Titre: Title:	Contributions to Constrained Mixed-Variable and Hierarchical Blackbox Optimization
Auteur: Author:	Edward Hallé-Hannan
Date:	2026
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Hallé-Hannan, E. (2026). Contributions to Constrained Mixed-Variable and Hierarchical Blackbox Optimization [Ph.D. thesis, Polytechnique Montréal]. PolyPublie. https://publications.polymtl.ca/76526/

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Contributions to Constrained Mixed-Variable and Hierarchical Blackbox Optimization

EDWARD HALLÉ-HANNAN
Département de mathématiques et de génie industriel

Thèse présentée en vue de l'obtention du diplôme de *Philosophiæ Doctor*
Mathématiques

Avril 2026

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Cette thèse intitulée :

Contributions to Constrained Mixed-Variable and Hierarchical Blackbox Optimization

présentée par **Edward HALLÉ-HANNAN**
en vue de l'obtention du diplôme de *Philosophiæ Doctor*
a été dûment acceptée par le jury d'examen constitué de :

Dominique ORBAN, président
Charles AUDET, membre et directeur de recherche
Youssef DIOUANE, membre et codirecteur de recherche
Sébastien LE DIGABEL, membre et codirecteur de recherche
Daniel ALOISE, membre
Katya SCHEINBERG, membre externe

DEDICATION

For my wife, Stéphanie,
and my brother, Damien

ACKNOWLEDGEMENTS

Four years fully dedicated to these research projects are symbolically coming to an end with this manuscript. I find it difficult to precisely describe how I feel, as it resembles a complex coffee blend of satisfaction, uncertainty about what comes next, tiredness, and happiness. One thing, however, is certain: I am filled with gratitude toward many people. This feeling is undoubtedly the dominant note of that coffee blend.

The first people who come to mind are, without surprise, my three advisors: Charles, Sébastien, and Youssef. I am forever grateful for their supervision, which was always grounded in openness, kindness, and wisdom. Their presence and guidance made this long and demanding journey infinitely more human, manageable and valuable. Charles, your intelligence never ceases to baffle me, a striking balance of logic and intuition. Sébastien, your focus on bringing research to people and amplifying its impact through communication and software says a lot about how caring and thoughtful a researcher you are. Youssef, your clear vision and ability to step back from the details to focus on core research ideas is something I deeply admire and consider immensely valuable in research. You are all exemplary mentors, and through you I have learned far more than just technical and research knowledge.

Next is Christophe, who was not officially my supervisor, but whom I had always implicitly considered as one from the moment he became involved in the research projects. It is clear to me that none of the implementation and numerical aspects of these projects could have been accomplished without Christophe's support and help. Christophe, I am deeply grateful for your patience and your dedication.

I send my thanks to the rising star Paul S., whose collaboration and expertise in surrogate modeling greatly contributed to these research projects. My friend, it is always a great pleasure to work with you, and I sincerely hope we will continue to do so in the future.

My sincere thanks also go to the members of the jury: Pr Orban, Pr Aloise, and Pr Scheinberg. I feel truly honored to have had a jury of such prestigious caliber. It was a privilege to have my work evaluated and discussed by these highly esteemed researchers. I also thank the faculty representative Pr Harvey.

Thanks to the many students and colleagues of GERAD whom I had the chance to meet throughout these years. They made this journey far more enjoyable, and through countless discussions and their research projects, I discovered many new topics and perspectives. You know who you are. I will not attempt to make a list, as there are simply too many of

you... and, quite frankly, to protect myself from inevitably forgetting someone! Many thanks the kind staff of GERAD for all the technical and administrative support over the years, especially Marie, Karine, and Marilyne.

I gratefully acknowledge the financial support of NSERC, FRQNT, and the Trottier Foundation throughout my PhD studies. Their support gave me the privilege of fully dedicating myself to these research projects. Beyond my own work, these organizations play an essential role in supporting and shaping the next generation of scientists in Québec and Canada.

Outside of my academic journey, I have spent, not nearly enough, time with my longtime friends, who provided me insightful perspectives far beyond academia. I have known now these friends more than half of my life. It has been a remarkable human experience to see each of us grow and mature[1] over the years. To avoid making anyone jealous, I shall proceed in alphabetical order: the charismatic and sharp Alexandre, the wise and kind Félix, the holistic and free-spirited François, the loyal and meticulous Frédérick, and the pragmatic and driven Pierre-Loup. Part of this group is also my brother and best friend Damien, the heart of the group, my lifelong teammate, and my eternal Warthog driver. Simply by observing you, I learned that regardless of the hardships, one can still enjoy the ride by staying authentic and genuine to who they are.

I was fortunate to have incredibly supportive parents who always encouraged me and pushed me to surpass myself. Dad, you always encouraged us to think critically, be curious, and play with ideas. Even though you are not officially an academic, I always felt like you were one... perhaps you were in another life. Mom, you are the person that believes in me the most. Although I often joke that you are biased, I truly appreciate your unlimited support.

Finally, saving the best for last, my wife and beloved Stéphanie. This achievement may carry my name, but the bigger story is that you were my Sam Gamgee throughout this entire journey. Even though your mathematical knowledge stops at basic calculus, this is an achievement that I wholeheartedly share with you. Thank you for your kindness, genuineness, and intelligence. You have supported more than I could have ever anticipated. You are simply a wonderful person, and I still have so much to learn from you.

[1] The use of the word “mature” becomes somewhat debatable once we grouped.

RÉSUMÉ

Cette thèse considère principalement des problèmes d'optimisation de boîtes noires sous contraintes qui sont mixtes. Les chapitres ultérieurs s'étendent aux problèmes hiérarchiques. La fonction objectif et les contraintes ne sont pas disponibles sous forme analytique et ne sont accessibles que par des évaluations, qui sont généralement des programmes informatiques ou des simulations. Ces évaluations peuvent être coûteuses en temps de calcul, ce qui motive le développement de méthodes d'optimisation capables de trouver de bonnes solutions en utilisant le moins d'évaluations possible. Les problèmes étudiés sont mixtes. Le domaine contient des variables de différents types, notamment des variables continues, entières et catégorielles. Les variables catégorielles sont particulièrement difficiles à traiter car elles prennent des valeurs qualitatives (catégories) qui possèdent peu de structure. Certains des problèmes étudiés sont également hiérarchiques. Dans ce contexte, les points ne partagent pas nécessairement les mêmes variables ou bornes. Ces caractéristiques conduisent à des domaines d'optimisation complexes combinant des fonctions boîte noire, des contraintes, différents types de variables et des structures hiérarchiques.

Ces problèmes apparaissent dans de nombreuses applications d'actualité, dont l'optimisation d'hyperparamètres en apprentissage automatique, la conception d'aéronefs et la configuration de systèmes d'ingénierie. Malgré leur importance pratique, les problèmes d'optimisation de boîtes noires mixtes et hiérarchiques restent relativement peu développés. En particulier, il existe peu de méthodes automatiques d'optimisation de boîtes noires mixtes disposant de garanties de convergence. De plus, les méthodes capables de gérer des niveaux arbitraires de hiérarchie ou de comparer des points définis sur des variables et des bornes différentes restent largement inexplorées. Cette thèse vise à combler plusieurs de ces lacunes en développant des outils de modélisation et des structures qui permettent de mieux optimiser ces problèmes. La thèse est de type par articles et contient quatre articles qui sont soit acceptés, soit en révision, soit soumis. Chaque article présente une contribution principale de la thèse.

La première contribution introduit CatMADS, une généralisation de l'algorithme MADS pour l'optimisation de boîtes noires sous contraintes comportant des variables catégorielles, entières et continues. Les variables catégorielles sont structurées à l'aide de voisinages basés sur des distances. Les voisinages sont utilisés pour développer des mécanismes de sonde pour les variables catégorielles. La méthode étend les garanties de convergence de MADS à l'aide de définitions généralisées de l'optimisation non lisse au contexte mixte. Le manuscrit est actuellement en révision à *SIAM Journal on Optimization*.

