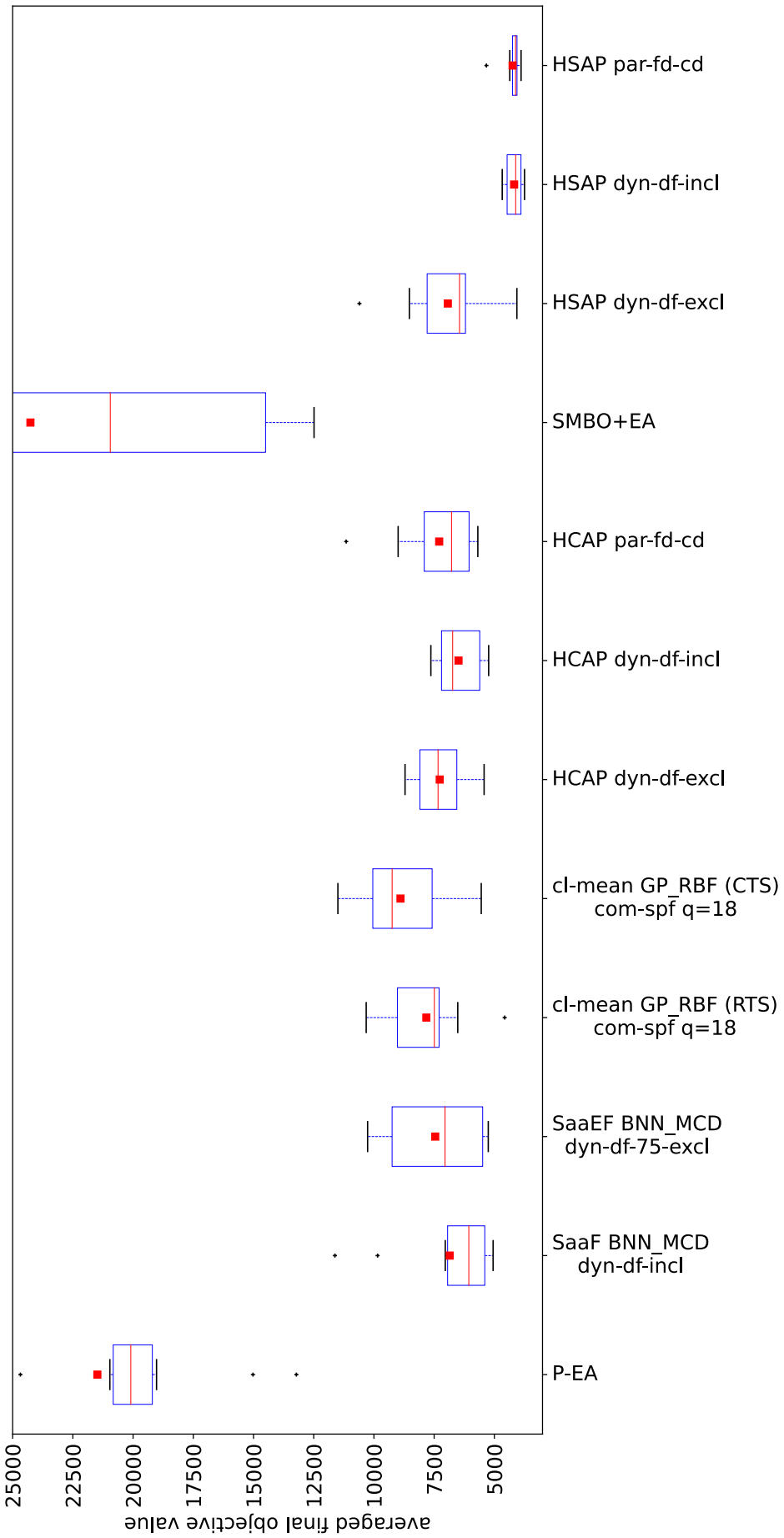

Figure 4.2: **Hybrid strategies** on the **Covid-19 contact reduction problem**. Distribution of the best objective values from the 10 runs of the experiment. Averaged values are depicted by red squares, median values by red dashes and variance information is given by the length of the boxes.

Table 4.4: **Parallel Hybrid methods**. Ranking of the best strategies according to the final objective value averaged over 10 runs. Ordering according to ascending average from top to bottom.

Strategy	Average
HSAP *dyn-df-incl*	4,178
HSAP *par-fd-cd*	4,241
HCAP *dyn-df-incl*	6,487
SaaF BNN_MCD *dyn-df-incl*	6,854
HSAP *dyn-df-excl*	6,931
HCAP *dyn-df-excl*	7,268
HCAP *par-fd-cd*	7,290
SaaEF BNN_MCD *dyn-df-75-excl*	7,455
cl-mean GP_RBF (RTS) *com-spf* q=18	7,824
cl-mean GP_RBF (CTS) *com-spf* q=18	8,897
P-EA	21,483
SMBO+EA	24,262

allows one to perform more simulations than *cl-mean* as indicates the length of the curves in Figure E.4. Secondly, the use of the informed operators enable a continuous improvement as soon as the IC-based AP has reached steady state. An attractive enhancement of HSAP would be to automatically detect the flatness in the convergence curve and trigger the AP switch. As we have already seen when designing ensemble of ECs, such a mechanism is not trivial to design, particularly because user-defined parameters must be avoided. However, exploiting the gradient of the curve is a potential lead that we plan to investigate in the future. HCAP outperforms SMBO+EA, SaaF and SaaEF in Figure E.4 while SaaF and SaaEF overtake SMBO+EA from 260 simulations. The bad performances of P-EA stressed by Figure E.4 demonstrate again the profit brought by surrogate models for both moderately and very expensive problems.

The length of the curves in Figure E.4 yields indications about the computational cost of the methods. Among the hybrid methods, SMBO+EA is the more computationally costly as the surrogate is trained on the entire database and ICs optimizations are run at each cycle. By reducing the training set size as in HCAP, more simulations are enabled and by reducing the computational effort dedicated to ICs optimization as in HSAP, the number of simulations gets closer to the one of P-SAEAs. A possible way to relieve the computational cost of HCAP would be to execute both APs in parallel.

4.3.3 Parallel scalability

We now study the behavior of the P-SBOAs when the number of computing cores n_{cores} is varied while the overall time granted to the search is maintained to 30 minutes. The way to fix the number of simulations per cycle q is not unique and may depend on n_{cores}. In a first set of experiments, the number of computing resources is decreased without modification of q with respect to the calibration adopted up to now. In a second set of experiments, the number of simulations per cycle is adjusted to fit the number of computing cores at hand ($q = n_{cores}$) therefore allowing to increase n_{cores}.

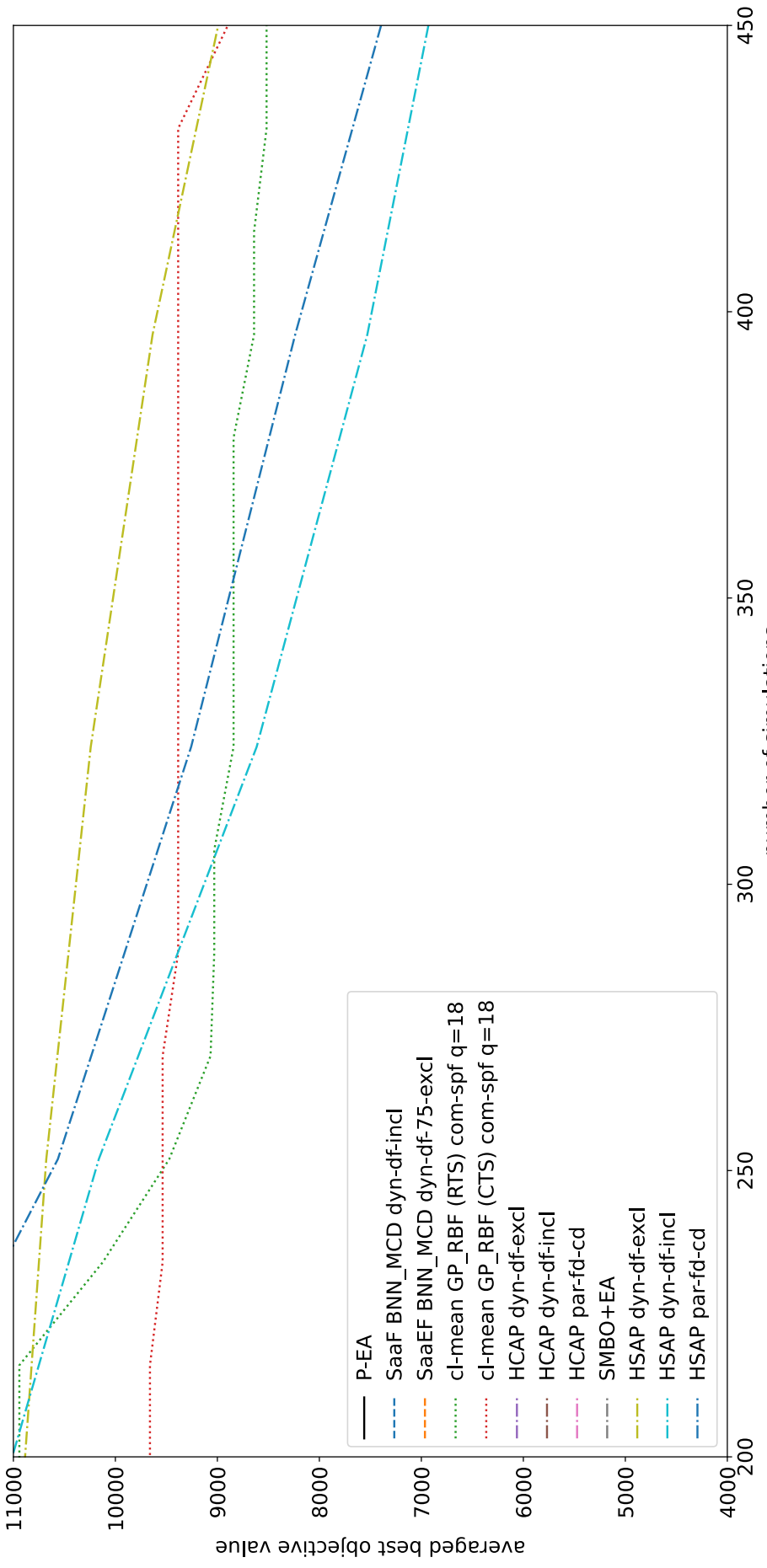

Figure 4.3: **Parallel Hybrid methods** applied to the **Covid-19 contact reduction** problem. Convergence profile in terms of best objective values averaged over the 10 runs of the experiment.

Table 4.5: **Parallel scalability (unaltered values for q).** Best objective values averaged over the 10 runs of the experiment for different numbers of available computing cores and without any modification of the algorithms.

Method \ n_{cores}	18	9	6	3	2
HSAP ($q_1 = 18; q_2 = 72$)	**4,178**	4,273	4,769	5,690	7,619
HCAP $q = 72$	6,487	6,849	8,663	13,890	20,438
SaaF $q = 72$	6,854	8,950	11,633	28,115	37,525
SaaEF $q = 72$	7,455	12,280	13,879	17,747	25,469
q-EGO (RTS) $q = 18$	7,824	8,040	8,712	8,063	13,705
q-EGO (CTS) $q = 18$	8,897	9,582	8,292	10,357	8,097
P-EA $q = 72$	21,483	17,514	20,256	32,438	46,719

In the first set of experiments, by setting $n_{cores} \in \{2, 3, 6, 9, 18\}$ the idleness of the computing units is completely avoided as q has been previously set to either 18 or 72 simulations per cycle. However, the realization of one cycle is more time-demanding which consequently shrinks the affordable number of simulations. The strategies retained for this analysis are those demonstrating the most interesting performances per category on the Covid-19 application:

- P-EA $q = 72$
- SaaF BNN_MCD *dyn-df-incl* $q = 72$
- SaaEF BNN_MCD *dyn-df-75-excl* $q = 72$
- q-EGO *cl-mean* GP_RBF (RTS) *com-spf* $q = 18$
- q-EGO *cl-mean* GP_RBF (CTS) *com-spf* $q = 18$
- HCAP *dyn-df-incl* $q = q_1 + q_2 = 9 + 63 = 72$
- HSAP *dyn-df-incl* $q_1 = 18$ $q_2 = 72$

The averaged best objective values reaped at the end of the experiments are gathered in Table 4.5. According to Table 4.5, HSAP is the best method for every value of n_{cores} and using three computing units is enough to outperform the competing approaches. Increasing the amount of energy invested to run the search, by means of raising the number of active computing cores, allows one to enhance the quality of the resolution for HSAP, HCAP, SaaF and SaaEF. Nonetheless, it does not seem to be always the case for q-EGO and P-EA. This latter observation should be attributed to the variability entailed by the aleatoric features of the approaches as the variation in n_{cores} does not trigger any modification of the algorithm. Besides, the reduction in affordable simulations induced by the reduction of the computing resources is not important enough to prematurely cut off the convergence of q-EGO as reflected by the convergence profiles in Figure E.5 in Section E of the appendix. In every case, the benefit of augmenting n_{cores} reaches a ceiling at 18 or 72 because q is fixed to either 18 or 72 in the considered methods.

In the second set of experiments, it is proposed to set q according to n_{cores} and to leverage higher number of cores $n_{cores} \in \{3, 9, 18, 72, 144\}$. For the APs based on IC optimization, the number of proposals per iteration is assumed to match the number of computing cores $q = n_{cores}$. In this situation, the challenge of maintaining diversity in the batch of newly candidates arises when the number of computing cores is high. When n_{cores}

Table 4.6: **Parallel scalability ($q = n_{cores}$).** Best objective values averaged over the 10 runs of the experiment for different numbers of available computing cores. The number of simulations per cycle q is fixed to n_{cores} for all the approaches.

Method \ n_{cores}	144	72	18	9	3
HSAP	**3,791**	3,865	4,617	7,118	8,950
HCAP	5,487	6,100	7,767	11,275	17,658
SaaF	6,463	7,728	6,779	9,951	21,641
q-EGO (RTS)	13,894	9,443	8,416	8,234	7,624
SaaEF	7,777	7,916	14,653	11,029	17,363
q-EGO (CTS)	12,820	8,981	8,822	9,579	9,954
P-EA	7,766	9,604	23,875	25,153	29,355
SMBOEA	18,011	17,892	22,865	29,479	31,613

is low, more computational efforts are engaged into full surrogate trainings thus limiting the affordable simulations but bringing the advantage of a more accurate surrogate. For the APs relying on reproduction operators, $q = n_{cores}$ is also applied. It is worth noting that this is not the usual practice as q has been set to $4.n_{cores}$ hitherto. The population size $n_{pop} = 72$ and the number of children $n_{chld} = 288$ are kept unaltered in P-SAEAs. In SaaEF, the number of predictions is set to $n_{pred} = 144 - n_{cores}$ while $n_{disc} = 144$ is unchanged. Proceeding in this way allows one to observe the effect of the proportion of predictions at the replacement step. In SaaF, the number of discardings per cycle is set to $n_{disc} = 288 - n_{cores}$. In HCAP, the following triplets are considered: $(n_{cores}, q_1, q_2) = (3, 1, 2), (9, 1, 8), (18, 2, 16), (72, 9, 63), (144, 18, 126)$. In P-EA, the number of children is set to the number of computing cores while the population size is not modified. Finally, the hybrid SMBO+EA from [Reh+18] is also included in the comparison.

The best results obtained in this thesis on the Covid-19 problem are provided by HSAP with $n_{cores} = 144$ according to Table 4.6. The excellent parallel scalability of HSAP and HCAP is reflected by the enhancement of the quality of the resolutions when n_{cores} increases as it is graphically illustrated in Figure E.7 and numerically exhibited in Table 4.6. For HSAP and $n_{cores} \leqslant 18$, the comparison between Table 4.5 and Table 4.6 commends to preserve the previous calibration $q = 72$ rather than setting the number of simulations per cycle to the number of available computing cores. If this recommendation is respected, for every values assumed for n_{cores} in these experiments, HSAP yields the best results in terms of mean, median and variance as reported in Table 4.5 and the distribution of the 10 final best objective values plotted in Figure E.6.

For q-EGO with *cl-mean*, the difficulty of maintaining diversity is highlighted by the box-plots of Figure E.7. For HCAP and *cl-mean* considering the complete database as training set, the runs carried out with $n_{cores} = 72$ output better decisions than the ones where $n_{cores} = 144$. More impressively, the performance of *cl-mean* with a reduced training set deteriorates as n_{cores} increases. Adding too much new solutions per cycle lead to propose unpromising candidates which wastes the budget by investing too much in surrogate training as one surrogate (partial) training is realized to obtain one new candidate. Figure E.7 indicates the satisfying parallel scalability of SaaF, SaaEF and P-EA. In SaaEF, the proportion of predicted solutions involved at the replacement step is lowered down when n_{cores} increases, thus enabling to mitigate the possible predictive inaccuracy. Future efforts should be engaged to study the resorting of surrogate error to set this proportion. The parallel scalability of SMBO+EA is good according to Figure E.7 but the associated performances are the worst among the investigated strategies as shown in Figure E.6 and Table 4.6.

Table 4.7: **Parallel scalability ($q = n_{cores}$).** Average number of simulations per search over the 10 runs of the experiment for different numbers of available computing cores. The number of simulations per cycle q is fixed to n_{cores} for all the approaches.

Method \ n_{cores}	144	72	18	9	3
P-EA	22,377	11,347	2,953	1,739	751
SaaF	8,654	5,522	1,818	1,134	483
SaaEF	8,481	4,975	1,792	1,121	488
HSAP	7,819	2,628	1,260	720	354
SMBOEA	3,960	3,355	1,324	1,079	650
HCAP	3,657	2,894	1,407	976	426
q-EGO (RTS)	792	770	673	639	491
q-EGO (CTS)	504	576	527	524	450

The number of affordable simulations increases when the number of computing cores increases for all the approaches except q-EGO with the complete training set (CTS) as Table 4.7 reveals. Because the surrogate is trained on the complete database, the last AP of q-EGO with CTS is not totally accomplished when $n_{cores} = 144$, therefore explaining the decrease in simulations compared to $n_{cores} \in \{9, 18, 72\}$. The computational expensiveness of the AP in q-EGO is directly related to q so these strategies present the lowest increase as indicated by Table 4.7. For $n_{cores} = 3$, SMBO+EA enables the highest number of simulations (650) among the surrogate-based approaches because both the generation and the simulation of the candidates are performed in parallel. This is not true anymore when n_{cores} grows as the production of the $n_{cores} - 2$ new decision vectors by reproduction is executed by a unique computing core. The number of simulations is high for the P-SAEAs and HSAP when $n_{cores} = 144$ as the participation of the AP based on IC optimization (*cl-mean*) is null or restricted.

In the context of a very expensive problem, maximizing the benefits of high number of computing cores is challenging as depicted by the convergence curves in Figure 4.4. For an AP built on an evolving population, running multiple generations appears more adequate even if a lot of candidates are only predicted as in SaaEF. For an AP set up on IC optimization, the lack of diversity in large new batches and the associated reduction of number of cycles imply a waste of the computational budget. In this configuration where the surrogate training is neglected, it is more convenient to employ q-EGO or HSAP with a curbed number of computing cores.

4.4 *A posteriori* landscape analysis

Landscape analysis is an entire research area that aims at characterizing the shape of the graph $(\mathcal{D}, f(\mathcal{D}))$ produced by the objective function f [Mal21]. Indications such as the abundance of basins of attraction, the extent of the infeasible region, the presence of flat regions or ridges are valuable to the designers of optimization algorithms as well as to the experts of the targeted problem. In this section, after revealing the best contact reduction strategy found so far, a landscape analysis is performed on the Covid-19 problem by analogy with the benchmark functions whose landscape features are known. The insight gained *a posteriori* explains the significance of relying on hybrid methods.

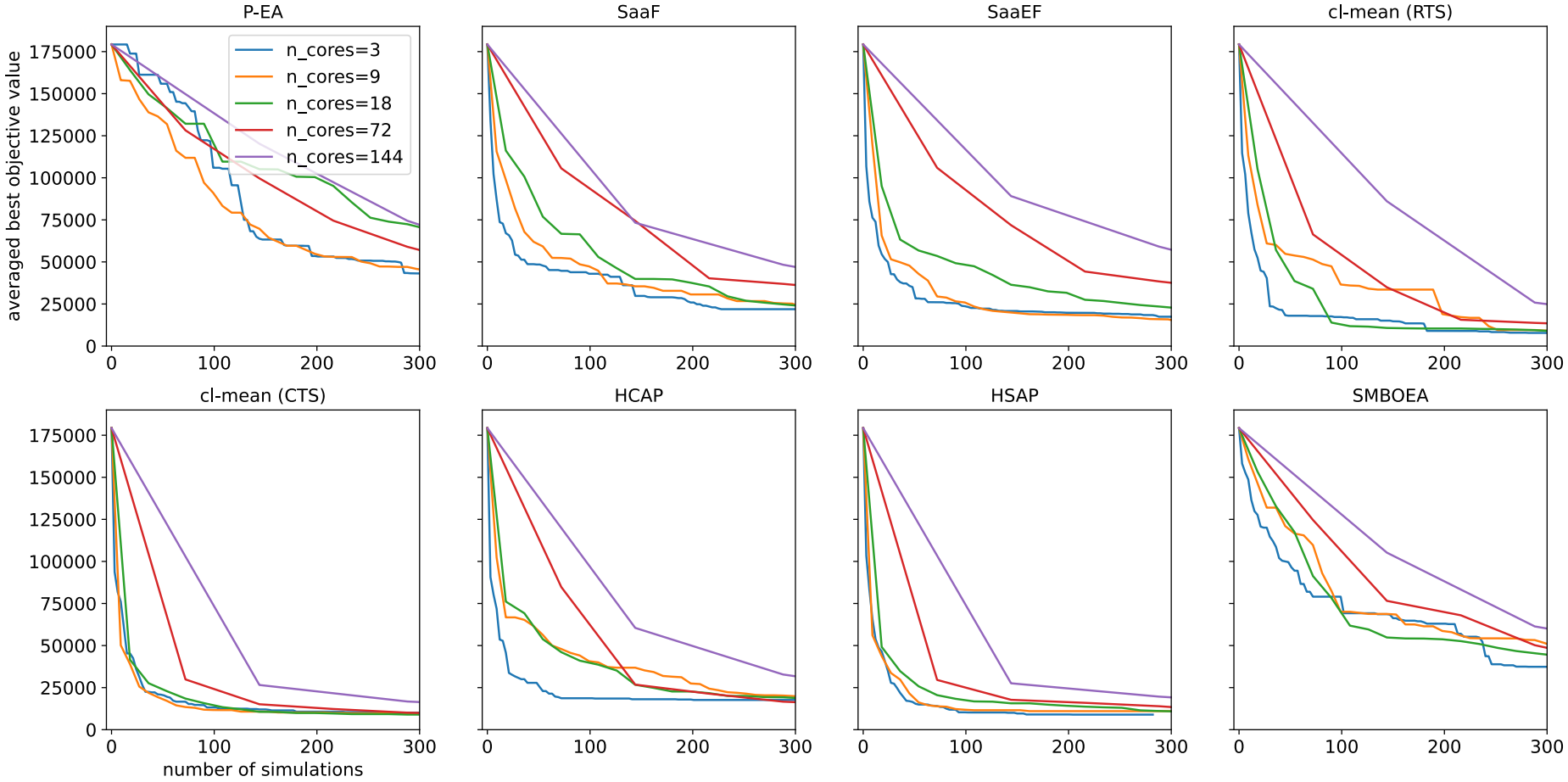

Figure 4.4: **Parallel scalability ($q = n_{cores}$).** Convergence profile in terms of best objective values averaged over the 10 runs of the experiment for different numbers of available computing cores. The number of simulations per cycle q is fixed to n_{cores} for all the approaches. The horizontal axis represents the number of simulations and the vertical axis represents the averaged best objective.

Table 4.8: **Best decision vector to the Covid-19 contact reduction problem.** x represents the contact mitigation factor.

Age-group	0-5	5-10	10-15	15-20	20-25	25-30	30-35	35-40	40-45	45-50
x	0.96	0.97	1.00	0.97	1.00	1.00	1.00	1.00	0.97	0.76
Age-group	50-55	55-60	60-65	65-70	70-75	75+				
x	0.00	0.00	0.00	0.00	0.00	0.00				

4.4.1 Reducing Covid-19-related death by contact reduction strategies

The overall minimum objective value found so far amounts to 3,536 and has been produced by the parallel hybrid HSAP method with the dynamic inclusive ensemble of ECs *dyn-df-incl* and $q = n_{cores} = 144$. According to the associated decision vector exposed in Table 4.8, contact for people aged less than 50 years-old should only be slightly reduced. Conversely, drastic contact reductions should be applied to seniors of 50+ years-old. Indeed, almost no restriction (less than 5%) is suggested for people from 0 to 45 years-old. Extreme contact restrictions are suggested for the elders beyond 50 years-old surely as they present the highest risk of medical complications.

4.4.2 Characterization of the landscape

An *a posteriori* analysis of the Covid-19 contact reduction problem is now possible as the experiments conducted so far have produced a huge amount of simulations. Among the 12,463,182 simulated solutions, 4,452,189 are infeasible, representing 36% of the whole set. The extent of the infeasible search space is expected to be greater than 36% because the sampling is biased towards the feasible region. Indeed, from the 720 initial simulated solutions obtained *via* LHS, only one is feasible. This is actually the reason why landscape analysis has not been performed prior to the optimization.

Relying on meta-models to characterize a search landscape is a technique that has been already proposed in [Mer+11; KT19]. On the one hand, the empirical comparisons led in Sub-section 2.5.4 suggested an analogy between the landscape of the Schwefel problem and the one of the Covid-19 application. On the other hand, the observations reported in Sub-section 3.4.4 indicate a similarity with the landscape underlying the Rosenbrock function. The dispersion metric is introduced hereafter to gain more knowledge about the landscape of the Covid-19 problem.

The dispersion metric [LW06] represents the multi-modality and the presence of global structure by measuring the average distance between the best solutions in the search space. The set of best candidates is defined by a proportion p_{DM} of the best solutions from the data set. It is stated in [KT19], that a database made of $50.d$ samples is sufficient to perform exploratory landscape analysis. Since $d = 16$ in this thesis, 800 samples are drawn for each problem. For Schwefel, 11,169,468 simulated solutions are available, 11,161,044 for Rastrigin, 11,276,316 for Rosenbrock and 8,010,993 feasible simulated solutions are accessible for the Covid-19 contact reduction problem. Each of these sets is divided into 800 clusters using the K-Means algorithm [AV07] implemented in Scikit-Learn [Ped+11] and the closest solution to each cluster's center is retained. The Flacco R package [KT19] is used to compute the dispersion metric for multiple values of p_{DM} and the outcomes are reported in Table 4.9. All the software libraries used in this thesis are listed in Table F.1.

Table 4.9: **Dispersion metric** based on a subset of 800 samples for the benchmark and the Covid-19 contact reduction problem. The dispersion metric is computed as the average distance between the best $\lfloor 800.p_{DM} \rfloor$ solutions divided by the average distance between the 800 solutions. Higher values characterize a harder optimization problem with respect to multi-modality and global structure.

p_{DM}	Schwefel	Covid-19	Rastrigin	Rosenbrock
0.02	0.7253051	0.6087193	0.3987407	0.2755931
0.05	0.7811394	0.7201954	0.4048308	0.3098171
0.1	0.8310808	0.7868463	0.4437553	0.3624034
0.25	0.8872491	0.8637474	0.5376603	0.4896596

High values of the dispersion metric indicate high dispersion of the best solutions in the search space and consequently imply the presence of multiple basins of attraction. For low values of p_{DM}, high values of the dispersion metric indicate a weak global structure in the sense that the multiple basins of attraction are far from each other. According to Table 4.9, the landscape associated to the Covid-19 problem is similar to the one of the Schwefel problem in terms of multi-modality and global structure.

Other tools built on the concept of nearest neighbors have been elaborated in [Ker+15] to bring out weak global structures. Let's denote $\mathcal{S}$ the 800-samples set generated previously for a given problem and let's define the distance to the nearest neighbors by:

$$d_{nn}(\boldsymbol{x}, \mathcal{S}) = min(\{d_2(\boldsymbol{x}, \boldsymbol{y}) | \boldsymbol{y} \in \mathcal{S} \backslash \{\boldsymbol{x}\}\}) \tag{4.2}$$

where $d_2(,)$ is the Euclidean distance. Let's define the distance to the better nearest neighbors by:

$$d_{nb}(\boldsymbol{x}, \mathcal{S}) = min(\{d_2(\boldsymbol{x}, \boldsymbol{y}) | f(\boldsymbol{y}) < f(\boldsymbol{x}) \text{ and } \boldsymbol{y} \in \mathcal{S}\}) \tag{4.3}$$

where $f()$ is the objective function. The set of the nearest neighbors distances is given by:

$$\mathcal{D}_{nn} = \{d_{nn}(\boldsymbol{x}, \mathcal{S}) | \boldsymbol{x} \in \mathcal{S}\} \tag{4.4}$$

and the set of the better nearest neighbors distances is given by:

$$\mathcal{D}_{nb} = \{d_{nb}(\boldsymbol{x}, \mathcal{S}) | \boldsymbol{x} \in \mathcal{S}\} \tag{4.5}$$

The two metrics used to compare the landscapes are:

$$nbf1 = \frac{sd(\mathcal{D}_{nn})}{sd(\mathcal{D}_{nb})} \qquad nbf2 = \frac{mean(\mathcal{D}_{nn})}{mean(\mathcal{D}_{nb})} \tag{4.6}$$

The first metric $nbf1$ is the ratio of the standard deviation of the two distance sets. For highly multi-modal problems or problems with a weak global structure, $sd(\mathcal{D}_{nb})$ is expected to be high so $ndf1 < 1$ while for problems with adequate global structure $sd(\mathcal{D}_{nn}) \approx sd(\mathcal{D}_{nb})$ is expected such that $ndf1 \approx 1$. The same reasoning applies for the second metric $nbf2$ when considering the ratio of the mean of the sets. Table 4.10 presents the nearest neighbors-related metrics computed for the benchmarks and the real-world problem. According to Table 4.10, the Covid-19 problem exhibits the less adequate topology followed by the Schwefel problem. The Rastrigin adequate global structure is detected by showing $nbf1 = 1$ for the associated samples set.

Table 4.10: **Nearest neighbors-related metrics**, as defined in (4.6), based on a subset of 800 samples for the benchmark and the Covid-19 contact reduction problems. Values closer to 1 indicate a more adequate global structure.

	Covid-19	Schwefel	Rastrigin	Rosenbrock
nbf1	0.863	0.967	1.000	0.998
nbf2	0.884	0.926	0.986	0.991

By the *a posteriori* landscape analysis conducted in this sub-section, it can be deduced that the constraint is severe and the landscape is multi-modal with a weak global structure. Adding the fact that the simulation is moderately expensive, this problem is undoubtedly tedious to solve. In such a critic case, the design and application of hybrid methods is thus relevant as they yield the best resolution of the problem.

4.5 Conclusion

This chapter started with the confrontation between P-SAEAs and P-SDAs. The APs based on IC optimization embedded into P-SDAs are more computationally expensive than the APs relying on reproduction operators inherent to P-SAEAs. Regarding the ECs, the more sophisticated ones require more computational efforts. In terms of resolution quality, the P-SAEAs where the surrogate is employed as an evaluator and/or a filter yields the best performance in case of moderately expensive problems on a landscape characterized by multi-modality and weak global structure. In this framework, it is advised to favor exploration at the onset of the search and exploitation afterwards through a dynamic EC where the indicator of exploration is the distance to the database of known solutions. From their side, P-SDAs are suitable to deal with low computational budgets and are able to exploit the predictive standard deviation as measure of uncertainty. The APs consisting in sampling sub-surrogates *via* Monte-Carlo Dropout or MCMC are notably to be put forward on multi-modal search landscapes with weak global structure.

Two brand-new parallel hybrid methods have been developed in this chapter to merge the best of both surrogate-based frameworks. The two kinds of APs are combined either concurrently at each cycle (HCAP) or successively during the search (HSAP). Each AP comes with its own EC and surrogate thus breeding multi-surrogate and multi-EC strategies. Numerical experiments conducted on the Covid-19 application reveal an enhancement of the quality of the resolution compared with a state-of-the-art parallel hybrid method and the best P-SBOAs identified so far. The new HSAP outstandingly stands at the first podium step for its various advantages. Indeed, it demonstrates a fast improvement for a low number of simulations such as P-SDAs and it achieves a continuous amelioration at latter stages of the search like P-SAEAs. Last but not the least, HSAP scales well with high numbers of computing cores. However, future endeavor should still be dedicated to automatically trigger the switch of AP during the search.

On the Covid-19 contact reduction problem, the best contact mitigation plan identified so far has been located by the new parallel hybrid HSAP method. According to this plan, contact should be restricted for people aged 50+ years-old as they represent the population more likely to grow complications. A landscape analysis run on the simulation-based objective function revealed a large infeasible region and a multi-modal landscape with weak global structure, confirming the difficulty of the problem and justifying the design of hybrid methods.

Chapter 5

Software platform for P-SBO

Contents

Related publications:

- Briffoteaux Guillaume, Melab Nouredine, Mezmaz Mohand, Tuyttens Daniel, "An adaptive evolution control based on confident regions for surrogate-assisted optimization" in *HPCS 2018 - International Conference on High Performance Computing & Simulation*, 2018, Orléans, France, hal-01922708, https://hal.archives-ouvertes.fr/hal-01922708/file/briffoteaux_melab_mezmaz_tuyttens.pdf

- Briffoteaux Guillaume, Ragonnet Romain, Tomenko Pierre, Mezmaz Mohand, Melab Nouredine, Tuyttens Daniel, "Comparing Parallel Surrogate-based and Surrogate-free Multi-Objective Optimization of COVID-19 vaccines allocation" in *OLA'2022 - International Conference on Optimization and Learning*, 2022, Syracuse, Italy.

5.1 Introduction

The investigations conducted in this chapter deal with the design of a scalable architecture for a software platform devoted to disseminate the techniques of P-SBO. The modular and flexible code structure of the pySBO platform is proposed along with an illustrated and exemplified documentation exposed on-line.

Section 5.2 motivates the importance of software architecture in the context of scientific research and raises the challenges to reach a scalable design. The design scalability is defined as the ability of a computer program to be modified, to be expanded and to cope with increased use [RXX11]. To respond to these challenges, existing tools are underlined for the convenient features they provide regarding code abstraction, utility, extensibility and accessibility. The Object-Oriented Programming (OOP) paradigm and the Python programming language are notably put forward. These tools are employed to build the pySBO Python platform for P-SBO whose modular structure is dissected in Section 5.3 through UML diagrams. An analysis of the Github repositories of related software is conducted to clarify the stance of pySBO in this branch of algorithms.

As a proof of concept and guidance to users, new algorithmic components are integrated into pySBO in Section 5.4. Multi-objective algorithms relying on custom evolutionary operators as well as the new problem to Covid-19 vaccine distribution are explained and implemented within the platform. The resolution of this novel simulation-based problem reported in Section 5.5 enables one to compare the various multi-objective algorithms and complete the demonstration of the usefulness of pySBO.

The pySBO platform and its documentation are available on-line at `https://github.com/GuillaumeBriffoteaux/pySBO` and `https://pysbo.readthedocs.io` respectively.

5.2 Scalable design

5.2.1 Motivations

Targeting as wider community as possible
The dissemination of surrogate-based optimization requires the availability of handy software tools easily understandable, extensible and customizable by a large scientific community. The software platform should assist researchers in the field of P-SBO to benchmark and design new algorithms. It should also facilitate ready-to-run methods and templates to help scientists from various domains to calibrate their model [Pal+22] and to build and solve simulation-based optimization problems in the search for optimal designs or plannings.

Diffusion of algorithmic tools
Taking care over the scientific software development is not always the priority in scientific research where the focus is prominently directed towards the development of new models. Nevertheless, leaving the implementation questions aside could slow down the diffusion of the research outputs and be notably detrimental to transversal research implicating multiple scientific fields.

Reducing the costs of software development
Reducing the execution time of computer programs is under the spotlight nowadays as denotes the infatuation for the computational capabilities of modern super-computers [TOP; Pan21]. The race for computing power is crucial for many domains such as artificial intelligence. Nevertheless, the costs attributed to software development and evolution should not be totally concealed by the computational costs. In [RXX11], the authors argue that

the time and costs entailed by software development are higher than the ones induced by computations ("Your time is worth more than your computer's time."). It is advised in [RXX11] to structure 80% of the code to simplify development and to diminish running time on the 20% of the code coinciding to the largest computational burden.

Scalable design
As early as the late 20th century, the design of scalable code architectures has been identified as a top-notch task of equally importance to super-computers building [NDS97]. When referring to software architecture, the design scalability is defined as the ability of a computer program to be modified, to be expanded and to cope with increased use [RXX11]. An architecture resorting to a modular decomposition is expected to lessen the dependencies between the algorithmic components and consequently boosts flexibility. A necessary condition to reach a scalable design is the availability of an exhaustive and exemplified documentation to favor the accessibility and alleviate the complexity related to code evolution.

5.2.2 Conceptual objectives

Code reusability
Code reusability is one of the primary objectives to design scalability. The reusability of algorithmic elements is defined in [FVW99] as their capability to be applied to various problems without requiring significant effort from the developer. Three levels of code reuse are pointed out in [Mel05]: *no reuse*, *reuse of code* and *reuse of code and design*.

The lowest level implies to implement from scratch and is indicated when the algorithm is trivial to program. The aptitude of the algorithm to treat a large range of applications may however demand a certain amount of work and experience feedback.

The medium level corresponds to libraries, that can be seen as extensions of a programming language. Libraries are often well-documented, computationally efficient and offer new functionalities *via* functions callable from the user side. For instance, the Keras library for Python has been used in this thesis to build ANNs [Cho15]. Albeit efficiently promoting code reuse, the internals of the libraries are hidden or tedious to comprehend at a first sight hence their characterization as black-box. As an example, the modification of the Keras library we have conducted to implement the BNN_MCD surrogate model has been quite time-consuming and resulted in poor performances regarding the computational costs of predictions.

Software frameworks are intended to overcome the shortcoming of libraries by enabling both code and design reusability through the decoupling of the invariant and specific parts of the code. The invariant part of a framework, also called "skeleton", is a collection of abstract and concrete classes representing pre-determined categories and actual algorithmic components paramount in the field of P-SBO. The specific part is rather written by the user to extend the framework according to its needs and/or to specify the problem at hand. The framework approach is chosen to design pySBO whose skeleton is inspired by the Pagmo/Pygmo platform dedicated to surrogate-free optimization [Ba19].

Utility and adaptation
The aim of the invariant part of the software platform is twofold. First, it should attract a large community of users by saving them time so they can focus more thoroughly on their specific problems. This is accomplished by offering a meaningful variety of ready-to-use convenient algorithmic tools (surrogates, ECs, problems, *etc.*) that have already proven to be valuable to P-SBO. It has been demonstrated in this thesis that the problem dependency is a key factor in the design of P-SBOAs, thus the algorithmic components in-

corporated into pySBO are numerous. Secondly, the skeleton should present access points to easily plug the specific code into. The invariant code should be protected enough to avoid dramatic disruption and, in the meanwhile, allowing flexibility of adaptation.

Accessibility
The accessibility is first conveyed by the easiness of use of the software platform. Getting the code and installing the associated dependencies should be straightforward and the execution of a starting example should be carried on quickly. The easiness of specialization and extension is the second major aspect. It goes hand in hand with a good comprehension of the code structure from a global picture of the platform to more refined details at the class level. The access points to special code insertion have to be properly highlighted. Finally, the lack of portability over different operating systems and hardware infrastructure may hinder the accessibility. The heterogeneity of modern computational systems notably with respect to memory organisation (distributed or shared) should be handled to best benefit from the parallel use of the computational resources.

5.2.3 The tools for scalable code architecture

Object-Oriented Programming (OOP)
The paradigm of OOP consists to split the code into classes representing the algorithmic elements that play a primordial role for the problem at hand [Del12]. A class gathers together attributes, whose concrete values define an actual object, and methods acting on these attributes. A class can be seen as a mould used to instantiate objects. Typically, the user does not need to be aware of the internals of the class as the interactions with the object are realized through its attached methods. As far as the interface is not reduced, the underlying code can evolve without severe restrictions therefore enhancing maintainability.

To accomplish the user's goal, the instantiated objects collaborate together *via* different kinds of association: inheritance, composition and aggregation. Composition and aggregation allows a class (the part) to be part of another class (the whole). Conversely to aggregation, the part can not exist independently from its whole in a composition relationship. A class (the child) can inherit all the attributes and methods from another class (the parent). The child class is then specialized by adding new functionalities to differentiate from its parent. Modularity and code reuse are greatly empowered by these associations as they segment the code and prevent duplication.

To represent a generic type (*e.g.* surrogate model), it is also possible to write an abstract class that only specifies the services of the type without dispensing the actual implementation. Abstract classes can not be instantiated but are intended to be derived by inheritance to concrete classes. An abstract type can however be used to characterize an input argument of a function thus bringing the advantage of deferring the choice for the concrete type at execution time. This latter aspect of OOP is called polymorphism. Inheritance and polymorphism aim at factorizing the code writing, moreover, abstract classes offer good access points to extend the platform.

Python programming language
Table 5.1 displays the number of questions in Stackoverflow and the number of projects per programming language when searching for "bayesian-optimization" in Github. The overwhelming majority of the repositories and a large number of questions are concerned with the Python programming language. Python is notably acclaimed for its easiness of handling for non-expert programmers. Indeed, in Python, no memory management has to be carried out unlike Java, C or C++, and no manual variable typing is required. Although Matlab and R show similar features in this respect, Matlab is a commercial black-box software and R is statistic-oriented while Python is free, open-source and general-purpose.

Table 5.1: **Representation of programming languages.** Number of Github repositories and number of Stackoverflow questions related to the programming languages.

Programming language	Github repositories "bayesian-optimization"	Stackoverflow questions
Python	1,154	1,928,701
Matlab	73	92,720
R	62	444,476
C++	36	760,929
Julia	14	10,299
C	8	377,712
Go	6	60,839
Java	0	1,839,322

The advantages of Python include OOP features for modularity, code portability through the interpreter, code indentation favoring readability and vectorization alleviating the amount of code. The assets of Python have brought the development of many libraries in particular in the domain of Machine Learning. In Table F.1, the libraries used in the experiments reported in this thesis are listed. Choosing Python is therefore expected to enhance accessibility of the pySBO platform. Nevertheless, everything comes at a price and the computational efficiency of Python does not come up to the mark. Indeed, special care should be devoted to the most expensive computational part of a program by resorting to libraries such as Numpy [Ha20], Numba [LPS15] or to interoperability with C [Gmy+20].

The mpi4py Python library for Message Passing Interface [GLS99] is employed to ensure the communication between the computing cores [DF21]. The parallel code runs in both shared or distributed memory systems without necessitating any modifications thus granting hardware portability.

Programming rules

A set of consistent programming rules has been set up and applied to write the pySBO framework in order to foster clarity and accessibility and to generate the documentation automatically. The Java convention of one class per file is notably adopted to impede code masking and to simplify the map of the platform. Every concrete class implements three specific methods: an initializer (`__init__`), a deleter (`__del__`) and a string method returning a description of the class (`__str__`). Besides, a portion of comments follows each class and function heading to explain its functioning and to precise the potential inputs and outputs. The automatic generation of a part of the documentation depends directly on the respect of this latter rule.

Documentation and distribution

The pySBO documentation rests on various contents merged together and organized by a bunch of *reStructuredText* files [GH18]. The map of the skeleton is drawn by a Unified Modelling Language (UML) diagram that references the objects along with their attributes, functions and the links between them thus displaying a global view of the platform. The comments reported in the Python code (docstrings) are transcribed in the documentation to give more finer details about the classes and functions. Widely used algorithms are implemented to serve either as starting examples, for benchmarking or as ready-to-use programs. Diagrams are recurrently employed to intuitively explain the algorithms and the underlying parallel aspects. Lastly, useful information as of installation

guidance and licence are communicated *via* the dedicated web-site. The *Read The Docs* utility effectively builds the documentation and freely deploys it on-line. Github is utilized to distribute and expose the pySBO platform to the open-source software community and to favor its collaborative evolution.

5.3 The modular structure of pySBO

5.3.1 From a global view to a finer description

For the sake of visualization clarity, the UML diagram of pySBO shown in Figure 5.1 only displays the classes and their links. The full UML diagram including the attributes and methods is deferred to the appendix section in Figure F.1. The symbols employed to depict the associations between classes are presented in the upper right corner of Figure 5.1. Along with the inheritance and aggregation relationships, the link called "dependency" is introduced to reflect cases where an object of a type defined by class A is used as an argument or a local variable in a method of class B.

From the global view of the platform exhibited in Figure 5.1, four collections of classes arise. To each collection is associated a color and a name chosen after the kind of algorithmic components the collection embeds. The Problem collection appears in red, the Evolution collection in yellow, the Surrogate collection in blue and the Evolution Control collection in green. Besides, a class called `Global_Var` contains public global variables used to store the best candidate and objective values found so far and the best hyper-volume and the associated reference point for multi-objective problems. Variable or function names preceding by the '+' symbol are accessible externally while the '-' symbol refers to entities local to the class where they are defined.