La deuxième contribution développe des voisinages catégoriels à partir de substituts des fonctions du problème. Des processus gaussiens sont utilisés pour capturer les similarités entre les variables catégorielles par rapport à la fonction objectif et aux contraintes. Les voisinages sont ensuite construits à partir de notions de dominance, établissant des compromis entre les similarités relatives à l'objectif et aux contraintes. Intégrée dans CatMADS, cette approche mène à une variante qui surpasse les solveurs de l'état de l'art. Le manuscrit correspondant est soumis au *Journal of Global Optimization*.

La troisième contribution traite des domaines hiérarchiques dont les points ne partagent pas nécessairement les mêmes variables ni les mêmes bornes. La distance méta est introduite afin de comparer ces points dans un espace étendu contenant toutes les variables possibles. Une bijection effectue une correspondance un-à-un entre un point du domaine original et un point de l'espace étendu. Cela permet de comparer des points appartenant à des sous-espaces de dimensions et de bornes différentes. La distance méta est publiée dans *Neurocomputing*.

À la suite du travail sur la distance méta, la contribution finale développe un cadre unifié pour les domaines mixtes et hiérarchiques. Ce cadre étend la distance méta et intègre plusieurs concepts de la littérature afin de modéliser un éventail plus large de situations, dont les incompatibilités entre variables. Le cadre est publié dans *Structural and Multidisciplinary Optimization*.

ABSTRACT

This thesis primarily considers constrained blackbox optimization problems with mixed variables, and extends to hierarchical domains in later chapters. The objective function and constraints are not available in analytical form and can only be accessed through evaluations, that are typically computer programs or simulations. Such evaluations may be time-consuming, which motivates the development of optimization methods capable of finding high-quality solutions using as few evaluations as possible. The problems considered are mixed-variable, meaning that the domain contains variables of different types, including continuous, integer, and categorical variables. Categorical variables are particularly challenging because their take qualitative values (categories) that lack natural or meaningful distances. Some of the problems studied are also hierarchical, meaning that points may involve different variables and be subject to different bounds. These characteristics lead to complex optimization domains that combine blackboxes, constraints, various variable types, and hierarchical structures.

These problems arise in many important applications, including hyperparameter optimization in machine learning, aircraft design, and the configuration of engineering systems. Despite their practical relevance, mixed-variable and hierarchical blackbox optimization problems remain relatively underdeveloped. In particular, there are few automatic mixed-variable blackbox optimization methods with convergence guarantees capable of handling such domains. Moreover, frameworks capable of tackling arbitrary levels of hierarchy or fully comparing points defined over different variables and bounds remain largely unexplored. This thesis aims to address several of these gaps by developing modeling and optimization mechanisms that better structure mixed-variable and hierarchical domains. The thesis is article-based and contains four articles that are either accepted, under revision, or submitted. Each article presents a main contribution of the thesis.

The first contribution introduces CatMADS, a generalization of the MADS algorithm for constrained blackbox optimization with categorical, integer, and continuous variables. Categorical variables are structured through distance-based neighborhoods that define polling mechanisms for categorical variables. The method extends convergence guarantees through generalized definitions of nonsmooth optimization while extending standard MADS mechanisms to the mixed-variable setting. The manuscript is currently in revision at *SIAM Journal on Optimization*.

The second contribution develops categorical neighborhoods constructed using surrogate

models. Gaussian processes are used to capture similarities between categorical variables with respect to both the objective and the constraints. The neighborhoods are then constructed based on notions of dominance, establishing trade-offs between similarities in the objective and the constraints. Integrated within CatMADS, this approach leads to a variant that outperforms state-of-the-art solvers. The corresponding manuscript is submitted to *Journal of Global Optimization*.

The third contribution addresses hierarchical optimization problems, where points in the domain may not share the same variables or bounds. A meta distance is introduced to compare such points by mapping hierarchical domains to an extended space containing all possible variables. This enables comparisons between points in subspaces of different dimensions and bounds. The meta distance is published in *Neurocomputing*.

Following this work, the final contribution develops a unified framework for hierarchical mixed-variable domains. This framework extends the meta distance and integrates several concepts from the literature to model a wider range of situations, such as incompatibilities between variables. The framework is published in *Structural and Multidisciplinary Optimization*.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1 INTRODUCTION

You may ask yourself
"Where does that highway go to?"
And you may ask yourself
"Am I right, am I wrong?"

— Talking Heads, *Once in a Lifetime*

This doctoral thesis lies at the intersection of mathematical optimization and machine learning. The core of the thesis is to develop new optimization concepts combining ideas from both fields. The underlying and fundamental goal is to address challenging blackbox optimization problems involving different types of variables.

1.1 Context: blackbox optimization

Optimization is a discipline of applied mathematics concerned with finding the most satisfactory solution to a given problem. Such a problem is essentially composed of decision variables that can be assigned different values, an objective function to be optimized, and constraints describing the feasibility of a point. The goal of optimization is to determine a solution (a point) that satisfies the constraints and minimizes[1] the objective function, such that

$$\min_{x \in \Omega} f(x), \tag{1.1}$$

where,

- $x \in \mathcal{X}$ is a point that contains the decision variables and lies in the domain $\mathcal{X}$,
- $f : \mathcal{X} \to \bar{\mathbb{R}}$ (with $\bar{\mathbb{R}} = \mathbb{R} \cup \{+\infty\}$) is the objective function, and
- $\Omega \subseteq \mathcal{X}$ is the feasible set defined by the constraints of the problem.

Optimization is divided into several branches, including linear optimization, combinatorial optimization, and nonsmooth optimization, to name a few. Each branch has its own formalism, optimality conditions, and methods, which exploit the available mathematical properties [124]. For example, in linear optimization, the simplex method [92] is an algorithm

[1] Without loss of generality, it is always possible to formulate an optimization problem as a minimization problem.

that exploits the linear form of the objective function f and the constraints by moving from vertex to vertex of the domain. The nature of the objective function f and of the feasible set Ω greatly influences which branch should be employed for a given problem [9].

In blackbox optimization, the objective function and constraints defining the feasible set Ω are supposed to be blackboxes. A blackbox is a process that only provides information through evaluating an input, represented as a point, and outputting a corresponding image.

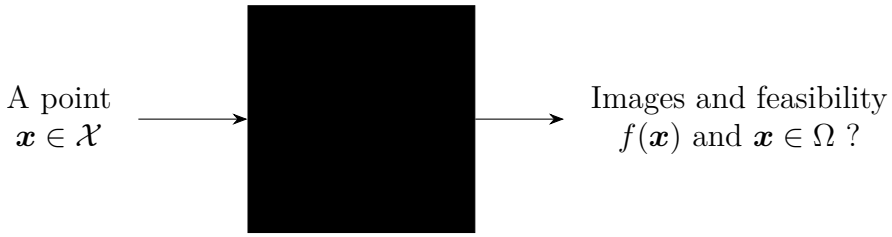


Figure 1.1 A visual representation of a blackbox.

Figure 1.1 illustrates a generic blackbox in which a point x is evaluated, then a objective value $f(\boldsymbol{x})$ and information on feasibility are output. Typically, blackboxes are computer simulations that are *expensive-to-evaluate*, where each evaluation is obtained by running a complex program. In [13], a simulation outputs the estimated energy produced by a solar power plant for given design variables, such as the number of solar panels or turbine types. In aerospace engineering, aircraft with improved environmental performance are frequently designed by optimizing simulations [63, 222]. These simulations take variables representing geometries and material choices, and output an energy consumption score. In [9, 48], several engineering applications related to blackbox optimization from the past two decades are described.

A blackbox is not necessarily a mathematical function since the same input may yield different outputs due to noise or variability in the evaluation process. The term *blackbox function* is used in this document, although it is a slight abuse of terminology.

The terms of *blackbox* and *blackbox optimization* used throughout the thesis are defined as follows.

Definition 1 (Blackbox and blackbox optimization [27])**.** *A* blackbox *is any process that, when provided an input, returns an output, but the inner workings of the process are not analytically available.*

Blackbox optimization (BBO) *is the study of applications, attributes, and solutions of optimization problems in which the values of one or more of the functions defining the problem are provided through blackbox(es). Such problems are referred to as BBO problems.*

BBO is closely related to *derivative-free optimization* [87]. The optimization methods developed in this thesis are in the class of algorithms that are derivative-free.

Definition 2 (Derivative-free method [27])**.** Derivative-free optimization (DFO) *is the study of optimization algorithms that do not directly use derivatives (or other first-order information). Such algorithms are referred to as DFO algorithms.*

Blackbox refers to a property of the optimization problem where the objective function and constraints can only be accessed through evaluations. DFO refers to a class of algorithms that do not rely on derivative information. Such algorithms are applied to blackbox problems, but also to problems for which derivatives exist yet are impractical or too costly to exploit. In short, the problems of interest are blackbox optimization problems, while the optimization methods are derivative-free.

1.2 Research problem

The main problem formulation in Equation (1.1) is now further detailed beyond the blackbox setting.

The constraints defining the feasible set Ω are assumed to be *relaxable* and *quantifiable*, meaning that they can be evaluated and return a meaningful value even when they are violated [162]. Constraint violations can thus be measured and handled explicitly within the optimization process. In contrast, an *unrelaxable* constraint must be satisfied for the blackbox to execute properly. Such constraints may correspond to bounds or *hidden constraints* that cause the evaluation to fail or produce invalid outputs. These latter constraints may arise when optimizing simulations or programs that are not fully reliable, for example when an unexpected division by zero is encountered. When a point $x \in \mathcal{X}$ triggers an unrelaxable constraint, it is considered invalid and assigned $f(\boldsymbol{x}) = +\infty$.

The problems considered are also *mixed-variable*. Each point in the domain is composed of variables of different types. A variable can take one of the following types:

- continuous, when it takes values in a real interval,
- integer, when it is discrete but still quantitative,
- categorical, when it is discrete and qualitative.

Categorical variables are notoriously difficult to handle, since they lack structure. For example, the variable representing the flavor of an ice cream, $x \in \{$“vanilla”, “chocolate”, “mint”$\}$,

lies in a set without much structure. Indeed, the categories are not ordered relative to one another, and the distances between them have no intrinsic mathematical meaning [130].

As of today, most mixed-variable derivative-free methods[2] that handle categorical variables stem from Bayesian optimization (BO) [90, 112, 194, 270] and metaheuristic approaches [95, 128, 248]. Both approaches lack convergence guarantees and robust local mechanisms. BO relies on surrogate models that are not exact approximations of the functions. Metaheuristics methods are based on sampling techniques that simply do not posses convergence guarantees in general.

In [4], the continuous direct search algorithm Mesh Adaptive Direct Search (MADS) is extended to handle discrete variables, namely integer and categorical ones. The extension, called Mixed-variable MADS (MV-MADS), employs manually defined neighborhoods to handle discrete variables. These neighborhoods are referred to as *user-defined*, emphasizing that they must be explicitly provided and are not constructed automatically. They must be tailored to each problem, and may require prior or expert knowledge that is not always available in a blackbox setting. The work [4] is primarily theoretical and incorporates discrete variables into the convergence analysis of MADS via user-defined neighborhoods. It does not provide numerical experiments or an implementation. Similarly, [167] also uses user-defined neighborhoods to generalize a continuous line-search descent method. The paper [167] focuses on a specific application involving the optimal design of a magnetic resonance device.

On top of being blackbox and mixed-variable problems, the later chapters of this thesis consider problems that are also *hierarchical*. In these problems, domains contain points that do not share the same variables and/or variables with different bounds. This can be modeled by a special type of variable, called *meta variable*, defined next.

Definition 3 (Meta variables (adapted from [25, 124]))**.** *A variable of any type is said to be* meta *if its values determine the inclusion of other variables or constraints, or affect the bounds of other variables.*

These special variable appear in many popular applications related to mixed-variable and hierarchical optimization. In deep learning [159], hyperparameter optimization includes a meta variable representing the number of hidden layers, which determines the number of variables associated with the neurons of the model. In aircraft design, the type of motor is a meta variable that activates its own subset of design variables [226]. In [167], the optimal design of a magnetic resonance imaging device involves a meta variable determining the number of magnets and thus the number of variables and constraints. Few methods have

[2] In mixed-variable problems, derivatives concern only the continuous variables.

been proposed to address such problems. However, they have important limitations.

In [194], Bayesian optimization is employed for mixed-variable and hierarchical domains. However, interactions between variables are modeled only within a single hierarchical level. Moreover, variables that are not shared between subproblems are not compared. The hierarchy considered is therefore limited, and some information between unshared variables is inevitably not exploited.

The user-defined neighborhoods employed in [4] and [167] have enough flexibility to handle hierarchical problems. Indeed, a neighborhood centered at a given point can contain points that do not share the same variables. However, this requires the user to explicitly encode the hierarchical structure and manually design appropriate neighborhoods, which can be quite difficult for some problems.

1.3 Main research gaps, objectives and contributions

The previous section highlights that multiple applications fall within the class of problems considered and identifies two major research gaps central to this thesis. First, there is currently no mixed-variable blackbox optimization method that is both automatic and equipped with convergence guarantees. As discussed in previous section, BO and metaheuristic approaches are automatic, but lack convergence guarantees. Methods based on user-defined neighborhoods possess theoretical guarantees, but are not automatic and are impractical in a blackbox optimization setting.

Second, existing hierarchical derivative-free methods do not handle multiple levels of hierarchy nor exploit all available information. Indeed, in practice, the basic approach of separating the hierarchical problem into many subproblems is still commonly used, although this is undesirable in a blackbox optimization context where evaluations are expensive and therefore limited. Bayesian optimization methods from the literature typically handle only a single hierarchical level and do not compare variables that are not shared between subproblems. Methods based on user-defined neighborhoods can in principle handle general hierarchical structures through specific implementations explicitly designed to encode the hierarchy. However, this can become a very complex task for users and typically involves arbitrary modeling choices. Such approaches are therefore deemed impractical and are discarded in favor of automatic hierarchical optimization methods.

These research gaps are addressed through the following objectives, each associated with a principal contribution presented in one of the articles composing this thesis.

Objective 1: Develop an efficient and automatic optimization method with convergence

guarantees for mixed-variable blackbox optimization problems. This is done by extending the MADS algorithm, which possesses convergence guarantees for quantitative variables, to the mixed-variable setting. The resulting method, called CatMADS, is presented in Chapter 4. This objective has a similar purpose to MV-MADS [4]. However, the proposed framework differs fundamentally from MV-MADS in several aspects. First, CatMADS uses categorical neighborhoods that are automatically for the problem at hand, whereas MV-MADS relies on manually designed user-defined neighborhoods. Second, integer variables are treated separately from categorical variables and are handled through the quantitative MADS framework rather than through neighborhoods. Consequently, most of the convergence analysis is redeveloped, improved, and extended to account for these more sophisticated mechanisms. Also, unlike MV-MADS, the proposed framework is accompanied by a complete and default implementation that can be used directly on mixed-variable optimization problems. This objective led to a paper currently under revision at *SIAM Journal on Optimization* (see arXiv:2506.06937). A prototype implementation and the computational experiments are available at `https://github.com/bbopt/CatMADS_prototype`.

Objective 2: Construct a problem-specific structure handling categorical variables in constrained optimization settings. The goal is to construct problem-specific and finely tuned categorical neighborhoods from data generated during the optimization process. More specifically, this is achieved using surrogate models of the objective and constraint functions. The resulting neighborhoods are presented in Chapter 5. They can be naturally integrated within the CatMADS framework from the previous objective. This work is presented in a manuscript submitted to *Journal of Global Optimization*. The implementation and the benchmarking scripts are accessible at `https://github.com/bbopt/surrogate_based_neighborhoods`.