The Problem collection further detailed in Figure F.2 deals with the specification of the optimization problem and, more particularly, the definition of the objective function. The `Problem` abstract class gathers the attributes common to all problems (number of decision variables and number of objectives) and imposes to any concrete child class to implement the objective function evaluation (`perform_real_evaluation`) and the test of feasibility (`is_feasible`). The abstract class `Box_Constrained` further imposes to write a method returning the bounds for each decision variable (`get_bounds`). The distinction typically made between single- and multi-objective problems is reported by extending `Box_Constrained` into `Single_Objective` and `Multi_Objective` respectively. The initializer verifies the number of objectives accordingly and a method plotting the objective function is demanded for any single-objective problem. The concrete problem classes provided within pySBO are well-known benchmark functions (Schwefel, Rastrigin, *etc.*) or test suites (DTLZ [Deb+01], CEC2013 [LQS13]). It is relatively easy to create a new problem by copy-pasting an existing class and modifying the methods. Besides, any box-constrained problem can be a part of an object, defined by the `DoE` class, serving for initial sampling of the search space. Both random sampling and LHS are provided but adding another sampling strategy is as obvious as creating a new method within the `DoE` class.

A problem is a part of a population of candidate solutions defined by the `Population` class in the Evolution collection relative to evolutionary computation (Figure F.3). All the categories of evolutionary operators are represented by an abstract class (`Selection`, `Crossover`, `Mutation` and `Replacement`) that expect the actual determination of the action realized by the operator (`perform_selection`, `perform_crossover`, `perform_mutation` and `perform_replacement`). Adding a new operator is possible by extending the adequate abstract class *via* inheritance. All the operators act on a population of solutions, making the `Population` class the central class of this collection. Arrays to store the decision vectors (`dvec`), the associated values of the objective(s) (`obj_vals`)

and the modes of evaluation (either `False` for surrogate prediction or `True` for real evaluation) compose a population object. Handy methods are already implemented such as input/output for logging and storing, sorting and splitting functionalities. Both single- and multi-objective populations are supported. The hyper-volume indicator is used to compare solutions in the multi-objective case (notably in `sort` and `update_best_sim`).

The structure of the collection of surrogates displayed in Figure F.4 is very basic. The `Surrogate` abstract class groups together the common features while the extended concrete classes are wrapper for regression or interpolation models built using Python libraries (Keras for `BNN_MCD`, GPyTorch for `GP` and `GP_MO`, *etc.*). A surrogate is specific to a problem, from hence the aggregation relationship with the `Problem` class. To construct a surrogate, the filename of the archive of simulations (`f_sim_archive`) and the number of samples used for training (`n_train_samples`) must be fed through the initializer. Two other filenames, `f_train_log` and `f_trained_model`, must be provided to store the training logs and the trained model respectively. The implementation of the actual prediction (`perform_prediction`), training (`perform_training`), data-normalization-related methods and the loading of a model trained in the past (`load_trained_model`) are deferred to the child classes. All the concrete classes except `GP_MO` allow for scalar predictions while `BNN_MCD` and `GP_MO` yield vector predictions.

The parent of all the classes defining ECs is the abstract `Evolution_Control` class from the collection presented in Figure F.5. It enforces to implement the method that returns the sorted indexes of the individuals in a population according to the criterion underlying the EC (`get_sorted_indexes`). This latter method is the most important when implementing a new EC. Most of the ECs are informed by the surrogate predictions and consequently embed an object of the `Surrogate` type. For such ECs, a method delivering the relative IC value (*e.g.* the value of EI) is demanded as we found it useful. Nevertheless, to build P-SDAs, it is most convenient to rely on the implementation of Algorithm 5 by resorting to `get_sorted_indexes` and the `Custom_Elitism` class from the Evolution collection. In this way, all types of ECs, and not only the informed ones, can be invoked. Indeed, it is not possible to compute the IC value for ensembles of ECs. The ensembles form another family of ECs embodied by the `Ensemble_EC` abstract class. The dual relationship of aggregation-inheritance between the `Evolution_Control` and `Ensemble_ECs` abstract classes conveys the fact that an ensemble of ECs is an EC composed of multiple ECs.

5.3.2 Related software

The review of publicly available codes is conducted by searching for "surrogate-based-optimization", "surrogate-assisted-optimization" and "surrogate-assisted-evolutionary -algorithm" within the topics of Github. Among the outcomes, the projects showing multiple algorithmic components and an on-line documentation are commented hereafter.

In [BD22], the authors introduce *pysamoo*, a Python library for surrogate-assisted single- and multi-objective optimization that encompasses directly applicable algorithms. The main discrepancy with pySBO is the black-box covering the internals of the code that prevents to compose new algorithms and complicate the definition of new problems. Nevertheless, the documentation available online is sound and illustrated by diagrams and examples. The *pySOT* Python platform proposed in [EBS19] is similar to pySBO in that it is extensible and exemplified. However, *pySOT* only enables one to compose single-objective P-SDAs while pySBO also includes P-SAEAs and multi-objective optimization. A remarkable feature of *pySOT* is the possibility of asynchronous parallel simulations, useful when the simulation duration varies significantly from a decision vector to another. In pySBO, the asynchronous management should be addressed by the user through directives proceeding from the Message Passing Interface. The Julia platform *Surrogates.jl*

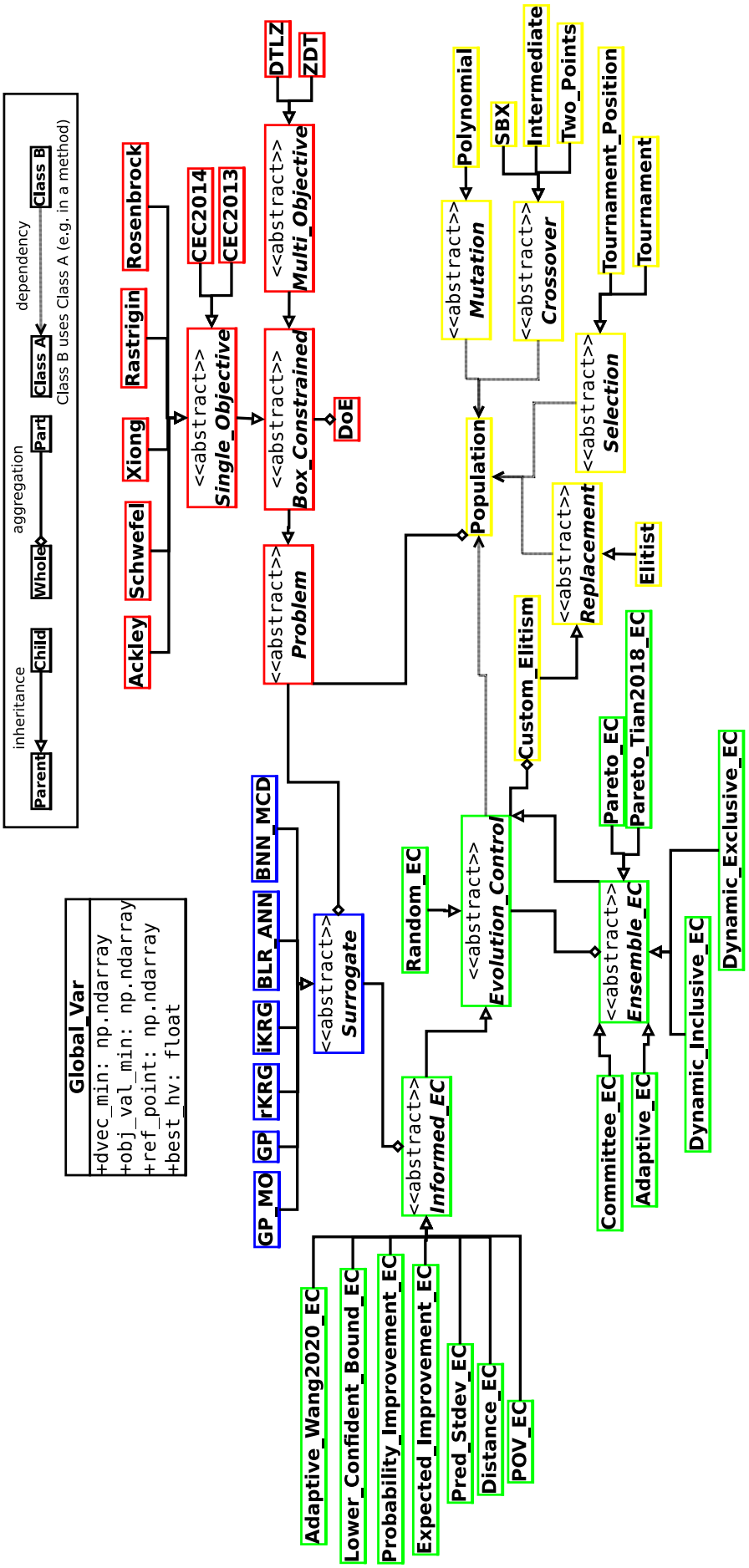

Figure 5.1: Global UML diagram of pySBO (classes only).

[Bes+22] is dedicated to single- and multi-objective SDAs but no indications are given on how to leveraging distributed computing. In the *EXPObench* repository [Bli+21] no pieces of code are especially supplied to build P-SBOAs. Instead, numerous approaches and problems are implemented in Python and a tutorial signifies how to add new strategies and instances for benchmarking. The C++ *NOMAD* library [MT22] is exclusively centered on the Mesh Adaptive Direct Search algorithm for constrained single-objective optimization [AD06]. Although substantially differing from pySBO, NOMAD is worth to cite for its exhaustive documentation. The Scikit-learn-based *SKSurrogates* platform for Python [Gha20] provides tools to implement SDAs to solve single-objective problems but without resorting to parallelism. The associated documentation exposes a comprehensive view of the code and detailed examples.

In the set of software referenced above, few projects (including *pysamoo* and pySBO) support multi-objective optimization and distributed computing. More importantly, they are almost all uniquely devoted to assemble SDAs in order to solve very expensive problems. Searching for "bayesian-optimization" in Github yields 1,529 results among which the Python *BoTorch* platform [Bal+19] that also focuses on SDAs with support for multi-objective and distributed computing. Our pySBO platform takes both very and moderately expensive instances into account by helping to build both SDAs and SAEAs. The development of software for P-SBO is currently exploding as exhibited by the huge number of repositories recently updated in Github.

5.4 Multi-objective test case

This section aims at demonstrating the definition of a new problem in pySBO and the integration of parallel surrogate-based and surrogate-free Multi-Objective (MO) algorithms into the platform. The optimization approaches are subsequently used to solve the new problem related with Covid-19 vaccine distribution.

5.4.1 Covid-19 vaccine distribution problem

Description of the problem
The vast vaccination programs implemented over the last year or so all around the world achieved reductions of Covid-19 hospitalizations and deaths [Vil+22]. However, access to vaccination remains challenging, especially for low- to middle-income countries that are not able to offer vaccination to all their citizens [SMD22]. The multi-objective problem we are concerned with consists in optimizing the age-specific vaccines allocation plan to limit the impact of the disease in Malaysia under a capped number of doses. The population is divided into 8 age-categories of 10-years band from 0-9 years old to 70+ years old and the impact is expressed in terms of total number of deaths and peak hospital occupancy. Moreover, the reduction of mobility restriction is considered as a target.

The simulation is realized in three phases by the AuTuMN software publicly available at `https://github.com/monash-emu/AuTuMN/`. The simulator is calibrated during the first phase with data accumulated from the beginning of the epidemic to the 1st of April 2021. The second phase starts at this latter date and lasts three months during which a daily limited number of doses is shared out among the population. Relaxation of mobility restrictions marks the kickoff of the third phase in the course of which a new distribution plan is applied involving the same number of daily available doses as in phase 2.

Proposed formulation

Decision variables $x_i \in [0,1]$ for $1 \leqslant i \leqslant 8$ and for $9 \leqslant i \leqslant 16$ represent the proportions of the available doses allocated to the 8 age-categories for phase 2 and phase 3 respectively. Variable $x_{17} \in [0,1]$ expresses the degree of relaxation of mobility restrictions where $x_{17} = 0$ leaves the restrictions unchanged and $x_{17} = 1$ means a return back to the pre-covid *era*. The following convex constraints convey the limitation of the number of doses during phases 2 and 3:

$$\sum_{i=1}^{8} x_i \leqslant 1 \text{ and } \sum_{i=9}^{16} x_i \leqslant 1 \tag{5.1}$$

The three-objective optimization problem consists in finding $\boldsymbol{x}^*$ such that

$$\boldsymbol{x}^* = \underset{\boldsymbol{x} \in [0,1]^{17} \text{ s.t. } (5.1)}{\arg\min} (g_1(\boldsymbol{x}), g_2(\boldsymbol{x}), 1 - \boldsymbol{x}_{17}) \tag{5.2}$$

where $g_1(\boldsymbol{x})$ is the simulated total number of deaths and $g_2(\boldsymbol{x})$ the simulated maximum number of occupied hospital beds during the period. Because the simulation involves the resolution of differential equations, g_1 and g_2 are non-linear.

Related formulations

The onset of the Covid-19 outbreak has been rapidly followed by the development of dedicated simulation software to predict the trajectory of the disease [Cha+20; Tra+21]. The availability of such tools enables one to inform authorities by formulating and solving optimization problems. In [Duq+20], a SEIR-model (Susceptible, Exposed, Infectious, Recovered) is deployed to simulate Covid-19 impacts. A single-objective (SO) problem is subsequently derived and handled by grid-search to regulate the alleviation of social restrictions. Multiple SO optimizations are carried out independently by a simplex or a line search algorithm in [Mat+21a; ML10; Mat+21b] to efficiently allocate doses of vaccines to the age-categories of a population. The number of infections, deaths and hospital admissions are considered as the possible objectives. The prioritization rules approved by the government of the studied cohort are integrated as constraints in the linear programming model presented in [Buh+21] to minimize mortality. In [Han+21], multiple indicators are combined into a scalar-valued objective function. The MO formulation exhibited in [AKN21] consists in maximizing the geographical diversity and social fairness of the distribution plan. Nevertheless, the MO problem is scalarized into a SO one that is then solved by a simplex algorithm. The approach used by Bubar *et al.* [Bub+21] is significantly different from ours, as the authors predefined a set of vaccination strategies and selected the most promising approach among them. In contrast, our continuous optimization approach automatically designs strategies in a fully flexible way. The optimization problem solved by McBryde *et al.* [McB+21] is closer to that presented in our work since a similar level of flexibility was allowed to design optimal vaccines allocation plans. However, the authors used a simpler Covid-19 model resulting in significantly shorter simulation times, such that optimization could be performed using more classical techniques. To the best of our knowledge, it has not been suggested yet to simultaneously minimize the number of deaths, peak hospital occupancy and the degree of mobility restriction through a MO-formulated problem. The fact that we consider the level of restrictions as one of the objectives to minimise represents a novelty compared to the previous works.

Integration in pySBO

A new concrete class, namely `Covid_vaccines`, given in Listing F.1, is derived from the abstract class `Multi_Objective` to implement the new problem at hand in pySBO. The new `__init__` (resp. `__del__`) method must first call upon the initializer (resp. the deleter) from the parent class where some preliminary instructions could have been

specified. In the present case, `Multi_Objective.__init__(self, 17, 3)` is executed to initialize the number of decision variables to 17 and the number of objectives to three in `Covid_vaccines.__init__`. The link with the external simulation library is established at initialization by instantiating an object from the AuTuMN code. This AuTuMN object is then used in the `perform_real_evaluation` function to evaluate the objective function. The constraints of the problems given by Equation (5.1) are injected to the `is_feasible` function while the bounds for the decision variables are stated in the `get_bounds` function.

5.4.2 Surrogate-free approaches

The major challenge in MO optimization is to balance convergence and diversity in the objective space. Convergence is related to the closeness to the Pareto Front (PF) [Tal09]. Diversity is indicated by an extended coverage of the objective space by the Non-Dominated Fronts (NDFs). General definitions valuable to MO optimization are given in Appendix C.

NSGA-II

The first highlighted MO algorithm is the Non-dominated Sorting Genetic Algorithm (NSGA-II) [Deb+02] exposed in Appendix C. This EA relies on two ingredients at the selection and replacement steps to balance between convergence and diversity. To promote convergence, solutions pertaining to better NDFs are better ranked (non-dominated sorting). To favor diversity, solutions composing the same NDF are distinguished by setting the promise as high as the crowding distance is high (crowding distance sorting). The non-dominated and crowding distance-based sorting has been a source of inspiration to design the Pareto-based bi-criterion EC with crowding distance presented in Sub-section 2.3.2. The complexity of NSGA-II is $\mathcal{O}(m.n_{pop}^2)$, where m is the number of objectives and n_{pop} the population size.

In pySBO, we rely on the Pygmo implementation of the non-dominated and crowding distance sorting for Python [Ba19]. The `sort` function of the `Population` class is written to handle single or multiple objectives in a transparent way for the user of the class. The selection step of NSGA-II consists of sorting the population and applying the tournament selection based on the position of the individuals into the population (lower index meaning a better promising individual). The elitist replacement implemented in pySBO is based on the sorting function of the population therefore implicitly resorting to non-dominated and crowding distance sorting in the MO scenario. The implementation of NSGA-II within pySBO is almost the same as the one of the genuine single-objective EA. The reference point for hyper-volume computation should notably be set through the public variable `Global_Var.ref_point`.

RVEA

The Reference Vector guided Evolutionary Algorithm (RVEA) has been recently proposed in [Che+16] to handle many-objective optimization problems. In RVEA, a set of reference vectors is introduced in order to decompose the objective space, and a new distance, termed angle penalized distance, is introduced to adaptively balance convergence and diversity during the search. The general structure of RVEA, which is presented in Algorithm 16, is roughly the same than the one of a traditional EA. The novelty in the algorithm structure relies on the methods used for the initialization and the update of the reference vectors (line 1 and 11 respectively) and the replacement step (line 9).

Initialization
The main goal of the set of reference vectors is to enhance diversity by uniformly decomposing the objective space into sub-populations. Each reference vector is the representative of one sub-population and each new candidate solution is affected to the sub-population whose representative reference vector is the closest. Consequently, the initial set of reference vectors must cover the objective space uniformly. To achieve this property, it is proposed to generate unit reference vectors $\boldsymbol{v_{1,j}}$ in the first quadrant through the simplex-lattice method [Cor02] (line 1 in Algorithm 16):

$$\begin{aligned} \boldsymbol{u_j} &= \left(u_j^1, \ldots, u_j^m\right) \text{ for } j \in \{1, \ldots, n_{ref}\} \\ u_j^k &\in \left\{\frac{0}{s_l}, \frac{1}{s_l}, \ldots, \frac{s_l}{s_l}\right\} \text{ such that } \sum_{k=1}^{m} u_j^k = 1 \\ \boldsymbol{v_{1,j}} &= \frac{\boldsymbol{u_j}}{||\boldsymbol{u_j}||} \end{aligned} \tag{5.3}$$

where $s_l \in \mathbb{N}^+$ determines the number of reference vectors by

$$n_{ref} = \binom{s_l + m - 1}{m - 1} \tag{5.4}$$

Selection and reproduction
The selection of parents consists in sampling randomly $\lfloor \frac{n_{ref}}{2} \rfloor$ pairs of parents from the current population (line 5). Each pair of parents is mated through cross-over and mutation (line 6) to generate a population of children.

Reference Vector guided replacement
The replacement step (line 9) is composed of three sub-steps. Firstly, the objective vectors from the population $\mathcal{P}_i$ are translated to fit in the first quadrant in the objective space. Secondly, the population is divided into sub-populations based on the distance to the reference vectors. Thirdly, one individual per sub-population is kept to form the new population $\mathcal{P}_{i+1}$.

The objective vector translation is realized thanks to the following formula:

$$\boldsymbol{y'_{i,l}} = \boldsymbol{y_{i,l}} - \boldsymbol{z_i^{min}} \text{ for } l \in \{1, \ldots, |\mathcal{P}_i|\} \tag{5.5}$$

where $\boldsymbol{y_{i,l}}$ is the objective vector associated to $\boldsymbol{x_{i,l}}$ (the l-th individual from $\mathcal{P}_i$) and $\boldsymbol{z_i^{min}} = \begin{pmatrix} z_{i,1}^{min} & \ldots & z_{i,m}^{min} \end{pmatrix}$ is the vector containing the minimum values known so far for each objective. $\boldsymbol{z_i^{min}}$ is also called the ideal point and the purpose of the translation is to move the objective vectors to the first quadrant where the ideal point is the origin. In other terms, vectors $\boldsymbol{y'_{i,l}}$ are located in the first quadrant where $\boldsymbol{z_i^{min}}$ is the origin.

Subsequently, the population $\mathcal{P}_i$ is divided into n_{ref} sub-populations $\mathcal{P}_{i,1}, \ldots, \mathcal{P}_{i,n_{ref}}$ where the representative of sub-population $\mathcal{P}_{i,j}$ is the reference vector $\boldsymbol{v_{i,j}}$. Determining the closest reference vector to a given translated objective vector $\boldsymbol{y'_{i,l}}$ amounts to determining the sub-population the individual $\boldsymbol{x_{i,l}}$ belongs to. It is worth noting that the population size may vary during the search because a sub-population may be empty. The acute angle between the reference vectors and the objective vector is a distance measure as a small angle value reflects a close proximity:

$$\mathcal{P}_{i,j^*} = \{\boldsymbol{x_{i,l}} | j^* = argmax_{j \in \{1,\ldots,n_{ref}\}} \cos\theta_{i,l,j}\} \tag{5.6}$$

where

$$\cos\theta_{i,l,j} = \frac{\boldsymbol{y'_{i,l}}.\boldsymbol{v_{i,j}}}{||\boldsymbol{y'_{i,l}}||} \tag{5.7}$$

Finally, for each sub-population, the individual minimizing the angle penalized distance is retained to be part of the new population. The angle penalized distance is given by

$$d_{i,l,j} = (1 + P(\theta_{i,l,j})).||\boldsymbol{y'_{i,l}}|| \tag{5.8}$$

where

$$P(\theta_{i,l,j}) = m.\left(\frac{i}{n_{gen}}\right)^2.\frac{\theta_{i,l,j}}{\gamma_{\boldsymbol{v_{i,j}}}} \tag{5.9}$$

where $\gamma_{\boldsymbol{v_{i,j}}}$ is the smallest angle value between reference vector $\boldsymbol{v_{i,j}}$ and the other reference vectors in $\mathcal{V}_i$. At the beginning of the search $\frac{i}{n_{gen}}$ is small thus $d_{i,l,j} \approx ||\boldsymbol{y'_{i,l}}||$. So, the angle penalized distance favors convergence since a small value for $||\boldsymbol{y'_{i,l}}||$ amounts for an objective vector $\boldsymbol{y_{i,l}}$ close to the ideal point. However, as the search proceeds, more importance is given to the term $\frac{\theta_{i,l,j}}{\gamma_{\boldsymbol{v_{i,j}}}}$ that is as small as $\boldsymbol{y'_{i,l}}$ is close to $\boldsymbol{v_{i,j}}$, thus indicating a better diversity. From the angle penalized distance definition, diversity is said to be good when the translated objective vectors are close to their associated reference vectors.

Reference Vector update
The last step of a RVEA cycle resides in updating the reference vectors. This step ensures obtaining a uniformly distributed NDF even for problems where the different objectives are scaled to different ranges. The update is realized according to the following formula:

$$\boldsymbol{v_{i+1,j}} = \frac{\boldsymbol{v_{1,j}} \odot (\boldsymbol{z^{max}_{i+1}} - \boldsymbol{z^{min}_{i+1}})}{||\boldsymbol{v_{1,j}} \odot (\boldsymbol{z^{max}_{i+1}} - \boldsymbol{z^{min}_{i+1}})||} \text{ for } j \in \{1, \ldots, n_{ref}\} \tag{5.10}$$

where $\boldsymbol{z^{max}_{i+1}}$ (resp. $\boldsymbol{z^{min}_{i+1}}$) is the vector made of the maximum (resp. minimum) objective values at the $i+1$ generation and $\odot$ is the element-wise product. The reference vector update should only be performed once in a while to ensure a stable convergence. The frequency of update f_{upd} is set to 0.1 as in [Che+16]. The complexity of RVEA is $\mathcal{O}(m.n^2_{ref})$.

Implementation
The new class called `Reference_Vectors_Set` is inserted into the Evolution collection of the platform to represent the set of reference vectors of RVEA. The associated UML diagram of this class is exhibited in Figure 5.2. The `__init__` function embeds the simplex-lattice method of Equation (5.3) to create a bi-dimensional array containing the reference vectors (`rv`). This latter array is made public to simplify the update process. The function realizing the guided replacement returns a new population, hence the dependency with the `Population` class. The aggregation relationship with the `Problem` class reflects the requirement of knowing the problem to build a population object. Besides, the number of objectives (stored within the problem object) is recurrently needed. Based on the new class, few lines of code are needed to convert the code for NSGA-II into the RVEA implementation in pySBO. It is particularly worth to remind that the initial set of reference vectors is to be conserved during the whole execution of RVEA as it is needed to update the current set (Equation (5.10)).

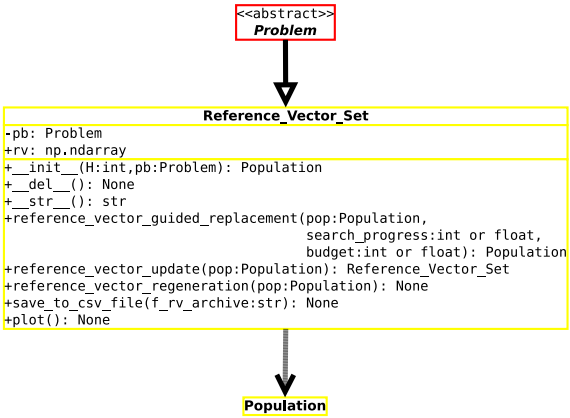

Figure 5.2: UML diagram of the *Reference_Vector_Set* class in pySBO.

RVEA*

In RVEA, the population size may decrease during the search due to empty sub-populations. In cases of degenerated or disconnected Pareto fronts a high number of sub-populations become empty and the NDF obtained at the end of the search may not be dense enough. To circumvent this hindrance it is proposed in [Che+16] to introduce an additional set of reference vectors $\mathcal{V}^*$ wherein useless reference vectors are regenerated at each cycle. The new RVEA variant is denoted RVEA* and is presented in Algorithm 17.

The additional set of reference vectors $\mathcal{V}^*$ is initialized like the original set $\mathcal{V}$ (line 2). Both sets are used to operate the reference vector guided replacement (line 10) by merging them and removing the possible duplicated vectors. The reference vector update described previously only applies to $\mathcal{V}$ (line 12) while $\mathcal{V}^*$ goes through regeneration at the end of each cycle (line 16).

The regeneration step consists to replace the reference vectors that would correspond to an empty sub-population when clustering the NDF of the current population $\mathcal{P}_{i+1}$. Four sub-steps are realized to carry out the regeneration. Firstly, the dominated solutions from the current population are discarded. Secondly, the objective values are translated according to Equation (5.5). Thirdly, the sub-populations are created just as in Equation (5.6). Finally, for each empty sub-population, the associated representative reference vector from $\mathcal{V}_i^*$ is replaced by a randomly sampled unit vector given by

$$\frac{\boldsymbol{u}}{||\boldsymbol{u}||} \in \mathbb{R}^m \text{ s.t. } u^k \in [0, z_{i+1}^{max,k}] \forall k \in \{1, \ldots, m\} \tag{5.11}$$

where $\boldsymbol{z_{i+1}^{max}}$ is the vector of the maximum objective values in the population.

The reference set $\mathcal{V}$ is maintained alongside $\mathcal{V}^*$ to prevent a premature discarding of a region in the objective space that would temporarily be empty. The regeneration procedure is implemented in the `Reference_Vector_Set` class and the code extension from RVEA to RVEA* is very limited within pySBO.

Custom Reproduction Operators

The constraints of the Covid-19 vaccine distribution problem are convex and analytically verifiable. Therefore, it is possible to design specific reproduction operators that directly generate feasible candidates. Assuming that every feasible solution can be reached, this technique has shown to be a reliable one [Mic+96]. The specific cross-over operator, called *distrib-X* and graphically exemplified in Figure 5.3, considers the two phases and the degree of relaxation independently. For two parents $\boldsymbol{x}$ and $\boldsymbol{y}$, the last decision variable for the two children $\boldsymbol{z}$ and $\boldsymbol{t}$ is set such that $z_{17} = x_{17}$ and $t_{17} = y_{17}$. Regarding the

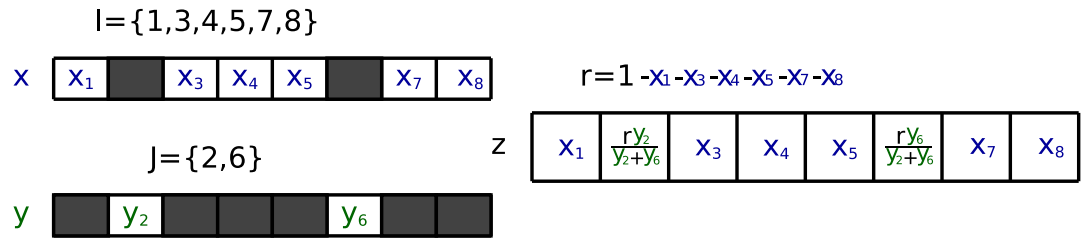

Figure 5.3: Illustration of the custom cross-over operator *distrib-X* for the Covid-19 vaccine distribution problem. Parents $\boldsymbol{x}$ and $\boldsymbol{y}$ father the offspring $\boldsymbol{z}$. For the sake of simplicity, only the decision variables related to phase 2 are displayed and only the first child $\boldsymbol{z}$ is shown.

second phase, let I and J be a random partition of $\{1, \ldots, 8\}$. For the age categories in I, $\boldsymbol{z}$ receives the proportion of vaccines from $\boldsymbol{x}$ ($z_i = x_i$ for $i \in I$). The remaining proportion of available doses at this step is $r = 1 - \sum_{i \in I} x_i$. For the age categories in J, the remaining proportion of doses is shared out according to the proportion allocated to the corresponding age categories in $\boldsymbol{y}$. In other terms, for $j \in J$, $z_j = \frac{r.y_j}{\sum_{j \in J} y_j}$. A similar treatment is applied to the variables associated to the third phase. The second child $\boldsymbol{t}$ is generated with an analogous procedure, where the roles of the parents are reversed.

The specific mutation operator, denoted *distrib-M*, disturbs a decision variable randomly chosen with uniform probability for $\{1, \ldots, 8\}$, $\{9, \ldots, 16\}$ and $\{17\}$. The last decision variable is mutated by polynomial mutation [Tal09]. For the remaining ones, two age categories of the same phase are randomly selected and a random amount of doses are transferred from the first category to the second one.

The well-known *intermediate* and the *2-points* cross-over operators [Tal09] can also be invoked to deal with the vaccine distribution problem. The *intermediate* strategy combines parents by random weighting average, while the *2-points* operator distributes portions of parents to the children. The portions are defined by two points with the first one separating phase 2 and phase 3 and the second one located between phase 3 and the relaxation decision variable x_{17}.

The incorporation of new reproduction operators within pySBO is achieved by following the same procedure than for adding a new problem. The access point for inheritance is now either the `Mutation` or the `Crossover` abstract class. In the Evolution collection displayed in Figure F.3, the general-purpose Intermediate and Two-Point cross-over operators are added alongside the SBX operator. The abstract class specifies the functions to be implemented. For each of these functions, the first instruction must be a call to the parent function if this latter executes some commands.

5.4.3 Surrogate-based algorithms

Additionally to the balance between convergence and diversity, the trade-off between exploitation and exploration is to be specified in surrogate-based optimization. Minimizing the predicted objective vectors (POVs) produced by the surrogate boosts exploitation of known promising regions of the search space. Conversely, maximizing the predictive uncertainty enhances exploration of unknown regions.

Surrogate-based methods for preventive treatment distribution

Despite the relative computational expensiveness of infectious disease transmission simulators, surrogate-based optimization has been rarely applied to the field. In [Bri+20], we harnessed surrogate models to determine the allocation of preventive treatments that minimize the number of deaths caused by tuberculosis in the Philippines. The identification of the regime for tuberculosis antibiotic treatments with lowest time and doses is formulated as a SO problem in [Cic+17] and solved by a method relying on a RBF surrogate model. The work presented in [Mii+21] deviates from this present study in that it aims to conceive a model prescribing the actions to perform according to a given situation. It is actually more related to artificial neural network hyper-parameters and architecture search. What is called "surrogate" in [Mii+21] is actually denominated "simulator" in simulation-based optimization.

AB-MOEA

The Adaptive Bayesian Multi-Objective Evolutionary Algorithm (AB-MOEA) has been recently designed to handle expensive MO optimization problems [Wan+20b]. This surrogate-based method, described in Algorithm 18, is made of three steps. The first step consists in proposing a set of new candidates by minimizing the POVs thanks to RVEA (line 5). As no predictive uncertainty is used, only exploitation is favored. The second step consists in re-evaluating the last population returned by RVEA thanks to an adaptive function $\boldsymbol{f_{ada}}$ (line 8) defined by

$$\boldsymbol{f_{ada}}(\boldsymbol{x}, \alpha) = (1 - \alpha)\hat{\boldsymbol{f}}(\boldsymbol{x})./\hat{\boldsymbol{f}}_{\boldsymbol{max}} + \alpha\hat{\boldsymbol{s}}^{\boldsymbol{2}}(\boldsymbol{x})./\hat{\boldsymbol{s}}^{\boldsymbol{2}}_{\boldsymbol{max}} \tag{5.12}$$

where

$$\alpha = -0.5\cos\left(\frac{b_c}{b}\pi\right) + 0.5 \tag{5.13}$$

where $\frac{b_c}{b}$ is the proportion of the budget already spent, $./$ is the element-wise division, $\hat{\boldsymbol{f}}_{\boldsymbol{max}}$ is the per-objective maximum POV observed in the last population returned by RVEA and $\hat{\boldsymbol{s}}^{\boldsymbol{2}}_{\boldsymbol{max}}$ is the per-objective maximum predictive variance. At the beginning of the search ($\alpha \approx 0$), $\boldsymbol{f_{ada}}$ favors convergence to the true PF by minimizing the POVs. As the search proceeds, α increases and so minimization of the predictive variance is included to reinforce exploitation. At the third step, q candidates are retained for simulation based on an adaptive sampling criterion described in Algorithm 19. The sampling criterion is similar to the reference vector guided replacement of RVEA: first the POVs are translated according to Equation (5.5), then for each POV an angle-based distance from the closest reference vector is computed (lines 4 to 8). During the first part of the search (when $\alpha < 0.5$), the distance is the angle to the set of reference vectors (line 5) thus favoring diversity. During the second part of the search, the distance is the angle penalized distance of Equation (5.8) (line 7) thus favoring both convergence and diversity. Afterwards, the candidates are sorted with a lower distance indicating a better promise. The sampling is realized per sub-population so that the lowest distances per sub-population are considered first.

The implementation of AB-MOEA resorts to the elements already available in pySBO such as the problems, the evolutionary operators, the surrogates and the implementation of RVEA. However, the implementation of AB-MOEA is naive in the sense that no particular effort have been dedicated yet to factorize the code corresponding to the new algorithmic components. In a nutshell, no new classes have been created. Future developments of the platform will target enhancements in this respect. For instance, the adaptive function $\boldsymbol{f_{ada}}$ is the same than the adaptive EC exposed by the `Adaptive_Wang2020_EC` class.

For the moment, the EC only accepts populations of single-objective individuals while $\boldsymbol{f_{ada}}$ is designed for multiple objectives. It is also worth to note that the angle penalized distance is employed in both RVEA and in the adaptive sampling criterion of AB-MOEA, therefore a dedicated class would be useful. Besides, in the first phase of the adaptive sampling criterion, another distance, namely the angle to the set of reference vectors is invoked. The use of different distances advocates the creation of a collection of classes representing distances and embedding the possibility to sort a population according to it.

SAEA-ME

The Surrogate-Assisted Evolutionary Algorithm for Medium Scale Expensive problems (SAEA-ME) shows to perform well on MO problems with more than 10 decision variables in [Rua+20]. In SAEA-ME, dissected in Algorithm 20, NSGA-II is used (line 5) to optimize a six-objective acquisition function where the three first objectives are the POVs and the last three objectives are the POVs minus the predictive variances (similar to LCB with $\lambda_{lcb} = 1$). From the set of proposed candidates, the $\lfloor \frac{q}{2} \rfloor$ ones showing the best hyper-volume contribution considering the POVs are first retained for parallel simulation (line 6 and 7). Then, the ones demonstrating the best hyper-volume contribution regarding the POVs minus two variances are simulated (line 8 and 9). The additional dimensionality reduction feature proposed in [Rua+20] is not considered here as it consumes computational budget and can be applied to any method.

In order to optimally reuse the code available in pySBO, two new classes are set up to implement SAEA-ME. The first one, called `MO_POV_LCB_IC`, implements the non-dominated and crowding distance sorting of a population for the problem $\min(\hat{\boldsymbol{f}}, \hat{\boldsymbol{f}} - \hat{\boldsymbol{s}}^{\boldsymbol{2}})$. The implementation of NSGA-II can consequently be directly reused by basing the selection and replacement step on this new sorting (in the same spirit as in Algorithm 5). The second novel class, called `MO_POV_LCB_EC`, serves to sort a population by placing first the $\lfloor \frac{q}{2} \rfloor$ individuals showing the best hyper-volume contribution for the problem $\min(\hat{\boldsymbol{f}})$ and then the $\lfloor \frac{q}{2} \rfloor$ ones for the problem $\min(\hat{\boldsymbol{f}} - 2\hat{\boldsymbol{s}}^{\boldsymbol{2}})$. The q candidates to simulate are thus easily identified from the set of proposals.

Gaussian Processes for multiple objectives

The multi-task Gaussian Process [BCW08] is implemented by the `GP_MO` class in pySBO *via* the GPyTorch library [Gar+18]. Using a multi-task GP to model multiple objectives has been realized in [Xia+14] to control quality in sheet metal forming. In a traditional regression GP [Ras06], a kernel function is specified to model the covariance between the inputs, thus allowing the model to learn the input-output mapping and to return predictions and predictive uncertainties. In the multi-task GP, inter-task dependencies are also taken into account in the hope of improving over the case where the tasks are decoupled. The BNN_MCD described in the first chapter of this thesis also supports multiple outputs such as the basic ANNs. It is embedded within pySBO through the `BNN_MCD` class.

5.5 Numerical experiments

5.5.1 Protocol

The five MO algorithms exhibited in the previous section are applied to the Covid-19 vaccine distribution problem. The computational budget is set to two hours on 18 computing cores, thus allowing 18 simulations to be realized in parallel. The simulation duration varies from one solution to another from 13 to 142 seconds on one computing core. Ten runs of the searches are carried out to ensure statistical robustness of the comparisons. The reference point for hyper-volume calculation is set to an upper bound for each objective $(32.10^6; 32.10^6; 1.5)$.

The surrogate-free approaches NSGA-II, RVEA and RVEA* are equipped with either the *distrib-X*, the *2-points* or the *intermediate* cross-over operator. For NSGA-II, the population size n_{pop} is set to 108 or 162, to avoid the idling of the computing cores and being close to the recommended value of 100 given in [Deb+02]. For RVEA and RVEA*, we choose $n_{pop} = 105$ or 171 to comply with the constraint imposed by the reference vectors initialization and to keep values close to those imposed for NSGA-II. Ten initial populations composed of 171 simulated solutions are generated to start the algorithms. Each initial population is made at 85% of solutions randomly sampled within the feasible search space and at 15% of candidates picked out on the boundary. When $n_{pop} < 171$, only the best n_{pop} candidates according to the non-dominated sorting defined in [Deb+02] are retained. For RVEA and RVEA*, a scaled version of the problem, where the first two objectives are divided by 1000, is also considered to demonstrate the effect of the objectives scales on the behavior of the methods.

The surrogate-based approaches AB-MOEA and SAEA-ME only integrate the *distrib-X* operator and use all the 171 initial samples as initial database. For RVEA in AB-MOEA, $n_{pop} = 105$ and the number of generations is fixed to 20 as recommended in [Wan+20b], while $n_{pop} = 76$ for NSGA-II in SAEA-ME according to the guidance provided in [Rua+20] and the population evolves for 100 generations. The surrogate is the multi-task GP and five different kernel functions are considered for comparison: the well-known Radial Basis Functions kernel is denoted *rbf*, the Matern kernel [Ste99] with hyper-parameter $\nu = 1.5$ and 2.5 are called *matern1.5* and *matern2.5* respectively and the Spectral Mixture kernel proposed in [WA13] is also raised with 2 and 4 components, denominated *sm2* and *sm4* respectively.

5.5.2 Empirical analysis

Table 5.2 shows the ranking of the algorithms according to the final hyper-volumes averaged over the ten runs. It can be observed in Table 5.2 that all the surrogate-based strategies outperform all those without surrogate. In particular, SAEA-ME with the *matern1.5* kernel is the best approach. The multi-task GP equipped with the *matern1.5* covariance function is preferred in both the SAEA-ME and AB-MOEA frameworks. Regarding the surrogate-free methods, NSGA-II with the *distrib-X* cross-over mechanism and $n_{pop} = 108$ yields the best averaged hyper-volume. It is worth noticing that the *distrib-X* operator, specifically designed for the problem at hand, is to put forward as it surpasses both the *intermediate* and the *2-points* strategies in all contexts. Among the RVEAs, the best variant is RVEA* with $n_{pop} = 105$ and the *distrib-X* cross-over thus indicating a possibly degenerated or disconnected PF. Indeed, the PF is certainly degenerated as indicates Figure 5.4 where are plotted the objective vectors from the ten final NDFs obtained by SAEA-ME with the *matern1.5* kernel. When analyzing the influence of objectives scales over the efficiency of RVEA and RVEA*, the conclusions drawn in [Che+16] are confirmed

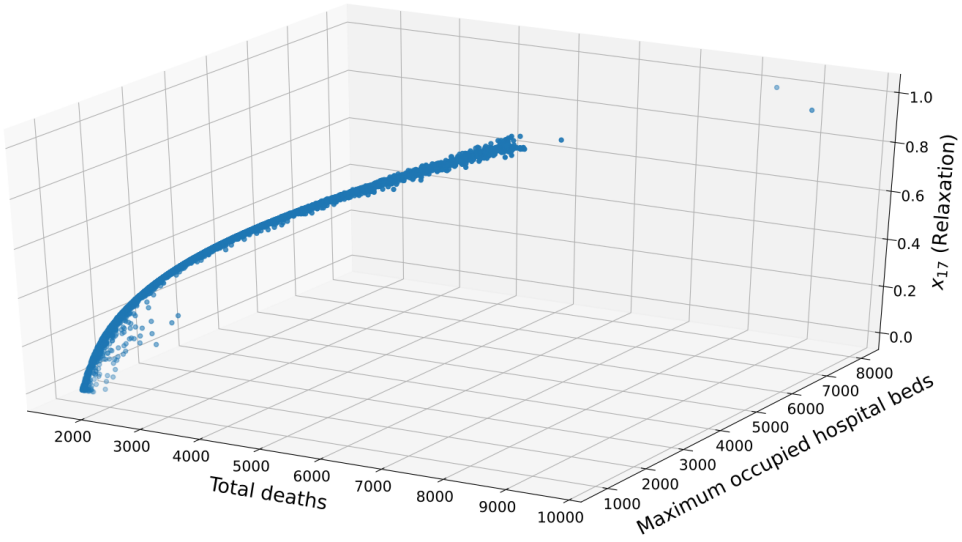

Figure 5.4: Best NDFs from the 10 runs for SAEA-ME with matern1.5 kernel on the Covid-19 vaccine distribution problem.

as both algorithms are more appropriate when objectives have similar scales. Indeed, the three objectives lie in $[1655; 13,762]$, $[843; 10,962]$ and $[0; 1]$, respectively. The previous ranges are approximated *a posteriori* based on 250,664 simulations performed in RVEA and RVEA* on the original problem. The necessity to adequately scale the objectives brings a disadvantage to RVEAs as the scaling weights are tedious to define especially in the context of black-box expensive simulations. Another drawback is the constraints on the population size preventing to totally impede the idling of computing cores in all scenarios.

Figure 5.5 monitors the averaged hyper-volume as the search proceeds for the best strategy per category according to Table 5.2. The hyper-volume improves sharply at the very beginning of the search for the surrogate-based methods and reaches convergence rapidly (around 300 to 500 simulations). NSGA-II improves much slower but seems not to have converged at the end of the execution. By the right extremities of the curves, it could be expected that the hyper-volume returned by NSGA-II exceeds the one from AB-MOEA for larger numbers of simulations. However, reiterating the experiments for a time budget of four hours has not allowed to verify this expectation. Figure 5.5 specifies that the impact of objectives scaling on RVEAs appears from around 300 simulations. In the setting of a capped computational budget, it is important to strongly favor convergence and exploitation at the onset of the search. SAEA-ME and AB-MOEA realizes this by minimizing the POVs at the top beginning of the execution. The difference between the two approaches lies in the incorporation of the predictive uncertainty. In SAEA-ME, a degree of exploration is maintained by maximization of the predictive variance. Conversely, minimization of the predictive uncertainty is involved at latter stages in AB-MOEA. In spite of the convergence-oriented strategy adopted by RVEAs at the early stages of the search, the embedded mechanism set up to handle many objectives is quite heavy and reveals to be unsuitable when the computational budget is restricted. Indeed, in [Che+16] the algorithms are run from 500 to 1,000 generations while 10 to 20 generations are allowed by our computational budget.

Table 5.2: Ranking of the MO surrogate-based and surrogate-free approaches according to the averaged final hyper-volumes over the 10 runs on the Covid-19 vaccine distribution problem.

Algorithm	Cross-over operator	Population size	GP kernel	Objectives scaling	Averaged final Hyper-volume ($\times 10^{10}$ $+1.535 \times 10^{15}$)
SAEA-ME	***distrib-X***	**76**	**matern1.5**	-	**80.1800**
SAEA-ME	*distrib-X*	76	*matern2.5*	-	80.1610
SAEA-ME	*distrib-X*	76	*rbf*	-	79.9541
SAEA-ME	*distrib-X*	76	*sm2*	-	79.6701
AB-MOEA	*distrib-X*	105	*matern1.5*	-	79.6200
AB-MOEA	*distrib-X*	105	*matern2.5*	-	79.5879
SAEA-ME	*distrib-X*	76	*sm4*	-	79.5789
AB-MOEA	*distrib-X*	105	*sm4*	-	79.4861
AB-MOEA	*distrib-X*	105	*sm2*	-	79.4841
AB-MOEA	*distrib-X*	105	*rbf*	-	79.4304
NSGA-II	*distrib-X*	108	-	-	79.3337
NSGA-II	*distrib-X*	162	-	-	79.1876
RVEA*	*distrib-X*	105	-	yes	77.2805
RVEA*	*distrib-X*	171	-	yes	77.2514
RVEA	*distrib-X*	171	-	yes	77.1287
RVEA	*distrib-X*	105	-	yes	77.0117
NSGA-II	*intermediate*	108	-	-	76.9946
NSGA-II	*intermediate*	162	-	-	76.8320
NSGA-II	*2-points*	162	-	-	75.6959
NSGA-II	*2-points*	108	-	-	75.5889
RVEA	*distrib-X*	105	-	-	75.5184
RVEA*	*intermediate*	171	-	yes	75.3816
RVEA*	*intermediate*	105	-	yes	75.2841
RVEA	*intermediate*	171	-	yes	75.2006
RVEA	*intermediate*	105	-	yes	75.1562
RVEA*	*distrib-X*	105	-	-	75.1555
RVEA*	*distrib-X*	171	-	-	75.1372
RVEA	*distrib-X*	171	-	-	75.0563
RVEA*	*2-points*	171	-	yes	74.9803
RVEA	*2-points*	171	-	yes	74.9195
RVEA	*2-points*	105	-	yes	74.7692
RVEA*	*2-points*	105	-	yes	74.7535
RVEA*	*intermediate*	105	-	-	74.5607
RVEA	*intermediate*	105	-	-	74.5585
RVEA	*intermediate*	171	-	-	74.4959
RVEA*	*intermediate*	171	-	-	74.4266
RVEA	*2-points*	171	-	-	74.3694
RVEA	*2-points*	105	-	-	74.3518
RVEA*	*2-points*	171	-	-	74.3264
RVEA*	*2-points*	105	-	-	74.2507

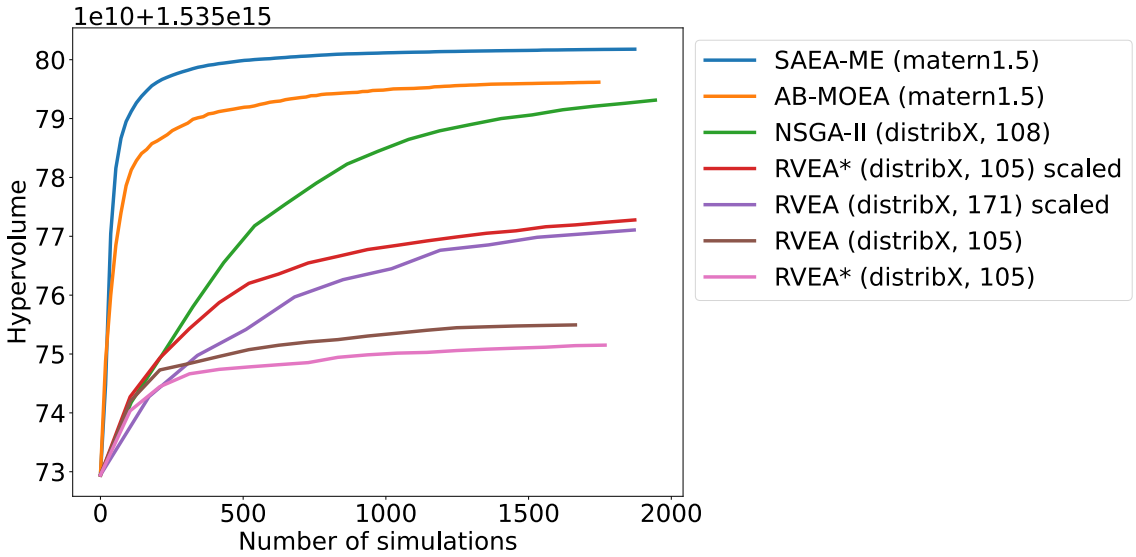

Figure 5.5: **MO optimization of Covid-19 vaccine distribution.** Averaged hypervolume according to the number of simulations.

Reducing the solving time of moderately expensive optimization problems where the simulation lasts less than five minutes may enable to manage optimization under uncertainty. As the calibration of the simulation tool is uncertain, multiple configurations of its parameters can be considered, resulting in multiple optimization exercises to be executed and thus enabling to gain insight about the variability of the results.

5.5.3 Resulting vaccine distribution plan

The optimal allocation plan implies providing 70% of the doses to the 10-19 years old age-group and 30% to the 20-29 age-group during phase 2 according to Figure 5.6. In phase 3, 70% of the doses are assigned to 20-29 years old individuals and 15% to both the 40-49 and 10-19 age-categories. This plan prioritizes the vaccination of younger adults as they are the most transmitting cohort because of their high contact rate in the population [PCJ17]. Nevertheless, the present results have to be considered with caution. Since our experiments date back to the beginning of 2021, few feedback about vaccination efficiency was available. It is assumed here that the vaccine reduces transmission although it might not be the case for the Omicron variant of concern that started to break through the world at the end of 2021. Our results are similar to those presented in [Wey+05; MG09] for influenza. From Figure 5.7 where the total number of deaths and the maximum number of occupied hospital beds are displayed with respect to the relaxation variable x_{17}, the alleviation of the physical distancing reveals to trigger an augmentation of the hospital occupancy and deaths.

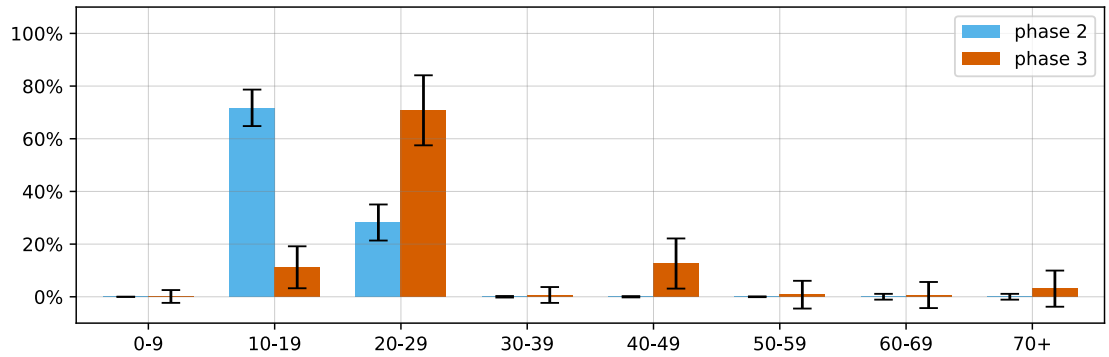

Figure 5.6: **vaccine distribution according to age-categories.** Averaged solutions from the best final NDFs returned by the 10 runs for SAEA-ME with *matern1.5* kernel.

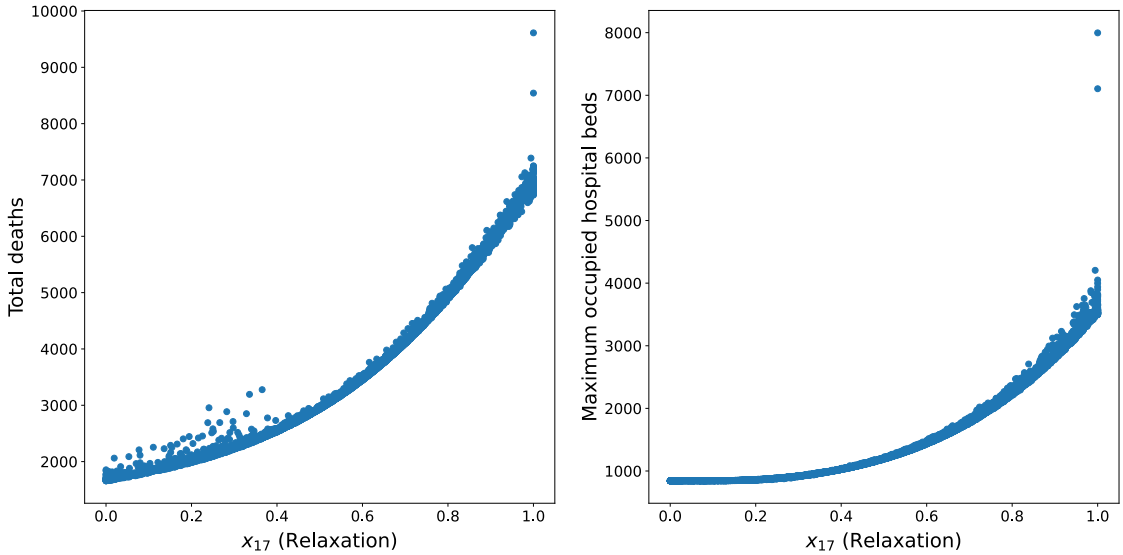

Figure 5.7: Total number of deaths and maximum number of occupied hospital beds according to relaxation of the physical distancing x_{17}. Best NDFs from the 10 runs for SAEA-ME with *matern1.5* kernel.

5.6 Conclusion

In this chapter, we have studied the design of a scalable software architecture to disseminate the techniques of P-SBO to the larger community. The resulting pySBO Python platform and its associated documentation are open-source and publicly available on-line through a Github repository and a dedicated web-page respectively. The abstract classes and the relationships offered by the object-oriented paradigm in Python ensure the extensibility and flexibility of the code. The numerous algorithmic components (ECs, surrogates, *etc.*) and the full approaches (q-EGO, SaaEF, *etc.*) already embedded into pySBO empower utility and reusability. The review of related software promotes pySBO for the ability to set up both P-SAEAs and P-SDAs for moderately and very expensive problems respectively. The support for multi-objective and inclusion of parallel computations are also key features that are uncommon in computer programs suite to P-SBO.

Multi-objective surrogate-based and surrogate-free algorithms like SAEA-ME, AB-MOEA and RVEA are implemented within pySBO as a proof of concept for extensibility. The multi-objective problem that is considered in the numerical experiments consists in allocating a limited number of vaccine doses to the age-groups of a population to minimize mortality, hospital occupancy and mobility restrictions. The handling of this Covid-19-related application demonstrates the utility of pySBO to the field of infectious disease and resource allocation. The comparison of the multi-objective approaches emphasizes the usefulness of pySBO to the domain of P-SBO.

At the time of writing this paragraph, pySBO is made of 55 classes plus 10 actual algorithms implemented for a total number of lines of code estimated to 5,000. Three users are known and the platform has been downloaded 17 times. For the moment, the optimization procedure in pySBO is mainly centered on evolutionary computations. Future developments will be devoted to add other optimizers such as Particle Swarm Optimization and to implement new P-SBOAs based on it.

Conclusions and perspectives

By combining the benefits of parallel computing, machine learning and optimization, P-SBOAs are built to solve black-box computationally expensive simulation-based optimization problems. In this thesis, we study the design of P-SAEAs, P-SDAs and hybrid algorithms according to the ruggedness of the search landscape and the affordable computational budgets. Particularly, three design aspects, the surrogate model, the acquisition process and the definition of promisingness represent the three dimensions of the design space of P-SBOAs as considered in this work. The ideal situation of a robust P-SBOA able to deal with many search landscapes and computational budgets serves as a guiding point at the horizon.

An important novelty brought by this thesis is to consider the distinction between moderately and very computationally expensive problems. To do so, a capped duration on a prefixed number of computing cores is selected as budget definition. The motivation is to realistically reproduce the situation faced by the users, especially when the surrogate training may become a significant workload.

As a first contribution, we proposed SaaEF, a P-SAEA where the surrogate is used both as an evaluator and a filter within an evolutionary algorithm. On the one hand, the surrogate evaluations and the parallel simulations allow to save computational budget. On the other hand, the filtering gives more opportunity to the reproduction operators to come up with promising new solutions. We choose the BNN_MCD as surrogate model to mimic the simulator in a fast way. Monte-Carlo Dropout consists in randomly deactivating neurons from a network thus generating sub-networks subsequently used to compute the predictive objective value. Three novel ways of defining the promisingness through ensembles of evolution controls are suggested. The dynamic and adaptive ensembles change the trade-off between exploration and exploitation during the search while the committee combines evolution controls by voting.

The second contribution consists in relying on sub-surrogates to create P-SDAs. The multiple sub-surrogates intervene in multiple simultaneous optimizations to acquire new candidates that are subsequently simulated in parallel. Sub-surrogate sampling emulates the use of ensembles of surrogates with the advantage of requiring only one surrogate update per acquisition process. Two strategies are proposed, differing on the way the sub-surrogates are constituted. In the first approach, sub-networks are sampled from a global neural network by Monte-Carlo Dropout (q-subnets). In the second approach, a Monte-Carlo Markov Chain sampling is triggered on the posterior distribution of the parameters of a Gaussian Process (q-post-HMC).

As a third contribution, the Hybrid Sequential Acquisition Process (HSAP) is proposed to retain the advantages of P-SAEAs and P-SDAs. First, to preserve the benefit of P-SDAs on tight computational budgets, about a hundred candidates are acquired through infill criterion optimization. Afterwards, a population is evolved through informed reproduction operators with the aim of continuously improving over the best located solution. Comparisons are conducted with a state-of-the-art method and HCAP, another hybrid

strategy we propose, where both acquisition processes are concurrently performed during the entire search. The HSAP algorithm shows the best parallel scalability and provides the best overall solution on the Covid-19 problem for a budget of 30 minutes on 144 computing cores.

The fourth contribution is the pySBO Python modular platform developed during this thesis to the dissemination of P-SBO tools. The software demonstrates an extensible and flexible skeleton filled with actual algorithmic components such as evolution controls and surrogates. The algorithms dissected in this work are provided within pySBO as ready-to-use programs. The accessibility is guaranteed by the open-source licence and the documentation available through a dedicated web-page. A multi-objective problem of Covid-19 vaccine distribution as well as recent surrogate-based and surrogate-free multi-objective algorithms are implemented and compared to exemplify the use and the extensibility of the platform.

One of the main objectives of this thesis was to investigate the design of P-SBOAs that prove robustness with respect to the computational budget and the search landscape characteristics. For moderately expensive problems, allowing more than around 1500 simulations, P-SAEAs are to be put forward. The most probable reason is that they better prevent being trapped in a local basin of attraction than P-SDAs as they deliver a higher degree of exploration. Besides, historical knowledge about the search landscape is embedded in two locations in P-SAEAs, in the evolving population and in the parameters of the trained surrogate. Recovering from a misguided search path may be easier than with P-SDAs where only the surrogate's parameters drive the search. However, for very expensive problems, P-SDAs are promoted. By optimizing the infill criterion, more efforts are engaged into exploitation than in P-SAEAs and good new solutions are obtained quickly. Regarding the promisingness in P-SAEAs, it seems often better to favor exploration at the beginning of the search and exploitation at the latter stages. The best metric for exploration is the distance to the database of already simulated solutions and the one for exploitation is the predicted objective value. The surrogate can therefore gain global knowledge of the landscape before being exploited to locate narrower regions of interest to scrutinize. In P-SDAs, the predictive uncertainty offered by the surrogate is a more informative metric of exploration as eliminating such uncertainty is of primary concern to detect new better solutions. The Pareto-based and adaptive evolution controls should be privileged in general in P-SDAs to aggregate the predictive uncertainty and the predicted objective.

On the Schwefel benchmark whose landscape demonstrates both multi-modality and weak global structure, BNN_MCD has shown a more adequate predictive capacity than the Gaussian Processes. However, the Gaussian Processes are more convenient on other landscapes. The training time of BNN_MCD remains low even when more training data accumulate as in the context of moderately expensive problems. For Gaussian Processes, restraining the training set to the last few simulations is an interesting way of limiting the training time and sometimes even improves the resolution quality. For a rugged landscape and a moderately expensive problem, allocating more budget to simulation than to surrogate training has been seen as a good idea.

Determining the suitable surrogate-optimizer coupling according to the search landscape characteristics is still an open question. Further comparisons including a broader range of benchmark problems must be conducted and landscape analysis tools should be employed in an attempt to categorize the landscapes. For what we have observed in our experiments related to P-SDAs, q-post-HMC and q-subnets can produce decisive results on the Schwefel problem for a tight and moderate computational budget respectively. Nevertheless, this observation is not retrieved on the Covid-19 application that has also been

qualified as multi-modal and weak global structure by *a posteriori* landscape analysis. More generally, for a very expensive problem, q-EGO seems to perform consistently well over all the optimization problems addressed in this thesis. For a moderately expensive problem, SaaEF is the more robust even if it is not the best strategy in all problems.

Merging the best of both surrogate-based frameworks, P-SAEAs and P-SDAs, HSAP is specifically designed to solve the Covid-19-related problem. The new algorithm integrates both BNN_MCD and a Gaussian Process as surrogates and dynamic and voting ensembles of ECs. HSAP is the best performing method on the moderately expensive problem of Covid-19 contact reduction and it scales very well to the number of computing cores.

The outcome of the resolution of the Covid-19 applications have been transmitted to the World Health Organization that redirected it to the studied countries. As these countries have no obligation of feedback on the utility of the formulated recommendations, it is tedious to evaluate the real impacts of our results.

As future research directions, we have identified some challenging perspectives summarized in the following:

- The dynamic ensembles of evolution controls have proven to be reliable in P-SAEAs for moderately expensive problems. The issue raised by these mechanisms are the need for the user to tweak the switch from exploration to exploitation. The attempts conducted to remove the user-defined parameters by setting adaptive ensembles of evolution controls have not borne fruit yet. The early stopping criterion will be enhanced in the future by integrating gradient information about the convergence curve.

- In q-subnets, the sub-networks are sampled randomly without preoccupation of the diversity. The lack of distinctiveness in the set of sub-surrogates may provoke a waste of computational budget by simulating multiple times the same solution. Improving the sampling in q-subnets is meaningful and a first lead would consist of setting groups of neurons that can not be deactivated at a same time.

- The characteristics of the search landscape have been identified as an impacting factor that guide the choice of algorithmic components such as surrogate models. This suggests to include landscape analysis metrics and surrogate predictive error to inform the optimization process. The dispersion metric as well as the distances to the nearest neighbors and the best nearest neighbors have been employed to quantify the multi-modality and the weakness of the global structure of search landscapes. These indicators should be considered in building P-SBOAs to empower robustness.

- The HSAP method developed in this thesis proves robustness with respect to the computational budget. In HSAP, the switch between the two acquisition processes that are successively leveraged during the search is fixed in the current version of the algorithm. Determining automatically the optimal step of the search to trigger the switch is a feature that worth some effort in the future. As in the case of adaptive evolution controls, relying on the gradient of the convergence curve is a potential direction.

- The optimizations realized in pySBO are mainly centered on evolutionary computations and synchronous parallel simulations. Future developments of the platform will target the inclusion of other meta-heuristics and parallel schemes for asynchronous simulations. The integration of Particle Swarm Optimization is already on its way and will allow to perform comparisons with P-SAEAs as the surrogate-optimizer couplings are similar.

Bibliography

[Aa15] M. Abadi and A. Agarwal et al. *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: https://www.tensorflow.org/.

[AD06] C. Audet and J. E. Dennis. "Mesh Adaptive Direct Search Algorithms for Constrained Optimization". In: *SIAM Journal on Optimization* 17.1 (2006), pp. 188–217. DOI: 10.1137/040603371.

[AKG09] V. Asouti, I. Kampolis, and K. Giannakoglou. "A grid-enabled asynchronous metamodel-assisted evolutionary algorithm for aerodynamic optimization". In: *Genetic Programming and Evolvable Machines* 10 (Dec. 2009), pp. 373–389. DOI: 10.1007/s10710-009-9090-5.

[AKN21] H. Anahideh, L. Kang, and N. Nezami. "Fair and diverse allocation of scarce resources". In: *Socio-Economic Planning Sciences* (2021), p. 101193. ISSN: 0038-0121. DOI: https://doi.org/10.1016/j.seps.2021.101193.

[AS16] T. Akhtar and C. A. Shoemaker. "Multi objective optimization of computationally expensive multi-modal functions with RBF surrogates and multi-rule selection". In: *Journal of Global Optimization* 64.1 (Jan. 2016), pp. 17–32. ISSN: 1573-2916. DOI: 10.1007/s10898-015-0270-y. URL: https://doi.org/10.1007/s10898-015-0270-y.

[AS19] T. Akhtar and C. A. Shoemaker. "Efficient Multi-Objective Optimization through Population-based Parallel Surrogate Search". In: *CoRR* (2019). URL: http://arxiv.org/abs/1903.02167.

[AV07] D. Arthur and S. Vassilvitskii. "k-means++: the advantages of careful seeding". In: *SODA '07: Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms*. New Orleans, Louisiana: Society for Industrial and Applied Mathematics, 2007, pp. 1027–1035. ISBN: 978-0-898716-24-5. URL: http://portal.acm.org/citation.cfm?id=1283494.

[Ba19] F. Biscani and al. *esa/pagmo2: pagmo 2.10*. Jan. 2019. DOI: 10.5281/zenodo.2529931.

[Bal+19] M. Balandat et al. "BoTorch: Programmable Bayesian Optimization in PyTorch". In: *CoRR* abs/1910.06403 (2019). arXiv: 1910.06403. URL: http://arxiv.org/abs/1910.06403.

[Bau+] M. Baudin et al. *pyDOE: The experimental design package for python*. https://pythonhosted.org/pyDOE/.

[BCW08] E. V. Bonilla, K. Chai, and C. Williams. "Multi-task Gaussian Process Prediction". In: *Advances in Neural Information Processing Systems*. Vol. 20. Curran Associates, Inc., 2008. URL: https://proceedings.neurips.cc/paper/2007/file/66368270ffd51418ec58bd793f2d9b1b-Paper.pdf.

[BD22] J. Blank and K. Deb. *pysamoo: Surrogate-Assisted Multi-Objective Optimization in Python*. 2022. arXiv: 2204.05855 [cs.NE].

[Beh+11] S. Behnel et al. "Cython: The best of both worlds". In: *Computing in Science & Engineering* 13.2 (2011), pp. 31–39.

[Ber+19] N. Berveglieri et al. "Surrogate-assisted multi-objective optimization based on decomposition: a comprehensive comparative analysis". In: *Genetic and Evolutionary Computation Conference (GECCO 2019)*. 2019.

[Bes+22] L. Bessi et al. *Surrogates.jl: Surrogate modeling and optimization for scientific machine learning (SciML)*. https://surrogates.sciml.ai/dev/. 2022.

[Bin+18] E. Bingham et al. "Pyro: Deep Universal Probabilistic Programming". In: *Journal of Machine Learning Research* (2018).

[Bis+14] B. Bischl et al. "MOI-MBO: Multiobjective Infill for Parallel Model-Based Optimization". In: vol. 8426. Feb. 2014, pp. 173–186. DOI: 10.1007/978-3-319-09584-4_17.

[Bis06] C. M. Bishop. *Pattern Recognition and Machine Learning*. Berlin, Heidelberg: Springer-Verlag, 2006. ISBN: 0387310738.

[Bli+21] L. Bliek et al. *EXPObench: Benchmarking Surrogate-based Optimisation Algorithms on Expensive Black-box Functions*. 2021. DOI: 10.48550/ARXIV.2106.04618.

[Bre+18] B. Breiderhoff et al. "Expensive optimization exemplified by ECG simulator optimization". In: 2018.

[Bri+20] G. Briffoteaux et al. "Evolution Control for parallel ANN-assisted simulation-based optimization application to Tuberculosis Transmission Control". In: *Future Generation Computer Systems* 113 (2020), pp. 454–467. ISSN: 0167-739X. DOI: https://doi.org/10.1016/j.future.2020.07.005.

[BSK05] D. Buche, N. N. Schraudolph, and P. Koumoutsakos. "Accelerating evolutionary algorithms with Gaussian process fitness function models". In: *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)* 35.2 (2005), pp. 183–194. DOI: 10.1109/TSMCC.2004.841917.

[Bub+21] K. M. Bubar et al. "Model-informed COVID-19 vaccine prioritization strategies by age and serostatus". In: *Science* 371.6532 (2021), pp. 916–921. DOI: 10.1126/science.abe6959.

[Buh+21] C. Buhat et al. "Using Constrained Optimization for the Allocation of COVID-19 Vaccines in the Philippines". In: *Applied health economics and health policy* 19(5) (2021), pp. 699–708. DOI: https://doi.org/10.1007/s40258-021-00667-z.

[Cal+20] J. M. Caldwell et al. *Modelling COVID-19 in the Philippines: technical description of the model*. Tech. rep. Monash University, 2020.

[Cap+05] F. Cappello et al. "Grid'5000: a large scale and highly reconfigurable grid experimental testbed". In: *The 6th IEEE/ACM International Workshop on Grid Computing, 2005.* 2005. DOI: 10.1109/GRID.2005.1542730.

[CH21] M. Claeson and S. Hanson. "COVID-19 and the Swedish enigma". In: *The Lancet* 397.10271 (2021), pp. 259–261. ISSN: 0140-6736. DOI: https://doi.org/10.1016/S0140-6736(20)32750-1.

[Cha+15] A. Chaudhuri et al. "Experimental Flapping Wing Optimization and Uncertainty Quantification Using Limited Samples". In: *Struct. Multidiscip. Optim.* 51.4 (Apr. 2015), pp. 957–970. ISSN: 1615-147X. DOI: 10.1007/s00158-014-1184-x. URL: https://doi.org/10.1007/s00158-014-1184-x.

[Cha+20] S. Chang et al. “Modelling transmission and control of the COVID-19 pandemic in Australia”. In: *Nature Communications* 11,5710 (Mar. 2020). DOI: https://doi.org/10.1038/s41467-020-19393-6.

[Che+16] R. Cheng et al. “A Reference Vector Guided Evolutionary Algorithm for Many-Objective Optimization”. In: *IEEE Transactions on Evolutionary Computation* 20.5 (2016), pp. 773–791. DOI: 10.1109/TEVC.2016.2519378.

[Cho15] F. Chollet. *Keras*. https://keras.io. 2015.

[Chu+17] T. Chugh et al. “A data-driven surrogate-assisted evolutionary algorithm applied to a many-objective blast furnace optimization problem”. In: *Materials and Manufacturing Processes* 32.10 (2017), pp. 1172–1178. URL: https://doi.org/10.1080/10426914.2016.1269923.

[Chu+19] T. Chugh et al. “A survey on handling computationally expensive multiobjective optimization problems with evolutionary algorithms”. In: *Soft Computing* 23 (May 2019). DOI: 10.1007/s00500-017-2965-0.

[Cic+17] J. M. Cicchese et al. “Applying Optimization Algorithms to Tuberculosis Antibiotic Treatment Regimens”. English. In: *Cellular and Molecular Bioengineering* 10,6 (Dec. 2017), pp. 523–535. DOI: https://doi.org/10.1007/s12195-017-0507-6.

[Cor02] J. A. Cornell. *Experiments with Mixtures: Designs, Models, and the Analysis of Mixture Data*. John Wiley & Sons, 2002.

[Deb+01] K. Deb et al. *Scalable Test Problems for Evolutionary Multi-Objective Optimization*. Tech. rep. Computer Engineering and Networks Laboratory (TIK), Swiss Federal Institute of Technology (ETH, 2001.

[Deb+02] K. Deb et al. “A fast and elitist multiobjective genetic algorithm: NSGA-II”. In: *IEEE Transactions on Evolutionary Computation* 6.2 (2002), pp. 182–197. DOI: 10.1109/4235.996017.

[Del12] C. Delannoy. *Programmer en Java*. Eyrolles, 2012. ISBN: 9782212135664. URL: https://books.google.be/books?id=1OStv%5C_D82I4C.

[DF21] L. Dalcin and Y. L. Fang. “mpi4py: Status Update After 12 Years of Development”. In: *Computing in Science Engineering* 23.4 (2021), pp. 47–54. DOI: 10.1109/MCSE.2021.3083216.

[Día+16] A. Díaz-Manríquez et al. “A Review of Surrogate Assisted Multiobjective Evolutionary Algorithms”. In: *Computational Intelligence and Neuroscience* 2016 (2016). doi: https://doi.org/10.1155/2016/9420460, p. 14. ISSN: Article ID 9420460.

[DN07] K. Deb and P. Nain. “An Evolutionary Multi-objective Adaptive Metamodeling Procedure Using Artificial Neural Networks”. In: *Evolutionary Computation in Dynamic and Uncertain Environments*. Vol. 51. doi: http://dx.doi.org/10.1007/978-3-540-49774-5_13. Berlin, Heidelberg: Springer, 2007. Chap. 13, pp. 297–322. ISBN: 978-3-540-49772-1.

[DTC17] A. Díaz-Manríquez, G. Toscano, and C. A. Coello Coello. “Comparison of metamodeling techniques in evolutionary algorithms”. In: *Soft Computing* 21.19 (Oct. 2017), pp. 5647–5663. DOI: 10.1007/s00500-016-2140-z.

[Duq+20] D. Duque et al. “Timing social distancing to avert unmanageable COVID-19 hospital surges”. In: *Proceedings of the National Academy of Sciences* 117.33 (2020), pp. 19873–19878. ISSN: 0027-8424. DOI: 10.1073/pnas.2009033117.

[EBS19] D. Eriksson, D. Bindel, and C. A. Shoemaker. "pySOT and POAP: An event-driven asynchronous framework for surrogate optimization". In: *arXiv preprint arXiv:1908.00420* (2019).

[EGN06] M. T. M. Emmerich, K. C. Giannakoglou, and B. Naujoks. "Single- and multiobjective evolutionary optimization assisted by Gaussian random field metamodels". In: *IEEE Transactions on Evolutionary Computation* 10.4 (2006), pp. 421–439. DOI: 10.1109/TEVC.2005.859463.

[Emm+02] M. Emmerich et al. "Metamodel—Assisted Evolution Strategies". In: Sept. 2002. ISBN: 978-3-540-44139-7. DOI: 10.1007/3-540-45712-7_35.

[Fen+15] Z. Feng et al. "A multiobjective optimization based framework to balance the global exploration and local exploitation in expensive optimization". In: *Journal of Global Optimization* 61.4 (Apr. 2015), pp. 677–694. DOI: 10.1007/s10898-014-0210-2.

[FSK08a] A. I. J. Forrester, A. Sóbester, and A. J. Keane. "Constructing a Surrogate". In: *Engineering Design via Surrogate Modelling*. John Wiley and Sons, Ltd, 2008. Chap. 2, pp. 33–76. ISBN: 9780470770801. DOI: https://doi.org/10.1002/9780470770801.ch2.

[FSK08b] A. I. J. Forrester, A. Sóbester, and A. J. Keane. *Engineering Design via Surrogate Modelling*. John Wiley and Sons, Ltd, 2008. ISBN: 9780470770801. DOI: https://doi.org/10.1002/9780470770801.

[FSK08c] A. I. J. Forrester, A. Sóbester, and A. J. Keane. "Exploring and Exploiting a Surrogate". In: *Engineering Design via Surrogate Modelling*. John Wiley and Sons, Ltd, 2008. Chap. 3, pp. 77–107. ISBN: 9780470770801. DOI: https://doi.org/10.1002/9780470770801.ch3.

[FSK08d] A. I. J. Forrester, A. Sóbester, and A. J. Keane. "Sampling Plans". In: *Engineering Design via Surrogate Modelling*. John Wiley and Sons, Ltd, 2008. Chap. 1, pp. 1–31. ISBN: 9780470770801. DOI: https://doi.org/10.1002/9780470770801.ch1.

[FVW99] A. Fink, S. Voß, and D. L. Woodruff. "Building Reusable Software Components for Heuristic Search". In: *Operations Research Proceedings 1998*. Berlin, Heidelberg: Springer Berlin Heidelberg, 1999, pp. 210–219. ISBN: 978-3-642-58409-1.

[Gal16] Y. Gal. "Uncertainty in Deep Learning". PhD thesis. University of Cambridge, 2016.

[Gar+18] J. R. Gardner et al. "GPyTorch: Blackbox Matrix-Matrix Gaussian Process Inference with GPU Acceleration". In: *Advances in Neural Information Processing Systems*. 2018.

[GBC16] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[GFL08] B. Glaz, P. P. Friedmann, and L. Liu. "Surrogate based optimization of helicopter rotor blades for vibration reduction in forward flight". In: *Structural and Multidisciplinary Optimization* 35.4 (Apr. 2008), pp. 341–363. ISSN: 1615-1488. DOI: 10.1007/s00158-007-0137-z.

[GH18] D. Greenfeld and E. Holscher. *The RestructuredText Book Documentation*. 2018. URL: https://sphinx-tutorial.readthedocs.io/_/downloads/en/latest/pdf/.

[Gha20] M. Ghasemi. *SKSurrogate: suite of tools to surrogate optimization for expensive functions based on Scikit-learn*. https://sksurrogate.readthedocs.io. 2020.

[GKW17] C. Grobler, S. Kok, and D. N. Wilke. "Simple intuitive multi-objective parallelization of efficient global optimization: simple-ego". In: *World Congress of Structural and Multidisciplinary Optimisation*. Springer. 2017, pp. 205–220.

[Gla+09] B. Glaz et al. "Multiple-Surrogate Approach to Helicopter Rotor Blade Vibration Reduction". In: *AIAA Journal* 47.1 (2009), pp. 271–282. URL: https://doi.org/10.2514/1.40291.

[GLS99] W. Gropp, E. Lusk, and A. Skjellum. *Using MPI: Portable Parallel Programming with the Message-passing Interface*. Scientific and engineering computation. MIT Press, 1999. ISBN: 9780262571326. URL: https://books.google.be/books?id=xpBZ0RyRb-oC.

[Gmy+20] J. Gmys et al. "A comparative study of high-productivity high-performance programming languages for parallel metaheuristics". In: *Swarm Evol. Comput.* 57 (2020), p. 100720. DOI: 10.1016/j.swevo.2020.100720.

[Goe+07] T. Goel et al. "Response surface approximation of Pareto optimal front in multi-objective optimization". In: *Computer Methods in Applied Mechanics and Engineering* 196.4 (2007), pp. 879–893. ISSN: 0045-7825. DOI: https://doi.org/10.1016/j.cma.2006.07.010.

[Gon+06] L. Gonzalez et al. "A generic framework for the design optimisation of multidisciplinary UAV intelligent systems using evolutionary computing". In: *Proceedings of the 44th AIAA Aerospace Sciences Meeting and Exhibit* (2006), pp. 1–19.

[Gra11] A. Graves. "Practical Variational Inference for Neural Networks". In: *Advances in Neural Information Processing Systems 24*. Curran Associates, Inc., 2011, pp. 2348–2356. URL: http://papers.nips.cc/paper/4329-practical-variational-inference-for-neural-networks.pdf.

[GRC10] D. Ginsbourger, R. Le Riche, and L. Carraro. "Kriging is well-suited to parallelize optimization". In: *Computational Intelligence in Expensive Optimization Problems*. Springer series in Evolutionary Learning and Optimization. springer, 2010, pp. 131–162. DOI: 10.1007/978-3-642-10701-6_6.

[GV05] A. Gaspar-Cunha and A. Vieira. "A Multi-Objective Evolutionary Algorithm Using Neural Networks to Approximate Fitness Evaluations." In: *International Journal of Computers, Systems and Signals* 6 (Jan. 2005), pp. 18–36.

[Ha20] C. R. Harris and al. "Array programming with NumPy". In: *Nature* 585.7825 (Sept. 2020), pp. 357–362. DOI: 10.1038/s41586-020-2649-2.

[Han+10] N. Hansen et al. *Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions*. Tech. rep. 2010.

[Han+21] S. Han et al. "Time-varying optimization of COVID-19 vaccine prioritization in the context of limited vaccination capacity". In: *Nature communications* 12.1 (Aug. 2021), p. 4673. ISSN: 2041-1723. DOI: 10.1038/s41467-021-24872-5.

[Her+17] J. M. Hernández-Lobato et al. *Parallel and Distributed Thompson Sampling for Large-scale Accelerated Exploration of Chemical Space*. 2017. arXiv: 1706.01825.

[Het89] H. W. Hethcote. "Three Basic Epidemiological Models". In: *Applied Mathematical Ecology. Biomathematics, vol 18*. Springer, Berlin, Heidelberg, 1989. ISBN: 978-3-642-64789-5. DOI: https://doi.org/10.1007/978-3-642-61317-3_5.

[HG14] M. D. Hoffman and A. Gelman. "The No-U-Turn Sampler: Adaptively Setting Path Lengths in Hamiltonian Monte Carlo". In: *J. Mach. Learn. Res.* 15.1 (Jan. 2014), pp. 1593–1623. ISSN: 1532-4435. DOI: 10.5555/2627435.2638586.

[HGW19] M. Hughes, M. Goerigk, and M. Wright. "A largest empty hypersphere metaheuristic for robust optimisation with implementation uncertainty". In: *Computers and Operations Research* 103 (2019), pp. 64–80. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2018.10.013.

[Hor+15] D. Horn et al. "Model-Based Multi-objective Optimization: Taxonomy, Multi-Point Proposal, Toolbox and Benchmark". In: *Evolutionary Multi-Criterion Optimization*. Cham: Springer International Publishing, 2015, pp. 64–78. ISBN: 978-3-319-15934-8.

[HSR17] A. Habib, H. K. Singh, and T. Ray. "A batch infill strategy for computationally expensive optimization problems". In: *Australasian Conference on Artificial Life and Computational Intelligence*. Springer. 2017, pp. 74–85.

[HST98] H. Haario, E. Saksman, and J. Tamminen. "An Adaptive Metropolis algorithm". In: *Bernoulli* 7 (1998), pp. 223–242.

[Hub+06] S. Huband et al. "A review of multiobjective test problems and a scalable test problem toolkit". In: *IEEE Transactions on Evolutionary Computation* 10.5 (Oct. 2006). doi: https://doi.org/10.1109/TEVC.2005.861417, pp. 477–506. ISSN: 1089-778X.

[Hun07] J. D. Hunter. "Matplotlib: A 2D graphics environment". In: *Computing in Science & Engineering* 9.3 (2007), pp. 90–95. DOI: 10.1109/MCSE.2007.55.

[Inn+15] M. S. Innocente et al. "Particle swarm algorithm with adaptive constraint handling and integrated surrogate model for the management of petroleum fields". In: *Applied Soft Computing* 34 (2015), pp. 463–484. ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2015.05.032.

[Jin05] Y. Jin. "A comprehensive survey of fitness approximation in evolutionary computation". In: *Soft Computing* 9.1 (Jan. 2005), pp. 3–12. ISSN: 1433-7479. DOI: 10.1007/s00500-003-0328-5.

[Jin11] Y. Jin. "Surrogate-assisted evolutionary computation: Recent advances and future challenges". In: *Swarm and Evolutionary Computation* 1.2 (2011). doi: https://doi.org/10.1016/j.swevo.2011.05.001, pp. 61–70. ISSN: 2210-6502.

[Jon01] D. Jones. "A Taxonomy of Global Optimization Methods Based on Response Surfaces". In: *J. of Global Optimization* 21 (Dec. 2001), pp. 345–383. DOI: 10.1023/A:1012771025575.

[JOS00] Y. Jin, M. Olhofer, and B. Sendhoff. "On Evolutionary Optimization with Approximate Fitness Functions". In: *Proceedings of the 2Nd Annual Conference on Genetic and Evolutionary Computation*. GECCO'00. Las Vegas, Nevada: Morgan Kaufmann Publishers Inc., 2000, pp. 786–793. ISBN: 1-55860-708-0. URL: http://dl.acm.org/citation.cfm?id=2933718.2933864.

[JOS01] Y. Jin, M. Olhofer, and B. Sendhoff. "Managing approximate models in evolutionary aerodynamic design optimization". In: *Proceedings of the 2001 Congress on Evolutionary Computation (IEEE Cat. No.01TH8546)*. Vol. 1. 2001, 592–599 vol. 1. DOI: 10.1109/CEC.2001.934445.

[JS04] Y. Jin and B. Sendhoff. “Reducing Fitness Evaluations Using Clustering Techniques and Neural Network Ensembles”. In: June 2004, pp. 688–699. ISBN: 978-3-540-22344-3. DOI: 10.1007/978-3-540-24854-5_71.

[JSW98] D. R. Jones, M. Schonlau, and W. J. Welch. “Efficient Global Optimization of Expensive Black-Box Functions”. In: *Journal of Global Optimization* 13.4 (Dec. 1998), pp. 455–492. ISSN: 1573-2916. DOI: 10.1023/A:1008306431147.

[Ker+15] P. Kerschke et al. “Detecting Funnel Structures by Means of Exploratory Landscape Analysis”. In: *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*. GECCO ’15. Madrid, Spain: Association for Computing Machinery, 2015, pp. 265–272. ISBN: 9781450334723. DOI: 10.1145/2739480.2754642.

[KT19] P. Kerschke and H. Trautmann. “Comprehensive Feature-Based Landscape Analysis of Continuous and Constrained Optimization Problems Using the R-Package Flacco”. In: *Applications in Statistical Computing: From Music Data Analysis to Industrial Quality Improvement*. Cham: Springer International Publishing, 2019, pp. 93–123. ISBN: 978-3-030-25147-5. DOI: 10.1007/978-3-030-25147-5_7.

[Kus63] H. J. Kushner. “A new method of locating the maximum point of an arbitrary multipeak curve in the presence of noise”. Undetermined. In: *Joint Automatic Control Conference* 1 (1963), pp. 69–79. DOI: 10.1109/JACC.1963.4168566.

[LC14] Y. Liu and M. Collette. “Improving surrogate-assisted variable fidelity multi-objective optimization using a clustering algorithm”. In: *Applied Soft Computing* 24 (2014), pp. 482–493. ISSN: 1568-4946. DOI: https://doi.org/10.1016/j.asoc.2014.07.022.

[LHS12] J. Liu, Z. H. Han, and W. Song. “Comparison of infill sampling criteria in Kriging-based aerodynamic optimization”. In: *28th Congress of the International Council of the Aeronautical Sciences 2012, ICAS 2012* 2 (Jan. 2012), pp. 1625–1634.

[Li+10] C. Li et al. “A modified global optimization method based on surrogate model and its application in packing profile optimization of injection molding process”. In: *The International Journal of Advanced Manufacturing Technology* 48.5 (May 2010), pp. 505–511. ISSN: 1433-3015. DOI: 10.1007/s00170-009-2302-6.

[Lin+18] X. Lin et al. “A Batched Scalable Multi-Objective Bayesian Optimization Algorithm”. Nov. 2018.

[Liu+17] J. Liu et al. “Efficient aerodynamic shape optimization of transonic wings using a parallel infilling strategy and surrogate models”. In: *Structural and Multidisciplinary Optimization* 55 (Mar. 2017). DOI: 10.1007/s00158-016-1546-7.

[LL05] Y. Lian and M. S. Liou. “Multi-objective optimization of transonic compressor blade using evolutionary algorithm”. In: *Journal of Propulsion and Power, vol. 21, no. 6* (2005), pp. 979–987.

[LPS15] S. K. Lam, A. Pitrou, and S. Seibert. “Numba: A llvm-based python jit compiler”. In: *Proceedings of the Second Workshop on the LLVM Compiler Infrastructure in HPC*. 2015, pp. 1–6.

[LQS13] J. J. Liang, B. Y. Qu, and P. N. Suganthan. *Problem Definitions and Evaluation Criteria for the CEC 2014 Special Session and Competition on Single Objective Real-Parameter Numerical Optimization*. Tech. rep. Computational Intelligence Laboratory Zhengzhou University and Nanyang Technological University, 2013.

[LSS10] I. Loshchilov, M. Schoenauer, and M. Sebag. "A Mono Surrogate for Multiobjective Optimization". In: *Genetic and Evolutionary Computation Conference 2010 (GECCO-2010)* (July 2010). DOI: 10.1145/1830483.1830571.

[LW06] M. Lunacek and D. Whitley. "The Dispersion Metric and the CMA Evolution Strategy". In: *Proceedings of the 8th Annual Conference on Genetic and Evolutionary Computation*. GECCO '06. Seattle, Washington, USA: Association for Computing Machinery, 2006, pp. 477–484. ISBN: 1595931864. DOI: 10.1145/1143997.1144085.

[LYC21] Y. Lou, S. Y. Yuen, and G. Chen. "Non-revisiting stochastic search revisited: Results, perspectives, and future directions". In: *Swarm and Evolutionary Computation* 61 (2021), p. 100828. ISSN: 2210-6502. DOI: https://doi.org/10.1016/j.swevo.2020.100828.

[Lyu+18] W. Lyu et al. "Batch Bayesian Optimization via Multi-objective Acquisition Ensemble for Automated Analog Circuit Design". In: *Proceedings of the 35th International Conference on Machine Learning*. Vol. 80. Proceedings of Machine Learning Research. PMLR, July 2018, pp. 3306–3314. URL: http://proceedings.mlr.press/v80/lyu18a.html.

[Mal21] K. M. Malan. "A Survey of Advances in Landscape Analysis for Optimisation". In: *Algorithms* 14.2 (Jan. 2021), p. 40. ISSN: 1999-4893. DOI: 10.3390/a14020040.

[Mat+21a] L. Matrajt et al. "Optimizing vaccine allocation for COVID-19 vaccines: potential role of single-dose vaccination". In: *Nature Communications* 12.3449 (2021). DOI: https://doi.org/10.1038/s41467-021-23761-1.

[Mat+21b] L. Matrajt et al. "Vaccine optimization for COVID-19: Who to vaccinate first?" In: *Science Advances* 7.6 (2021), eabf1374. DOI: 10.1126/sciadv.abf1374.