Objective 3: Introduce a structure comparing any pair of points in hierarchical domains, even if they do not share the same variables or are subjected to the same bounds. This objective builds on the notion of meta variables from the literature, which provide a modeling framework for defining hierarchical optimization problems. However, this framework alone does not provide any structure for comparing points with different variables or bounds, and no optimization method is introduced at this stage. In Chapter 6, a distance comparing points with different variables and subject to different bounds is presented, which is a necessary first step toward performing hierarchical optimization. The results addressing this objective are published in *Neurocomputing* (see doi:10.1016/j.neucom.2025.131208). The associated code and numerical experiments are available at `https://github.com/bbopt/graph_distance`.

Objective 4: Build an optimization framework handling complex hierarchical dependencies and exploiting all available hierarchical information. The framework is obtained by gener-

alizing the distance introduced in the previous objective using concepts from the existing literature. It is then extended to Bayesian optimization in order to develop a hierarchical optimization method. The resulting framework is presented in Chapter 7, and led to an article published in *Structural and Multidisciplinary Optimization* (see doi:10.1007/s00158-026-04249-2). The framework is implemented in the open-source Python package surrogate modeling toolbox (SMT) [226].

1.4 Plan

This thesis follows an article-based format. The literature review of the thesis is presented in Chapter 2. The four articles composing this thesis are presented respectively in Chapters 4 to 7, and each chapter addresses one of the objectives stated in the previous section. Additional contributions that are not part of a ready-to-publish manuscript are discussed in Chapter 8. Finally, the last chapter concludes the thesis by summarizing the contributions, discussing the limitations, and proposing potential improvements for future work.

CHAPTER 2 LITERATURE REVIEW

But then they sent me away
To teach me how to be sensible
Logical, oh, responsible, practical

— Supertramp, *The Logical Song*

This literature reviews covers blackbox optimization approaches for mixed-variable and hierarchical problems. The main approaches considered in the current literature are direct search, Bayesian optimization (BO), and metaheuristics. Each is presented in its own subsection, first in the simpler continuous setting and then through its extensions to mixed-variable problems. Section 2.1 presents direct search approach and key definitions from nonsmooth optimization. Next, Section 2.2 details the BO approach and focuses on Gaussian processes (GPs) as surrogate models guiding the optimization. Finally, Section 2.3 provides an overview of two commonly used metaheuristics in mixed-variable optimization: genetic algorithms and covariance matrix adaptation evolution strategies.

2.1 Direct search

Unconstrained direct search methods with a strict decrease are iterative algorithms that start from an initial point $x_{(0)} \in \mathbb{R}^n$ and seek candidate points with smaller objective function values. Let $k \in \mathbb{N}$ denote the current iteration and $x_{(k)} \in \mathbb{R}^n$ the incumbent, that is the best known solution at start of iteration k. A candidate point $\boldsymbol{x}$ becomes the incumbent if $f(\boldsymbol{x}) < f(\boldsymbol{x}_{(k)})$.

2.1.1 Coordinate search and Latin hypercube sampling: first methods

The first direct search method is *Coordinate Search* (CS) [102], developed in 1952. CS is not a state-of-the-art method. However, its presentation introduces several key concepts. For clarity, constraints are introduced later in Section 2.1.4.

CS introduces the optimization mechanism called the *poll*. It is a local mechanism that varies each component of the incumbent $\boldsymbol{x}_{(k)}$ by a parameter called the *step size* $\delta_{(k)} \in]0, \infty[$. These variations are performed in the directions $\pm\boldsymbol{e}_i$, where $\boldsymbol{e}_i$ denotes the i-th canonical vector with $i \in \mathbb{R}^n$. The resulting directions are called the *coordinate directions*. The set of

candidate points generated using these coordinate directions is referred to as the *poll set*, or simply the *poll* for short.

For a given iteration $k > 0$, it is expressed as

$$P_{(k)} = \left\{ \boldsymbol{x}_{(k)} \pm \delta_{(k)} \boldsymbol{e}_i \ : \ i \in \{1, 2, \ldots, n\} \right\} \subset \mathbb{R}^n \tag{2.1}$$

The main steps of CS are presented in Algorithm 1.

Algorithm 1: Coordinate search (CS).

0. **Initialization**. Choose an initial point $x_{(0)} \in \mathbb{R}^n$, an initial step size $\delta_{(0)} \in]0, \infty[$, a stopping threshold $\epsilon_{\text{stop}} \in [0, \infty[$, and an evaluation budget $N_{\text{budget}} \in \mathbb{N}$. Set $k = 0$.
1. **Stopping check**. If $\delta_{(k)} < \epsilon_{\text{stop}}$ or $k > N_{\text{budget}}$, then **STOP**
2. **Poll**. Evaluate candidate points in the poll set

$$P_{(k)} = \left\{ \boldsymbol{x}_{(k)} \pm \delta_{(k)} \boldsymbol{e}_i : i \in \{1, 2, \ldots, n\} \right\}$$

3. **Update**.
   If a point $\boldsymbol{x} \in P_{(k)}$ with $f(\boldsymbol{x}) < f(\boldsymbol{x}_{(k)})$ is found (success), set $\boldsymbol{x}_{(k+1)} \leftarrow x$ and $\delta_{(k+1)} \leftarrow \delta_{(k)}$
   Otherwise (unsucessful), set $\boldsymbol{x}_{(k+1)} \leftarrow \boldsymbol{x}_{(k)}$ and $\delta_{(k+1)} \leftarrow \delta_{(k)}/2$
4. **Iteration**. Set $k \leftarrow k + 1$ and **GO TO** Step 1.

The initial step size $\delta_{(0)}$, the stopping criterion ϵ_{stop}, and the evaluation budget N_{budget} are parameters chosen at initialization. The main loop is executed while the step size $\delta_{(k)}$ remains greater than the stopping threshold ϵ_{stop} and the iteration counter k does not exceed the budget N_{budget}. The poll seeks a better candidate point. If it finds a better point, it is *successful* and the step size is unchanged. Otherwise, it is *unsuccessful*, and the step size is reduced by half.

Several implementation details must be specified to execute the CS algorithm. First, two strategies are commonly used when a better solution is found. The first is the complete strategy, which evaluates all candidate points, even if a better point is found, and sets the next incumbent solution such that $\boldsymbol{x}_{(k+1)} \in \operatorname{argmin}\{f(\boldsymbol{x}) \ : \ \boldsymbol{x} \in P_{(k)} \cup \{\boldsymbol{x}_{(k)}\}\}$ The second strategy is called opportunistic. This strategy stops the poll as soon as a better candidate point is found. In practice, the opportunistic strategy is preferred, as it empirically reduces the number of evaluations required for convergence [220]. Next, an initial point $\boldsymbol{x}_{(0)}$ must also be provided. Without prior knowledge of the problem, statistical sampling methods are typically used. Typically, a fraction of the evaluation budget N_{budget} is allocated to determine an initial solution $\boldsymbol{x}_{(0)}$. This phase is commonly referred to as a "Design of Experiment" (DoE). A widely used technique is the Latin Hypercube Sampling (LHS) [174]. An example

of Latin hypercube sampling in 2D is shown in Figure 2.1.

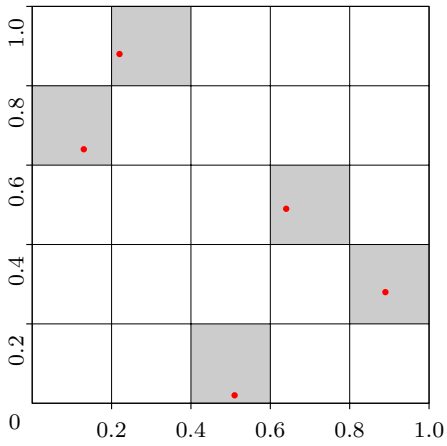


Figure 2.1 Latin hypercube sampling in $[0,1]^2 \subset \mathbb{R}^2$.

In Figure 2.1, each component is divided into 5 equally spaced intervals. Consequently, the two-dimensional space is partitioned into 25 squares. Each row and each column contains exactly one shaded square among the 5 squares. One point is sampled randomly within each shaded square and shown in red in Figure 2.1. Latin hypercube sampling in n dimensions is a direct generalization of this example. In short, $p \in \mathbb{N}$ points are sampled randomly, where each point comes from a hypercube among the p^n hypercubes partitioning the space.

2.1.2 Generalized pattern search: improvements

CS has three major limitations. First, the poll directions are restricted to the coordinate directions. Second, CS can perform poorly when the initial step size $\delta_{(0)}$ is too small. Finally, CS has no exploration mechanism concerning global optimization. Generalized Pattern Search (GPS) [251] was developed to address these limitations. The poll directions of GPS are more flexible than those of CS. Let $\mathbb{D}$ denote a predetermined set of directions. For CS, this set is simply

$$\mathbb{D} = \{ \pm \boldsymbol{e}_i : i \in \{1, 2, \ldots, n\} \}.$$

For GPS, this set of directions must be a positive spanning set, defined next.

Definition 4 (Positive spanning set [94]). *A finite set of vectors $\mathbb{D}$ is said to be a positive spanning set of $\mathbb{R}^n$ if the set of all nonnegative combinations of vectors in $\mathbb{D}$ spans $\mathbb{R}^n$, that is*

$$pspan(\mathbb{D}) = \left\{ \sum_{i=1}^{|\mathbb{D}|} \lambda_i \boldsymbol{d}_i \ : \ \lambda_i \geq 0, \ \boldsymbol{d}_i \in \mathbb{D}, \ \textit{for } i \in \{1, 2, \ldots, |\mathbb{D}|\} \right\} = \mathbb{R}^n. \tag{2.2}$$

Positive spanning sets are related to descent directions. Any open half-space of $\mathbb{R}^n$ contains at least one direction from any positive spanning set. If a function f is differentiable at a point $\boldsymbol{x} \in \mathbb{R}^n$, then there exists a direction $\boldsymbol{d} \in \mathbb{D}$ that is a descent direction of f at $\boldsymbol{x}$. Positive spanning sets can therefore capture descent directions in differentiable cases. GPS strategically uses these sets of directions for polling.

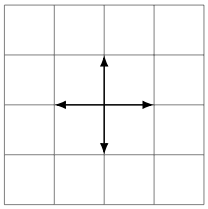
By construction, the candidate points produced by a poll of GPS must lie on the mesh defined below.

Definition 5 (Mesh)**.** *The mesh discretizes $\mathbb{R}^n$ with a step size $\delta_{(k)} \in]0, \infty[$ and a matrix $D \in \mathbb{R}^{n \times p}$ whose columns are directions from a positive spanning set $\mathbb{D}$, such that*

$$M_{(k)} = \left\{ \boldsymbol{x}_{(k)} + \delta_{(k)} D \boldsymbol{y} \ : \ \boldsymbol{y} \in \mathbb{N}^p \right\} \subset \mathbb{R}^n. \tag{2.3}$$

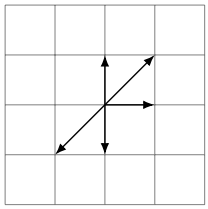
The matrix D must be expressed as $D = GZ$, where $G \in \mathbb{R}^{n \times n}$ is an invertible matrix and $Z \in \mathbb{Z}^{n \times p}$ is a matrix whose columns form a positive spanning set.

In Definition 5, the matrix G determines the form of the discretization, while Z contains the directions of this discretized space. This provides more flexibility than CS: this is illustrated in Figure 2.2. Figure 2.2a corresponds to CS with coordinate directions. Figure 2.2b reproduces the lattice of Figure 2.2a, but includes additional directions. Finally, Figure 2.2c illustrates that the product $D = GZ$ provides flexibility both in the directions and in the discretization of the space, which in this case is based on equilateral triangles.



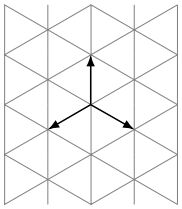
(a) $G = I$,

$$Z = \begin{bmatrix} 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \end{bmatrix}$$



(b) $G = I$,

$$Z = \begin{bmatrix} 1 & 0 & 1 & 0 & -1 \\ 0 & 1 & 1 & -1 & -1 \end{bmatrix}$$



(c) $G = \begin{bmatrix} \sqrt{3} & 0 \\ -1 & 2 \end{bmatrix}$,

$$Z = \begin{bmatrix} 1 & 0 & -1 \\ 0 & 1 & -1 \end{bmatrix}$$

Figure 2.2 Three examples of space discretization with GPS, where $\delta_{(k)} = 1$ (inspired by [26, Chapter 7, p.118]).

A standard choice the mesh is to set $G = I$ and $Z = [I \ -I]$ as in Figure 2.2a. For GPS, the

poll set is generated from a positive spanning set of directions $\mathbb{D}_{(k)}$, such that

$$P_{(k)} = \{\boldsymbol{x}_{(k)} + \delta_{(k)}\boldsymbol{d} \ : \ \boldsymbol{d} \in \mathbb{D}_{(k)}\} \subset M_{(k)},$$

where $\mathbb{D}_{(k)} \subseteq \mathbb{D}$ is a positive spanning set. The poll is performed using a subset of directions $\mathbb{D}_{(k)}$, which are selected from the predetermined set of directions $\mathbb{D}$.

Another important difference between CS and GPS is the management of the step size $\delta_{(k)}$. GPS introduces an additional parameter $\tau \in]0,1[\cap\mathbb{Q}$ that controls the increase and decrease of $\delta_{(k)}$. If a better candidate point is found (success), then the step size is increased according to $\delta_{(k+1)} \leftarrow \tau^{-1}\delta_{(k)}$. Otherwise, the step size is decreased with $\delta_{(k+1)} \leftarrow \tau\delta_{(k)}$. This mechanism can correct an initial step size $\delta_{(0)}$ that is too small. It can also sometimes increase the convergence speed.

Note that if the level set

$$L(\boldsymbol{x}_{(0)}) = \{\boldsymbol{x} \in \mathcal{X} : f(\boldsymbol{x}) \leq f(\boldsymbol{x}_{(0)})\}$$

is bounded, then for any iteration k, the incumbent $\boldsymbol{x}_{(k)}$ remain in a bounded set. In this case, the mesh $M_{(k)}$ discretizing $\mathcal{X}$ contains finitely many points. If the algorithm runs indefinitely, it inevitably exhausts all mesh points that can be evaluated. Consequently, GPS produces only unsuccessful iterations, and the mesh size parameter $\delta_{(k)}$ converges to zero. This is formalized with the following property.

Property 1. *Suppose the level set $L(\boldsymbol{x}_{(0)})$ is bounded, and let U denote the set of indices of the unsuccessful iterations generated by GPS. For any infinite subset $K \subseteq U$, the mesh parameter satisfies*

$$\liminf_{k \in K} \delta_{(k)} = 0. \tag{2.4}$$

On its own, a poll may ignore promising regions outside its reach. Moreover, the points generated by a poll are restricted to the current step size. The third improvement of GPS is the *search step*. This step allows points to be evaluated with greater flexibility. The only restriction of this step is that the selected points must lie on the mesh. It can be used as an exploration mechanism to compensate for the poll, which acts as an *intensification* mechanism that focuses on improving the incumbent locally. Algorithmically, the search step is optional and often improves the overall quality of the solution. There exist many generic and inexpensive search strategies, such as the Nelder–Mead search [34] or Variable Neighborhood Search strategies [18]. More sophisticated strategies can also be used for

hybridization, *e.g.*, BO (see Section 2.2).