[MC10] S. Zapotecas Martínez and C. A. Coello Coello. "[ACM Press the 12th annual conference - Portland, Oregon, USA (2010.07.07-2010.07.11)] Proceedings of the 12th annual conference on Genetic and evolutionary computation - GECCO '10 - A multi-objective meta-model assisted memetic algorithm with non gradient-based local search". In: 2010. ISBN: 9781450300728. DOI: 10.1145/1830483.1830581.

[McB+21] E. S. McBryde et al. "Modelling direct and herd protection effects of vaccination against the SARS-CoV-2 Delta variant in Australia". In: *Medical Journal of Australia* 215.9 (2021), pp. 427–432. DOI: https://doi.org/10.5694/mja2.51263.

[Med22] Medicalxpress. *Weaker virus? Herd immunity? Omicron sparks cautious hopes*. https://medicalxpress.com/news/2022-01-weaker-virus-herd-immunity-omicron.html. 2022.

[Mel05] N. Melab. "Contributions à la résolution de problèmes d'optimisation combinatoire sur grilles de calcul". https://pepite.univ-lille.fr/ori-oai-search/notice/view/univ-lille-16397?resultBackUrl=. Habilitation à diriger des recherches. Université de Lille, Nov. 2005.

[Mer+11] O. Mersmann et al. “Exploratory Landscape Analysis”. In: *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*. GECCO ’11. Dublin, Ireland: Association for Computing Machinery, 2011, pp. 829–836. ISBN: 9781450305570. DOI: 10.1145/2001576.2001690.

[Met+53] N. Metropolis et al. “Equation of State Calculations by Fast Computing Machines”. In: *The Journal of Chemical Physics* 21.6 (1953), pp. 1087–1092. DOI: 10.1063/1.1699114.

[MG09] J. Medlock and A. P. Galvani. “Optimizing Influenza Vaccine Distribution”. In: *Science* 325.5948 (2009), pp. 1705–1708. DOI: 10.1126/science.1175570.

[Mic+96] Z. Michalewicz et al. “Evolutionary algorithms for constrained engineering problems”. In: *Computers and Industrial Engineering* 30.4 (1996), pp. 851–870. ISSN: 0360-8352. DOI: https://doi.org/10.1016/0360-8352(96)00037-X.

[Mii+21] R. Miikkulainen et al. “From Prediction to Prescription: Evolutionary Optimization of Nonpharmaceutical Interventions in the COVID-19 Pandemic”. In: *IEEE Transactions on Evolutionary Computation* 25.2 (2021), pp. 386–401. DOI: 10.1109/TEVC.2021.3063217.

[ML10] L. Matrajt and I. Longini. “Optimizing Vaccine Allocation at Different Points in Time during an Epidemic”. In: *PloS one* 5 (Nov. 2010), e13767. DOI: 10.1371/journal.pone.0013767.

[Mla+15] M. Mlakar et al. “GP-DEMO: Differential Evolution for Multiobjective Optimization based on Gaussian Process models”. In: *European Journal of Operational Research* 243.2 (2015). doi: https://doi.org/10.1016/j.ejor.2014.04.011, pp. 347–361. ISSN: 0377-2217.

[MM11] K. Mitra and S. Majumder. “Successive approximate model based multiobjective optimization for an industrial straight grate iron ore induration process using evolutionary algorithm”. In: *Chemical Engineering Science* 66 (Aug. 2011), pp. 3471–3481. DOI: 10.1016/j.ces.2011.03.041.

[MP11] J. Muller and R. Piché. “Mixture surrogate models based on Dempster-Shafer theory for global optimization problems”. In: *J. Global Optimization* 51 (Sept. 2011), pp. 79–104. DOI: 10.1007/s10898-010-9620-y.

[MS14] J. Müller and C.A. Shoemaker. “Influence of ensemble surrogate models and sampling strategy on the solution quality of algorithms for computationally expensive black-box global optimization problems”. In: *Journal of Global Optimization* 60.2 (Oct. 2014). doi: https://doi.org/10.1007/s10898-014-0184-0, pp. 123–144. ISSN: 1573-2916.

[MT22] V. R. Montplaisir and C. Tribes. *NOMAD = Nonlinear Optimization by Mesh Adaptive Direct Search*. https://github.com/bbopt/nomad. 2022.

[NDS97] C. Norton, V. Decyk, and B. Szymanski. “High Performance Object-Oriented Scientific Programming in Fortran 90”. In: Jan. 1997.

[Nea96] R. M. Neal. *Bayesian Learning for Neural Networks*. Berlin, Heidelberg: Springer-Verlag, 1996. ISBN: 0387947248. DOI: 10.5555/525544.

[NKS98] P. B. Nair, A. J. Keaney, and R. P. Shimpiz. “Combining Approximation Concepts with Genetic Algorithm-based Structural Optimization Procedures”. In: 1998.

[ODM17] J. O'Neill, S. Delany, and B. MacNamee. "Model-Free and Model-Based Active Learning for Regression". In: vol. 513. Jan. 2017, pp. 375–386. ISBN: 978-3-319-46561-6. DOI: 10.1007/978-3-319-46562-3_24.

[OJS03] T. Okabe, Y. Jin, and B. Sendhoff. "A critical survey of performance indices for multi-objective optimisation". In: *Evolutionary Computation, 2003. CEC '03. The 2003 Congress on*. Vol. 2. doi: https://doi.org/10.1109/CEC.2003.1299759. Dec. 2003, 878–885 Vol.2.

[ONK03] Y. Ong, P. Nair, and A. Keane. "Evolutionary Optimization of Computationally Expensive Problems via Surrogate Modeling". In: *AIAA Journal* (Apr. 2003). DOI: 10.2514/2.1999.

[Pal+22] N. Kugalur Palanisamy et al. "Identification of the Parameter Values of the Constitutive and Friction Models in Machining Using EGO Algorithm: Application to Ti6Al4V". In: *Metals* 12.6 (2022). ISSN: 2075-4701. DOI: 10.3390/met12060976.

[Pan21] A. Pannier. *Strategic Calculation: High-Performance Computing and Quantum Computing in Europe's Quest for Technological Power*. https://www.ifri.org/sites/default/files/atoms/files/pannier_strategic_calculation_2021.pdf. Études de l'Ifri. Oct. 2021.

[PCJ17] K. Prem, A. R. Cook, and M. Jit. "Projecting social contact matrices in 152 countries using contact surveys and demographic data". In: *PLOS Computational Biology* 13.9 (Sept. 2017), pp. 1–21. DOI: 10.1371/journal.pcbi.1005697.

[Ped+11] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[Pol+00] C. Poloni et al. "Hybridization of a multi-objective genetic algorithm, a neural network and a classical optimizer for a complex design problem in fluid dynamics". In: *Computer Methods in Applied Mechanics and Engineering* 186.2 (2000). doi: https://doi.org/10.1016/S0045-7825(99)00394-1, pp. 403–420. ISSN: 0045-7825.

[PPJ19] D. Phan, N. Pradhan, and M. Jankowiak. "Composable Effects for Flexible and Accelerated Probabilistic Programming in NumPyro". In: *arXiv preprint arXiv:1912.11554* (2019).

[PR15] C. Paulson and G. Ragkousis. *pyKriging: A Python Kriging Toolkit*. Version v0.1.0-alpha. July 2015. DOI: 10.5281/zenodo.21389.

[Ras06] C. E. Rasmussen. *Gaussian processes for machine learning*. MIT Press, 2006.

[Reh+18] F. Rehback et al. "Comparison of Parallel Surrogate-Assisted Optimization Approaches". In: *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO '18. Kyoto, Japan: Association for Computing Machinery, 2018, pp. 1348–1355. ISBN: 9781450356183. DOI: 10.1145/3205455.3205587.

[Reh+20] F. Rehbach et al. "Expected Improvement versus Predicted Value in Surrogate-Based Optimization". In: *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*. GECCO '20. Cancún, Mexico: Association for Computing Machinery, 2020, pp. 868–876. ISBN: 9781450371285. DOI: 10.1145/3377930.3389816.

[RH00] K. Rasheed and H. Hirsh. "Informed operators: Speeding up genetic-algorithm-based design optimization using reduced models". In: (May 2000).

[Ric+16] J. Richter et al. “Faster Model-Based Optimization Through Resource-Aware Scheduling Strategies”. In: vol. 10079. May 2016, pp. 267–273. ISBN: 978-3-319-50348-6. DOI: 10.1007/978-3-319-50349-3_22.

[RK17] S. Ratnoo and D. Kamboj. “A non-revisiting Genetic Algorithm with adaptive mutation for Function Optimization”. In: *International Journal of Advanced Research in Computer Science* 2.6 (2017), pp. 504–507. ISSN: 0976-5697. DOI: 10.26483/ijarcs.v2i6.933. URL: http://ijarcs.info/index.php/Ijarcs/article/view/933.

[Ron98] S. Ronald. “Duplicate genotypes in a genetic algorithm”. In: *1998 IEEE International Conference on Evolutionary Computation Proceedings. IEEE World Congress on Computational Intelligence (Cat. No.98TH8360)*. 1998, pp. 793–798. DOI: 10.1109/ICEC.1998.700153.

[Ros+13] A. Rosales-Pérez et al. “A hybrid surrogate-based approach for evolutionary multi-objective optimization”. In: *2013 IEEE Congress on Evolutionary Computation*. June 2013, pp. 2548–2555. DOI: 10.1109/CEC.2013.6557876.

[RS05] R. G. Regis and C. A. Shoemaker. “Constrained Global Optimization of Expensive Black Box Functions Using Radial Basis Functions”. In: *Journal of Global Optimization* 31.1 (Jan. 2005), pp. 153–171. ISSN: 1573-2916. DOI: 10.1007/s10898-004-0570-0.

[RS07] R. Regis and C. Shoemaker. “A Stochastic Radial Basis Function Method for the Global Optimization of Expensive Functions”. In: *INFORMS Journal on Computing* 19 (Nov. 2007), pp. 497–509. DOI: 10.1287/ijoc.1060.0182.

[Rua+20] X. Ruan et al. “Surrogate Assisted Evolutionary Algorithm for Medium Scale Multi-Objective Optimisation Problems”. In: *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*. GECCO ’20. Cancún, Mexico: Association for Computing Machinery, 2020, pp. 560–568. ISBN: 9781450371285. DOI: 10.1145/3377930.3390191.

[Rud16] S. Ruder. “An overview of gradient descent optimization algorithms”. In: *CoRR* abs/1609.04747 (2016). arXiv: 1609.04747. URL: http://arxiv.org/abs/1609.04747.

[RV20] S. Rojas-Gonzalez and I. Van Nieuwenhuyse. “A survey on kriging-based infill algorithms for multiobjective simulation optimization”. In: *Computers and Operations Research* 116 (2020), p. 104869. ISSN: 0305-0548. DOI: https://doi.org/10.1016/j.cor.2019.104869.

[RXX11] D. Rouson, J. Xia, and X. Xu. *Scientific Software Design: The Object-Oriented Way*. Cambridge University Press, 2011. ISBN: 9781139498784. URL: https://books.google.fr/books?id=xedE5KkHVn4C.

[SC04] J. J. Alonso S. Choi and H. S. Chung. “Design of a low-boom supersonic business jet using evolutionary algorithms and an adaptive unstructured mesh method”. In: *Proceedings of the 45th AIAA/ASME/ASCE/AHS/ASC Structures, Structural Dynamics, and Materials Conference* (Apr. 2004), pp. 2692–2706.

[Scu10] D. Sculley. “Web-Scale k-Means Clustering”. In: *Proceedings of the 19th International Conference on World Wide Web*. WWW ’10. Raleigh, North Carolina, USA: Association for Computing Machinery, 2010, pp. 1177–1178. ISBN: 9781605587998. DOI: 10.1145/1772690.1772862.

[SLA12] J. Snoek, H. Larochelle, and R. P. Adams. "Practical Bayesian Optimization of Machine Learning Algorithms". In: *Proceedings of the 25th International Conference on Neural Information Processing Systems - Volume 2*. NIPS'12. Lake Tahoe, Nevada: Curran Associates Inc., 2012, pp. 2951–2959.

[SLK04] A. Sobester, S. J. Leary, and A. Keane. "A parallel updating scheme for approximating and optimizing high fidelity computer simulations". In: *Structural and Multidisciplinary Optimization* 27 (Jan. 2004), pp. 371–383. DOI: 10.1007/s00158-004-0397-9.

[SMD22] M. Sheel, S. McEwen, and S. E. Davies. "Brand inequity in access to COVID-19 vaccines". In: *The Lancet Regional Health - Western Pacific* 18 (2022), p. 100366. ISSN: 2666-6065. DOI: https://doi.org/10.1016/j.lanwpc.2021.100366.

[SMS19] R. El Shawi, M. Maher, and S. Sakr. "Automated Machine Learning: State-of-The-Art and Open Challenges". In: *CoRR* abs/1906.02287 (2019). arXiv: 1906.02287. URL: http://arxiv.org/abs/1906.02287.

[Sno+15] J. Snoek et al. "Scalable Bayesian Optimization Using Deep Neural Networks". In: *Statistics* (Feb. 2015).

[SR10] L. Shi and K. Rasheed. "A Survey of Fitness Approximation Methods Applied in Evolutionary Algorithms". In: *Computational Intelligence in Expensive Optimization Problems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 3–28. ISBN: 978-3-642-10701-6. DOI: 10.1007/978-3-642-10701-6_1.

[Sri+14] N. Srivastava et al. "Dropout: A Simple Way to Prevent Neural Networks from Overfitting". In: *Journal of Machine Learning Research* 15 (2014), pp. 1929–1958. URL: http://jmlr.org/papers/v15/srivastava14a.html.

[Ste99] M.L. Stein. *Interpolation of Spatial Data: Some Theory for Kriging*. Springer Series in Statistics. Springer New York, 1999. ISBN: 9780387986296. URL: https://books.google.be/books?id=5n%5C_XuL2Wx1EC.

[Syb+08] A. Syberfeldt et al. "A parallel surrogate-assisted multi-objective evolutionary algorithm for computationally expensive optimization problems". In: *2008 IEEE Congress on Evolutionary Computation (IEEE World Congress on Computational Intelligence)*. doi: https://doi.org/10.1109/CEC.2008.4631228. June 2008, pp. 3177–3184.

[Tal09] E. G. Talbi. *Metaheuristics: From Design to Implementation*. Wiley Series on Parallel and Distributed Computing. Wiley, 2009. ISBN: 9780470496909. URL: https://books.google.fr/books?id=SIsa6zi5XV8C.

[Tia+19] J. Tian et al. "Multiobjective Infill Criterion Driven Gaussian Process-Assisted Particle Swarm Optimization of High-Dimensional Expensive Problems". In: *IEEE Transactions on Evolutionary Computation* 23.3 (June 2019), pp. 459–472. ISSN: 1941-0026. DOI: 10.1109/TEVC.2018.2869247.

[Ton+21] Hao Tong et al. "Surrogate models in evolutionary single-objective optimization: A new taxonomy and experimental study". In: *Information Sciences* 562 (2021), pp. 414–437. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2021.03.002.

[TOP] TOP500. *TOP500 The List*. URL: https://www.top500.org/.

[Tra+21] J. M. Trauer et al. "Understanding how Victoria, Australia gained control of its second COVID-19 wave". In: *Nature Communications* 12.6266 (2021). DOI: https://doi.org/10.1038/s41467-021-26558-4.

[Va20] P. Virtanen and al. “SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python”. In: *Nature Methods* 17 (2020), pp. 261–272. DOI: 10.1038/s41592-019-0686-2.

[VHW13] F. A. C. Viana, R. T. Haftka, and L. T. Watson. “Efficient global optimization algorithm assisted by multiple surrogate techniques”. In: *Journal of Global Optimization* 56.2 (2013), pp. 669–689.

[Vil+13] D. Villanueva et al. “Locating Multiple Candidate Designs with Dynamic Local Surrogates”. In: (May 2013).

[Vil+22] T. N. Vilches et al. “COVID-19 hospitalizations and deaths averted under an accelerated vaccination program in northeastern and southern regions of the USA”. In: *The Lancet Regional Health - Americas* 6 (2022), p. 100147. ISSN: 2667-193X. DOI: https://doi.org/10.1016/j.lana.2021.100147.

[WA13] Andrew Gordon Wilson and Ryan Prescott Adams. *Gaussian Process Kernels for Pattern Discovery and Extrapolation*. 2013. arXiv: 1302.4245.

[Wan+16] H. Wang et al. “A comparative study of expected improvement- assisted global optimization with different surrogates”. In: *Engineering Optimization* (Jan. 2016). DOI: 10.1080/0305215X.2015.1115645.

[Wan+20a] J. Wang et al. “Parallel Bayesian Global Optimization of Expensive Functions”. In: *Operations Research* 68.6 (2020), pp. 1850–1865. DOI: 10.1287/opre.2019.1966.

[Wan+20b] X. Wang et al. “An adaptive Bayesian approach to surrogate-assisted evolutionary multi-objective optimization”. In: *Information Sciences* 519 (2020), pp. 317–331. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2020.01.048.

[Wey+05] D. Weycker et al. “Population-wide benefits of routine vaccination of children against influenza”. In: *Vaccine* 23.10 (2005), pp. 1284–1293. ISSN: 0264-410X. DOI: https://doi.org/10.1016/j.vaccine.2004.08.044.

[WJD17] H. Wang, Y. Jin, and J. Doherty. “Committee-Based Active Learning for Surrogate-Assisted Particle Swarm Optimization of Expensive Problems”. In: *IEEE Transactions on Cybernetics* 47.9 (Sept. 2017), pp. 2664–2677. ISSN: 2168-2267. DOI: 10.1109/TCYB.2017.2710978.

[Xia+14] W. Xia et al. “A multi-objective optimization method based on Gaussian process simultaneous modeling for quality control in sheet metal forming”. In: *The International Journal of Advanced Manufacturing Technology* 72 (2014), pp. 1333–1346.

[YC09] S. Y. Yuen and C. K. Chow. “A Genetic Algorithm That Adaptively Mutates and Never Revisits”. In: *Trans. Evol. Comp* 13.2 (Apr. 2009), pp. 454–472. ISSN: 1089-778X. DOI: 10.1109/TEVC.2008.2003008.

[Zho+07] Z. Zhou et al. “Memetic algorithm using multi-surrogates for computationally expensive optimization problems”. In: *Soft Comput.* 11 (Aug. 2007), pp. 957–971. DOI: 10.1007/s00500-006-0145-8.

[ZWC14] J. Zhu, Y. Wang, and M. Collette. “A multi-objective variable-fidelity optimization method for genetic algorithms”. In: *Engineering Optimization* 46 (Apr. 2014). DOI: 10.1080/0305215X.2013.786063.

List of Figures

List of Tables

Appendix A

Parallel Surrogate-based optimization

Table A.1: Terminology adopted in Machine Learning and Optimization

Symbol	Machine Learning	Optimization
$f()$	black-box mapping	objective function
$\mathcal{D}$	input space	search space
d	input dimension	search space dimension, number of decision variables
x $(d = 1)$	input	decision variable, candidate solution
$\boldsymbol{x}$ $(d > 1)$	input vector	decision vector, candidate solution
$\mathcal{M}$	output space	objective space
m	output dimension	objective space dimension, number of objectives
y $(m = 1)$	output, target	objective value
$\boldsymbol{y}$ $(m > 1)$	output vector, target vector	objective vector
$\hat{f}()$	approximation	surrogate
n	training set size	surrogate training set size

Appendix B

Parallel Surrogate-assisted Evolutionary computations

Algorithm 13 MCDropout prediction with one-hidden-layer ANN

Input

$\boldsymbol{x}$: candidate to predict
n_{sub}: number of sub-networks
p_{drop}: probability of dropping out neurons
$(W^{(1)}, \boldsymbol{w^{(2)}})$: weights trained with Dropout
$\boldsymbol{h}()$: activation function

1: **for** $1 \leqslant i \leqslant n_{sub}$ **do**
2: $\quad \boldsymbol{\epsilon^{(1)}} \leftarrow$ Bernouilli_sampling(p_{drop})
3: $\quad \hat{f}_i \leftarrow \boldsymbol{\epsilon^{(1)}} \odot \boldsymbol{w^{(2)}}.\boldsymbol{h}(diag(\boldsymbol{\epsilon^{(1)}}).W^{(1)}.\boldsymbol{x})$
4: **end for**
5: $\hat{f} \leftarrow \frac{1}{n_{sub}} \sum_{i=1}^{n_{sub}} \hat{f}_i$
6: $\hat{s} \leftarrow \sqrt{\frac{1}{n_{sub}} \sum_{i=1}^{n_{sub}} (\hat{f}_i - \hat{f})^2}$
7: **return** $(\hat{f}, \hat{s})$

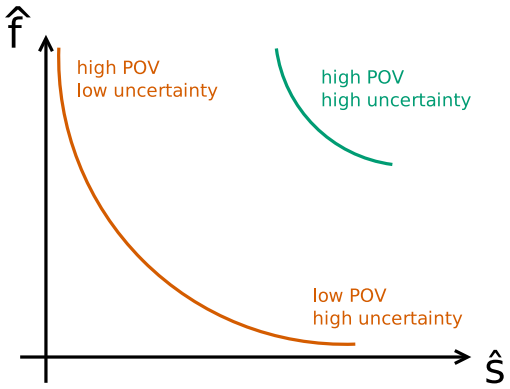

Figure B.1: Illustration of *par-tian-fs*. The orange line represents the first NDF and the green line represents the last NDF.

Algorithm 14 SaaEF

Input

simulator: real objective function
surrogate: surrogate model
budget: computational budget for the search
n_{pop}: population size
n_{chld}: number of new candidates issued per cycle
q: number of simulations per batch
n_{pred}: number of predictions per batch
n_{disc}: number of discarding per batch

1: $database \leftarrow$ LHS($simulator$, n_{pop})
2: $surrogate \leftarrow$ training($database$)
3: $\mathcal{P} \leftarrow database$ ▷ initial population
4: $(\boldsymbol{x}_{min}, y_{min}) \leftarrow$ get_best_cost($database$)
5: **while** $budget \neq 0$ **do**
6: $\mathcal{P}_p \leftarrow$ selection($\mathcal{P}$, n_{chld}) ▷ population of parents
7: $\mathcal{P}_c \leftarrow$ reproduction($\mathcal{P}_p$, n_{chld}) ▷ population of children
8: $(\mathcal{B}_{sim}, \mathcal{B}_{pred}) \leftarrow$ evolution_control($\mathcal{P}_c$, $surrogate$, q, n_{pred}, n_{disc})
9: parallel_simulation($simulator$, $\mathcal{B}_{sim}$)
10: $database \leftarrow database \cup \mathcal{B}_{sim}$
11: $surrogate \leftarrow$ training($database$)
12: prediction($surrogate$, $\mathcal{B}_{pred}$)
13: $\mathcal{P} \leftarrow$ replacement($\mathcal{P}$, $\mathcal{B}_{sim}$, $\mathcal{B}_{pred}$, n_{pop})
14: $(\boldsymbol{x}_{min}, y_{min}) \leftarrow$ get_best_cost($database$)
15: $budget \leftarrow$ get_remaining_budget($budget$, elapsed_time)
16: **end while**
17: **return** $\boldsymbol{x}_{min}$, y_{min}

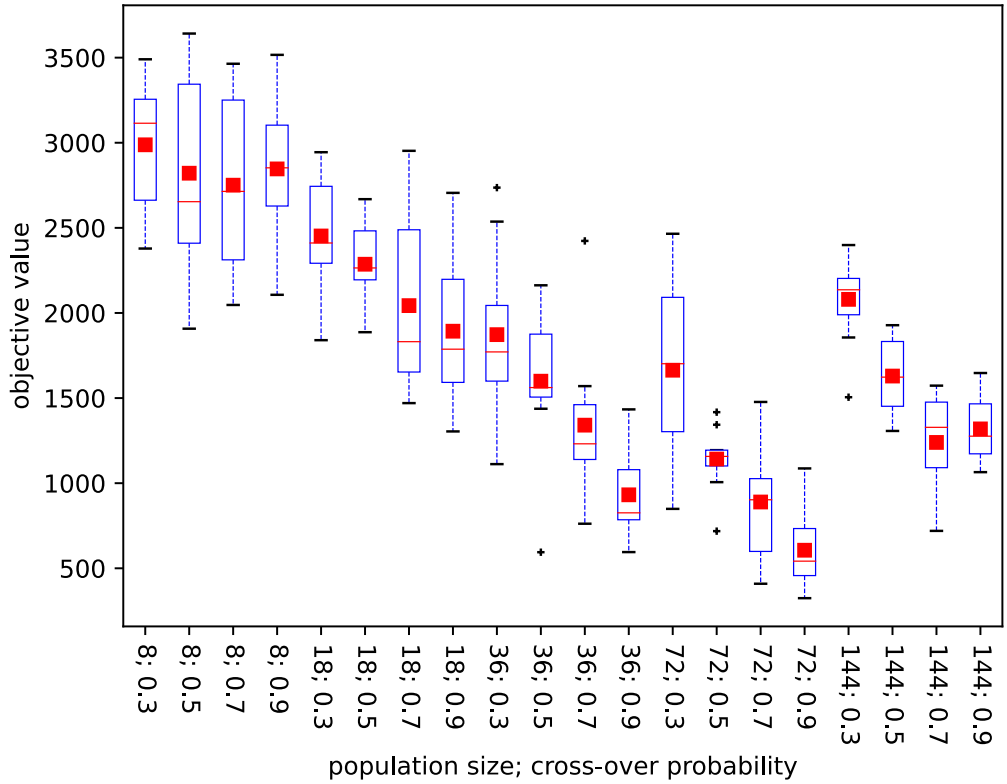

Figure B.2: **Calibration of parallel EA**. Distribution of the best objective values from the 10 repetitions of the experiments on the **Schwefel** problem. Average values are depicted by red squares, median values by red dashes and variance information is given by the length of the boxes.

Table B.1: **Calibration of BNN_MCD**. Best NDF according to minimization of the validation mean squared error (VMSE) and the negative validation average log-likelihood (NVALL) on the **Schwefel** problem. Ordering according to ascending VMSE from top to bottom. The retained configuration appears in bold.

n_{hl}	m_u	λ_{decay}	l	p_{drop}	VMSE	NAVLL
1	512	1e-3	1e-2	0.05	0.0281	6.0782
1	1024	1e-3	1e-2	0.1	0.0282	5.8554
10	2048	1e-3	1e-2	0.3	0.0284	5.2803
10	1024	1e-3	1e-2	0.3	0.0285	5.273
10	2048	1e-3	1e-2	0.5	0.0287	4.5629
1	1024	1e-3	1e-2	0.5	0.0289	4.5159
8	4096	1e-2	1e-2	0.005	0.0289	2.6749
8	4096	1e-2	1e-2	0.1	0.0289	2.5988
2	4096	1e-2	1e-2	0.1	0.0291	2.5987
1	1024	1e-2	1e-2	0.1	0.0291	2.591
10	4096	1e-2	1e-2	0.3	0.0292	2.4188
8	2048	1e-2	1e-2	0.3	0.0293	2.4183
1	2048	1e-2	1e-2	0.3	0.0293	2.4124
2	2048	1e-2	1e-2	0.5	0.0295	2.196
10	2048	1e-2	1e-2	0.5	0.0297	2.1956
1	2048	1e-1	1e-2	0.005	0.0299	1.2783
1	4096	1e-1	1e-2	0.005	0.0299	1.2782
1	4096	1e-1	1e-2	0.05	0.0301	1.2542
1	**1024**	**1e-1**	**1e-2**	**0.1**	0.0304	1.2257
2	2048	1e-1	1e-2	0.1	0.0305	1.2257
1	512	1e-1	1e-2	0.1	0.0308	1.2256
1	2048	1e-1	1e-2	0.3	0.0309	1.0946
1	512	1e-1	1e-2	0.3	0.0311	1.0946
1	2048	1e-1	1e-2	0.5	0.0312	0.9208
1	4096	1e-1	1e-2	0.5	0.0317	0.9208
2	4096	1e-1	1e-2	0.5	0.0321	0.9207
8	4096	1	1e-2	0.005	0.033	0.1026
10	4096	1	1e-2	0.05	0.0332	0.0793
5	4096	1	1e-2	0.1	0.0335	0.0521
10	4096	1	1e-2	0.3	0.034	-0.074
2	2048	1	1e-2	0.3	0.0348	-0.074
2	2048	1	1e-2	0.5	0.035	-0.2428
10	4096	1	1e-2	0.5	0.0351	-0.2428
1	256	1	1e-1	0.3	0.0925	-1.227
1	256	1	1e-1	0.5	0.1254	-1.3953
1	256	1	1	0.05	2667.34	-2.2039
1	256	1	1	0.3	3001.3754	-2.3104
1	256	1	1	0.5	3230.1306	-2.4348

Table B.2: **Calibration of BNN_MCD**. Best NDF according to minimization of the validation mean squared error (VMSE) and the negative validation average log-likelihood (NVALL) on the **Rastrigin** problem. Ordering according to ascending VMSE from top to bottom.

n_{hl}	m_u	λ_{decay}	l	p_{drop}	VMSE	NAVLL
8	2048	1e-3	1e-2	0.1	0.0323	6.3019
10	4096	1e-3	1e-2	0.3	0.0324	5.5486
10	2048	1e-3	1e-2	0.3	0.0328	5.546
10	4096	1e-3	1e-2	0.5	0.033	4.7496
10	4096	1e-2	1e-2	0.05	0.0332	2.6757
8	4096	1e-2	1e-2	0.05	0.0332	2.6757
10	4096	1e-2	1e-2	0.1	0.0334	2.6337
5	1024	1e-2	1e-2	0.1	0.0336	2.6328
2	4096	1e-2	1e-2	0.3	0.0337	2.4471
8	2048	1e-2	1e-2	0.3	0.0338	2.4454
10	2048	1e-2	1e-2	0.3	0.0339	2.4452
2	4096	1e-2	1e-2	0.5	0.0339	2.2152
1	4096	1e-1	1e-2	0.005	0.0345	1.2823
2	2048	1e-1	1e-2	0.05	0.0347	1.2577
2	4096	1e-1	1e-2	0.1	0.0348	1.229
1	4096	1e-1	1e-2	0.3	0.0356	1.0974
1	2048	1e-1	1e-2	0.3	0.0356	1.0972
5	4096	1e-1	1e-2	0.3	0.0364	1.0972
1	1024	1e-1	1e-2	0.5	0.0365	0.9229
1	2048	1e-1	1e-2	0.5	0.0372	0.9229
8	4096	1e-1	1e-2	0.5	0.0375	0.9228
10	4096	1e-1	1e-2	0.5	0.0378	0.9228
1	4096	1	1e-2	0.005	0.0384	0.1029
1	4096	1	1e-2	0.1	0.0385	0.0524
2	4096	1	1e-2	0.1	0.0399	0.0524
1	4096	1	1e-2	0.3	0.0403	-0.0736
5	4096	1	1e-2	0.3	0.0406	-0.0737
1	4096	1	1e-2	0.5	0.0411	-0.2426
2	4096	1	1e-2	0.5	0.0416	-0.2426
1	2048	1	1e-2	0.5	0.042	-0.2426
8	1024	1	1e-2	0.5	0.0461	-0.2426
1	256	1	1e-1	0.3	0.1234	-1.227
1	256	1	1e-1	0.5	0.129	-1.3953
1	512	1	1e-1	0.5	0.2099	-1.3953
1	1024	1	1e-1	0.5	0.4058	-1.3953
1	256	1	1	0.1	2634.2219	-2.2082
1	256	1	1	0.3	2967.1152	-2.2736
1	256	1	1	0.5	3345.4199	-2.3886

Table B.3: **Calibration of BNN_MCD**. Best NDF according to minimization of the validation mean squared error (VMSE) and the negative validation average log-likelihood (NVALL) on the **Rosenbrock** problem. Ordering according to ascending VMSE from top to bottom. The retained configuration appears in bold.

n_{hl}	m_u	λ_{decay}	l	p_{drop}	VMSE	NAVLL
1	2048	1e-3	1e-2	0.005	0.0235	5.6153
1	2048	1e-3	1e-2	0.05	0.0236	5.5204
1	4096	1e-3	1e-2	0.1	0.0243	5.2677
1	4096	1e-3	1e-2	0.3	0.0245	4.7687
1	2048	1e-3	1e-2	0.5	0.0252	4.2843
1	4096	1e-3	1e-2	0.5	0.0256	4.2509
2	4096	1e-3	1e-2	0.5	0.0309	4.2273
1	512	1e-2	1e-2	0.005	0.0311	2.6421
1	2048	1e-2	1e-2	0.1	0.0312	2.569
1	4096	1e-2	1e-2	0.1	0.0316	2.569
1	2048	1e-2	1e-2	0.3	0.0319	2.3986
1	4096	1e-2	1e-2	0.3	0.0319	2.3972
1	2048	1e-2	1e-2	0.5	0.0323	2.1841
1	4096	1e-2	1e-2	0.5	0.0325	2.1835
1	256	1e-2	1e-1	0.05	0.0385	1.2509
1	**1024**	**1e-1**	**1e-2**	**0.1**	0.0431	1.2362
1	1024	1e-1	1e-2	0.3	0.0436	1.1028
1	4096	1e-1	1e-2	0.3	0.0437	1.1026
1	512	1e-1	1e-2	0.3	0.0441	1.1025
1	1024	1e-1	1e-2	0.5	0.0443	0.9264
1	4096	1	1e-2	0.05	0.045	0.0805
2	4096	1	1e-2	0.05	0.0457	0.0805
2	4096	1	1e-2	0.1	0.0459	0.0532
1	4096	1	1e-2	0.3	0.0461	-0.0731
1	2048	1	1e-2	0.3	0.0461	-0.0731
1	1024	1	1e-2	0.3	0.0467	-0.0731
1	4096	1	1e-2	0.5	0.0468	-0.2421
1	512	1	1e-2	0.5	0.0477	-0.2421
8	2048	1	1e-2	0.5	0.0486	-0.2422
1	256	1e-1	1e-1	0.5	0.0659	-0.2423
1	512	1e-1	1e-1	0.5	0.0756	-0.2423
1	256	1	1e-1	0.1	0.1148	-1.1012
1	256	1	1e-1	0.5	0.132	-1.3952
1	512	1	1e-1	0.5	0.2207	-1.3953
1	2048	1	1e-1	0.5	0.7305	-1.3953
1	4096	1	1e-1	0.5	1.4349	-1.3953
2	4096	1	1e-1	0.5	424.7706	-1.3953
1	256	1	1	0.05	2658.1091	-2.1998
1	256	1	1	0.3	2677.07	-2.3144
1	256	1	1	0.5	3048.7062	-2.4446

Table B.4: **Calibration of BNN_MCD**. BNN_MCD configurations that appear in the NDF of 2 or 3 **benchmark problems**. The retained configuration appears in bold.

n_{hl}	m_u	λ_{decay}	l	p_{drop}	Occurrences
1	256	1	1	0.5	3
1	256	1	1	0.3	3
1	256	1	1e-1	0.5	3
1	512	1e-1	1e-2	0.3	2
1	2048	1e-1	1e-2	0.3	2
2	4096	1	1e-2	0.1	2
1	1024	1e-1	1e-2	0.5	2
1	4096	1	1e-2	0.3	2
1	4096	1e-1	1e-2	0.005	2
10	2048	1e-3	1e-2	0.3	2
1	2048	1e-2	1e-2	0.3	2
1	2048	1e-1	1e-2	0.5	2
1	**1024**	**1e-1**	**1e-2**	**0.1**	2
1	256	1	1	0.05	2
1	256	1	1e-1	0.3	2
1	512	1	1e-1	0.5	2
8	2048	1e-2	1e-2	0.3	2
1	4096	1	1e-2	0.5	2
1	4096	1e-1	1e-2	0.3	2

Table B.5: **Calibration of parallel EA**. Statistics of the distribution of the best objective values from the 10 repetitions of the experiments on the **Schwefel** problem. Ordering according to ascending average best objective value from top to bottom. The best value for each column appears in bold.

n_{pop}	p_c	Average	Median	Minimum	Variance
72	0.9	**606.16**	**542.0**	**324.5**	47011.53
72	0.7	889.47	902.62	409.36	118702.65
36	0.9	931.69	825.79	595.2	67856.63
72	0.5	1141.93	1157.35	718.28	**32334.32**
144	0.7	1239.44	1328.12	719.76	76650.15
144	0.9	1318.88	1276.17	1064.83	35878.64
36	0.7	1340.92	1231.3	761.9	175525.18
36	0.5	1599.24	1561.78	594.54	160909.86
144	0.5	1629.41	1623.04	1306.89	43485.15
72	0.3	1663.67	1701.82	849.11	259982.01
36	0.3	1872.68	1771.19	1111.97	208935.32
18	0.9	1892.87	1787.23	1304.08	161087.18
18	0.7	2043.18	1831.31	1470.45	273082.05
144	0.3	2079.94	2136.23	1505.05	62635.81
18	0.5	2286.32	2264.47	1887.0	55069.82
18	0.3	2452.32	2411.11	1840.13	109126.93
8	0.7	2750.68	2714.23	2047.0	278623.58
8	0.5	2821.03	2653.74	1907.67	312750.77
8	0.9	2846.3	2852.68	2106.74	173903.74
8	0.3	2987.9	3114.64	2378.45	144095.13

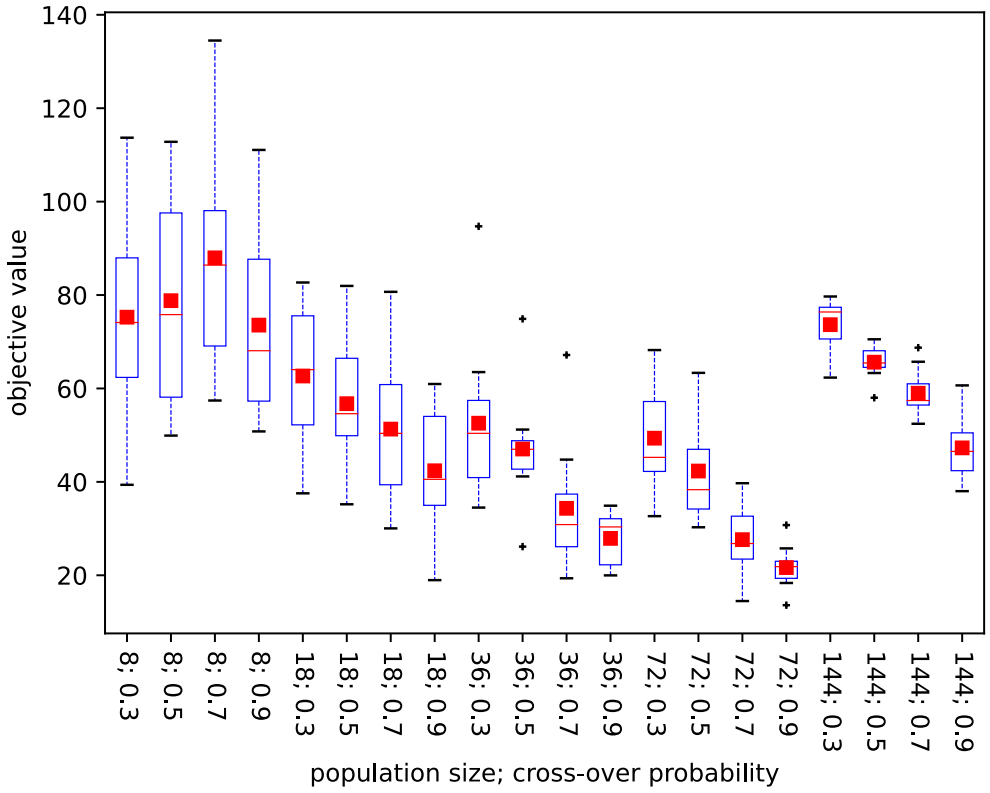

Figure B.3: **Calibration of parallel EA**. Distribution of the best objective values from the 10 repetitions of the experiments on the **Rastrigin** problem. Average values are depicted by red squares, median values by red dashes and variance information is given by the length of the boxes.

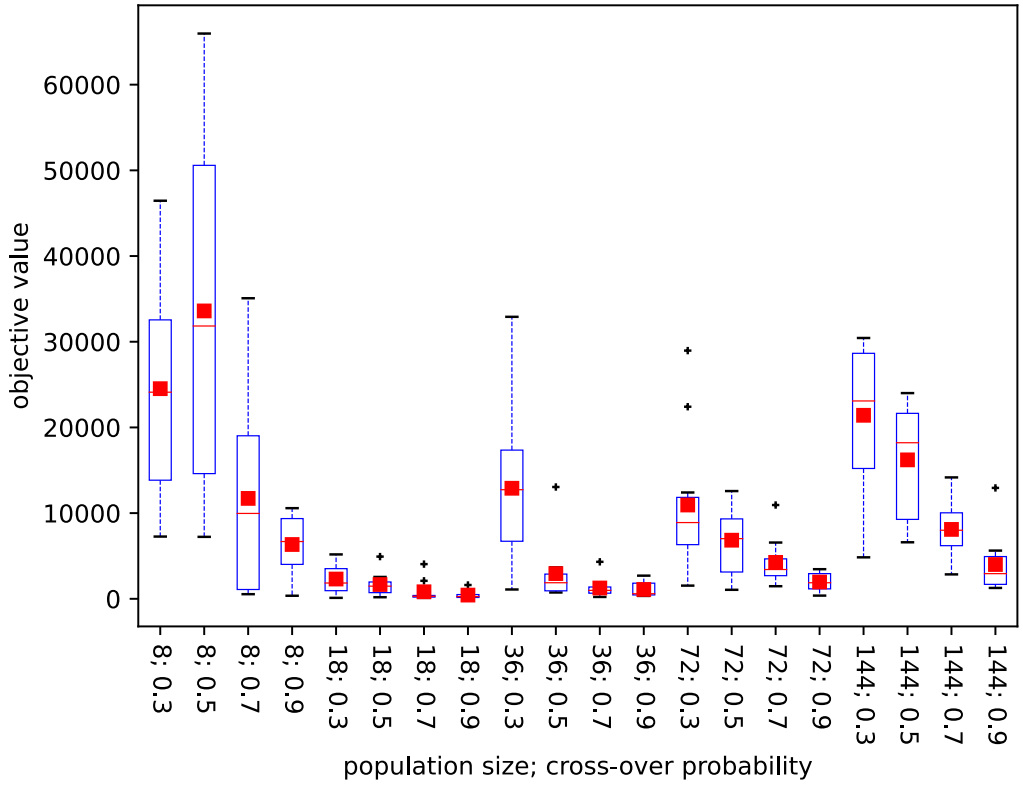

Figure B.4: **Calibration of parallel EA**. Distribution of the best objective values from the 10 repetitions of the experiments on the **Rosenbrock** problem. Average values are depicted by red squares, median values by red dashes and variance information is given by the length of the boxes.