2.1.3 Mesh adaptive direct search: state of the art

GPS remains fundamentally limited by the predetermined directions. The MADS algorithm [21], introduced in 2006, generalizes GPS by improving the poll step. In MADS, the directions are generated dynamically at each iteration using generic routines that produce positive spanning sets. The algorithm equips the mesh $M_{(k)}$ with a *frame* $F_{(k)}$ centered around the current solution $x_{(k)}$. The role of the mesh size $\delta_{(k)}$ is split in two. The mesh size $\delta_{(k)}$ still characterizes the mesh $M_{(k)}$, but the *frame size* $\Delta_{(k)}$ controls the length of the polling directions. In MADS, the poll points must lie on the mesh and within the frame.

Definition 6 (Frame)**.** *At a given iteration $k > 0$, the frame centered at the incumbent $\boldsymbol{x}_{(k)}$ is the set*

$$F_{(k)} = \{\boldsymbol{x} \in M_{(k)} : \|\boldsymbol{x} - \boldsymbol{x}_{(k)}\|_\infty \leq \Delta_{(k)}\} \subset M_{(k)}, \tag{2.5}$$

where $\Delta_{(k)}$ is the frame size satisfying $\delta_{(k)} \leq \Delta_{(k)}$.

The poll is finally presented in a definition. The definition is given in the MADS framework, since it is the most general.

Definition 7 (Poll)**.** *For a given incumbent $\boldsymbol{x}_{(k)} \in \mathcal{X}$, the poll set $P_{(k)}$ is defined as*

$$P_{(k)} = \{\boldsymbol{x}_{(k)} + \delta_{(k)}\boldsymbol{d} : \boldsymbol{d} \in \mathbb{D}_{(k)}\} \subset F_{(k)} \subset M_{(k)}, \tag{2.6}$$

where $\mathbb{D}_{(k)}$ is a positive spanning set of directions satisfying $\|d\|_\infty \leq 1$ for any $d \in \mathbb{D}_{(k)}$.

Note that by construction

$$\|\boldsymbol{x}_{(k)} + \delta_{(k)}\boldsymbol{d} - \boldsymbol{x}_{(k)}\|_\infty = \delta_{(k)}\|\boldsymbol{d}\|_\infty \leq \delta_{(k)} \leq \Delta_{(k)}, \tag{2.7}$$

and therefore $P_{(k)} \subset F_{(k)}$. To ensure that $\delta_{(k)} \leq \Delta_{(k)}$, the mesh size can be set as

$$\delta_{(k)} = \min\left\{\Delta_{(k)}, (\Delta_{(k)})^2\right\}. \tag{2.8}$$

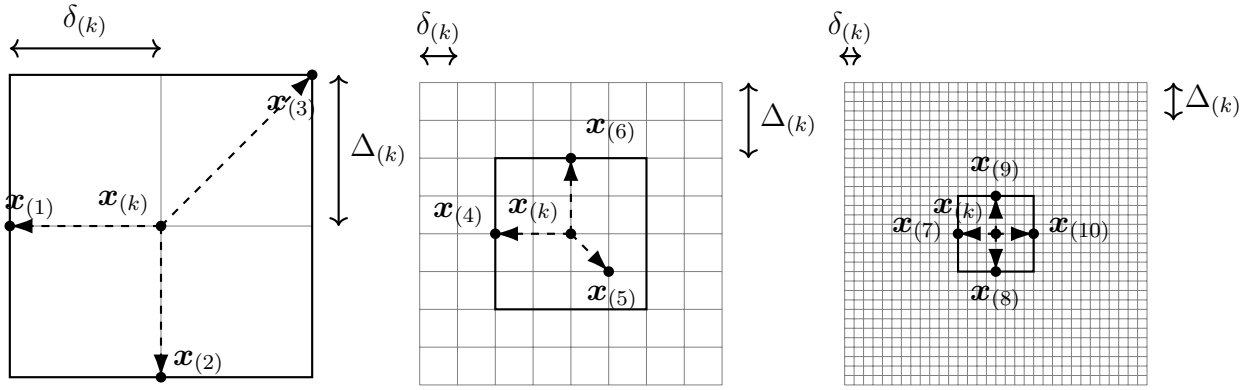
As for CS and GPS, the mesh size is reduced when MADS yields an unsuccessful iteration. Since Property 1 also holds for MADS, the inequality $0 \leq \delta_{(k)} \leq \Delta_{(k)}$ ensures the following property.

Property 2. *Suppose the level set $L(\boldsymbol{x}_{(0)})$ is bounded, and let U denote the set of indices of*

the unsuccessful iterations generated by MADS. *For any infinite subset* $K \subseteq U$,

$$\lim_{k \in K} \delta_{(k)} = 0 \quad \textit{if and only if} \quad \lim_{k \in K} \Delta_{(k)} = 0.$$

As the mesh and frame sizes converge, the mesh size $\delta_{(k)}$ in Equation (2.8) decreases to zero faster than the frame size $\Delta_{(k)}$. This mechanism allows MADS to generate a sequence of directions that becomes increasingly dense as the algorithm progresses. This is illustrated in Figure 2.3.



(a) $\Delta_{(k)} = 1$, $\delta_{(k)} = 1$ (b) $\Delta_{(k)} = 1/2$, $\delta_{(k)} = 1/4$ (c) $\Delta_{(k)} = 1/4$, $\delta_{(k)} = 1/16$

Figure 2.3 Three consecutive unsuccessful polls of MADS at $\boldsymbol{x}_{(k)} \in \mathbb{R}^2$.

In Figure 2.3, the poll generates candidate points with positive spanning sets that are represented by dotted arrows starting from $\boldsymbol{x}_{(k)}$. The candidate points $\boldsymbol{x}_{(1)}, \boldsymbol{x}_{(2)}, \ldots, \boldsymbol{x}_{(10)}$ lie on the mesh (grey grid) and are within the frame (black square). The number of admissible candidates increases as the algorithm progresses. In Figure 2.3a, there are $3 \times 3 = 9$ grid intersections lie in the frame, hence 8 potential candidates. In Figure 2.3b, the mesh shrinks more rapidly than the frame, yielding 25 grid intersections in the frame and therefore 24 admissible points after excluding the center.

Several techniques exist for constructing positive spanning sets, including QRMads [255] and OrthoMADS [5]. The latter generates a set of $2n$ orthogonal directions with the following steps:

1. a normalized vector $\boldsymbol{v} \neq 0 \in \mathbb{R}^n$ is generated randomly;
2. an orthonormal basis of $\mathbb{R}^n$ is constructed using the Householder matrix

$$H = I - 2vv^\top;$$

3. a matrix whose columns form a positive spanning set is constructed as

$$D_{(k)} \approx [H \ -H].$$

The approximation symbol indicates that the columns must be projected (rounded) onto the mesh $M_{(k)}$.

The MADS algorithm for the unconstrained case is presented in Algorithm 2. As for GPS, the points provided by a search step must lie on the mesh $M_{(k)}$ to preserve convergence guarantees. Typically, the mesh and frame parameters are initialized with $\tau = 1/2$ and $\delta_{(0)} = 1 = \Delta_{(0)}$.

Algorithm 2: Unconstrained MADS.

---

0. **Initialization**.
   1. Choose initial mesh and frame sizes $\delta_{(0)}, \Delta_{(0)} \in]0, \infty[$ with $\Delta_{(0)} \geq \delta_{(0)}$.
   2. Choose a rational parameter $\tau \in]0, 1[\cap \mathbb{Q}$.
   3. Choose a stopping threshold $\epsilon_{\text{stop}} \in [0, \infty[$.
   4. Choose an evaluation budget $N_{\text{budget}} \in \mathbb{N}$.
   5. Choose $D = GZ$, where $G \in \mathbb{R}^{n \times n}$ is invertible and $Z \in \mathbb{Z}^{n \times p}$ is a positive spanning matrix.
   6. Choose an initial point $\boldsymbol{x}_{(0)} \in \mathcal{X}$.
   7. Set $k = 0$.

1. **Stopping check**. If $\delta_{(k)} < \epsilon_{\text{stop}}$ or $k > N_{\text{budget}}$, then **STOP**

2. **Search** (optional). Evaluate a finite set of candidate points $S_{(k)} \subset M_{(k)}$
   - If a point $\boldsymbol{x} \in S_{(k)}$ with $f(\boldsymbol{x}) < f(\boldsymbol{x}_{(k)})$ is found, then go to Step 4

3. **Poll**. Construct a positive spanning set within the frame $F_{(k)}$ centered at $\boldsymbol{x}_{(k)}$, yielding candidate points $P_{(k)} \subset F_{(k)} \subset M_{(k)}$. Evaluate points in $P_{(k)}$

4. **Update**.
   - If a point $\boldsymbol{x} \in S_{(k)} \cup P_{(k)}$ with $f(\boldsymbol{x}) < f(\boldsymbol{x}_{(k)})$ is found, set

$$\boldsymbol{x}_{(k+1)} \leftarrow \boldsymbol{x} \qquad \text{and} \qquad \Delta_{(k+1)} \leftarrow \tau^{-1}\Delta_{(k)}.$$

   - otherwise, set

$$\boldsymbol{x}_{(k+1)} \leftarrow \boldsymbol{x}_{(k)} \qquad \text{and} \qquad \Delta_{(k+1)} \leftarrow \tau\Delta_{(k)}.$$

   Then set

$$\delta_{(k+1)} \leftarrow \min\left\{\Delta_{(k+1)}, \left(\Delta_{(k+1)}\right)^2\right\} \qquad \text{and} \qquad k \leftarrow k+1$$

   and **GO TO** Step 1.

2.1.4 Handling constraints with the progressive barrier

An important advantage of MADS is its ability to handle constrained problems efficiently using the progressive barrier [22]. The progressive barrier exploits information from relaxable constraints. A constraint is *relaxable* if it can be unsatisfied while still returning a measure of its infeasibility. It is written as $c_j(\boldsymbol{x}) \leq 0$, where $c_j : \mathcal{X} \to \mathbb{R}$ denotes its left-hand side and J is the set of indices of relaxable constraints. Such a constraint is unsatisfied whenever $c_j(\boldsymbol{x}) > 0$. In contrast, a constraint is *unrelaxable* if the blackbox fails when it is not satisfied. Unrelaxable constraints typically correspond to bound constraints or hidden constraints that cause the blackbox evaluation to fail.

The information provided by relaxable constraints is incorporated through an aggregation function $h : \mathcal{X} \to \mathbb{R}$. This function is inspired by filters in [105] and is defined as follows:

$$h(\boldsymbol{x}) = \begin{cases} \sum\limits_{j \in J} \left(\max\{0, c_j(\boldsymbol{x})\}\right)^2 & \text{if } \boldsymbol{x} \in \mathcal{X}, \\ \infty & \text{otherwise.} \end{cases} \tag{2.9}$$

By definition, the aggregation function satisfies $h(\boldsymbol{x}) \geq 0 \ \forall \boldsymbol{x} \in \mathcal{X}$, and $h(\boldsymbol{x}) = 0 \Leftrightarrow \boldsymbol{x} \in \Omega$ (feasible). A point $\boldsymbol{x}$ does not satisfy at least one unrelaxable constraint if and only if $h(\boldsymbol{x}) = \infty$.

The main idea of MADS with the progressive barrier is to perform two MADS polls: one centered at the *feasible incumbent* $\boldsymbol{x}_{\text{FEA}} \in \Omega$, and another centered at a promising *infeasible incumbent* $\boldsymbol{x}_{\text{INF}} \in \mathcal{X} \setminus \Omega$. For readability, the iteration subscript k is omitted when the subscripts FEA and INF are used. The infeasible incumbent is updated to progressively approach feasibility throughout the iterations. The barrier $h_{(k)} \geq 0$ controls the accepted infeasibility of the infeasible incumbent at iteration k, such that $h(\boldsymbol{x}_{\text{INF}}) \leq h_{(k)}$. The feasible and infeasible incumbents are defined as

$$\boldsymbol{x}_{\text{FEA}} \in \operatorname{argmin}\left\{f(\boldsymbol{x}) \ : \ \boldsymbol{x} \in \Omega \cap \mathbb{X}\right\} \quad \text{and} \quad \boldsymbol{x}_{\text{INF}} \in \operatorname{argmin}\left\{f(\boldsymbol{x}) \ : \ 0 < h(\boldsymbol{x}) \leq h_{(k)}^{\max}, \boldsymbol{x} \in \mathbb{X}\right\}, \tag{2.10}$$

where $\mathbb{X} \subset \mathcal{X}$, called the *set of known points*, is the set of points whose objective and constraint functions have been evaluated since the algorithm was launched. The barrier $h_{(k)}^{\max}$ rejects all candidate points $\boldsymbol{x} \in P_{(k)}^{\inf}$ such that $h(\boldsymbol{x}) > h_{(k)}^{\max}$.

Figure 2.4 illustrates a set of points $\mathbb{X}$ mapped in the (h, f) space with barrier $h_{(k)}^{\max}$. The feasible incumbent $\boldsymbol{x}_{\text{FEA}}$ lies on the vertical axis since $h(\boldsymbol{x}_{\text{FEA}}) = 0$.

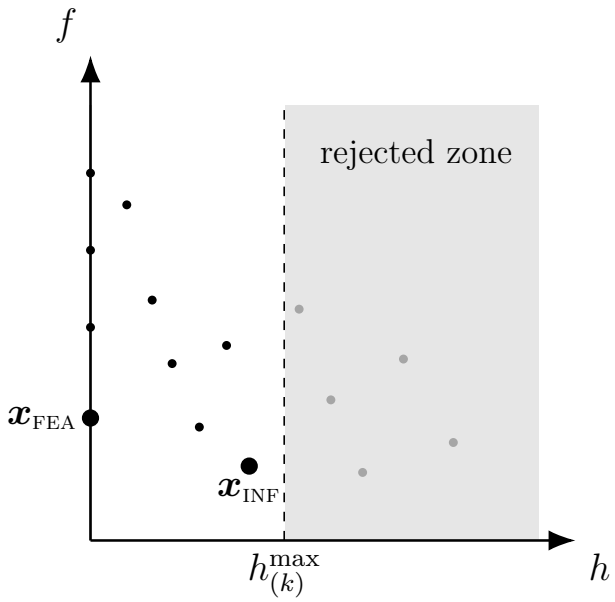
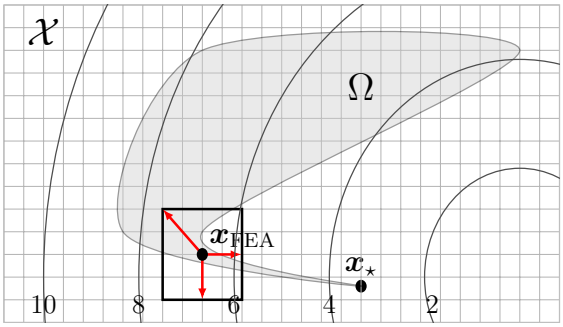
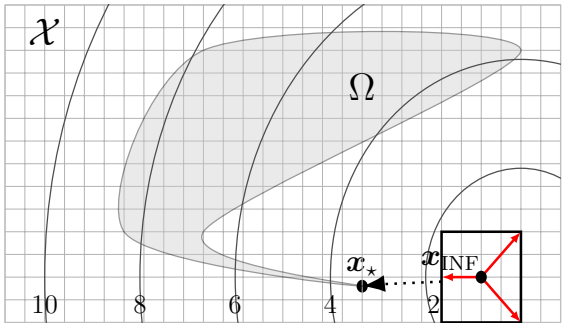


Figure 2.4 Evaluated point in $\mathbb{X}$ with their objective and aggregation values mapped in the (f, h) space.

Again, the feasible poll P_{FEA} and the infeasible poll P_{INF} are performed around the feasible and infeasible incumbents, respectively. The poll is the union of these two sets, such that $P_{(k)} = P_{\text{FEA}} \cup P_{\text{INF}}$. The infeasible poll can identify promising solutions that are difficult to reach when enforcing feasibility at all times. Figure 2.5 compares the feasible and infeasible polls on a two-dimensional constrained problem. The global minimum $\boldsymbol{x}_\star$ is difficult to reach using only feasible polls, which impose that new incumbents remain in Ω.