Table B.6: **Calibration of parallel EA**. Statistics of the distribution of the best objective values from the 10 repetitions of the experiments on the **Rastrigin** problem. Ordering according to ascending average best objective value from top to bottom. The best value for each column appears in bold.

n_{pop}	p_c	Average	Median	Minimum	Variance
72	0.9	**21.65**	**21.85**	**13.6**	18.79
72	0.7	27.63	26.79	14.48	48.61
36	0.9	27.9	30.35	19.97	28.24
36	0.7	34.33	30.85	19.35	168.97
72	0.5	42.32	38.33	30.28	114.27
18	0.9	42.37	40.54	18.96	174.72
36	0.5	47.03	46.97	26.13	130.18
144	0.9	47.27	46.52	38.01	40.08
72	0.3	49.35	45.24	32.66	119.86
18	0.7	51.29	50.38	30.04	215.73
36	0.3	52.56	50.39	34.49	283.84
18	0.5	56.74	54.59	35.2	183.72
144	0.7	58.97	57.42	52.43	23.21
18	0.3	62.66	64.03	37.54	204.24
144	0.5	65.62	65.45	58.01	**10.98**
8	0.9	73.56	68.07	50.8	357.22
144	0.3	73.65	76.37	62.32	32.3
8	0.3	75.26	74.12	39.37	444.26
8	0.5	78.79	75.8	49.9	507.6
8	0.7	87.96	86.41	57.41	586.07

Table B.7: **Calibration of parallel EA**. Statistics of the distribution of the best objective values from the 10 repetitions of the experiments on the **Rosenbrock** problem. Ordering according to ascending average best objective value from top to bottom. The best value for each column appears in bold.

n_{pop}	p_c	Average	Median	Minimum	Variance
18	0.9	**433.91**	**259.33**	**74.26**	**191251.85**
18	0.7	820.67	295.8	110.05	1466702.09
36	0.9	1092.0	592.06	336.37	658936.44
36	0.7	1261.31	984.69	216.17	1191134.25
18	0.5	1665.09	1477.32	184.67	1661523.63
72	0.9	1959.02	1872.94	373.53	1082062.41
18	0.3	2307.74	1843.04	110.43	2586536.83
36	0.5	2942.54	1875.14	736.89	12259372.39
144	0.9	4001.54	2935.33	1267.18	11060605.09
72	0.7	4252.02	3417.06	1469.76	6778061.45
8	0.9	6337.49	6684.22	358.06	11193763.83
72	0.5	6839.66	7021.64	1050.14	15688530.4
144	0.7	8099.69	8001.32	2850.41	11235759.19
72	0.3	10946.07	8896.91	1538.18	66445555.65
8	0.7	11717.1	9962.93	539.94	120357282.62
36	0.3	12901.81	12731.52	1088.82	78500881.02
144	0.5	16214.5	18213.14	6596.36	45565006.41
144	0.3	21415.34	23089.57	4840.79	63520417.44
8	0.3	24522.5	24112.23	7262.69	155115614.92
8	0.5	33595.52	31836.06	7229.95	395710854.33

Table B.8: **Calibration of SaaEF with BNN_MCD**. Statistics of the distribution of the best objective values from the 10 repetitions of the experiments on the **Schwefel** problem. Ordering according to ascending average best objective value from top to bottom. The best value for each column appears in bold.

n_{chld}	(δ_{ES}, n_{ES}) cross. val.	Average	Median	Minimum	Variance	q
288	$(10^{-8}, 32)$ yes	**516.82**	**480.87**	190.58	66999.17	72
-	No surrogate	607.91	503.25	253.29	71295.35	-
144	$(10^{-8}, 32)$ yes	685.72	558.1	276.4	210240.3	36
288	$(10^{-8}, 32)$ no	872.15	828.71	483.86	58040.28	72
288	$(10^{-4}, 8)$ yes	893.4	820.97	502.66	80785.52	72
288	$(10^{-4}, 8)$ no	1011.34	926.27	650.28	73748.54	72
144	$(10^{-4}, 8)$ yes	1060.04	1004.16	649.05	65682.39	36
144	$(10^{-8}, 32)$ no	1085.84	1103.41	686.69	66753.01	36
144	$(10^{-4}, 8)$ no	1125.48	1085.87	721.91	89206.08	36

Table B.9: **Calibration of SaaEF with BNN_MCD**. Statistics of the distribution of the best objective values from the 10 repetitions of the experiments on the **Rastrigin** problem. Ordering according to ascending average best objective value from top to bottom. The best value for each column appears in bold.

n_{chld}	(δ_{ES}, n_{ES}) cross. val.	Average	Median	Minimum	Variance	q
-	No surrogate	**23.30**	**22.66**	**12.99**	30.58	-
144	$(10^{-8}, 32)$ no	29.31	28.47	16.02	86.36	36
288	$(10^{-8}, 32)$ yes	29.4	27.0	21.39	44.37	72
288	$(10^{-4}, 8)$ yes	31.06	34.95	15.98	68.34	72
288	$(10^{-8}, 32)$ no	33.41	31.92	19.56	67.76	72
144	$(10^{-8}, 32)$ yes	34.76	36.64	23.65	34.01	36
144	$(10^{-4}, 8)$ no	35.38	34.6	19.21	166.75	36
144	$(10^{-4}, 8)$ yes	35.68	35.87	28.79	**17.78**	36
288	$(10^{-4}, 8)$ no	38.08	33.3	21.93	193.38	72

Table B.10: **Calibration of SaaEF with BNN_MCD**. Statistics of the distribution of the best objective values from the 10 repetitions of the experiments on the **Rosenbrock** problem. Ordering according to ascending average best objective value from top to bottom. The best value for each column appears in bold.

n_{chld}	(δ_{ES}, n_{ES}) cross. val.	Average	Median	Minimum	Variance	q
144	$(10^{-8}, 32)$ yes	**1096.8**	810.92	446.3	488068.36	36
144	$(10^{-4}, 8)$ yes	1105.28	**755.04**	339.37	1006032.14	36
-	No surrogate	1191.14	757.03	246.71	1557771.06	-
288	$(10^{-8}, 32)$ yes	1359.59	1268.25	745.12	**336182.97**	72
288	$(10^{-4}, 8)$ yes	1407.53	1052.41	**198.69**	869984.37	72
288	$(10^{-4}, 8)$ no	1414.96	1293.39	280.94	732488.17	72
288	$(10^{-8}, 32)$ no	1989.58	1973.39	528.56	1482968.13	72
144	$(10^{-4}, 8)$ no	2019.27	1034.21	392.79	4621452.92	36
144	$(10^{-8}, 32)$ no	4387.67	3873.49	1389.24	5623346.82	36

Table B.11: **P-SAEAs with SaaEF** on the **Schwefel** problem. Top-5 ECs according to the average, median and minimum best objective value (10 independent repetitions). Ordering per cell according to ascending value from top to bottom. Surrogates ordered according to decreasing performance from top to bottom.

EC	Average	EC	Median	EC	Min
BNN_MCD_5					
dyn-df-incl	131.95	*dyn-dpf-excl*	115.37	*dyn-df-excl*	33.37
dyn-dpf-excl	136.3	*par-fd-cd*	124.48	*par-fs-hvc*	81.27
par-fd-cd	167.55	*dyn-df-incl*	133.46	*par-fd-cd*	85.73
dyn-df-excl	168.64	*dyn-df-excl*	143.08	*dyn-dpf-excl*	86.8
com-dpf	255.12	*par-fd-hvc*	213.62	*par-fd-hvc*	87.71
GP_RBF					
dyn-spf-excl	206.19	*dyn-dpf-excl*	194.67	*dyn-sf-excl*	42.6
dyn-dpf-excl	209.51	*dyn-spf-excl*	220.85	*dyn-spf-excl*	49.94
dyn-df-75-excl	233.11	*dyn-sf-incl*	222.1	*com-dpf*	53.55
dyn-sf-excl	256.31	*dyn-sf-75-excl*	242.18	*dyn-df-75-excl*	63.89
dyn-sf-incl	261.79	*dyn-df-excl*	246.16	*dyn-dpf-excl*	74.35
rKRG_18					
dyn-df-75-excl	216.06	*dyn-dpf-excl*	205.45	*dyn-df-75-excl*	61.1
dyn-dpf-excl	316.37	*dyn-df-75-excl*	231.9	*dyn-dpf-excl*	97.37
dyn-df-incl	372.3	*dyn-df-incl*	272.28	*dist*	149.85
dyn-df-excl	376.67	*dyn-df-excl*	360.51	*par-fd-hvc*	152.98
stdev	415.78	*stdev*	378.03	*dyn-df-excl*	171.34
ANN_BLR					
dyn-df-incl	267.78	*dyn-df-incl*	302.64	*par-fd-cd*	37.78
dyn-df-75-excl	381.19	*dyn-dpf-excl*	368.73	*dyn-df-incl*	38.11
dyn-dpf-excl	381.88	*dyn-df-excl*	385.51	*dyn-dpf-excl*	87.79
dist	454.32	*dyn-df-75-excl*	399.24	*dyn-df-75-excl*	121.54
dyn-df-excl	476.63	*dist*	456.36	*dist*	181.56

Table B.12: **P-SAEAs with SaaEF** on the **Rastrigin** problem. Top-5 ECs according to the average, median and minimum best objective value (10 independent repetitions). Ordering per cell according to ascending value from top to bottom. Surrogates ordered according to decreasing performance from top to bottom.

EC	Average	EC	Median	EC	Min
		GP_RBF			
dyn-dpf-excl	19.08	*dyn-dpf-excl*	14.95	*dyn-dpf-excl*	10.79
par-fd-cd	19.65	*com-dpf*	16.65	*dyn-fpd-excl*	12.4
com-dpf	20.42	*par-fd-cd*	18.05	*dyn-df-incl*	13.58
dyn-df-excl	21.25	*dyn-df-excl*	21.07	*par-fd-cd*	13.7
dyn-df-incl	22.96	*dyn-df-incl*	22.65	*dyn-df-excl*	14.09
		rKRG_36			
dyn-spf-excl	22.66	*dyn-spf-excl*	20.41	*dyn-spf-excl*	12.07
par-fs-cd	24.47	*par-fs-cd*	23.55	*par-fs-cd*	13.84
dyn-sf-incl	25.36	*dyn-sf-incl*	24.08	*com-spf*	14.2
par-fs-hvc	26.87	*dyn-dpf-excl*	26.13	*dyn-df-excl*	14.35
ei	28.09	*par-fs-hvc*	26.8	*par-fs-hvc*	14.72
		ANN_BLR			
dyn-sf-excl	22.72	*dyn-sf-excl*	22.52	*par-fs-hvc*	12.66
dyn-sf-incl	23.74	*dyn-sf-incl*	23.25	*rand*	12.78
dyn-spf-excl	25.41	*par-fs-hvc*	24.47	*dyn-sf-excl*	13.71
dyn-dpf-excl	25.72	*dyn-df-incl*	24.58	*bp*	16.09
par-fs-hvc	25.87	*dyn-spf-excl*	24.88	*dyn-sf-75-excl*	17.84
		BNN_MCD_5			
dyn-df-excl	23.4	*dyn-spf-excl*	21.39	*dyn-sf-75-excl*	11.89
dyn-spf-excl	23.56	*pi*	23.12	*ada-wang-max*	12.89
dyn-sf-incl	23.61	*dyn-df-excl*	23.6	*com-spf*	13.52
pi	25.18	*dyn-sf-incl*	24.5	*dyn-fps-excl*	13.56
rand	25.23	*ada-wang-min*	24.92	*dyn-sf-incl*	13.77

Table B.13: **P-SAEAs with SaaEF** on the **Rosenbrock** problem. Top-5 ECs according to the average, median and minimum best objective value (10 independent repetitions). Ordering per cell according to ascending value from top to bottom. Surrogates ordered according to decreasing performance from top to bottom.

EC	Average	EC	Median	EC	Min
		GP_RBF			
par-fd-hvc	233.93	*par-fd-cd*	157.34	*par-fd-hvc*	41.95
dyn-fpd-excl	263.56	*dyn-fpd-excl*	182.28	*lcb*	65.82
dist	272.13	*dyn-fd-incl*	184.74	*pi*	69.01
par-fs-hvc	306.48	*par-fs-hvc*	206.18	*dyn-fd-excl*	70.13
par-fd-cd	316.21	*par-fd-hvc*	207.13	*par-fd-cd*	72.85
		rKRG_18			
par-fd-hvc	414.44	*par-fd-hvc*	326.97	*dyn-sf-excl*	61.25
par-fs-hvc	435.21	*dyn-dpf-excl*	379.4	*com-dpf*	104.12
dyn-dpf-excl	470.8	*pi*	385.36	*dyn-spf-excl*	122.29
par-fd-cd	498.11	*dyn-df-excl*	388.25	*dyn-dpf-excl*	145.01
dist	568.01	*par-fs-hvc*	398.37	*dyn-df-excl*	148.92
		BNN_MCD_5			
dist	713.71	*dyn-fpd-excl*	581.12	*dyn-df-75-excl*	219.35
com-dpf	728.97	*dyn-fd-excl*	630.66	*dyn-fd-excl*	228.24
dyn-fd-incl	741.21	*dyn-fd-incl*	669.79	*ei*	233.89
dyn-fpd-excl	795.48	*com-dpf*	706.4	*dyn-fpd-excl*	267.99
dyn-df-75-excl	884.51	*dist*	728.55	*ada-wang-min*	271.43
		ANN_BLR			
dyn-dpf-excl	2063.52	*dyn-dpf-excl*	1797.39	*dyn-fs-excl*	378.89
par-fd-hvc	2748.7	*par-fd-hvc*	2522.99	*bp*	595.32
par-fd-cd	3247.48	*par-fd-cd*	2783.59	*dyn-dpf-excl*	705.46
bp	3368.46	*ada-wang-min*	3195.2	*lcb*	725.55
dyn-df-excl	3833.16	*dyn-fpd-excl*	3344.09	*dyn-df-75-excl*	988.39

Table B.14: **P-SAEAs with SaaEF** on the **Covid-19 contact reduction** problem. Top-5 ECs according to the average, median and minimum best objective value (10 independent repetitions). Ordering per column according to ascending value from top to bottom. Surrogates ordered according to decreasing performance from top to bottom.

EC	Average	EC	Median	EC	Min
		BNN_MCD_5			
dyn-df-75-excl	7455	*dist*	6284	*dist*	4385
dyn-df-excl	7679	*dyn-df-excl*	6693	*dyn-df-excl*	4876
dist	7837	*dyn-df-75-excl*	7047	*ada-df*	5020
ada-df	8232	*dyn-df-incl*	7468	*par-fd-hvc*	5103
dyn-df-incl	8983	*ada-df*	7645	*dyn-df-75-excl*	5252
		rKRG_18			
dyn-df-incl	9254	*dyn-df-incl*	8552	*dist*	5933
dyn-df-75-excl	10107	*dyn-df-75-excl*	9425	*dyn-dpf-excl*	6419
dyn-df-excl	10901	*dyn-df-excl*	9969	*dyn-df-incl*	6622
dist	11395	*dist*	10370	*com-dpf*	6698
dyn-dpf-excl	11847	*dyn-dpf-excl*	11851	*dyn-df-excl*	7085
		GP_RBF			
ada-df	16595	*ada-df*	15824	*dyn-fd-incl*	8517
ada-dpf	17205	*ada-dpf*	16599	*dyn-sf-incl*	9792
dyn-dpf-excl	19067	*dist*	16634	*ada-df*	10023
dist	20471	*dyn-df-incl*	17595	*pi*	10891
dyn-sf-incl	20498	*dyn-dpf-excl*	19064	*dyn-fps-excl*	11246
		ANN_BLR			
dyn-df-excl	18449	*dyn-df-excl*	17546	*ada-dpf*	10487
dyn-df-75-excl	19601	*ada-df*	18105	*ada-df*	10674
dyn-dpf-excl	20195	*dyn-df-75-excl*	18698	*par-fd-hvc*	10941
ada-df	20201	*dyn-dpf-excl*	19164	*ada-wang-max*	10953
dyn-df-incl	21175	*dyn-df-incl*	19246	*dyn-df-incl*	11089

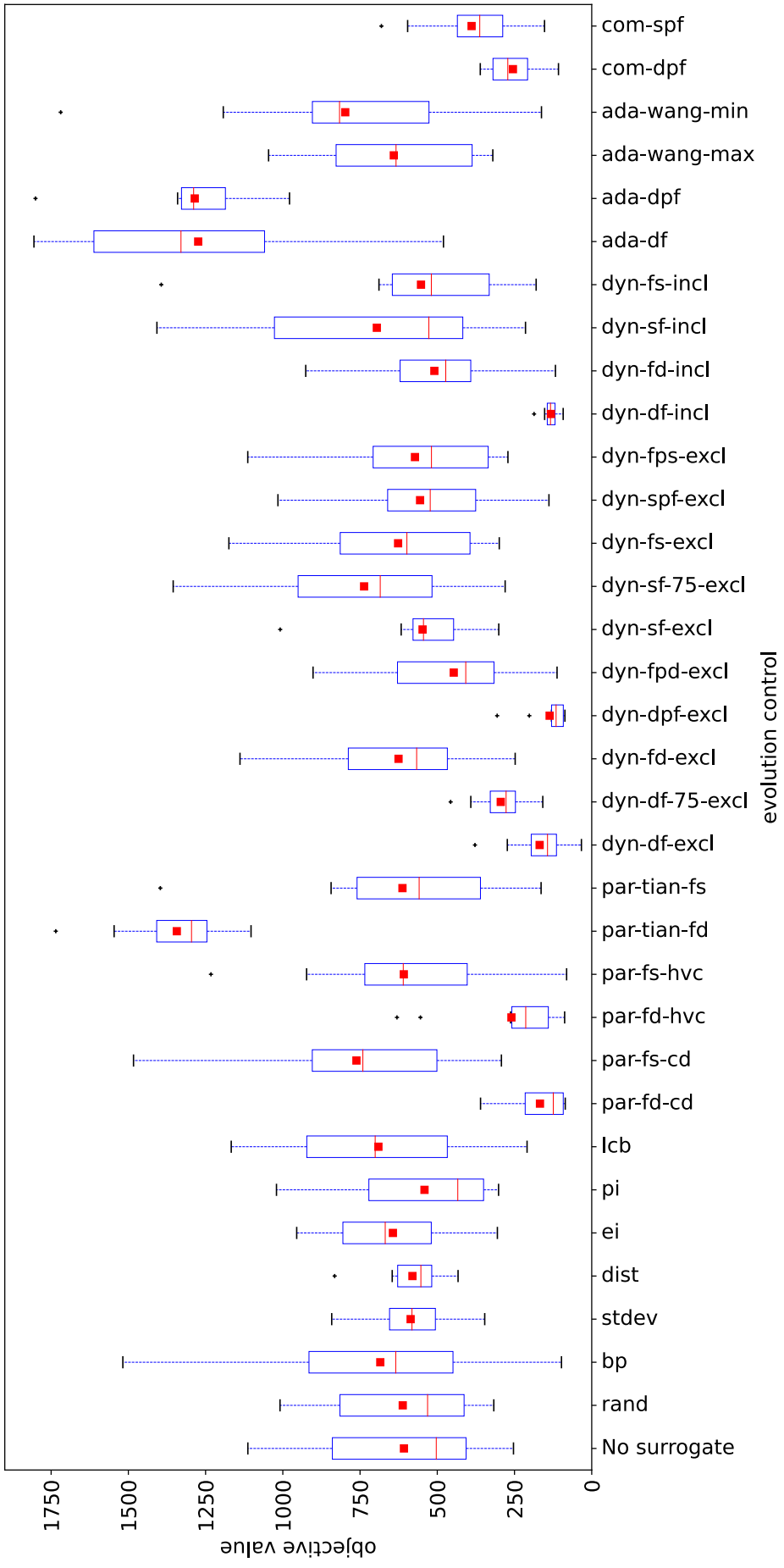

Figure B.5: **P-SAEAs with SaaEF and BNN_MCD_5** on the **Schwefel** problem. Distribution of the best objective values from the 10 repetitions of the experiment. Averaged values are depicted by red squares, median values by red dashes and variance information is given by the length of the boxes.

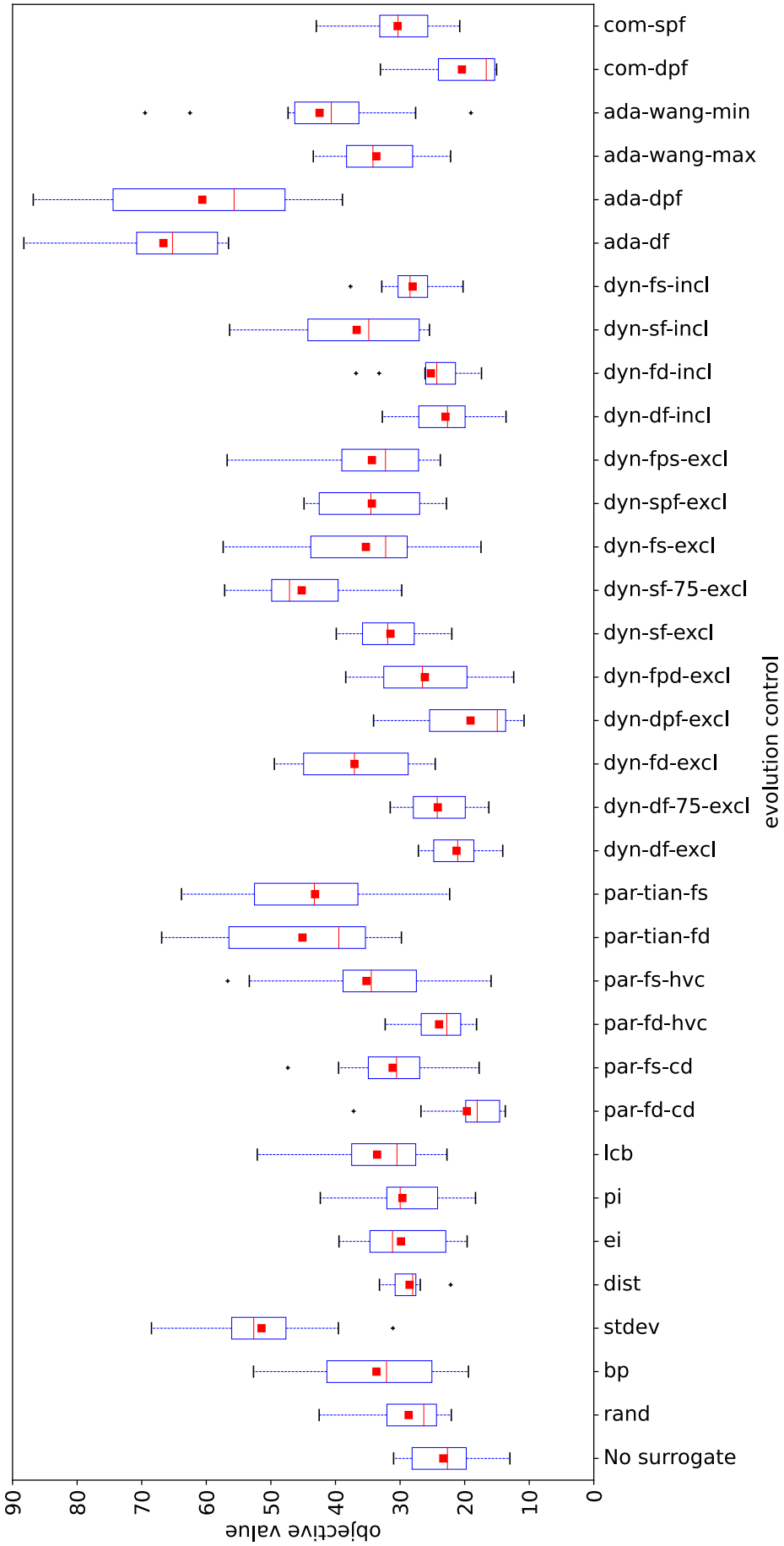

Figure B.6: **P-SAEAs with SaaEF and GP_RBF** on the **Rastrigin** problem. Distribution of the best objective values from the 10 repetitions of the experiment. Averaged values are depicted by red squares, median values by red dashes and variance information is given by the length of the boxes.

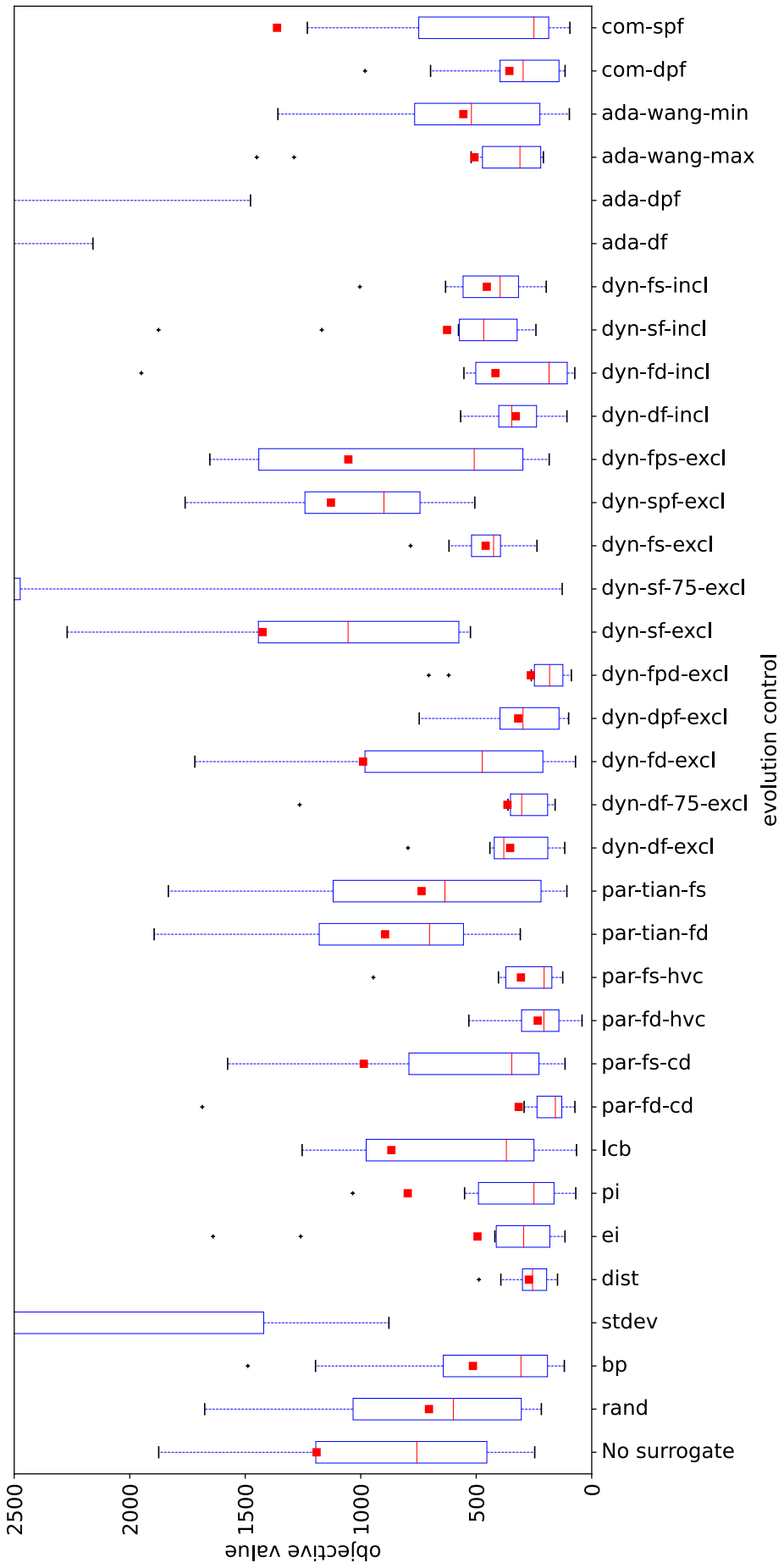

Figure B.7: **P-SAEAs with SaaEF and GP_RBF** on the **Rosenbrock** problem. Distribution of the best objective values from the 10 repetitions of the experiment. Averaged values are depicted by red squares, median values by red dashes and variance information is given by the length of the boxes.

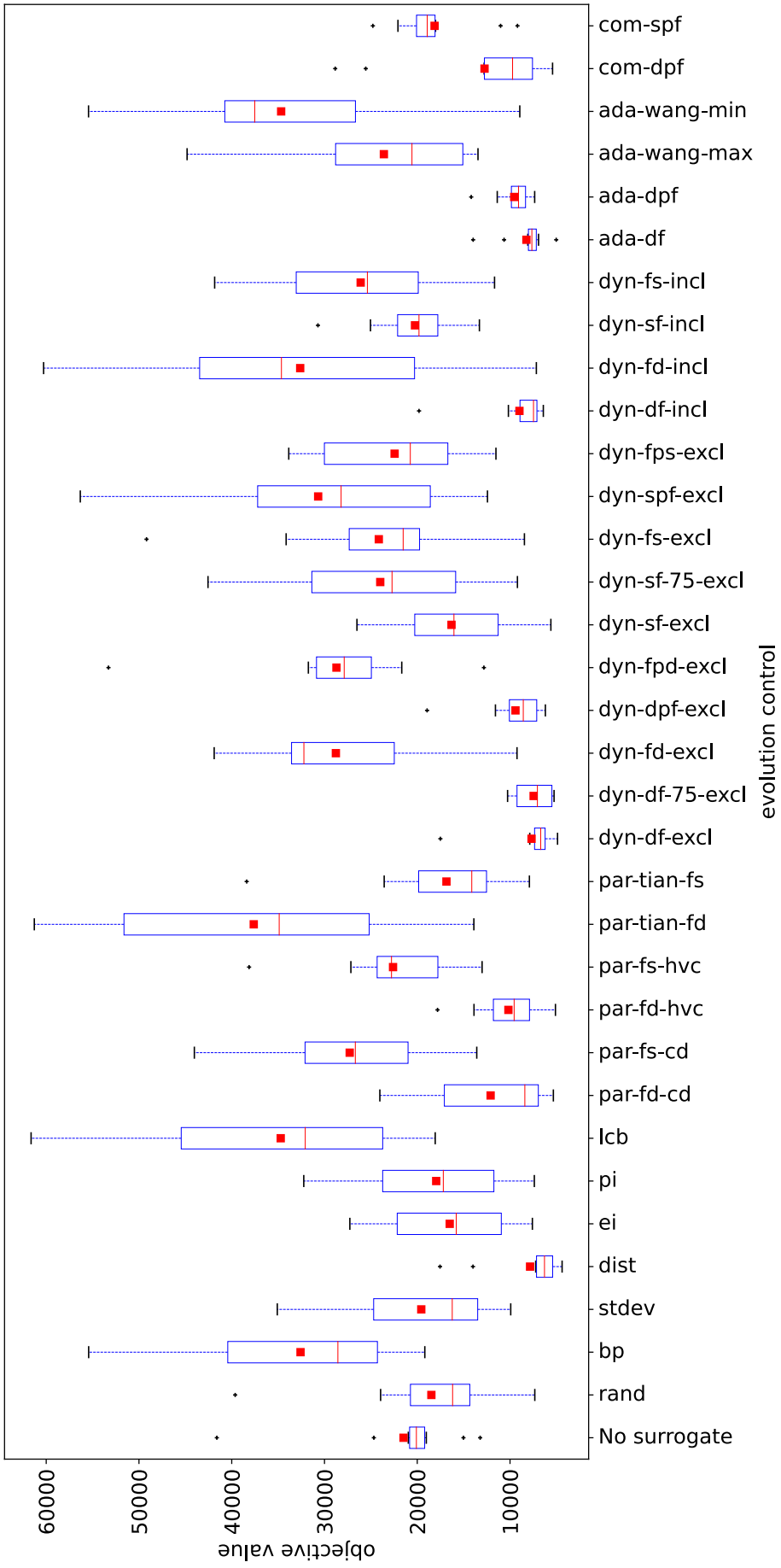

Figure B.8: **P-SAEAs with SaaEF and BNN_MCD_5** on the **Covid-19 contact reduction problem.** Distribution of the best objective values from the 10 repetitions of the experiment. Averaged values are depicted by red squares, median values by red dashes and variance information is given by the length of the boxes.

Appendix C

Multi-objective optimization

In multi-objective optimization, the comparison between two solutions is performed according to multiple criteria simultaneously [OJS03; Hub+06]. Given two decision vectors $\boldsymbol{x^{(1)}}$ and $\boldsymbol{x^{(2)}}$ and X a set of decision vectors containing $\boldsymbol{x^{(1)}}$, it is stated that:

- $\boldsymbol{x^{(1)}}$ dominates $\boldsymbol{x^{(2)}}$ if $\boldsymbol{x^{(1)}}$ is at least better than $\boldsymbol{x^{(2)}}$ regarding one criterion and as good as $\boldsymbol{x^{(2)}}$ regarding the remaining criteria;
- $\boldsymbol{x^{(1)}}$ and $\boldsymbol{x^{(2)}}$ are incomparable if no solution dominates the other and there exists at least one criterion whose value differs from $\boldsymbol{x^{(1)}}$ to $\boldsymbol{x^{(2)}}$;
- $\boldsymbol{x^{(1)}}$ is a non-dominated solution of X if there is no solution in X that dominates $\boldsymbol{x^{(1)}}$.

Given a population of solutions, the subset composed of non-dominated individuals, such that one non-dominated individual is incomparable with all other non-dominated individuals, is called the Non-Dominated Set (NDS). The corresponding population of objective vectors is called the Non-Dominated Front (NDF). The NDSs are ranked in decreasing order of interest such that each solution of the i-th NDS dominates each solution of the $(i+1)$-th NDS. The rank is called the Non-Dominated Rank (NDR). The overall best NDF for a given problem is called the Pareto Front (PF) of this problem and the corresponding set in the search space is called the Pareto Set (PS).

To distinguish between solutions with the same NDR, it is proposed to rely on two metrics: the crowding distance [Deb+02] and the hyper-volume contribution [Rua+20]. For a given non-extreme solution, its crowding distance $cd()$ is the average distance, in the objective space, between its closest neighbors of the same NDR along each objective. For a bi-dimensional objective space, the crowding distance is the average side length of the rectangle formed based on the two nearest neighbors of the studied solution as depicted in Figure C.1. For each objective function, solutions having the smallest and the largest function value are called extreme solutions such as $\boldsymbol{x^{(1)}}$ and $\boldsymbol{x^{(3)}}$ in Figure C.1. Extreme solutions are assigned an infinite crowding distance. A higher value of the crowding distance exhibits a solution with a wider empty neighborhood, which is attractive to enhance the distribution of the NDF.

The hyper-volume $hv()$ is a metric originally developed to compare NDFs. For a bi-dimensional objective space, given a reference point O', the hyper-volume of a NDF with respect to O' is the area of the surface delimited by O' and the NDF as depicted in Figure C.2. Larger values of the hyper-volume indicate better objectives values (convergence) and a larger number of candidates (diversity) within the NDF. The definition of hyper-volume is easily generalized to higher dimensions. In Figure C.2, the hatched area represents the hyper-volume contribution $hvc()$ of the candidate $\boldsymbol{x^{(2)}}$. The hyper-volume contribution

is defined as the difference between the hyper-volume of the entire NDF and the hyper-volume of the NDF without the considered solution. For the example presented in Figure C.2:

$$hvc(\boldsymbol{x^{(2)}}) = hv(\boldsymbol{x^{(1)}}, \boldsymbol{x^{(2)}}, \boldsymbol{x^{(3)}}, O') - hv(\boldsymbol{x^{(1)}}, \boldsymbol{x^{(3)}}, O') \tag{C.1}$$

The difference between the effects of the crowding distance and the hyper-volume contribution is quite subtle. The crowding distance only takes into account the solutions with the same NDR while the hyper-volume takes into account all the solutions. In Figure C.1, the crowding distance of $\boldsymbol{x^{(2)}}$ is not influenced by the solution $\boldsymbol{x^{(4)}}$, while in Figure C.2, the hyper-volume contribution of $\boldsymbol{x^{(2)}}$ is influenced by $\boldsymbol{x^{(4)}}$.

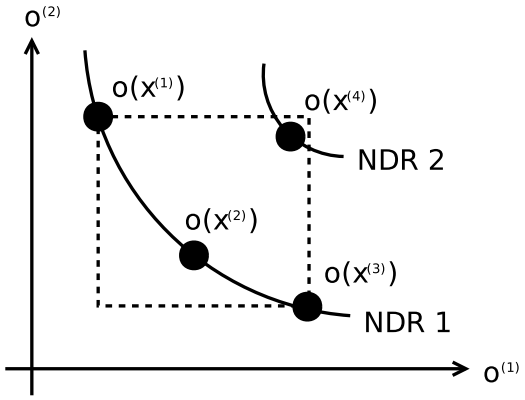

Figure C.1: Illustration of the crowding distance in a bi-objective space $\boldsymbol{o} = (o^{(1)}, o^{(2)})$. The crowding distance of $\boldsymbol{x^{(2)}}$ is the average side length of the rectangle drawn with dashed lines. The crowding distance of extreme solutions $\boldsymbol{x^{(1)}}$ and $\boldsymbol{x^{(2)}}$ is infinite.

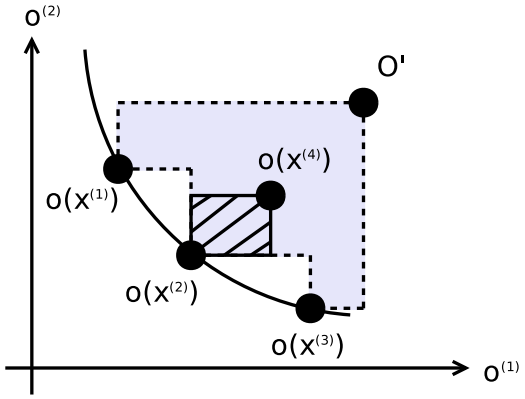

Figure C.2: Illustration of the hyper-volume for a reference point O' and a NDS made of candidates $\boldsymbol{x^{(1)}}$, $\boldsymbol{x^{(2)}}$ and $\boldsymbol{x^{(3)}}$. The hyper-volume is the surface of the shaded area in the bi-objective space $\boldsymbol{o} = (o^{(1)}, o^{(2)})$. The hatched area represents the contribution of $\boldsymbol{x^{(2)}}$ to the hyper-volume.

Algorithm 15 Non-dominated Sorting Genetic Algorithm II

Input
n_{pop}: population size
n_{gen}: number of generations

$\mathcal{P} \leftarrow$ initial_sampling(n_{pop})
for $i = 1 : n_{gen}$ **do**
  $\mathcal{P} \leftarrow$ non_dominated_and_crowded_distance_sorting($\mathcal{P}$)
  $\mathcal{P}_{par} \leftarrow$ tournament_position($\mathcal{P}$, n_{pop}) ▷ population of parents
  $\mathcal{P}_{chld} \leftarrow$ reproduction($\mathcal{P}_{par}$) ▷ population of children
  $\mathcal{P} \leftarrow \mathcal{P} \cup \mathcal{P}_{chld}$
  $\mathcal{P} \leftarrow$ non_dominated_and_crowded_distance_sorting($\mathcal{P}$)
  $\mathcal{P} \leftarrow$ elitist_position($\mathcal{P}$, n_{pop})
end for
return best NDF from $\mathcal{P}$

Algorithm 16 Reference Vector guided Evolutionary Algorithm

Input
n_{ref}: number of reference vectors
n_{gen}: maximum number of generations
f_{upd}: frequency of update

$\mathcal{V}_1 \leftarrow$ simplex_lattice(n_{ref}) ▷ initial set of reference vectors
$\mathcal{P}_1 \leftarrow$ initial_sampling(n_{ref})
evaluation($\mathcal{P}_1$)
for $i = 1 : n_{gen}$ **do**
  $\mathcal{P}_i^{par} \leftarrow$ select_parents($\mathcal{P}_i$)
  $\mathcal{P}_i^{chld} \leftarrow$ reproduction($\mathcal{P}_i^{par}$)
  evaluation($\mathcal{P}_i^{chld}$)
  $\mathcal{P}_i \leftarrow \mathcal{P}_i \cup \mathcal{P}_i^{chld}$
  $\mathcal{P}_{i+1} \leftarrow$ reference_vector_guided_replacement(i, $\mathcal{P}_i$, $\mathcal{V}_i$)
  if $i \bmod \lfloor n_{gen} . f_{upd} \rfloor == 0$ **then**
    $\mathcal{V}_{i+1} \leftarrow$ reference_vector_update(i, $\mathcal{P}_{i+1}$, $\mathcal{V}_i$, $\mathcal{V}_1$)
  else
    $\mathcal{V}_{i+1} \leftarrow \mathcal{V}_i$
  end if
end for
return best NDF from $\mathcal{P}_{n_{gen}+1}$

Algorithm 17 RVEA*

Input
n_{ref}: number of reference vectors
n_{gen}: maximum number of generations
f_{upd}: frequency of update

$\mathcal{V}_1 \leftarrow$ simplex_lattice(n_{ref})
$\mathcal{V}_1^* \leftarrow$ copy($\mathcal{V}_1$) ▷ additional set of reference vectors
$\mathcal{P}_1 \leftarrow$ initial_sampling(n_{ref})
evaluation($\mathcal{P}_1$)
for $i = 1 : n_{gen}$ **do**
  $\mathcal{P}_i^{par} \leftarrow$ select_parents($\mathcal{P}_i$)
  $\mathcal{P}_i^{chld} \leftarrow$ reproduction($\mathcal{P}_i^{par}$)
  evaluation($\mathcal{P}_i^{chld}$)
  $\mathcal{P}_i \leftarrow \mathcal{P}_i \cup \mathcal{P}_i^{chld}$
  $\mathcal{P}_{i+1} \leftarrow$ reference_vector_guided_replacement(i, $\mathcal{P}_i$, $\mathcal{V}_i \cup (\mathcal{V}_i^* \backslash \mathcal{V}_i)$)
  if $i \bmod \lfloor n_{gen} . f_{upd} \rfloor == 0$ **then**
    $\mathcal{V}_{i+1} \leftarrow$ reference_vector_update(i, $\mathcal{P}_{i+1}$, $\mathcal{V}_i$, $\mathcal{V}_1$)
  else
    $\mathcal{V}_{i+1} \leftarrow \mathcal{V}_i$
  end if
  $\mathcal{V}_{i+1}^* \leftarrow$ regeneration($\mathcal{P}_{i+1}$, $\mathcal{V}_i^*$)
end for
return best NDF from $\mathcal{P}_{n_{gen}+1}$

Algorithm 18 Adaptive Bayesian Multi-Objective Evolutionary Algorithm

Input
simulator: real objective function
surrogate: surrogate model
$\hat{\boldsymbol{f}}$: surrogate model predictive function
budget: budget for the search
q: number of simulations per cycle

database $\leftarrow$ initial_sampling(*simulator*)
surrogate $\leftarrow$ training(*database*)
$b_c \leftarrow 0$
while $b_c \leqslant$ *budget* **do**
  $(\mathcal{B}, \mathcal{V}) \leftarrow$ RVEA(105, 20, 0.1) ▷ last population and reference vector set from Algorithm 16 for the problem min $\hat{\boldsymbol{f}}$
  update(b_c)
  $\alpha \leftarrow -0.5 \cos\left(\frac{b_c}{budget}\pi\right) + 0.5$
  evaluate($\mathcal{B}$, $\boldsymbol{f_{ada}}$, α)
  $\mathcal{B}_{sim} \leftarrow$ adaptive_sampling_criterion($\mathcal{B}$, $\mathcal{V}$, α, q, b_c, *budget*) ▷ Algorithm 19
  parallel_evaluation(*simulator*, $\mathcal{B}_{sim}$)
  database $\leftarrow$ *database* $\cup \mathcal{B}_{sim}$
  surrogate $\leftarrow$ training(*database*)
end while
return best NDF from *database*

Algorithm 19 Adaptive Sampling Criterion in AB-MOEA

Input
$\mathcal{B}$: set of candidates
$\mathcal{V}$: set of reference vectors
α: adaptive parameter
q: number of candidates to retain
m: number of objectives
b_c: budget already spent
budget: total budget

for $i = 1 : |\mathcal{B}|$ **do**
  $\boldsymbol{y'_i} \leftarrow$ translate($\boldsymbol{y_i}$)
  $j \leftarrow$ sub_population_index($\mathcal{V}$, $\boldsymbol{y'_i}$)
  if $\alpha < 0.5$ **then**
    $d_i \leftarrow m\frac{\theta(\boldsymbol{y'_i},\boldsymbol{v_j})}{\gamma_{\boldsymbol{v_j}}}$
  else
    $d_i \leftarrow (1 + P(\theta(\boldsymbol{y'_i}, \boldsymbol{v_j}), m, b_c, budget)).||\boldsymbol{y'_i}||$
  end if
end for
$\mathcal{B} \leftarrow$ sort_per_sub_population($\boldsymbol{d}$, $\mathcal{B}$)
return q first candidates from $\mathcal{B}$

Algorithm 20 Surrogate-Assisted Evolutionary Algorithm for Medium Scale Expensive problems

Input
simulator: real objective function
surrogate: surrogate model
$\hat{\boldsymbol{f}}$: surrogate model predictive function
$\hat{\boldsymbol{s}}$: predictive standard deviation
budget: budget for the search
q: number of simulations per cycle

database $\leftarrow$ initial_sampling(*simulator*)
surrogate $\leftarrow$ training(*database*)
$b_c \leftarrow 0$
while *budget*$\neq 0$ **do**
  $\mathcal{B} \leftarrow$ NSGA-II(76, 100, $\hat{\boldsymbol{f}}$, $\hat{\boldsymbol{s}}$, *database*) ▷ last population from Algorithm 15 for the problem min($\hat{\boldsymbol{f}}$, $\hat{\boldsymbol{f}} - \hat{\boldsymbol{s}}^{\boldsymbol{2}}$). Initial population initialized with the last 76 solutions from *database*.
  $\mathcal{B} \leftarrow$ HVC_sorting($\hat{\boldsymbol{f}}$, $\mathcal{B}$)
  $\mathcal{B}_{sim} \leftarrow$ elitist_position($\mathcal{B}$, $\lfloor \frac{q}{2} \rfloor$)
  $\mathcal{B} \leftarrow$ HVC_sorting($\hat{\boldsymbol{f}} - 2\hat{\boldsymbol{s}}^{\boldsymbol{2}}$, $\mathcal{B}$)
  $\mathcal{B}_{sim} \leftarrow \mathcal{B}_{sim} \cup$ elitist_position($\mathcal{B}$, $\lfloor \frac{q}{2} \rfloor$)
  parallel_evaluation(*simulator*, $\mathcal{B}_{sim}$)
  database $\leftarrow$ *database* $\cup \mathcal{B}_{sim}$
  surrogate $\leftarrow$ training(*database*)
  budget $\leftarrow$ get_remaining_budget(*budget*, elapsed_time)
end while
return best NDF from *database*

Appendix D

Parallel Surrogate-driven algorithms

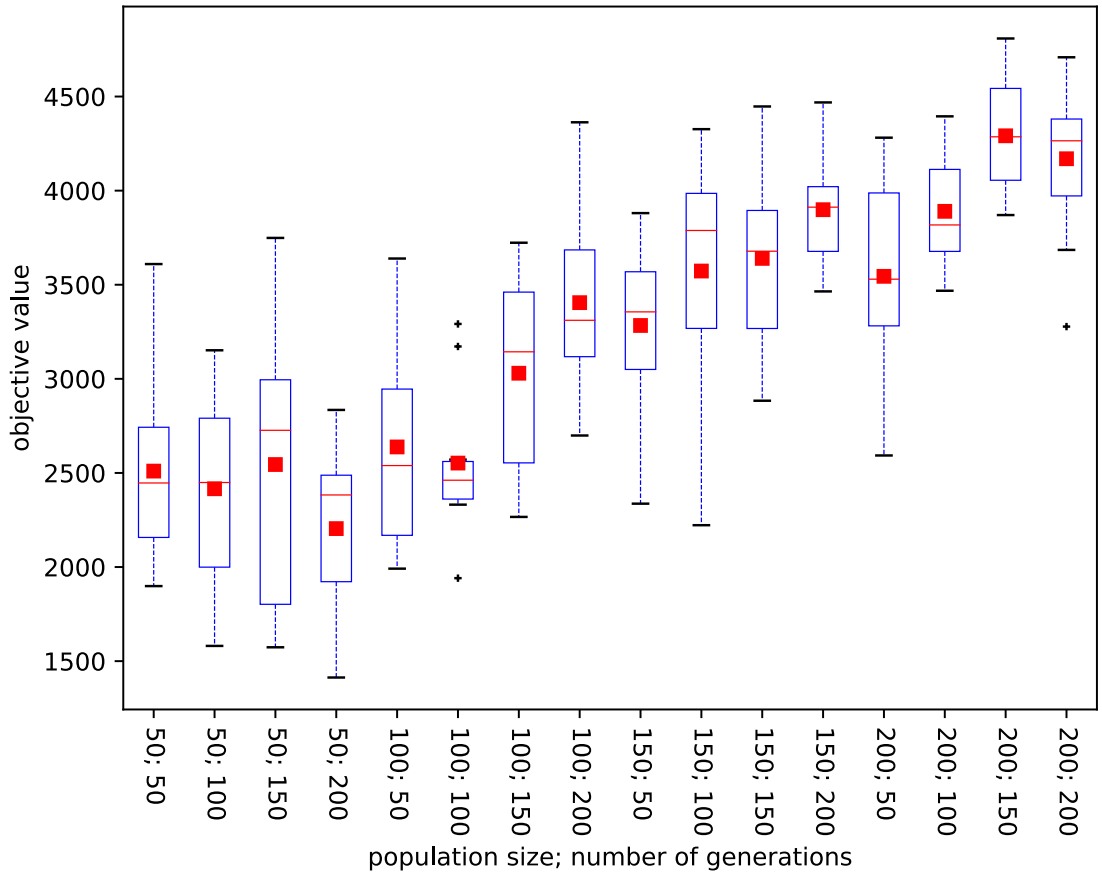

Figure D.1: **Calibration of EA with EC-based selection and replacement in q-EGO**. Distribution of the best objective values from the 10 repetitions of the experiments on the **Schwefel** problem. Average values are depicted by red squares, median values by red dashes and variance information is given by the length of the boxes.

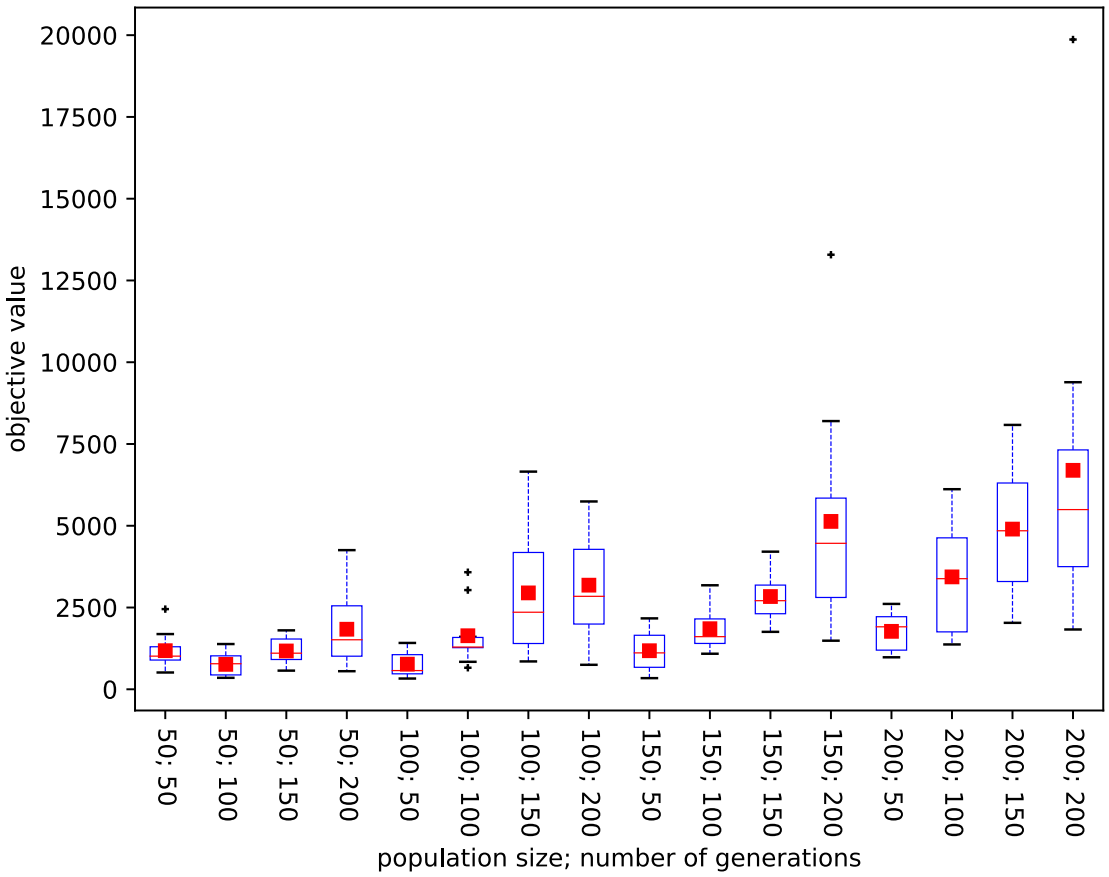

Figure D.2: **Calibration of EA with EC-based selection and replacement in q-EGO**. Distribution of the best objective values from the 10 repetitions of the experiments on the **Rosenbrock** problem. Average values are depicted by red squares, median values by red dashes and variance information is given by the length of the boxes.

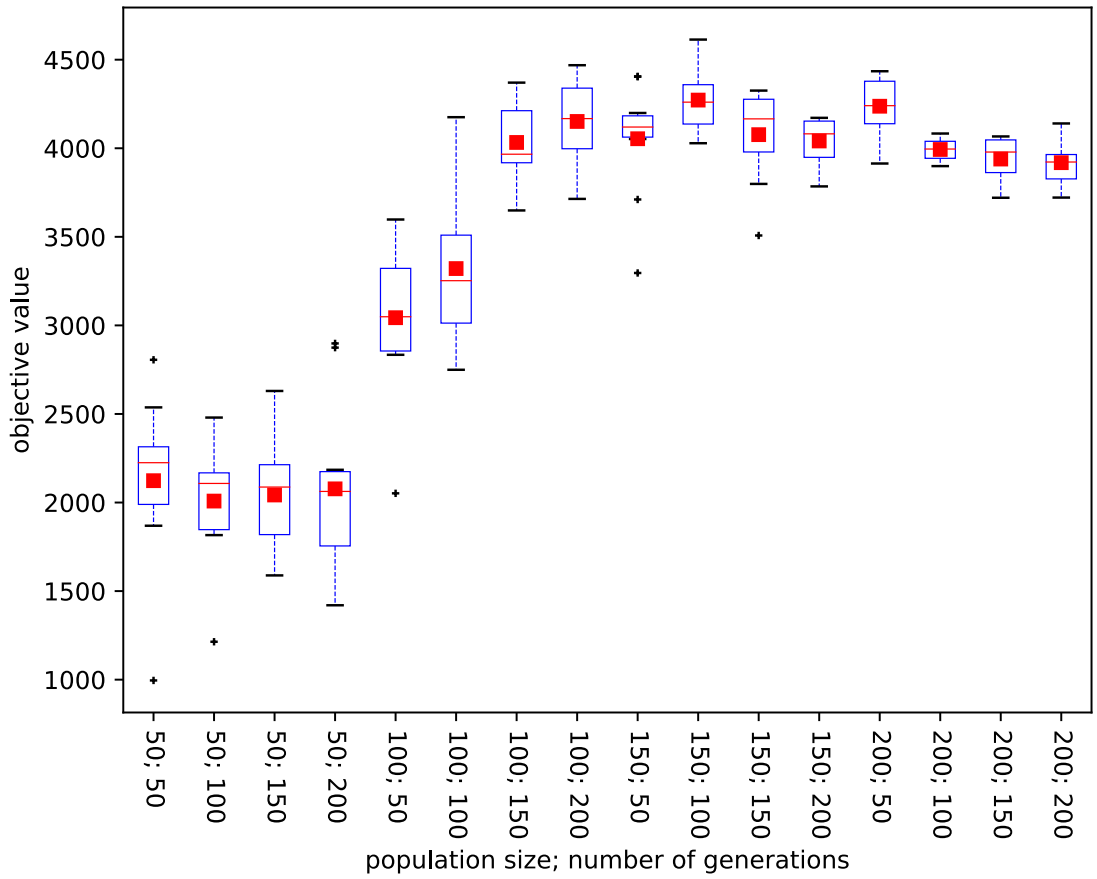

Figure D.3: **Calibration of EA with EC-based selection and replacement in *q-subnets***. Distribution of the best objective values from the 10 repetitions of the experiments on the **Schwefel** problem. Average values are depicted by red squares, median values by red dashes and variance information is given by the length of the boxes.

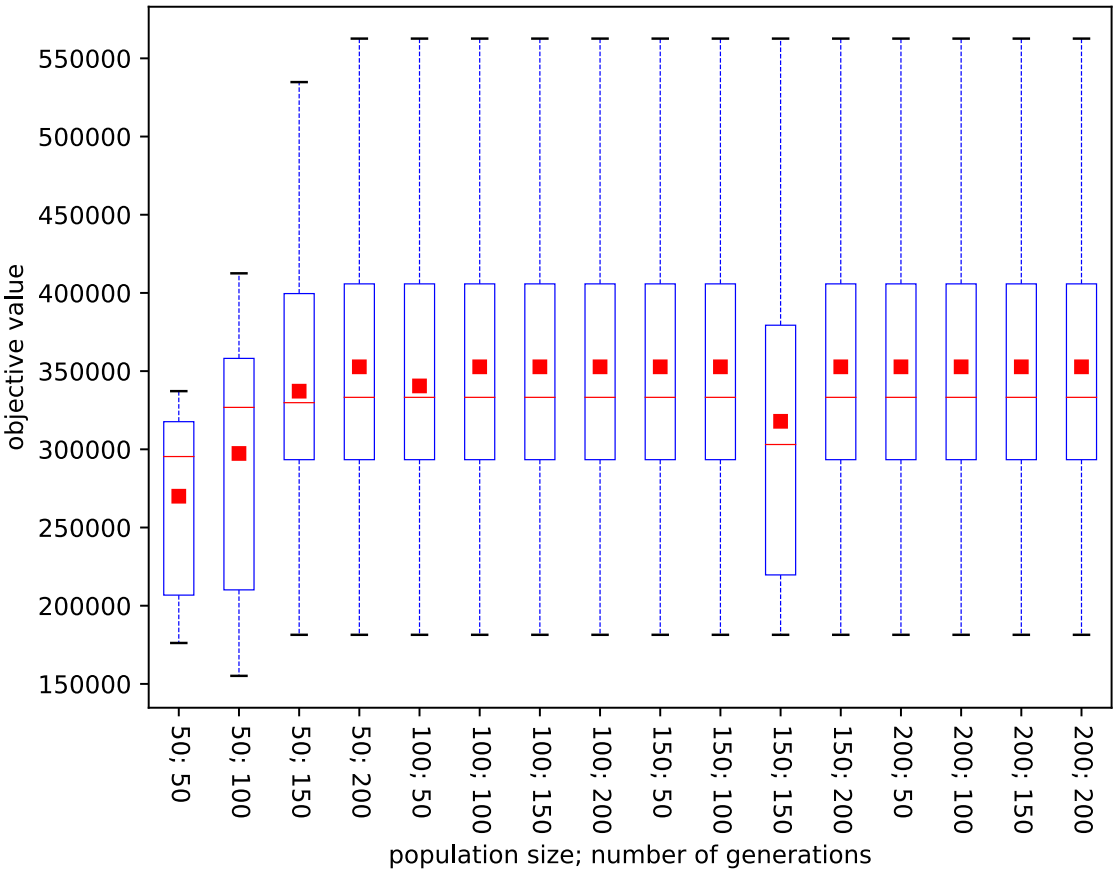

Figure D.4: **Calibration of EA with EC-based selection and replacement in *q-subnets***. Distribution of the best objective values from the 10 repetitions of the experiments on the **Rosenbrock** problem. Average values are depicted by red squares, median values by red dashes and variance information is given by the length of the boxes.

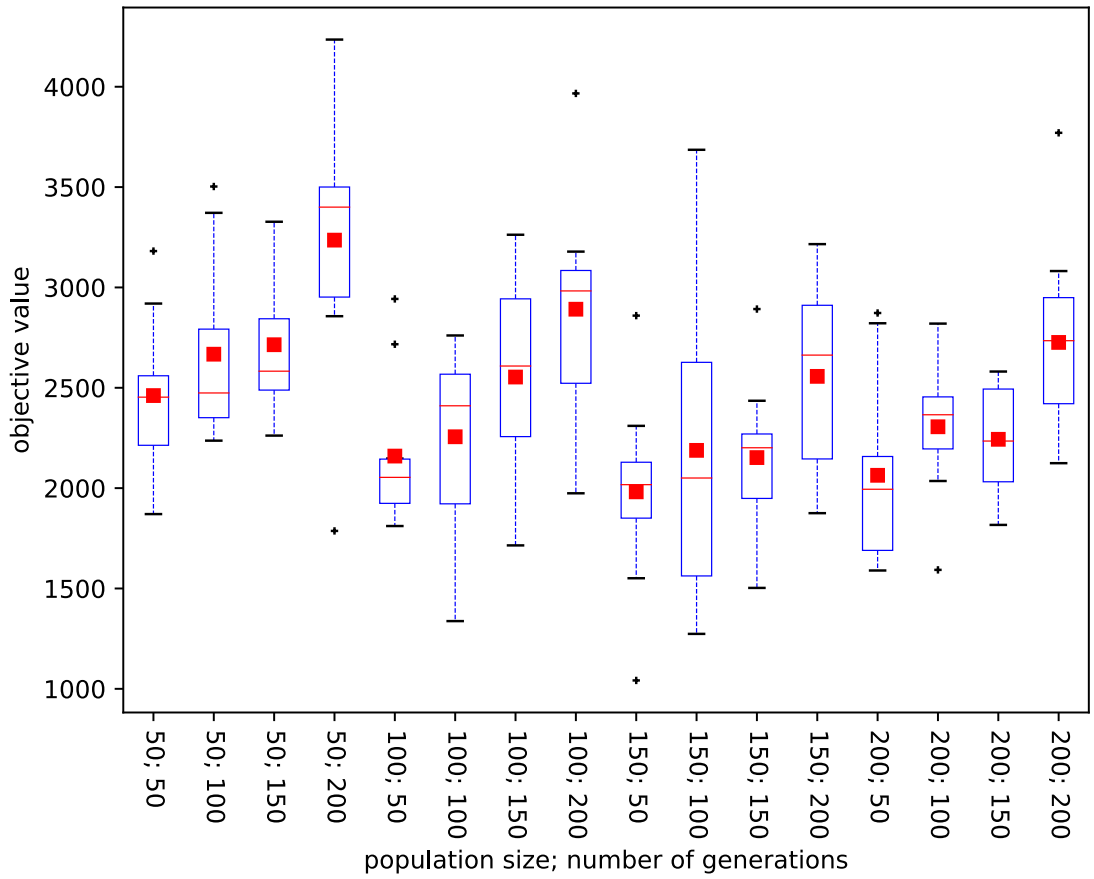

Figure D.5: **Calibration of EA with EC-based selection and replacement in *q-Pareto***. Distribution of the best objective values from the 10 repetitions of the experiments on the **Schwefel** problem. Average values are depicted by red squares, median values by red dashes and variance information is given by the length of the boxes.

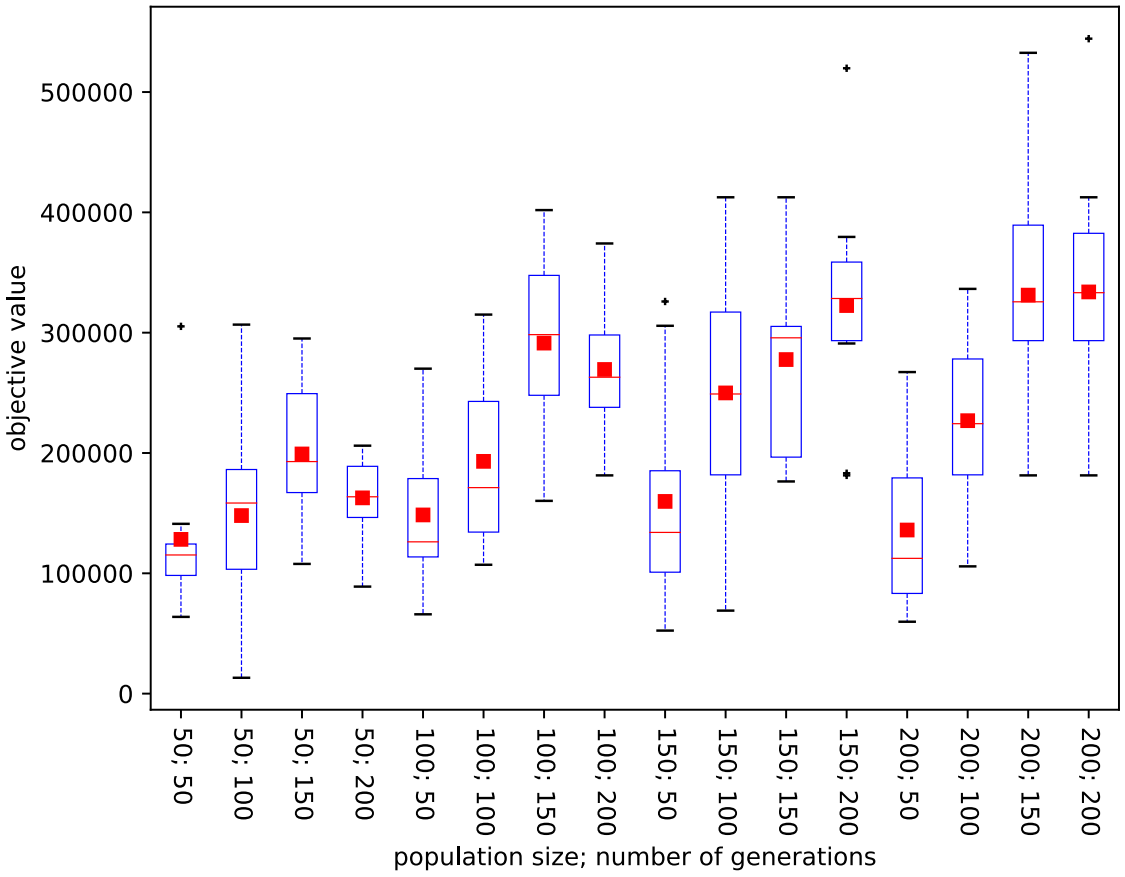

Figure D.6: **Calibration of EA with EC-based selection and replacement in *q-Pareto***. Distribution of the best objective values from the 10 repetitions of the experiments on the **Rosenbrock** problem. Average values are depicted by red squares, median values by red dashes and variance information is given by the length of the boxes.

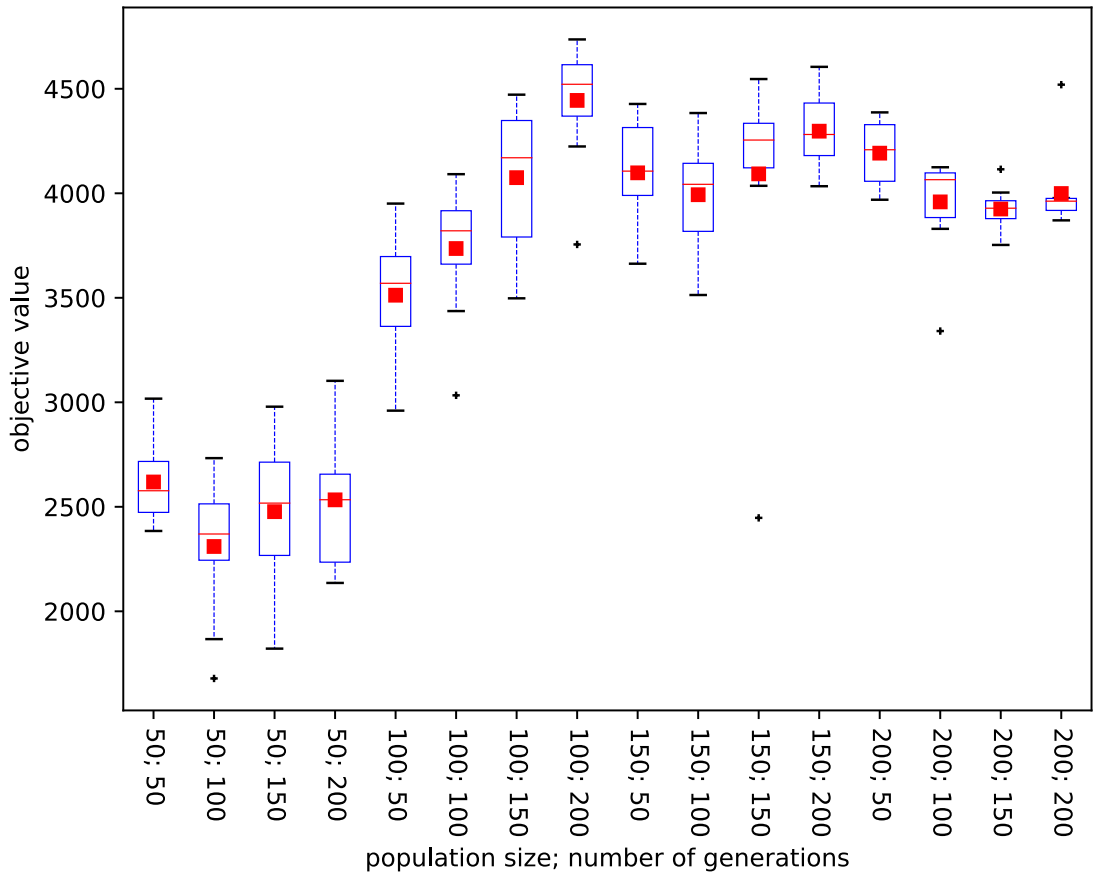

Figure D.7: **Calibration of EA with EC-based selection and replacement in q-post-HMC with GP_HMC**. Distribution of the best objective values from the 10 repetitions of the experiments on the **Schwefel** problem. Average values are depicted by red squares, median values by red dashes and variance information is given by the length of the boxes.

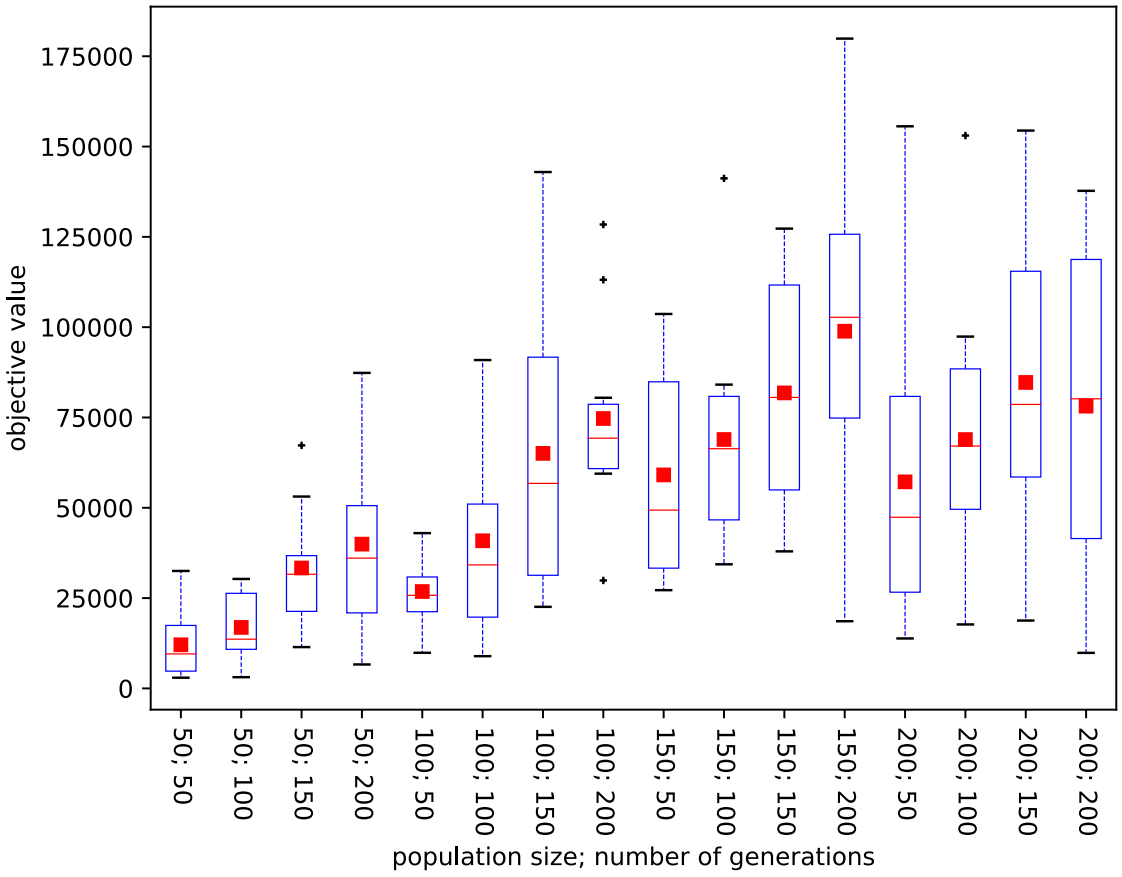

Figure D.8: **Calibration of EA with EC-based selection and replacement in q-post-HMC with GP_HMC**. Distribution of the best objective values from the 10 repetitions of the experiments on the **Rosenbrock** problem. Average values are depicted by red squares, median values by red dashes and variance information is given by the length of the boxes.

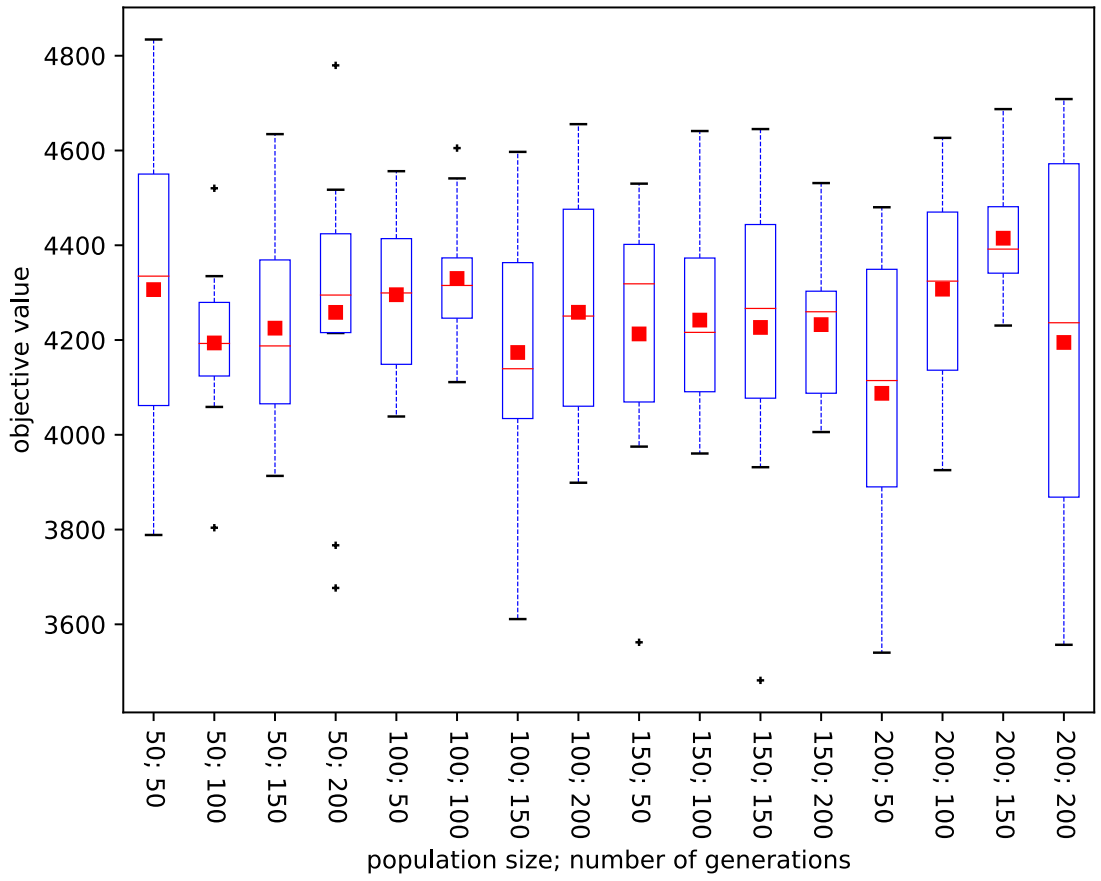

Figure D.9: **Calibration of EA with EC-based selection and replacement in q-post-HMC with BNN_HMC**. Distribution of the best objective values from the 10 repetitions of the experiments on the **Schwefel** problem. Average values are depicted by red squares, median values by red dashes and variance information is given by the length of the boxes.

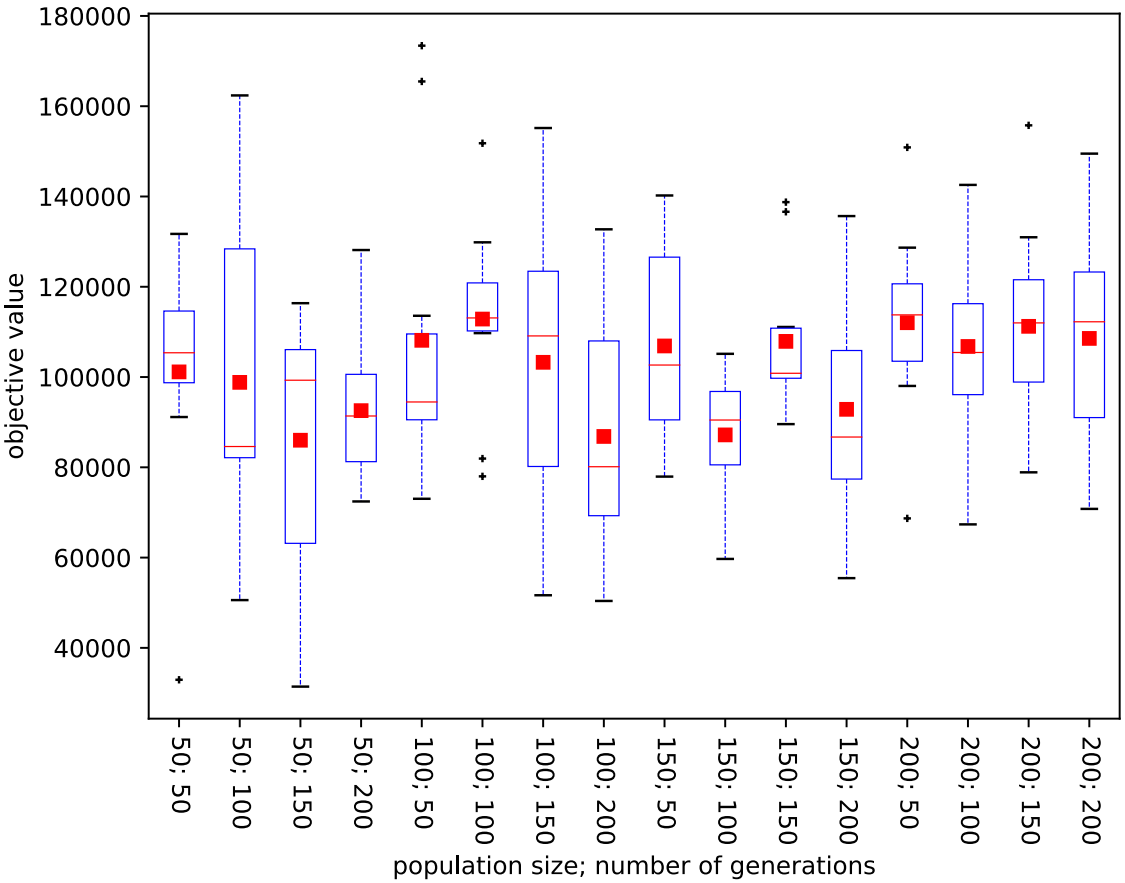

Figure D.10: **Calibration of EA with EC-based selection and replacement in q-post-HMC with BNN_HMC**. Distribution of the best objective values from the 10 repetitions of the experiments on the **Rosenbrock** problem. Average values are depicted by red squares, median values by red dashes and variance information is given by the length of the boxes.

Table D.1: **P-SDAs** applied to the **Schwefel** problem. Top (surrogate,EC) pairs per AP according to the average best objective value (10 independent repetitions). Ordering per cell according to ascending value from top to bottom. APs ordered according to decreasing performance from top to bottom.

Surrogate	EC	Average	Surrogate	EC	Average
q=18			q=72		
q-subnets					
BNN_MCD	***pov***	**1965.45**	BNN_MCD	*pov*	2483.44
BNN_MCD	*par-fd-cd*	3880.78	BNN_MCD	*par-fd-cd*	3666.19
q-Pareto					
BNN_MCD	*par-fs-hvc*	2015.06	BNN_MCD	*par-fs-cd*	2019.45
BNN_MCD	*par-fs-cd*	2192.29	BNN_MCD	*par-fs-hvc*	2105.19
BNN_MCD	*par-tian-fs*	2204.09	BNN_MCD	*par-tian-fs*	2279.5
BNN_MCD	*par-fd-cd*	3227.54	BNN_MCD	*par-fd-cd*	3426.36
BNN_MCD	*par-fd-hvc*	3292.93	GP_RBF	*par-tian-fs*	3459.56
q-post-HMC					
GP_HMC	*pov*	2345.8	GP_HMC	*pov*	3638.16
GP_HMC	*par-fd-cd*	3877.84	GP_HMC	*par-fd-cd*	3796.09
BNN_HMC	*par-fd-cd*	3989.17	BNN_HMC	*par-fd-cd*	3936.38
BNN_HMC	*pov*	4237.76	BNN_HMC	*pov*	4320.74
q-EGO *cl-mean*					
GP_RBF	*ada-wang-min*	2478.06	GP_RBF	*par-tian-fs*	3240.52
BNN_MCD	*lcb*	2723.47	GP_RBF	*dyn-df-excl*	3401.49
GP_RBF	*par-tian-fs*	2730.8	GP_RBF	*ada-wang-max*	3454.15
GP_RBF	*ei*	2913.49	GP_RBF	*ada-wang-min*	3458.33
GP_RBF	*pov*	2925.1	GP_RBF	*pov*	3459.91
q-EGO *sb*					
GP_RBF	*ada-wang-min*	2945.8	GP_RBF	*pov*	3491.5
BNN_MCD	*lcb*	2995.06	GP_RBF	*pi*	3528.64
GP_RBF	*lcb*	3067.79	GP_RBF	*lcb*	3546.29
BNN_MCD	*pov*	3119.08	GP_RBF	*dyn-df-75-excl*	3581.97
GP_RBF	*ei*	3177.17	GP_RBF	*dyn-df-excl*	3596.12

Table D.2: **P-SDAs** applied to the **Rastrigin** problem. Top (surrogate,EC) pairs per AP according to the average best objective value (10 independent repetitions). Ordering per cell according to ascending value from top to bottom. APs ordered according to decreasing performance from top to bottom.

Surrogate	EC	Average	Surrogate	EC	Average
q=18			q=72		
q-Pareto					
rKRG	***par-tian-fd***	**77.83**	GP_RBF	*par-tian-fs*	93.37
GP_RBF	*par-tian-fs*	99.01	rKRG	*par-tian-fd*	96.82
GP_RBF	*par-tian-fd*	101.8	iKRG	*par-tian-fd*	102.94
iKRG	*par-tian-fd*	103.49	rKRG	*par-tian-fs*	129.48
rKRG	*par-tian-fs*	123.78	GP_RBF	*par-tian-fd*	131.16
q-EGO *cl-mean*					
GP_RBF	*ada-wang-min*	90.59	GP_RBF	*par-tian-fs*	114.34
GP_RBF	*par-tian-fs*	92.92	GP_RBF	*ada-wang-min*	119.57
GP_RBF	*pov*	102.42	GP_RBF	*pov*	123.53
GP_RBF	*dyn-fd-excl*	103.78	GP_RBF	*dyn-fs-excl*	126.27
GP_RBF	*dyn-fs-excl*	111.25	GP_RBF	*par-tian-fd*	126.43
q-post-HMC					
GP_HMC	*pov*	121.85	GP_HMC	*pov*	119.87
GP_HMC	*par-fd-cd*	125.36	BNN_HMC	*pov*	164.32
BNN_HMC	*pov*	168.61	BNN_HMC	*par-fd-cd*	199.94
BNN_HMC	*par-fd-cd*	195.28	GP_HMC	*par-fd-cd*	200.92
q-EGO *sb*					
GP_RBF	*ada-wang-min*	112.33	GP_RBF	*ada-wang-min*	121.21
GP_RBF	*par-tian-fs*	118.45	GP_RBF	*par-tian-fs*	127.73
GP_RBF	*pi*	133.14	GP_RBF	*par-tian-fd*	138.64
rKRG	*par-tian-fd*	134.44	GP_RBF	*pi*	143.53
GP_RBF	*par-tian-fd*	139.36	rKRG	*par-tian-fd*	154.25
q-subnets					
BNN_MCD	*pov*	200.92	BNN_MCD	*pov*	200.92
BNN_MCD	*par-fd-cd*	200.92	BNN_MCD	*par-fd-cd*	200.92

Table D.3: **P-SDAs** applied to the **Rosenbrock** problem. Top (surrogate,EC) pairs per AP according to the average best objective value (10 independent repetitions). Ordering per cell according to ascending value from top to bottom. APs ordered according to decreasing performance from top to bottom.

Surrogate	EC	Average	Surrogate	EC	Average
	q=18			q=72	
		q-EGO *sb*			
GP_RBF	***ada-wang-min***	**472.02**	GP_RBF	*pov*	1409.49
GP_RBF	*pi*	1629.54	GP_RBF	*pi*	1563.03
GP_RBF	*par-tian-fs*	1826.74	GP_RBF	*par-tian-fs*	1749.29
GP_RBF	*dyn-fs-excl*	4119.73	GP_RBF	*ada-wang-min*	1798.29
GP_RBF	*dyn-sf-excl*	4338.11	GP_RBF	*dyn-fd-excl*	2079.98
		q-Pareto			
rKRG	*par-tian-fd*	572.57	GP_RBF	*par-tian-fs*	639.77
GP_RBF	*par-tian-fs*	713.99	GP_RBF	*par-tian-fd*	940.72
GP_RBF	*par-tian-fd*	1199.01	rKRG	*par-tian-fd*	6174.1
iKRG	*par-tian-fd*	6563.32	iKRG	*par-tian-fd*	29045.14
ANN_BLR	*par-tian-fd*	33617.6	GP_RBF	*par-fd-hvc*	54025.08
		q-EGO *cl-mean*			
GP_RBF	*ada-wang-min*	1109.17	GP_RBF	*par-tian-fs*	6579.41
GP_RBF	*par-tian-fs*	1515.04	GP_RBF	*pov*	6972.94
GP_RBF	*pov*	2276.48	GP_RBF	*dyn-fps-excl*	8067.16
GP_RBF	*pi*	2532.63	GP_RBF	*dyn-spf-excl*	8341.52
GP_RBF	*dyn-fs-excl*	2749.46	GP_RBF	*pi*	8386.29
		q-post-HMC			
GP_HMC	*pov*	18228.89	GP_HMC	*pov*	73350.55
GP_HMC	*par-fd-cd*	23781.63	BNN_HMC	*pov*	105167.53
BNN_HMC	*pov*	82980.38	GP_HMC	*par-fd-cd*	177484.34
BNN_HMC	*par-fd-cd*	220068.63	BNN_HMC	*par-fd-cd*	259459.82
		q-subnets			
BNN_MCD	*pov*	306678.75	BNN_MCD	*pov*	319621.31
BNN_MCD	*par-fd-cd*	352757.01	BNN_MCD	*par-fd-cd*	336275.42

Table D.4: **P-SDAs** applied to the **Covid-19 contact reduction** problem. Top (surrogate,EC) pairs per AP according to the average best objective value (10 independent repetitions). Ordering per cell according to ascending value from top to bottom. APs ordered according to decreasing performance from top to bottom.

Surrogate	EC	Average	Surrogate	EC	Average
q=18			q=72		
q-EGO *cl-mean*					
GP_RBF	***com-spf***	**7824.31**	GP_RBF	*com-spf*	12682.04
GP_RBF	*par-fs-hvc*	9225.48	GP_RBF	*dyn-sf-75-excl*	13082.08
GP_RBF	*par-fs-cd*	10117.98	GP_RBF	*par-fs-hvc*	13303.13
GP_RBF	*par-fd-hvc*	10168.57	GP_RBF	*stdev*	13366.71
GP_RBF	*com-dpf*	11067.32	GP_RBF	*dyn-spf-excl*	14583.25
q-post-HMC					
GP_HMC	*par-fd-cd*	8778.04	GP_HMC	*par-fd-cd*	8648.29
GP_HMC	*pov*	13260.55	GP_HMC	*pov*	46031.48
BNN_HMC	*par-fd-cd*	39999.3	BNN_HMC	*par-fd-cd*	68382.66
BNN_HMC	*pov*	179248.38	BNN_HMC	*pov*	179248.38
q-subnets					
BNN_MCD	*par-fd-cd*	36953.01	BNN_MCD	*par-fd-cd*	11531.22
BNN_MCD	*pov*	61471.69	BNN_MCD	*pov*	78781.72
q-Pareto					
GP_RBF	*par-fs-cd*	16717.64	GP_RBF	*par-fs-cd*	44576.87
GP_RBF	*par-fs-hvc*	17495.94	GP_RBF	*par-fs-hvc*	60930.42
GP_RBF	*par-fd-cd*	22935.7	GP_RBF	*par-fd-hvc*	65451.73
GP_RBF	*par-fd-hvc*	32776.52	GP_RBF	*par-fd-cd*	71554.54
BNN_MCD	*par-fs-cd*	57509.38	BNN_MCD	*par-fd-cd*	121757.66

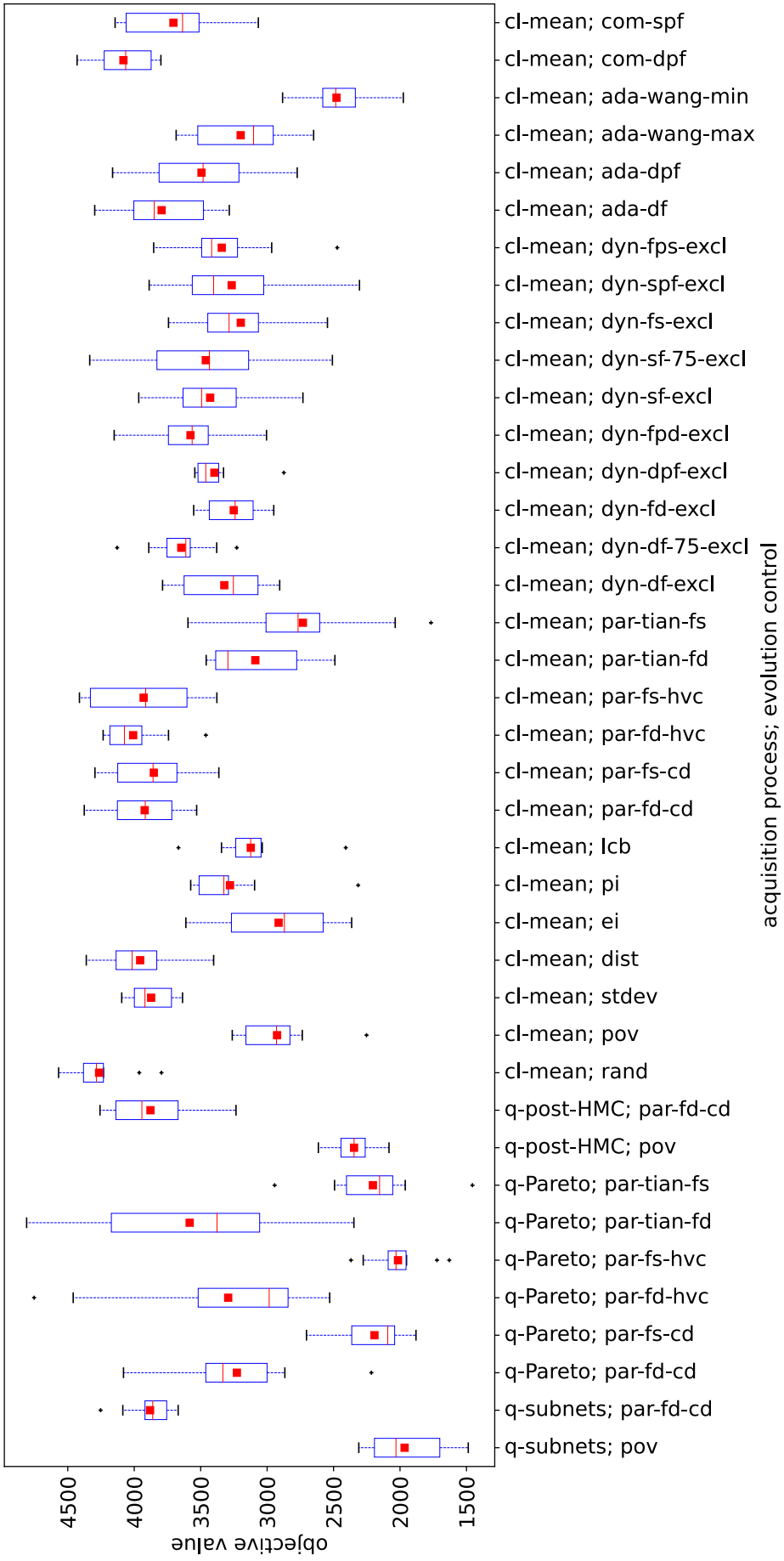

Figure D.11: **Some P-SDAs** on the **Schwefel** problem. The BNN_MCD surrogate is used in q-Pareto, GP_RBF in *cl-mean* and GP_HMC in q-post-HMC. Distribution of the best objective values from the 10 repetitions of the experiment. Averaged values are depicted by red squares, median values by red dashes and variance information is given by the length of the boxes.

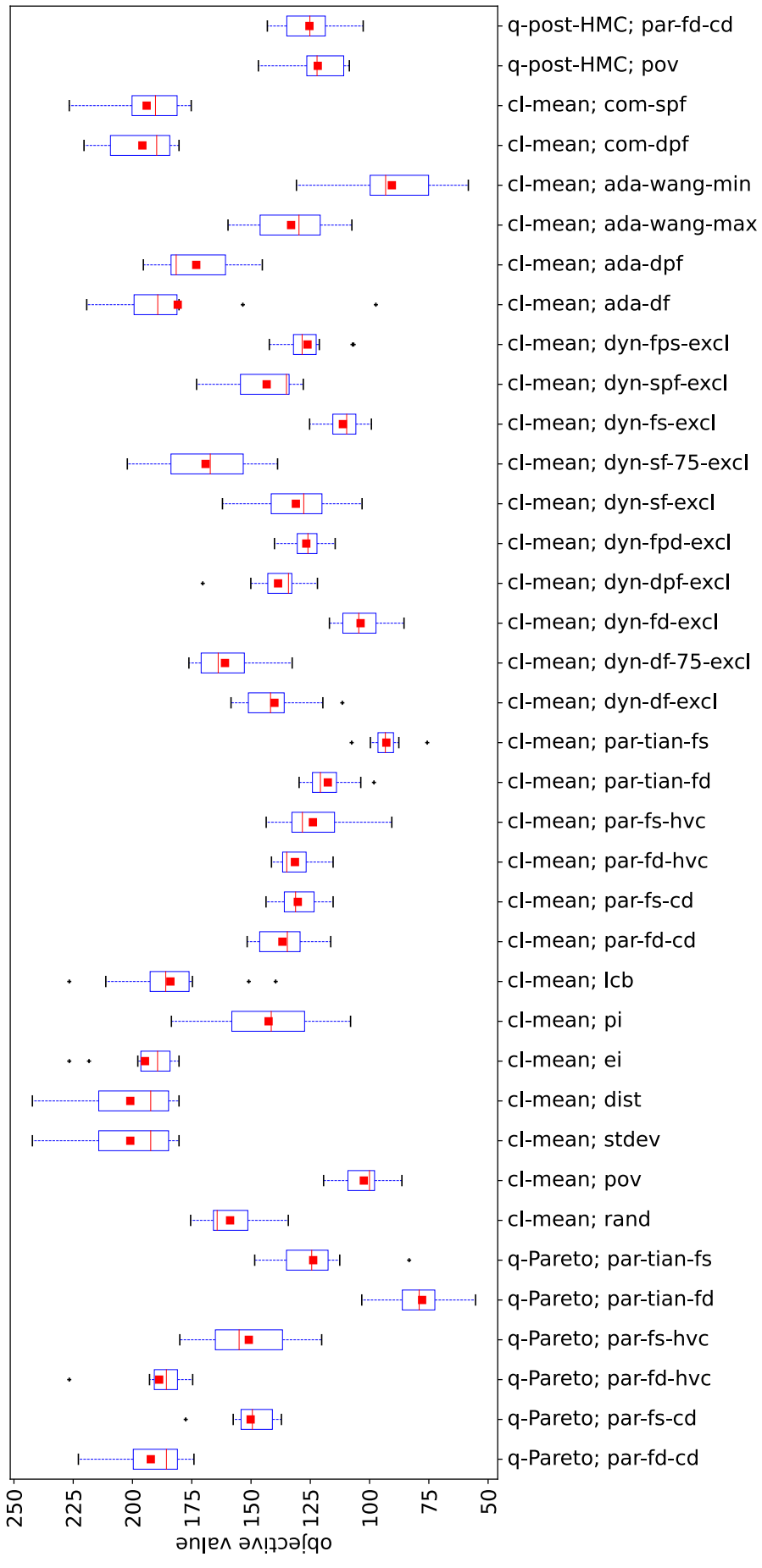

Figure D.12: **Some P-SDAs** on the **Rastrigin** problem. The rKRG surrogate is used in q-Pareto, GP_RBF in *cl-mean* and GP_HMC in q-post-HMC. Distribution of the best objective values from the 10 repetitions of the experiment. Averaged values are depicted by red squares, median values by red dashes and variance information is given by the length of the boxes.

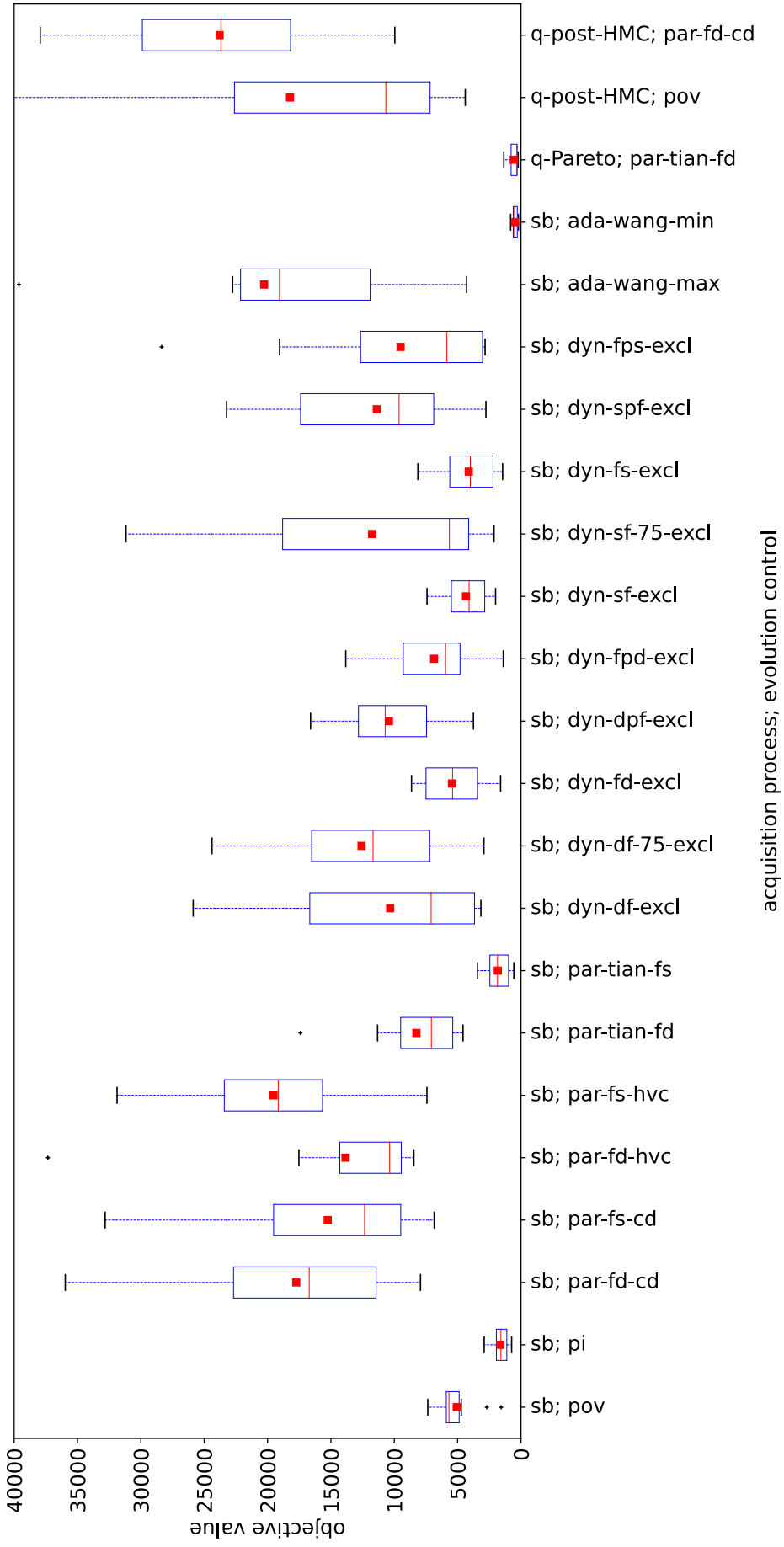

Figure D.13: **Some P-SDAs** on the **Rosenbrock** problem. The GP_RBF surrogate is used in *sb*, rKRG in q-Pareto and GP_HMC in q-post-HMC. Distribution of the best objective values from the 10 repetitions of the experiment. Averaged values are depicted by red squares, median values by red dashes and variance information is given by the length of the boxes.

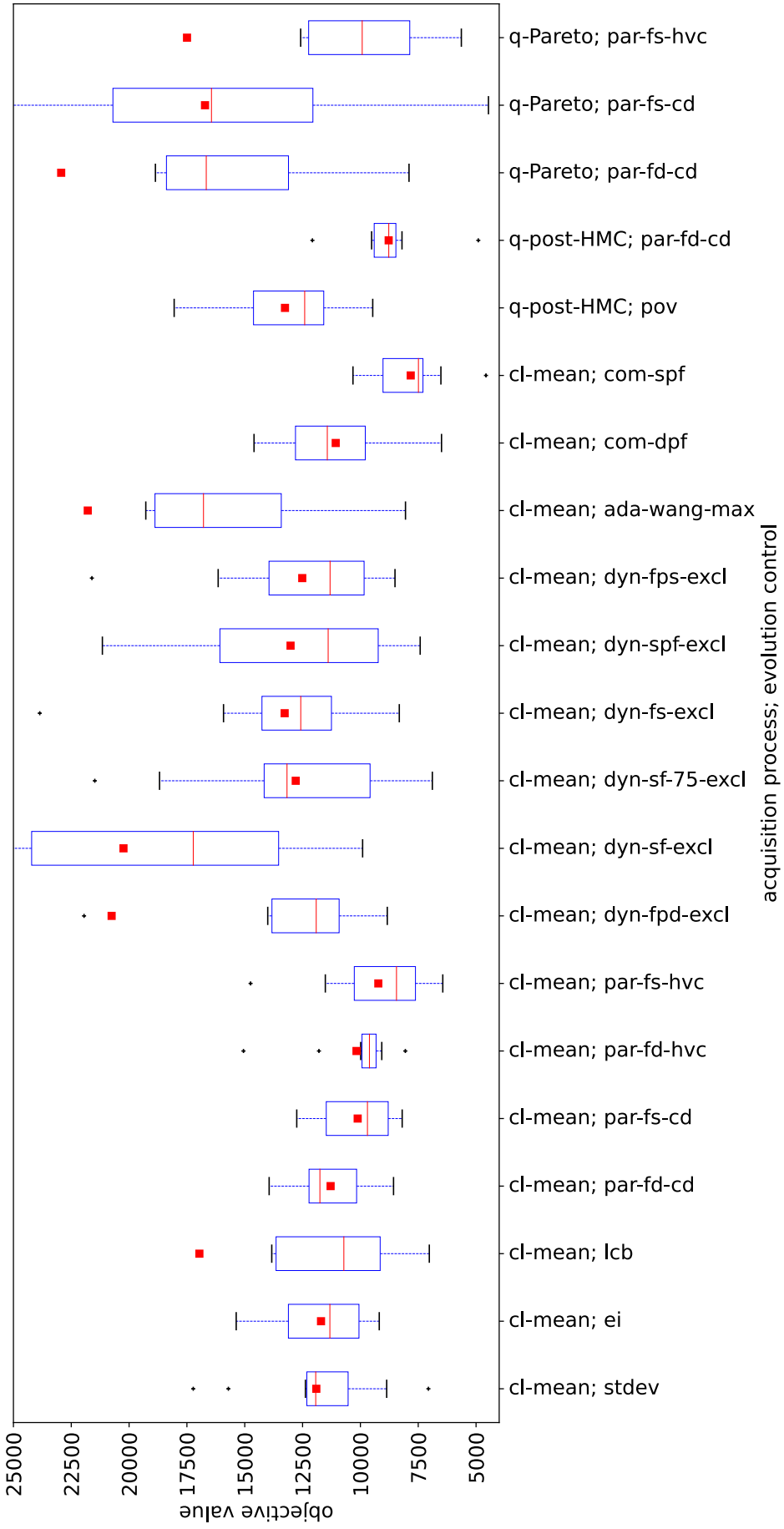

Figure D.14: **Some P-SDAs** on the **Covid-19 contact reduction** problem. The GP_RBF surrogate is used in *cl-mean* and q-Pareto and GP_HMC is used in q-post-HMC. Distribution of the best objective values from the 10 repetitions of the experiment. Averaged values are depicted by red squares, median values by red dashes and variance information is given by the length of the boxes.

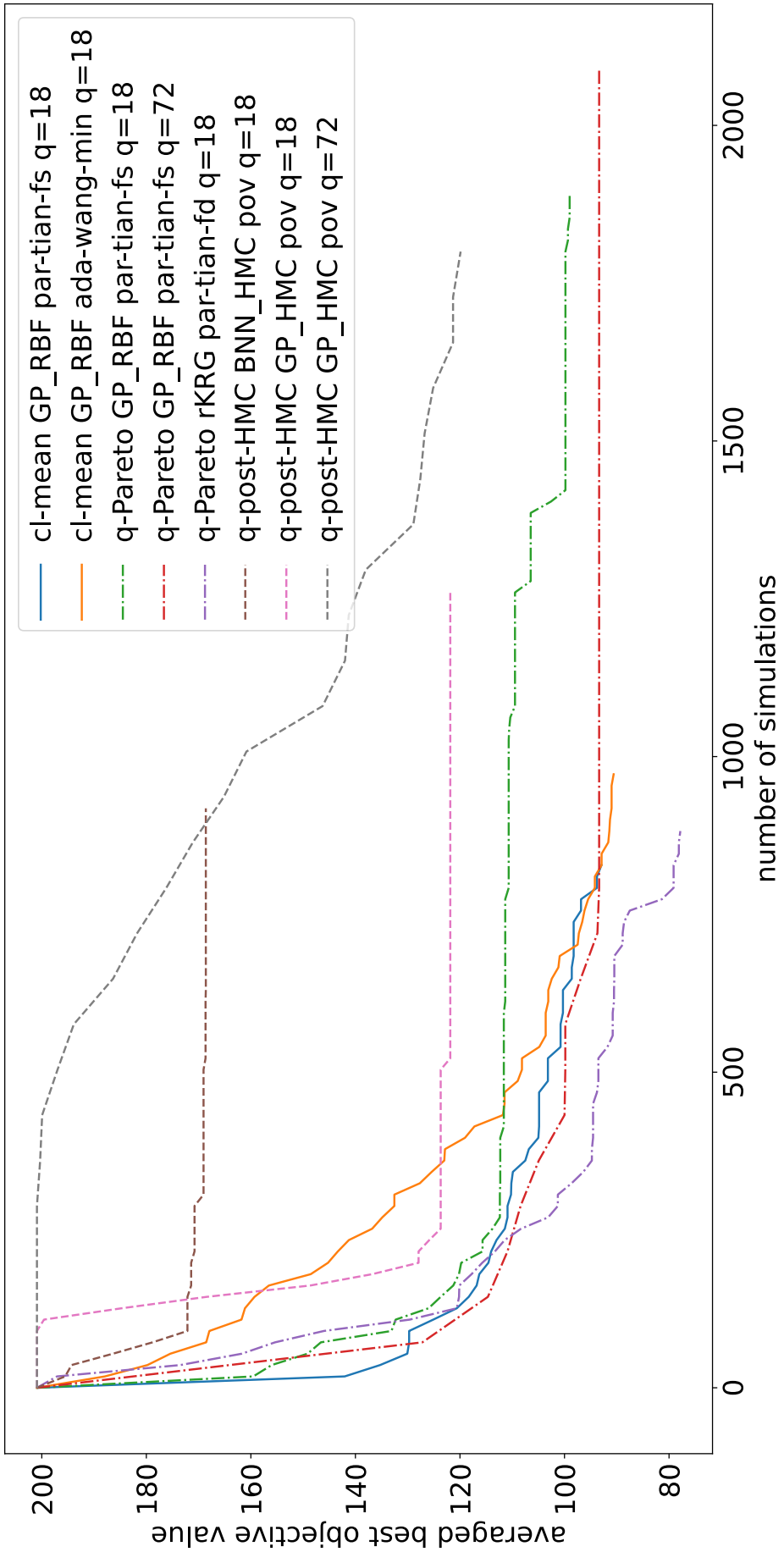

Figure D.15: **Best P-SDAs** applied to the **Rastrigin** problem. Convergence profile in terms of best objective values averaged over the 10 repetitions of the experiment.

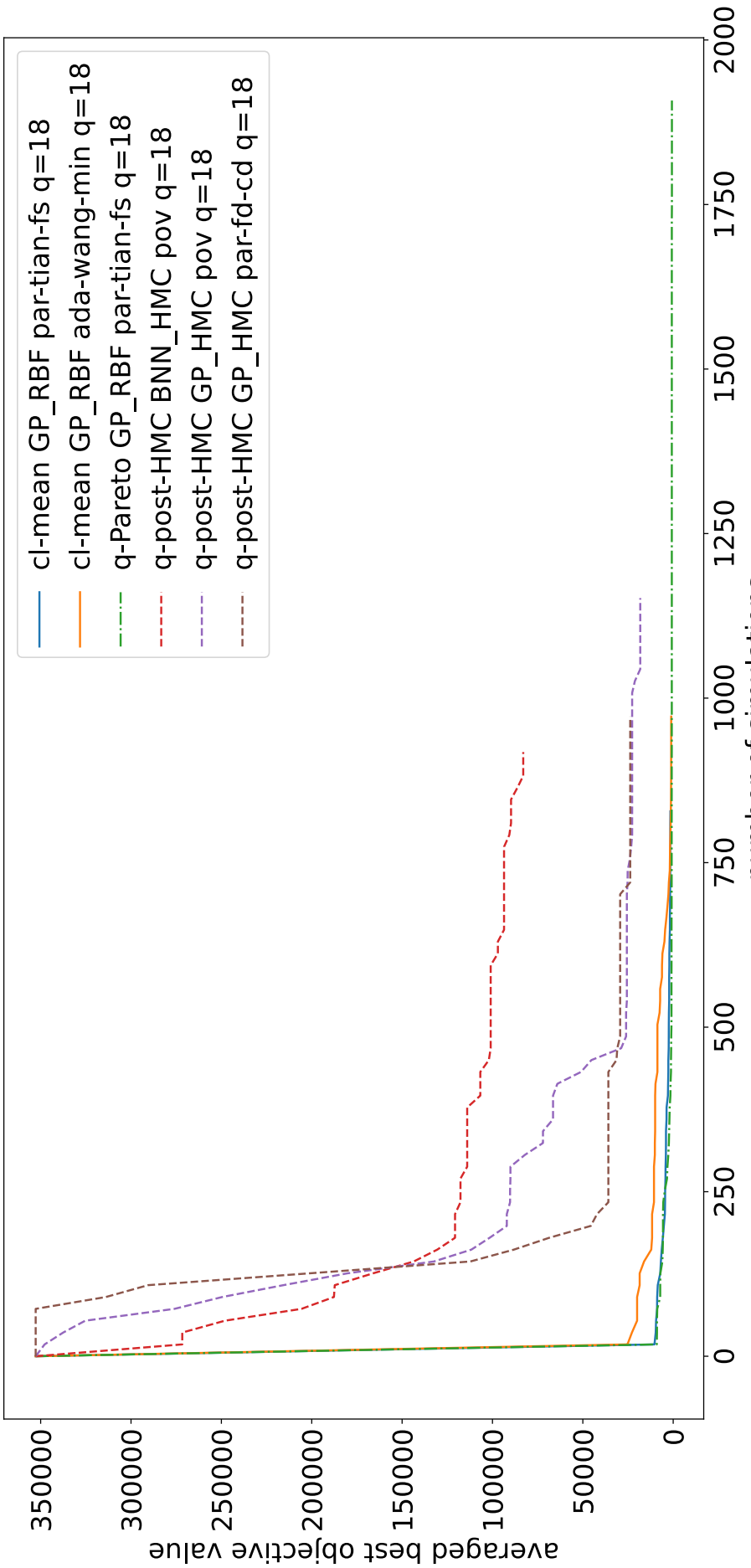

Figure D.16: **Best P-SDAs** applied to the **Rosenbrock** problem. Convergence profile in terms of best objective values averaged over the 10 repetitions of the experiment.

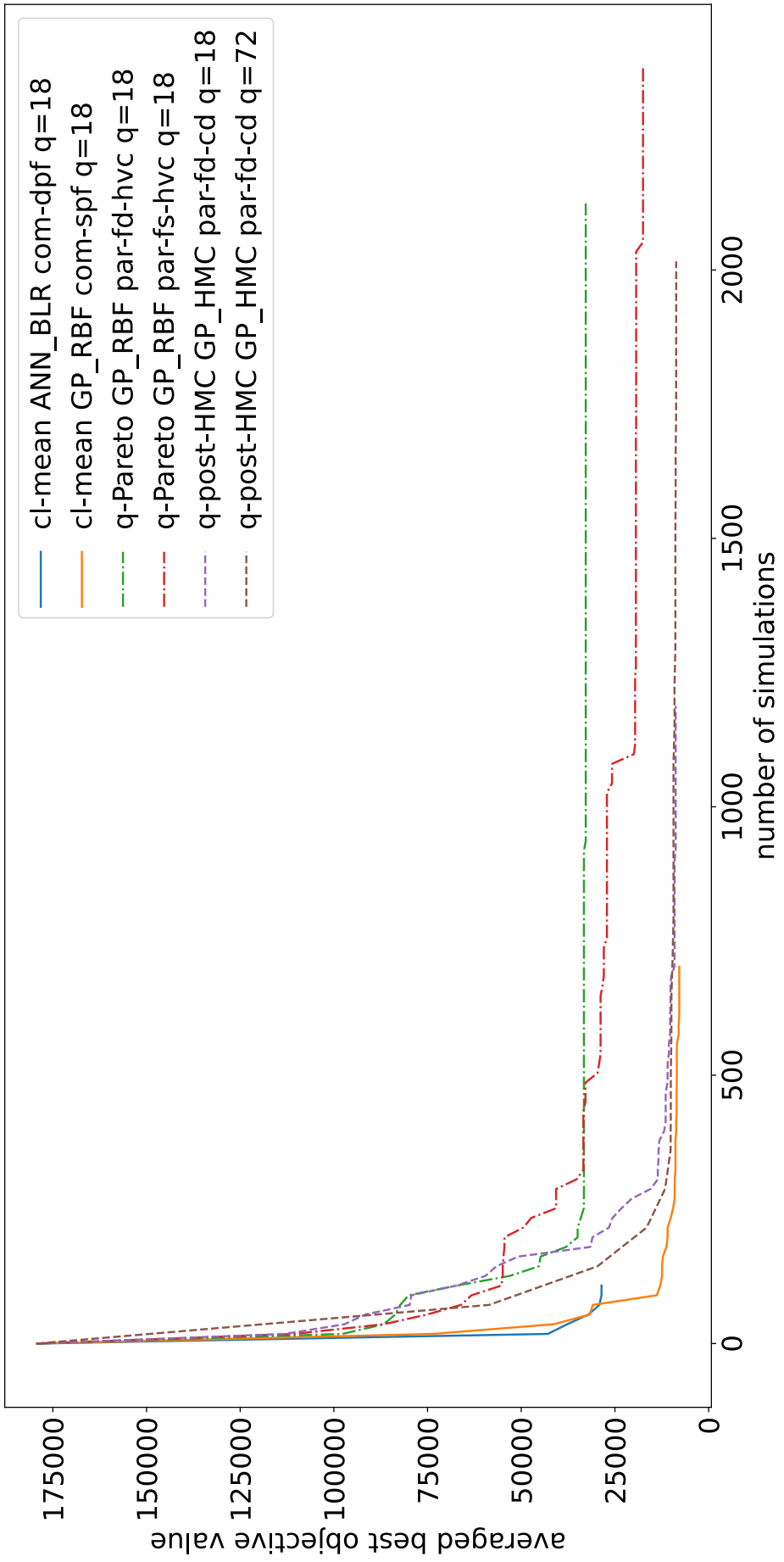

Figure D.17: **Best P-SDAs** applied to the **Covid-19 contact reduction** problem. Convergence profile in terms of best objective values averaged over the 10 repetitions of the experiment.

Table D.5: **P-SDAs** applied to the **Schwefel** problem. The surrogates are **trained on the complete database of simulated candidates**. Top (surrogate,EC) pairs per AP according to the average best objective value (10 independent repetitions). Ordering per cell according to ascending value from top to bottom. APs ordered according to decreasing performance from top to bottom.

Surrogate	EC	Average
q=18		
q-post-HMC		
GP_HMC	***pov***	**2431.86**
BNN_HMC	*par-fd-cd*	4018.91
GP_HMC	*par-fd-cd*	4123.49
BNN_HMC	*pov*	4290.25
q-EGO *cl-mean*		
GP_RBF	*dyn-df-excl*	2435.37
GP_RBF	*ada-dpf*	2821.83
GP_RBF	*ada-df*	2886.64
GP_RBF	*par-tian-fd*	3014.26
GP_RBF	*dyn-df-75-excl*	3130.79
q-Pareto		
ANN_BLR	*par-fd-cd*	3521.33
GP_RBF	*par-fd-cd*	3632.55
ANN_BLR	*par-fd-hvc*	3794.06
GP_RBF	*par-fd-hvc*	3872.34
GP_RBF	*par-tian-fd*	3917.57

Table D.6: **P-SDAs** applied to the **Rastrigin** problem. The surrogates are **trained on the complete database of simulated candidates**. Top (surrogate,EC) pairs per AP according to the average best objective value (10 independent repetitions). Ordering per cell according to ascending value from top to bottom. APs ordered according to decreasing performance from top to bottom.

Surrogate	EC	Average
q=18		
q-EGO *cl-mean*		
GP_RBF	***lcb***	**84.66**
GP_RBF	*pov*	90.82
GP_RBF	*ada-wang-min*	94.64
GP_RBF	*dyn-fd-excl*	97.24
GP_RBF	*dyn-fs-excl*	97.57
q-post-HMC		
GP_HMC	*pov*	98.4
GP_HMC	*par-fd-cd*	100.46
BNN_HMC	*pov*	169.02
BNN_HMC	*par-fd-cd*	194.4
q-Pareto		
GP_RBF	*par-tian-fs*	100.1
GP_RBF	*par-tian-fd*	126.49
ANN_BLR	*par-tian-fd*	138.86
GP_RBF	*par-fd-cd*	143.25
GP_RBF	*par-fs-cd*	155.13

Table D.7: **P-SDAs** applied to the **Rosenbrock** problem. The surrogates are **trained on the complete database of simulated candidates**. Top (surrogate,EC) pairs per AP according to the average best objective value (10 independent repetitions). Ordering per cell according to ascending value from top to bottom. APs ordered according to decreasing performance from top to bottom.

Surrogate	EC	Average
	q=18	
	q-EGO *cl-mean*	
GP_RBF	***ada-df***	**962.09**
GP_RBF	*dyn-df-75-excl*	1027.11
GP_RBF	*dyn-df-excl*	1146.64
GP_RBF	*ada-dpf*	1202.51
GP_RBF	*dyn-sf-excl*	2310.55
	q-Pareto	
GP_RBF	*par-tian-fs*	2779.66
GP_RBF	*par-tian-fd*	8192.03
GP_RBF	*par-fd-hvc*	12603.82
GP_RBF	*par-fd-cd*	17783.92
GP_RBF	*par-fs-hvc*	18340.68
	q-post-HMC	
GP_HMC	*par-fd-cd*	8183.66
GP_HMC	*pov*	36484.01
BNN_HMC	*pov*	105077.64
BNN_HMC	*par-fd-cd*	234481.45

Table D.8: **P-SDAs** applied to the **Covid-19 contact reduction** problem. The surrogates are **trained on the complete database of simulated candidates**. Top (surrogate,EC) pairs per AP according to the average best objective value (10 independent repetitions). Ordering per cell according to ascending value from top to bottom. APs ordered according to decreasing performance from top to bottom.

Surrogate	EC	Average
	q=18	
	q-Pareto	
GP_RBF	***par-fs-hvc***	**8298.09**
GP_RBF	*par-fs-cd*	9294.49
GP_RBF	*par-fd-cd*	11267.75
GP_RBF	*par-fd-hvc*	12026.66
GP_RBF	*par-tian-fd*	73839.82
	q-EGO *cl-mean*	
GP_RBF	*com-spf*	8897.23
GP_RBF	*par-fs-hvc*	10459.41
GP_RBF	*par-fs-cd*	10997.18
GP_RBF	*stdev*	11671.04
GP_RBF	*dyn-sf-75-excl*	12519.88
	q-post-HMC	
GP_HMC	*par-fd-cd*	13258.49
BNN_HMC	*par-fd-cd*	50113.88
GP_HMC	*pov*	80087.67
BNN_HMC	*pov*	179248.38

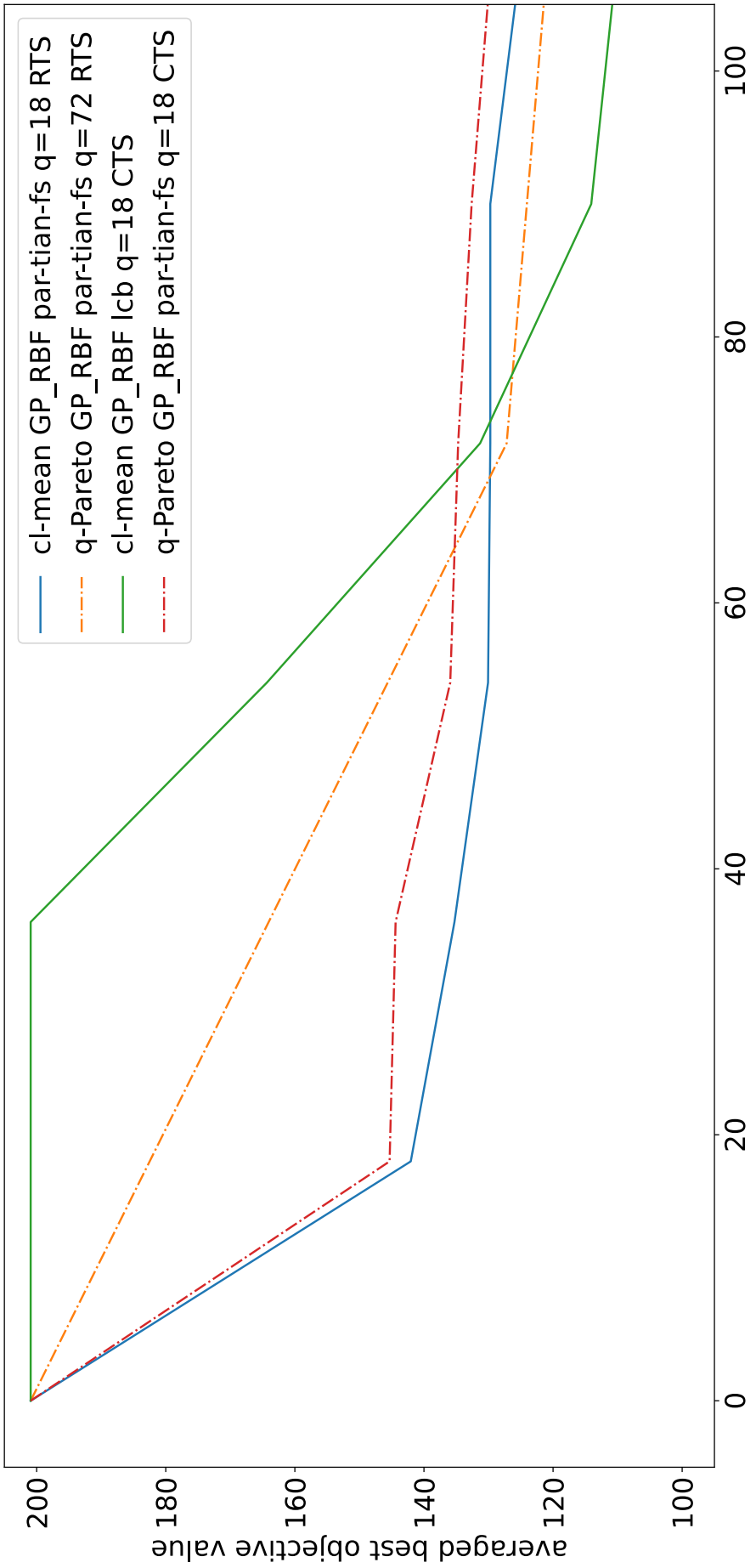

Figure D.18: **Best P-SDAs** applied to the **Rastrigin** problem. Convergence profile in terms of best objective values averaged over the 10 repetitions of the experiment. RTS: reduced training set. CTS: complete training set.

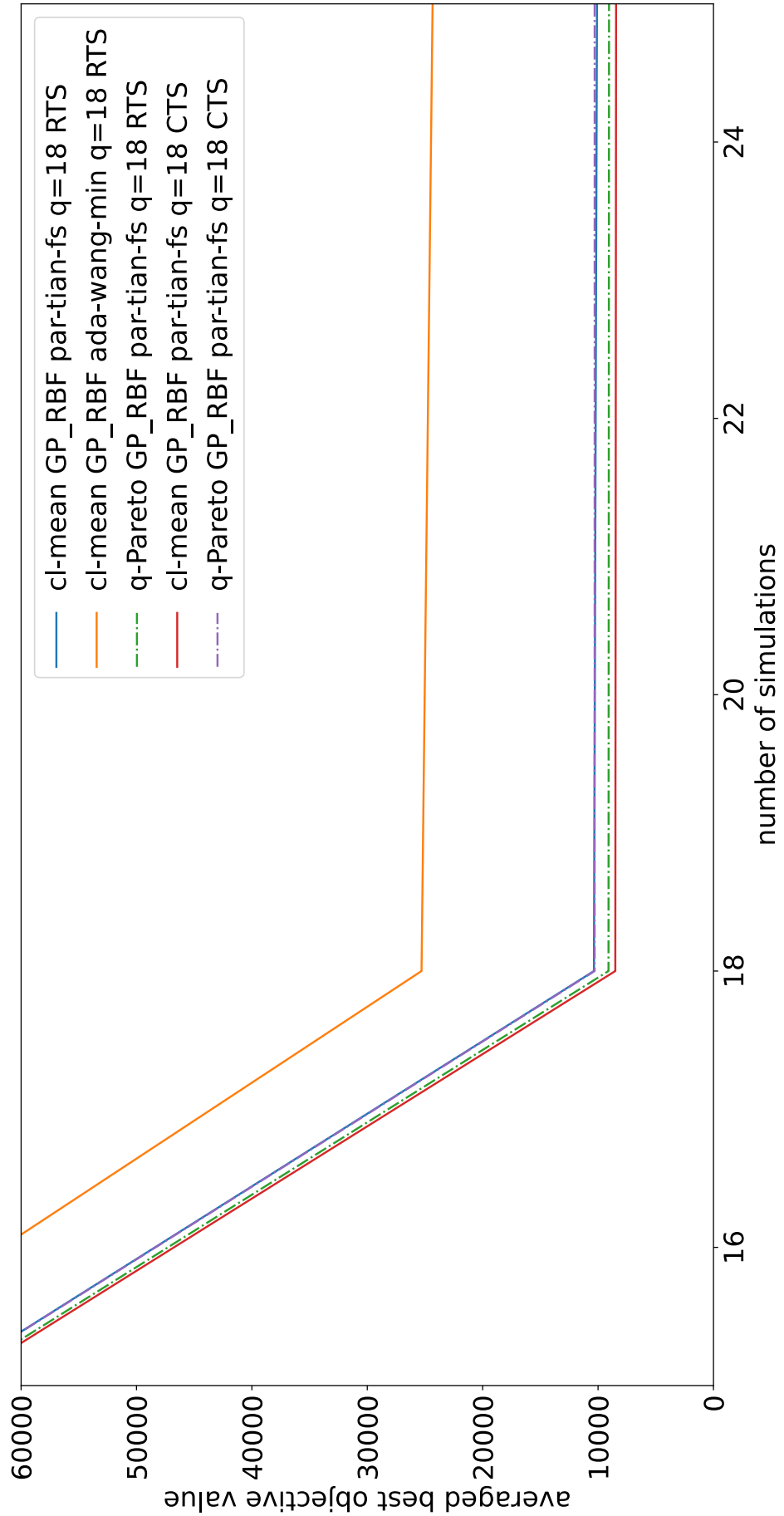

Figure D.19: **Best P-SDAs** applied to the **Rosenbrock** problem. Convergence profile in terms of best objective values averaged over the 10 repetitions of the experiment. RTS: reduced training set. CTS: complete training set.

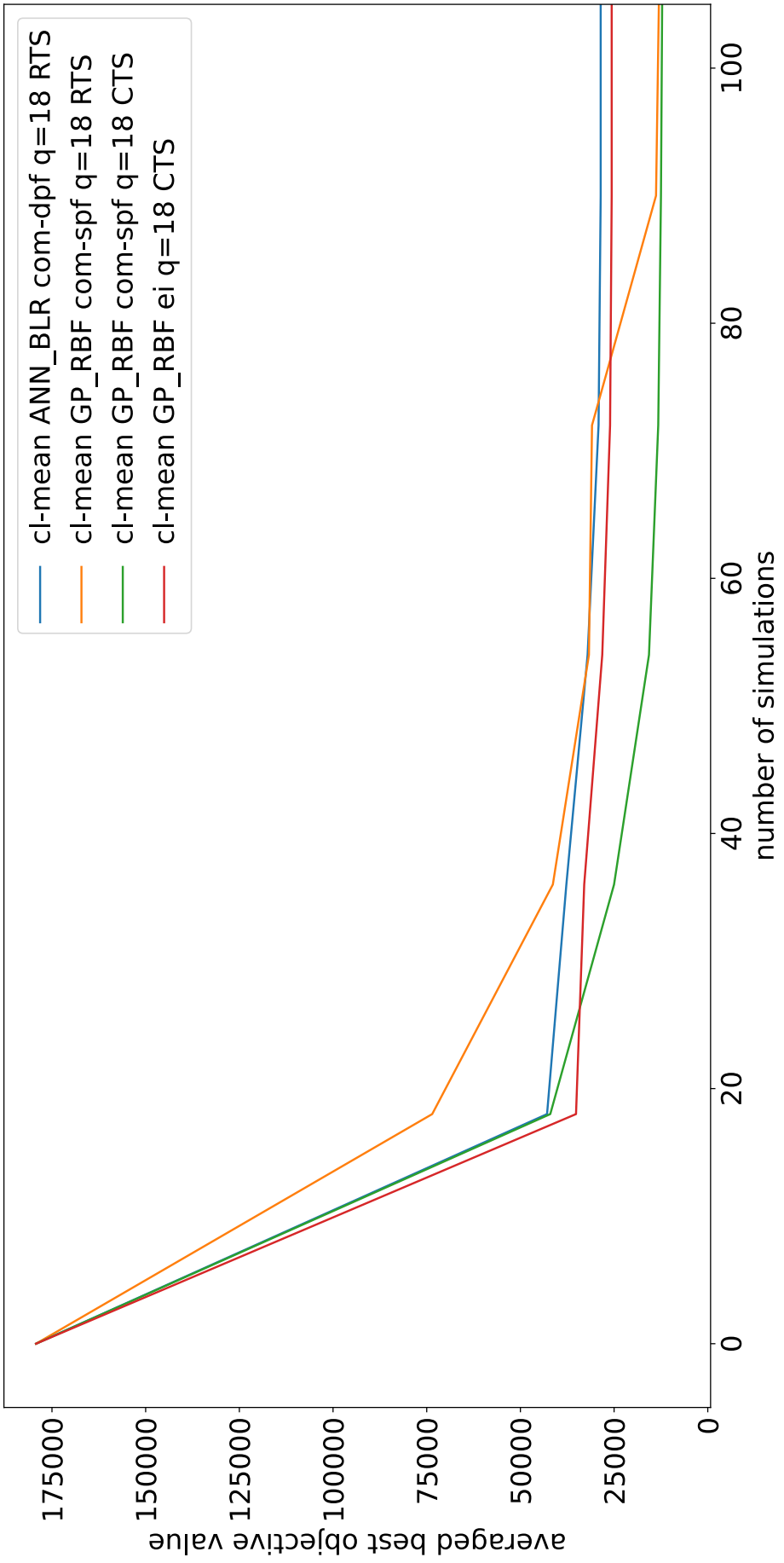

Figure D.20: **Best P-SDAs** applied to the **Covid-19** problem. Convergence profile in terms of best objective values averaged over the 10 repetitions of the experiment. RTS: reduced training set. CTS: complete training set.

Appendix E

Parallel Hybrid methods

Algorithm 21 Framework of SMBO+EA from [Reh+18]

Input

simulator: real objective function
GP_RBF: surrogate model
budget: computational budget for the search
EI: first infill criterion
pov: second infill criterion
$n_{pop1} = 100$: population size for AP1 and AP2
$n_{gen} = 50$: number of generations for AP1 and AP2
$n_{pop2} = 72$: population size for AP3
$q = 16$: number of candidates to simulate per cycle for AP3

1: $database \leftarrow$ LHS+parallel_simulations($simulator$, n_{pop2})
2: $surrogate \leftarrow$ training($database$)
3: $\mathcal{P} \leftarrow database$ ▷ initial population
4: **while** $budget \neq 0$ **do**
5: **BEGIN parallel section**
6: **IN 1 core:**
7: $\boldsymbol{x}^{(1)} \leftarrow$ 1-point_Acquisition_Process(EI, $surrogate$, n_{pop1}, n_{gen}) ▷ Algorithm 5
8: $y^{(1)} \leftarrow simulator(\boldsymbol{x}^{(1)})$
9: **IN 1 core:**
10: $\boldsymbol{x}^{(2)} \leftarrow$ 1-point_Acquisition_Process(pov, $surrogate$, n_{pop1}, n_{gen}) ▷ Algorithm 5
11: $y^{(2)} \leftarrow simulator(\boldsymbol{x}^{(2)})$
12: **IN 16 cores:**
13: $\mathcal{P}_p \leftarrow$ selection($\mathcal{P}$, q) ▷ population of parents
14: $\mathcal{B}_{sim} \leftarrow$ reproduction($\mathcal{P}_p$, q) ▷ population of children
15: parallel_simulations($simulator$, $\mathcal{B}_{sim}$)
16: **END parallel section**
17: $\mathcal{B}_{sim} \leftarrow \mathcal{B}_{sim} \cup (\boldsymbol{x}^{(1)}, y^{(1)}) \cup (\boldsymbol{x}^{(2)}, y^{(2)})$
18: $database \leftarrow database \cup \mathcal{B}_{sim}$
19: $surrogate \leftarrow$ training($database$)
20: $\mathcal{P} \leftarrow$ replacement($\mathcal{P}$, $\mathcal{B}_{sim}$, n_{pop2})
21: $budget \leftarrow$ get_remaining_budget($budget$, elapsed_time)
22: **end while**
23: ($\boldsymbol{x}_{min}$, y_{min}) $\leftarrow$ get_best_cost($database$)
24: **return** $\boldsymbol{x}_{min}$, y_{min}

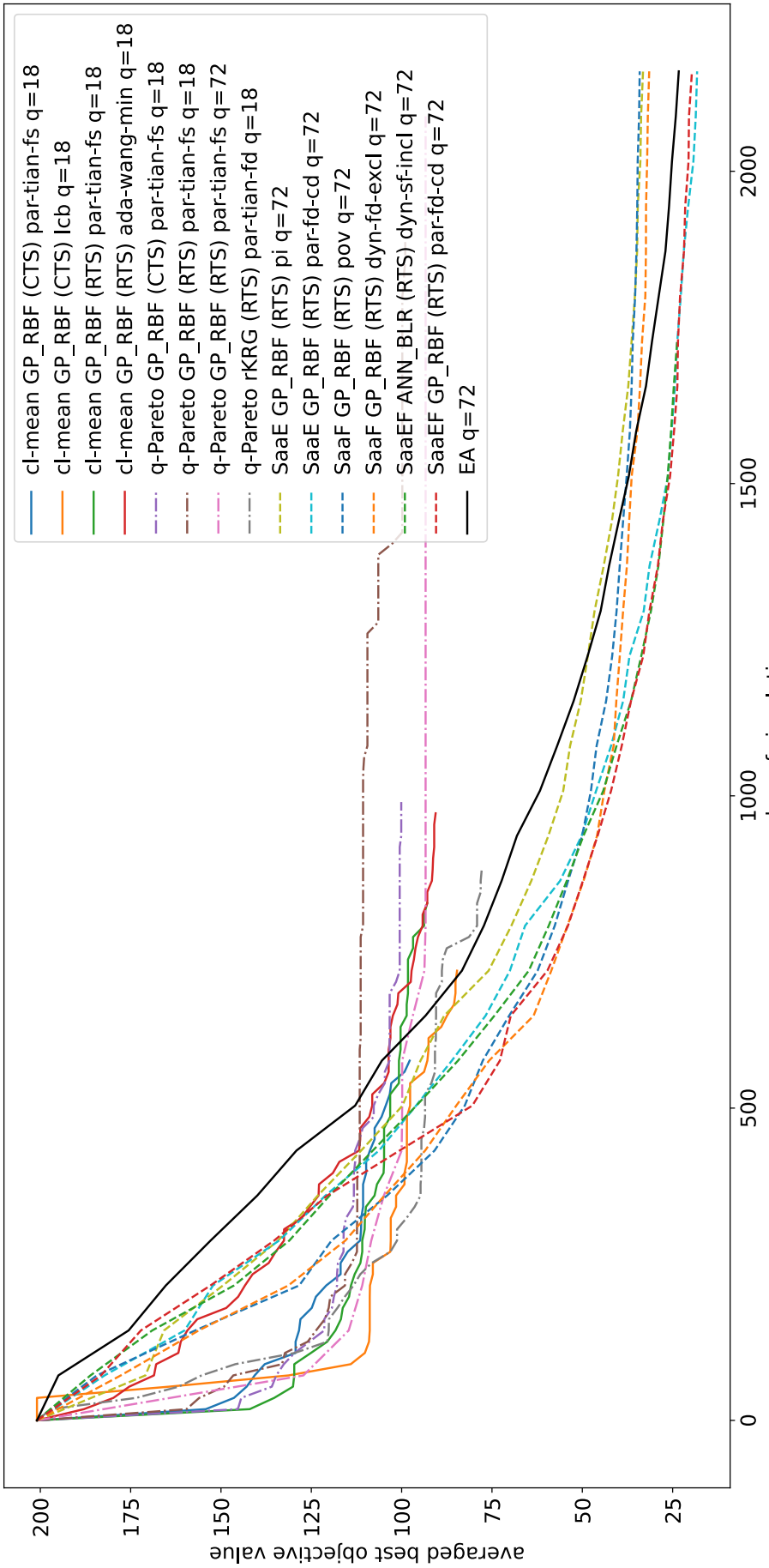

Figure E.1: **P-SAEAs *versus* P-SDAs** application to the **Rastrigin** problem. Convergence profile in terms of best objective values averaged over the 10 repetitions of the experiment.

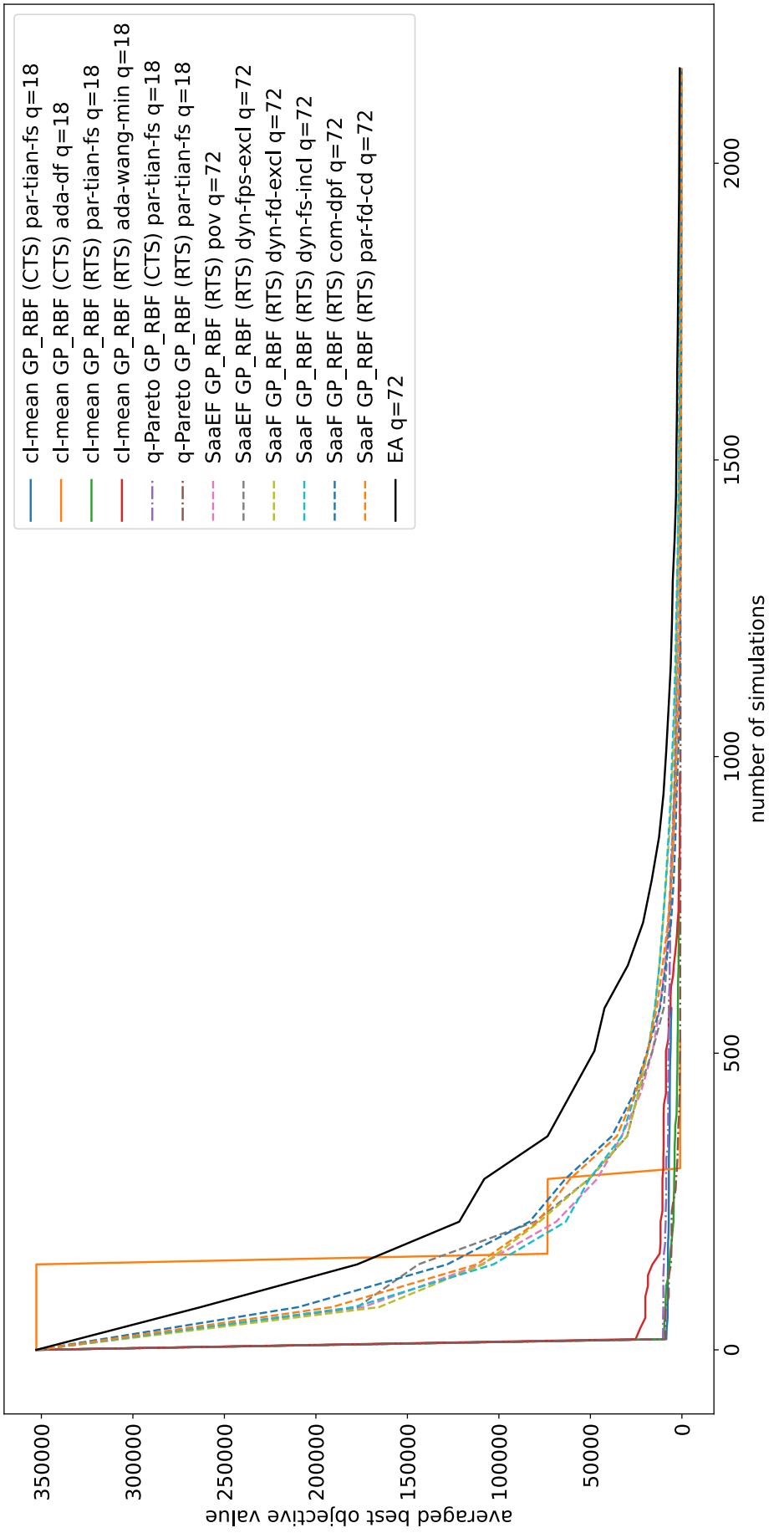

Figure E.2: **P-SAEAs *versus* P-SDAs** application to the **Rosenbrock** problem. Convergence profile in terms of best objective values averaged over the 10 repetitions of the experiment.

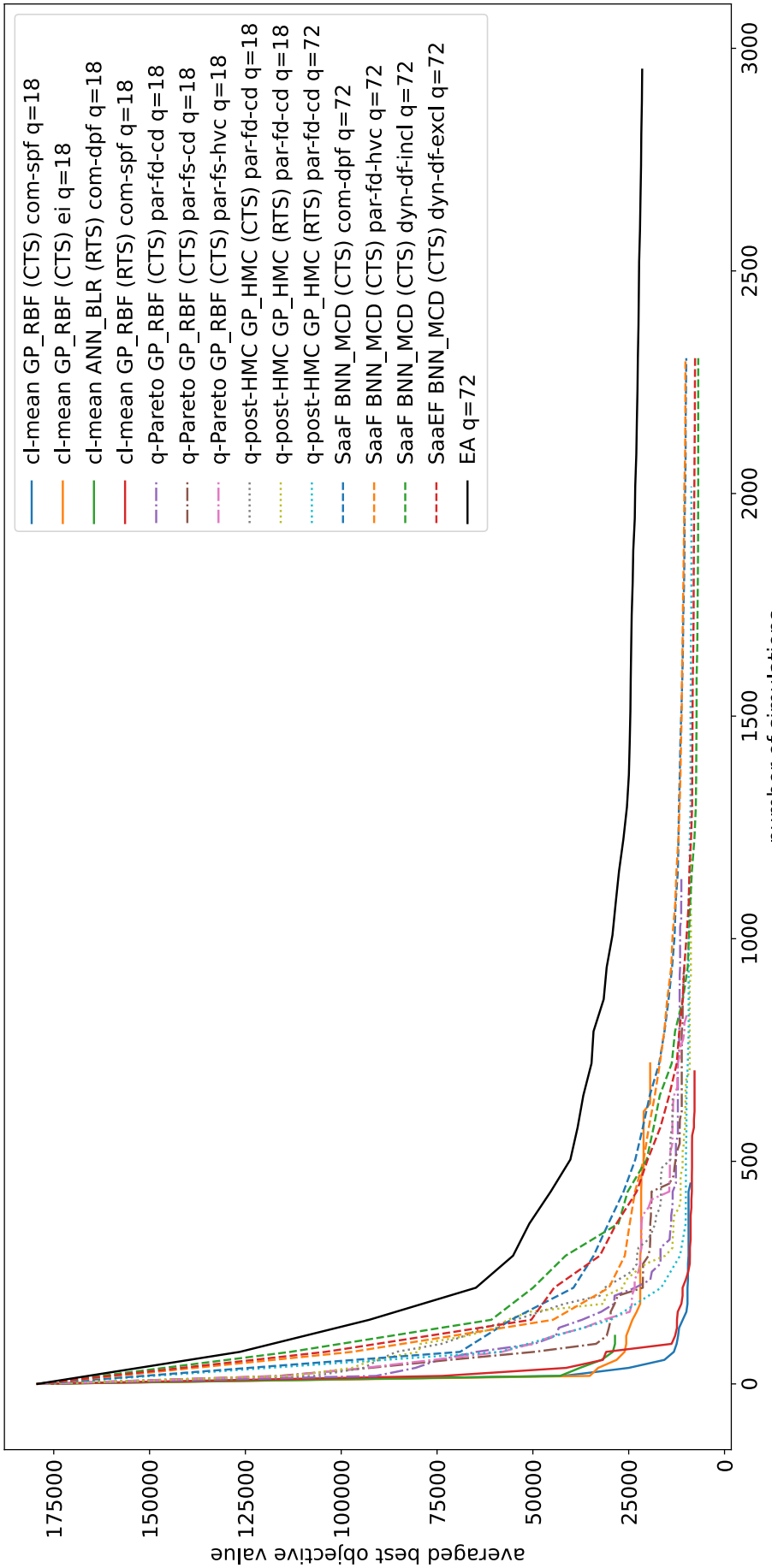

Figure E.3: **P-SAEAs *versus* P-SDAs** application to the **Covid-19 contact reduction** problem. Convergence profile in terms of best objective values averaged over the 10 repetitions of the experiment.

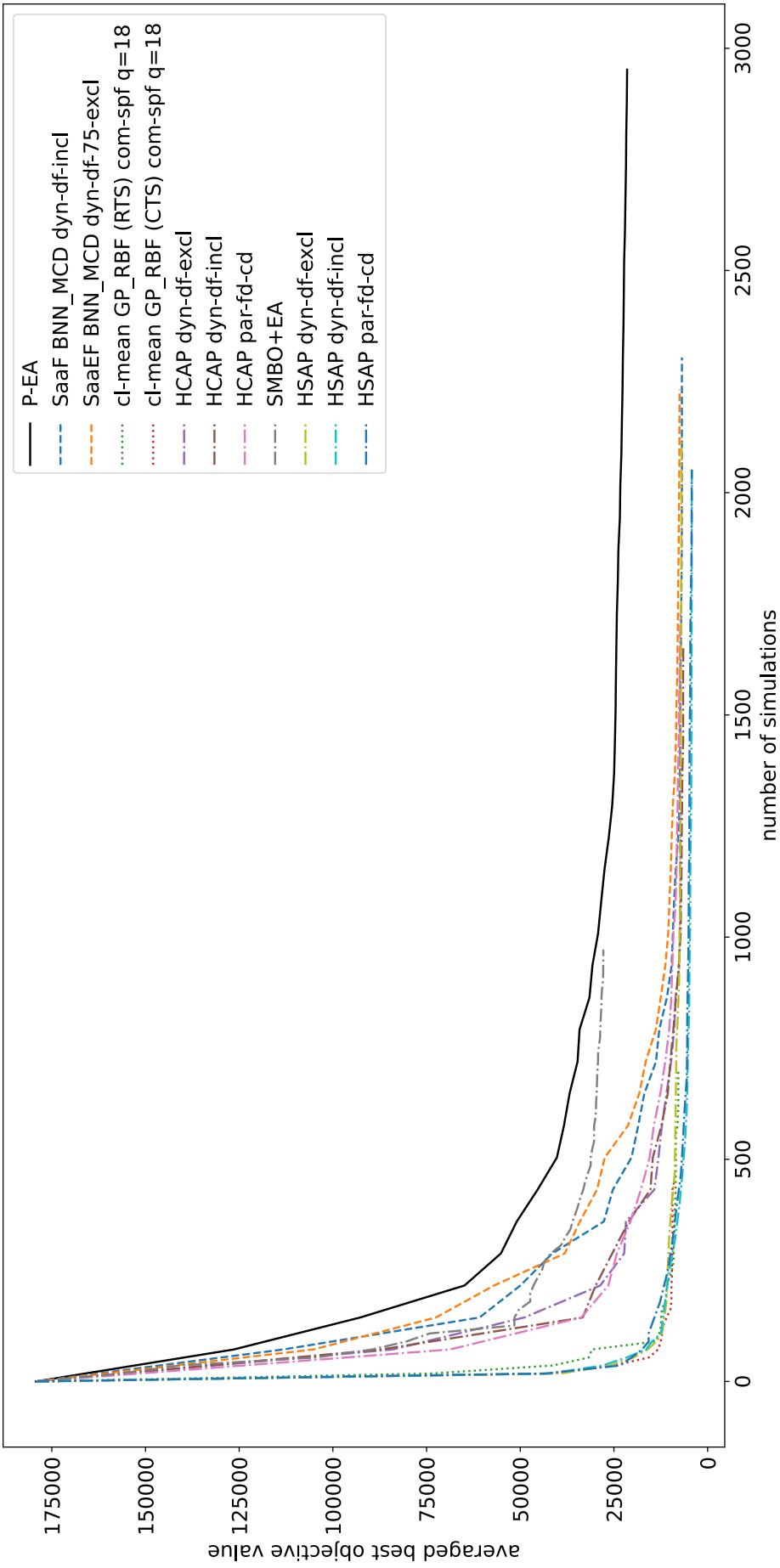

Figure E.4: **Parallel Hybrid methods** applied to the **Covid-19 contact reduction** problem. Convergence profile in terms of best objective values averaged over the 10 repetitions of the experiment.

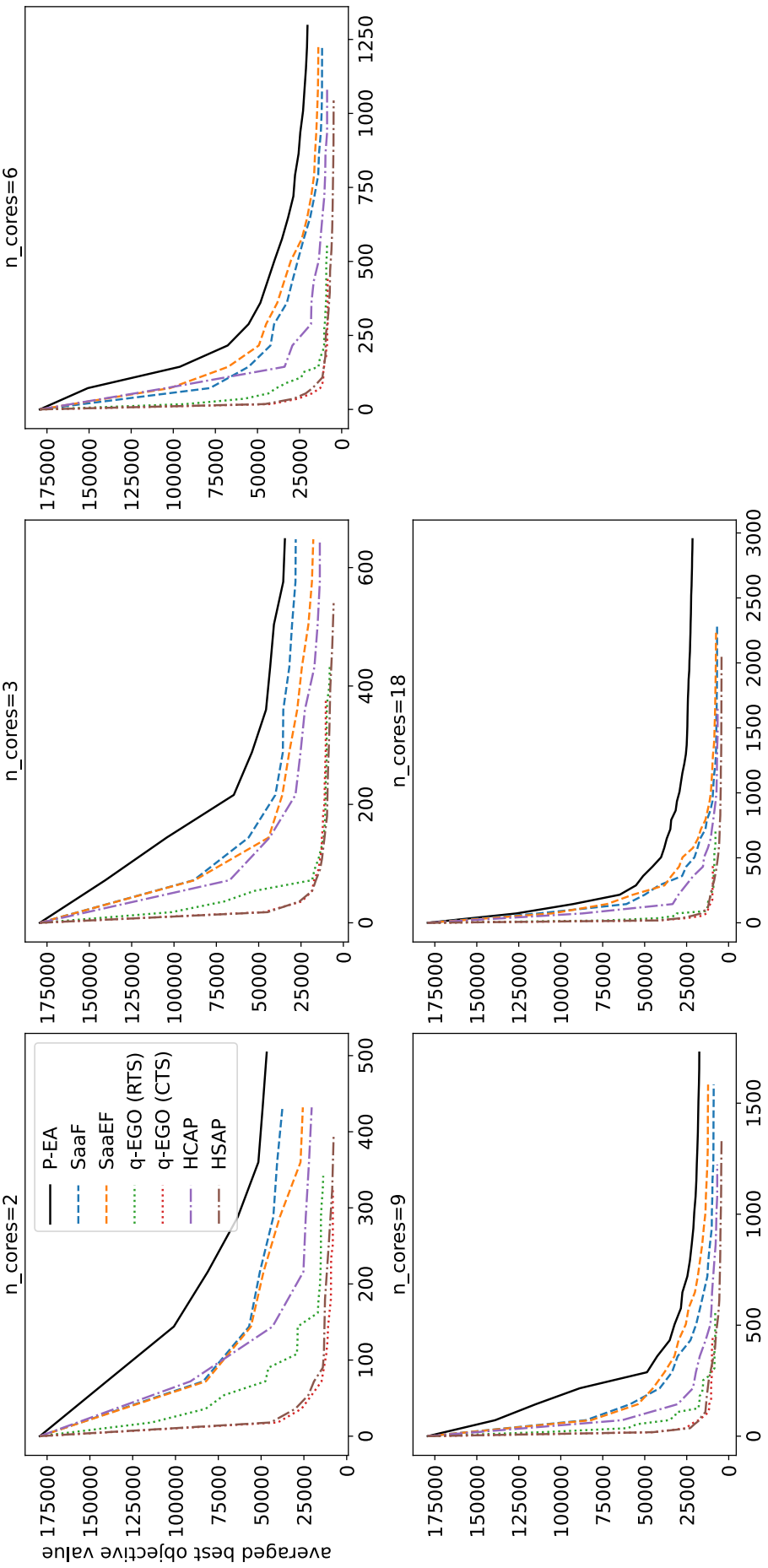

Figure E.5: **Parallel scalability (unaltered values for q).** Convergence profile in terms of best objective values averaged over the 10 repetitions of the experiment for different numbers of available computing cores and without any modification of the algorithms.

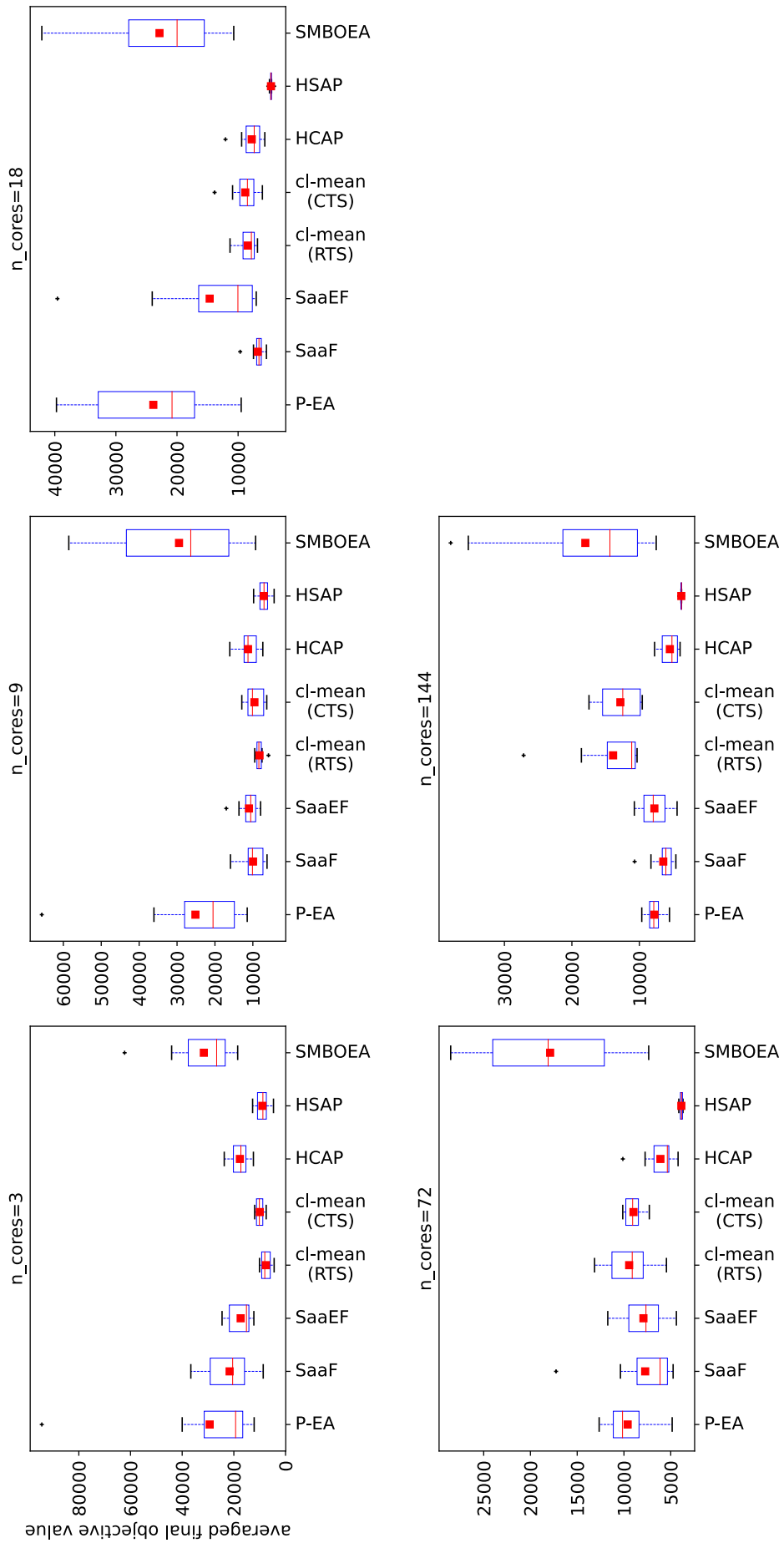

Figure E.6: **Parallel scalability ($q = n_{cores}$).** Distribution of the best objective values from the 10 repetitions of the experiment. The number of simulations per cycle q is fixed to n_{cores} for all the approaches. Averaged values are depicted by red squares, median values by red dashes and variance information is given by the length of the boxes.

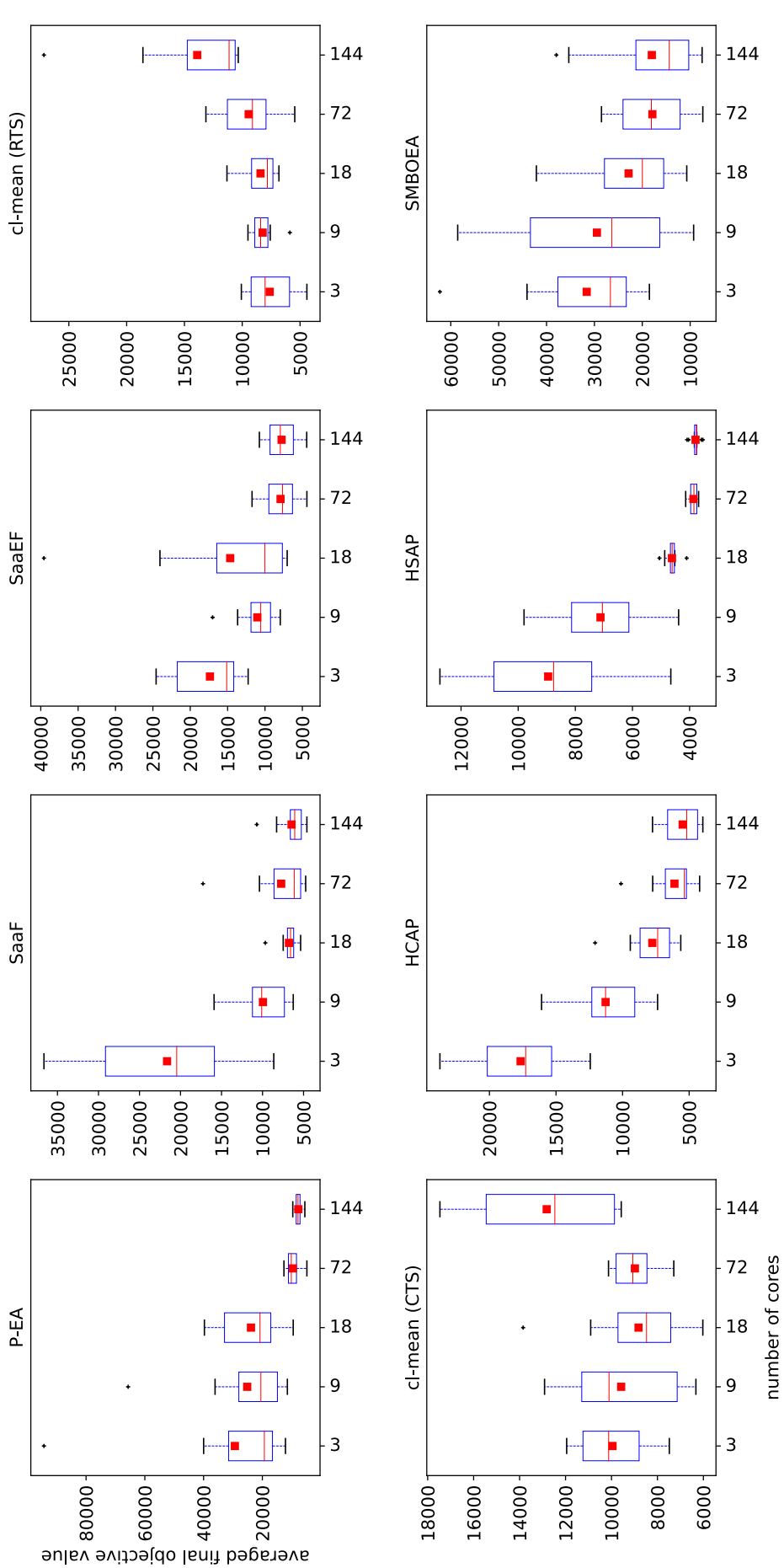

Figure E.7: **Parallel scalability ($q = n_{cores}$).** Distribution of the best objective values from the 10 repetitions of the experiment. The number of simulations per cycle q is fixed to n_{cores} for all the approaches. Averaged values are depicted by red squares, median values by red dashes and variance information is given by the length of the boxes.

Appendix F

pySBO

Listing F.1: Python code of the *COVID_vaccines* class in pySBO.

```
import time
import numpy as np
import sys
from AuTuMN.apps.covid_19.vaccine_optimisation.vaccine_opti
    import initialise_opti_object
from Problems.Multi_Objective import Multi_Objective

class COVID_vaccines(Multi_Objective):

    def __init__(self, country):
        Multi_Objective.__init__(self, 17, 3)
        assert type(country)==str
        self.__opti_object = initialise_opti_object(country)

    def __del__(self):
        Multi_Objective.__del__(self)
        del self.__opti_object

    def __str__(self):
        return "COVID vaccines problem\n  "+str(self.n_dvar)+"
            decision variables\n  "+str(self.n_obj)+" objectives"


    def perform_real_evaluation(self, candidates):
        assert self.is_feasible(candidates)
        if candidates.ndim==1:
            candidates = np.array([candidates])

        costs = np.zeros((candidates.shape[0], self.n_obj))
        for i,cand in enumerate(candidates):
            t_start = time.time()
            costs[i] = self.__opti_object.evaluate_objective(
                cand) # deaths, hospital, relaxation
            costs[i][2] = 1. - costs[i][2]
            t_end = time.time()
            print("Simulation time: "+str(t_end-t_start))
```

```
34
35          return costs
36
37
38      def get_bounds(self):
39          bounds=np.ones((2,self.n_dvar))
40          bounds[0,:]*=0.0
41          bounds[1,:]*=1.0
42          return bounds
43
44
45      def is_feasible(self, candidates):
46          feasible=False
47          if Multi_Objective.is_feasible(self, candidates)==True:
48              [lower_bounds, upper_bounds]=self.get_bounds()
49              phase2_less98 = np.where(np.sum(candidates[:,0:8],
                    axis=1)<=0.98)[0]
50              phase3_less98 = np.where(np.sum(candidates[:,8:16],
                    axis=1)<=0.98)[0]
51              feasible=(lower_bounds<=candidates).all() and (
                    candidates<=upper_bounds).all() and (np.sum(
                    candidates[:,0:8], axis=1)<1.0+1.e-8).all() and (
                    np.sum(candidates[:,8:16], axis=1)<1.0+1.e-8).all
                    ()
52              if feasible:
53                  if phase2_less98.size!=0:
54                      print("[COVID_vaccines.py] WARNING: 
                            candidate(s) "+str(phase2_less98)+" use 
                            less than 98% of vaccines in phase 2")
55                  if phase3_less98.size!=0:
56                      print("[COVID_vaccines.py] WARNING: 
                            candidate(s) "+str(phase3_less98)+" use 
                            less than 98% of vaccines in phase 3")
57
58          return feasible
```

Table F.1: List of libraries used in the numerical experiments reported in this thesis.

Name	Tool	Language	Reference
GPyTorch	GPs	Python	[Gar+18]
Flacco	dispersion metric	R	[KT19]
Keras and Tensorflow	ANNs	Python	[Cho15] and [Aa15]
Matplotlib	visualization	Python	[Hun07]
mpi4py	parallel communication	Python	[DF21]
Numpy	vector calculations	Python	[Ha20]
pybnn	ANN_BLR	Python	[Sno+15]
pyDOE	LHS	Python	[Bau+]
Pygmo	benchmark, hyper-volume	Python	[Ba19]
pyKriging	Kriging	Python	[PR15]
Pyro and NumPyro	GP_HMC, BNN_HMC	Python	[Bin+18] and [PPJ19]
Scikit-Learn	K-Means	Python	[Ped+11]
Scipy	distance calculations	Python	[Va20]

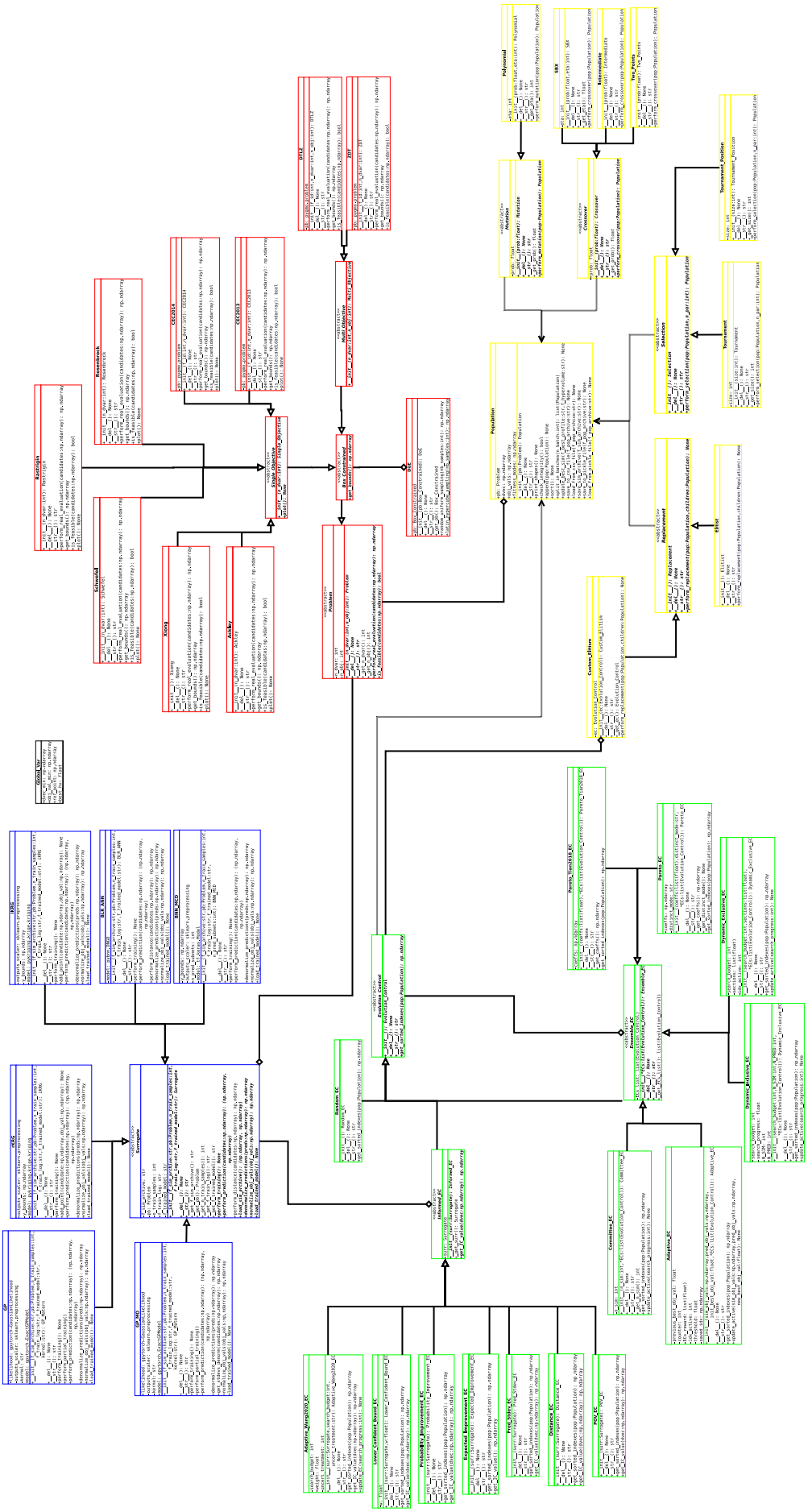

Figure F.1: Global UML diagram of pySBO. Available at https://pysbo.readthedocs.io/en/latest/_downloads/28999b287a4b8574c9e66583a29410af/UML.svg

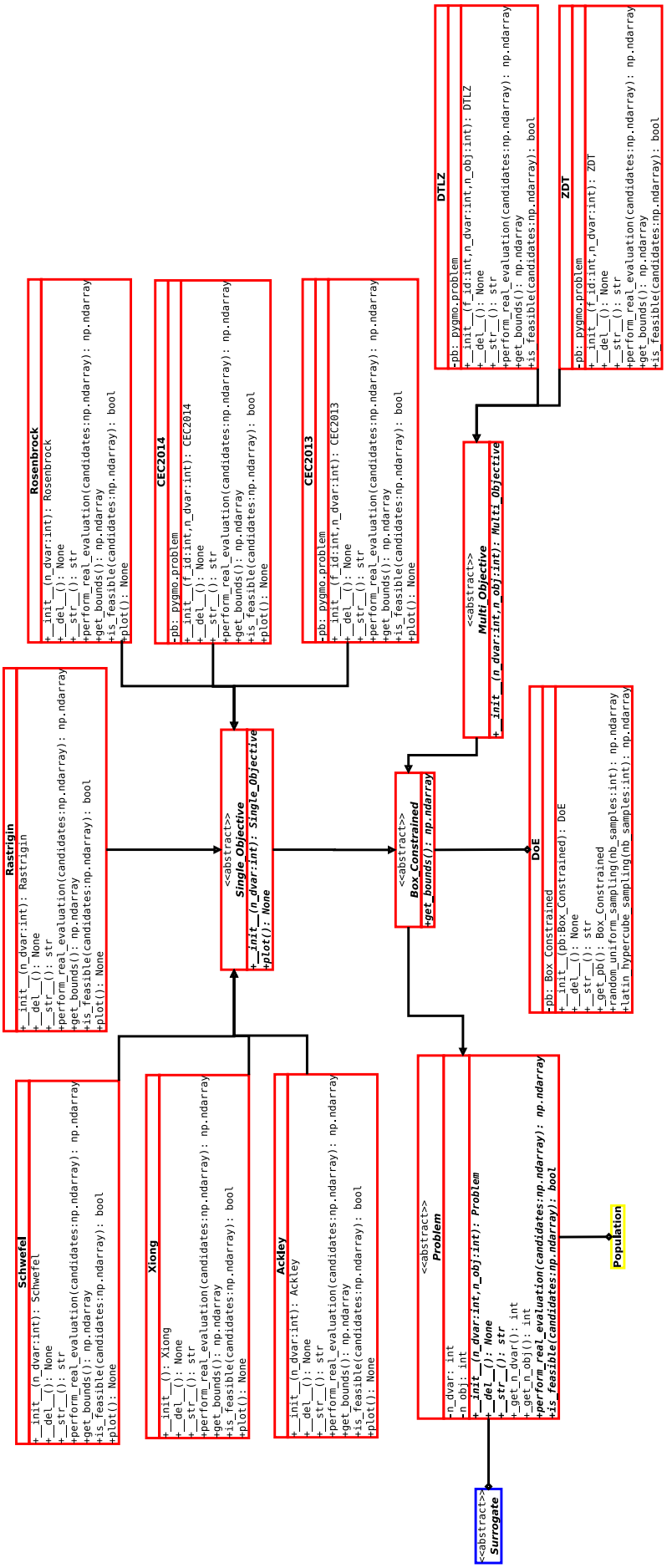

Figure F.2: Global UML diagram of the Problem collection of classes.

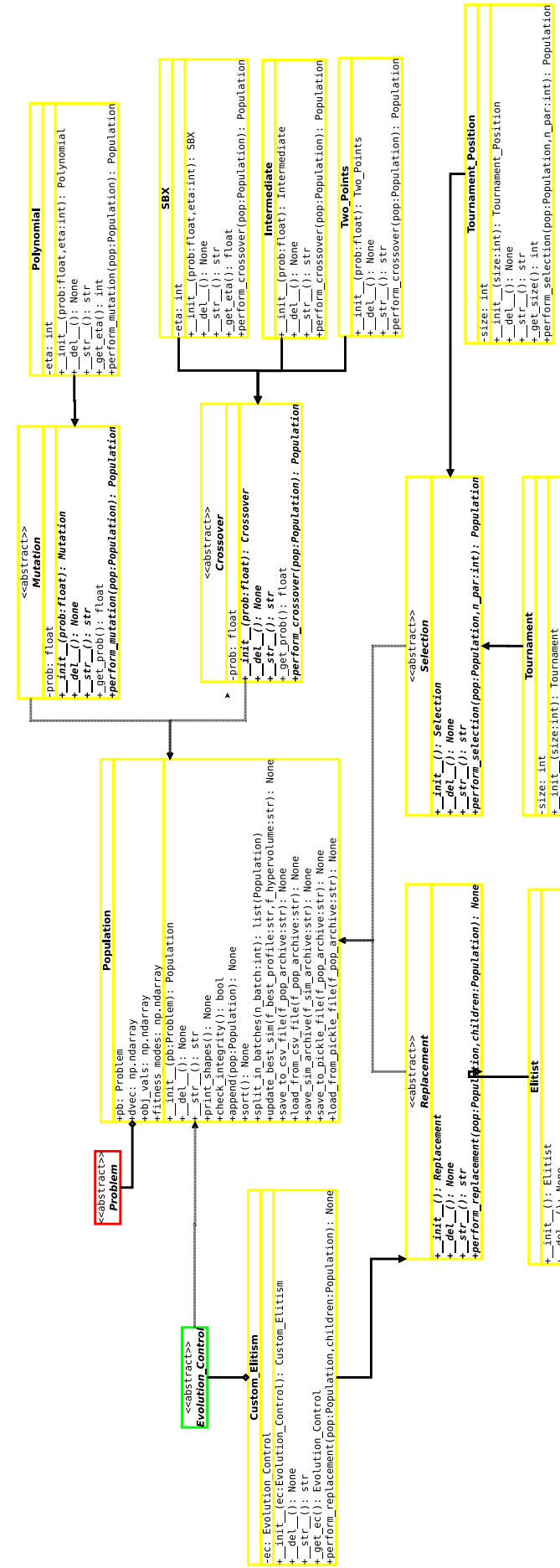

Figure F.3: Global UML diagram of the Evolution collection of classes.

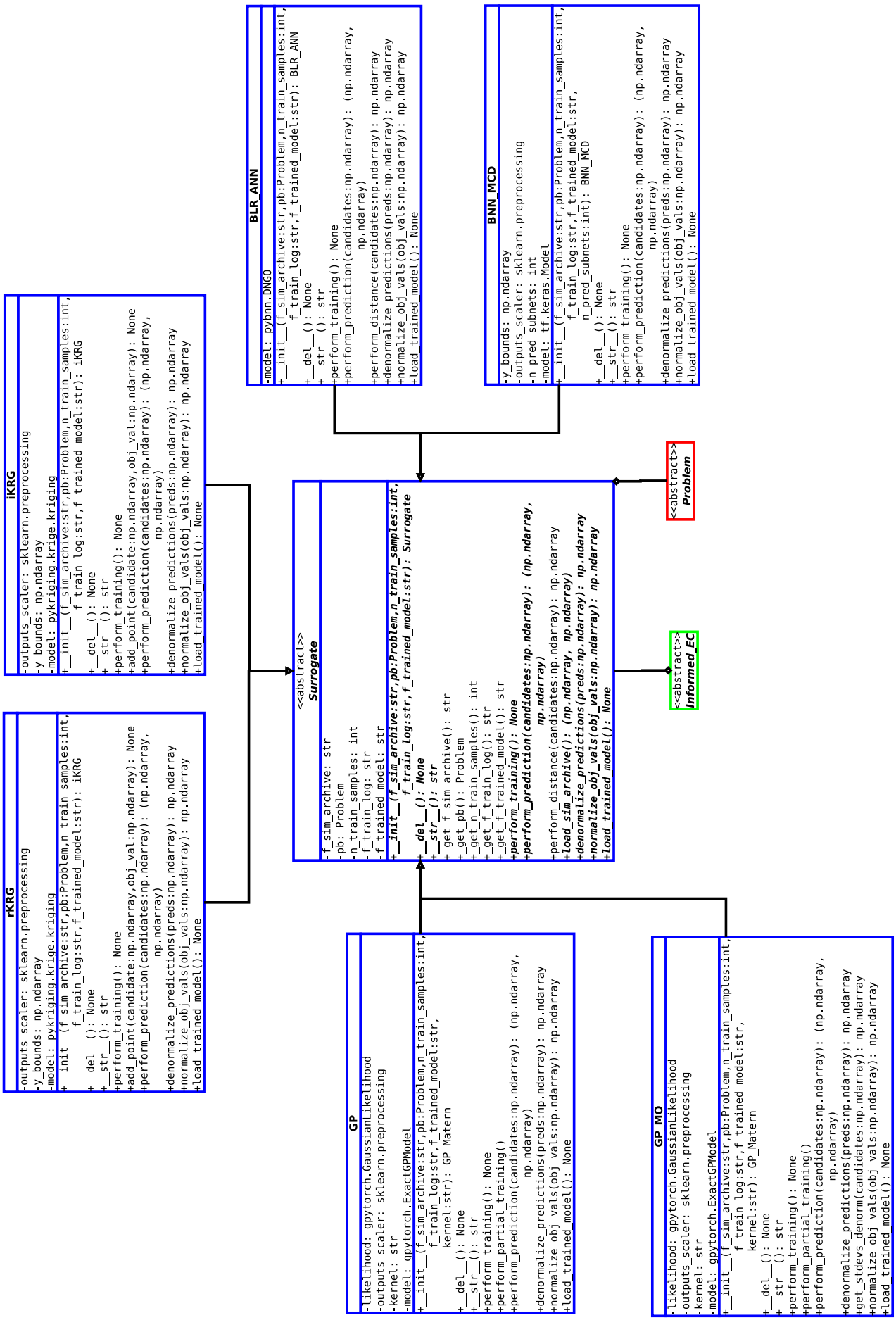

Figure F.4: Global UML diagram of the Surrogate collection of classes.

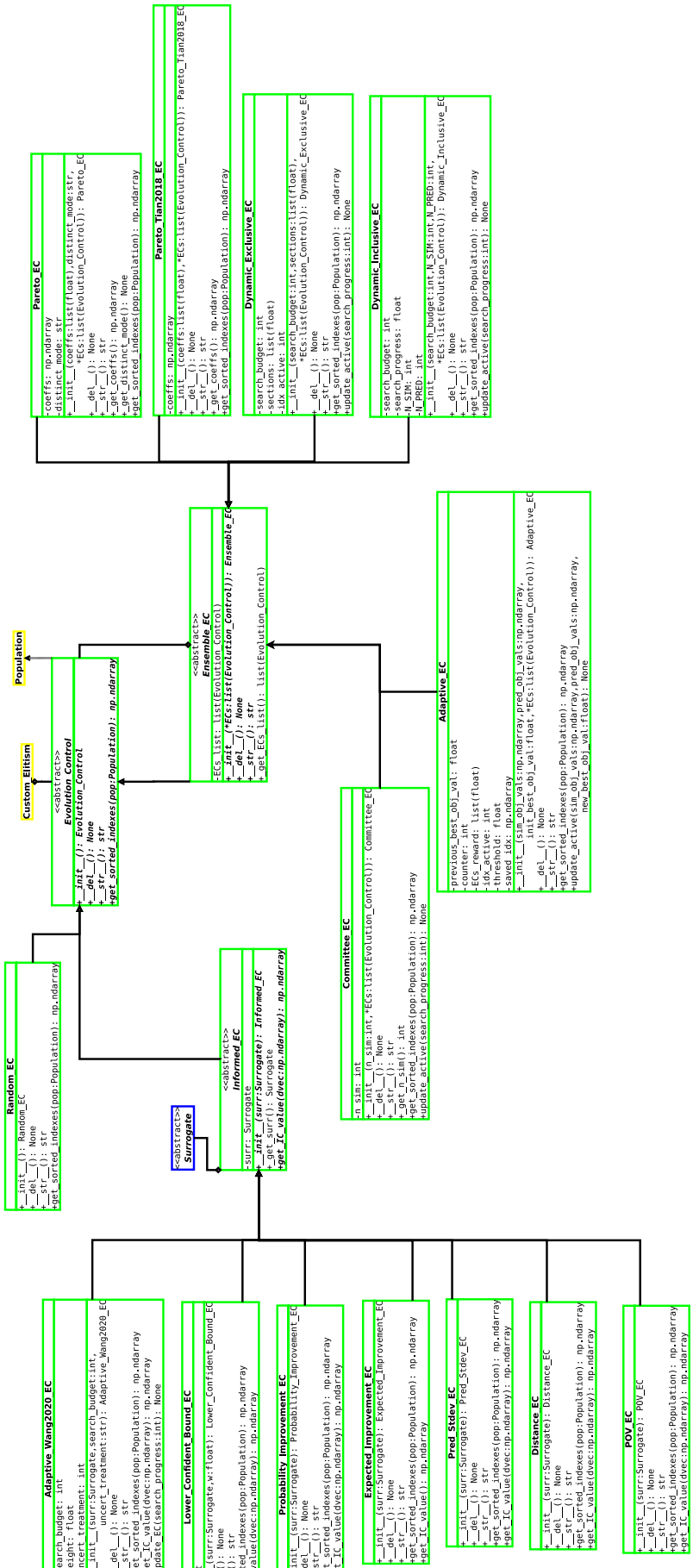

Figure F.5: Global UML diagram of the Evolution Control collection of classes.