(a) Feasible poll at $\boldsymbol{x}_{\text{FEA}}$.



(b) Infeasible poll at $\boldsymbol{x}_{\text{INF}}$.

Figure 2.5 A feasible and infeasible polls on a two-dimensional constrained example.

Figure 2.5a illustrates that the feasible poll must pass through a narrow feasible region to reach the minimum $\boldsymbol{x}_\star$. To move along this region, the mesh must be sufficiently fine.

Figure 2.5b shows an infeasible poll with more flexible paths toward the minimum $\boldsymbol{x}_\star$. As the barrier is reduced toward zero, the infeasible incumbent $\boldsymbol{x}_{\text{INF}}$ approaches the feasible region and possibly $\boldsymbol{x}_\star$.

The incumbent updates rely on dominance notions inspired by bi-objective optimization.

Definition 8 (Dominance in constrained optimization)**.**

- *Feasible dominance. Let* $\boldsymbol{x}, \boldsymbol{y} \in \Omega$*. The point* $\boldsymbol{x}$ *dominates* $\boldsymbol{y}$*, noted* $x \prec_f y$*, if*

$$h(\boldsymbol{x}) = 0 = h(\boldsymbol{y}) \quad \textit{and} \quad f(\boldsymbol{x}) < f(\boldsymbol{y}).$$

- *Infeasible dominance. Let* $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X} \setminus \Omega$*. The point* $\boldsymbol{x}$ *dominates* $\boldsymbol{y}$*, noted* $x \prec_h y$*, if*

$$h(\boldsymbol{x}) \leq h(\boldsymbol{y}) \textit{ and } f(\boldsymbol{x}) \leq f(\boldsymbol{y})$$

with at least one strict inequality, i.e.

$$(h(\boldsymbol{x}) \leq h(\boldsymbol{y}) \textit{ and } f(\boldsymbol{x}) < f(\boldsymbol{y})) \quad \textit{or} \quad (h(\boldsymbol{x}) < h(\boldsymbol{y}) \textit{ and } f(\boldsymbol{x}) \leq f(\boldsymbol{y})).$$

A point $\boldsymbol{x} \in P_{\text{FEA}}$ replaces $\boldsymbol{x}_{\text{FEA}}$ if $\boldsymbol{x} \prec_f \boldsymbol{x}_{\text{FEA}}$, and a point $\boldsymbol{x} \in P_{\text{INF}}$ replaces $\boldsymbol{x}_{\text{INF}}$ if $\boldsymbol{x} \prec_h \boldsymbol{x}_{\text{INF}}$.

With MADS and the progressive barrier, an iteration may have three possible outcomes depending on the relation between the evaluated points and the incumbents. Recall that the points produced by the search are contained in $S_{(k)}$.

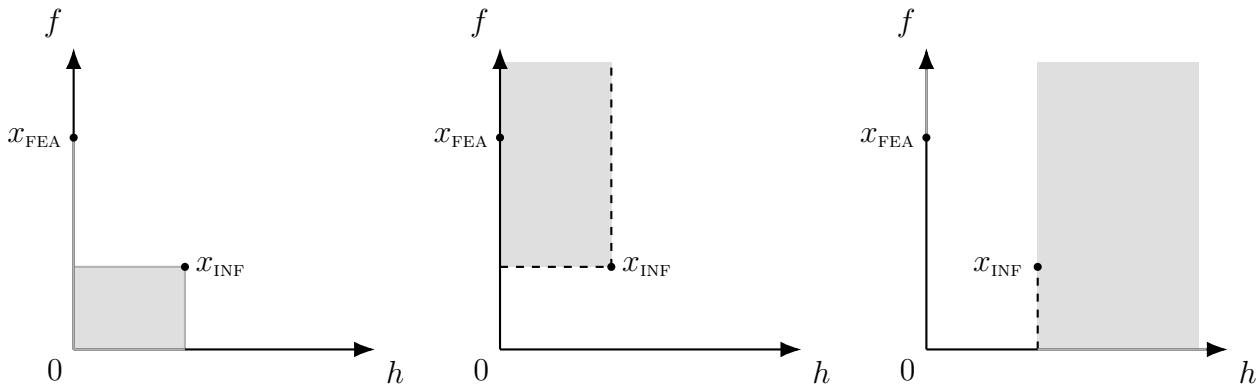
1. A *dominating iteration* (successful) occurs when a point $\boldsymbol{y} \in S_{(k)} \cup P_{(k)}$ dominates either the feasible incumbent $\boldsymbol{x}_{\text{FEA}}$ or the infeasible incumbent $\boldsymbol{x}_{\text{INF}}$. Visually, this corresponds to a point located either below $\boldsymbol{x}_{\text{FEA}}$ on the feasible axis or in the region dominating $\boldsymbol{x}_{\text{INF}}$ in the (h, f) plane, as illustrated in Figure 2.6a. In this case, both the frame size $\Delta_{(k+1)}$ and the mesh size $\delta_{(k+1)}$ are increased, and the barrier threshold is updated to $h^{\max}_{(k+1)} = h(\boldsymbol{x}_{\text{INF}})$.

2. An *improving iteration* (partially successful) occurs when no point dominates the incumbents but a point $\boldsymbol{y} \in S_{(k)} \cup P_{(k)}$ satisfies $0 < h(\boldsymbol{y}) < h(\boldsymbol{x}_{\text{INF}})$. Such a point improves the feasibility of the infeasible incumbent without dominating it, as illustrated in Figure 2.6b. In this case, the barrier threshold is updated according to

$$h^{\max}_{(k+1)} = \max\{h(\boldsymbol{v}) : h(\boldsymbol{v}) < h(\boldsymbol{x}_{\text{INF}}),\ \boldsymbol{v} \in \mathbb{X} \cup S_{(k)} \cup P_{(k)}\}.$$

   The mesh and frame parameters remain unchanged.

3. An iteration is *unsuccessful* when neither of the previous conditions occurs. In that case, candidate points lie either above the feasible incumbent or beyond the barrier threshold, as illustrated in Figure 2.6c. The frame and mesh sizes are reduced, and the barrier threshold is maintained at $h^{\max}_{(k+1)} = h(\boldsymbol{x}_{\text{INF}})$.



(a) Dominating iteration (b) Improving iteration (c) Unsuccessful iteration

Figure 2.6 Three possible outcomes of an iteration with the progressive barrier (inspired by [26]).

Algorithmically, extending MADS to the constrained case with the progressive barrier amounts to performing two polls, introducing the barrier parameter $h^{\max}_{(k)}$, and modifying the update rules.

2.1.5 Overview of theoretical results

This section presents the main theoretical results of MADS. These results are later generalized and extended to the mixed-variable setting in Chapter 5, which presents the second article. These results are fundamental to this thesis.

The following definition is a building block of the convergence analysis of MADS.

Definition 9 (Refined subsequence and refined point)**.** *Let U be the set of indices of the unsuccessful iterations generated by an instance of* CatMADS*. If K is an infinite subset of U such that*

$$\begin{cases} \lim_{k\in K} \boldsymbol{x}_{(k)} = \boldsymbol{x}_\star & \text{for some } \boldsymbol{x}_\star \in \mathcal{X}, \\ \lim_{k\in K} \delta_{(k)} = 0 \end{cases}$$

then the subsequence $\{\boldsymbol{x}_{(k)}\}_{k\in K}$ is called a refining subsequence*, and $\boldsymbol{x}_\star \in \mathcal{X}$ is called a* refined point*.*

The existence of a refining subsequence and a corresponding refined point is ensured under the following assumptions:

1. an initial point $\boldsymbol{x}_{(0)} \in \mathcal{X}$ with $f(\boldsymbol{x}_{(0)}) \in \mathbb{R}$ is known (A1);
2. the level set $L_{(\boldsymbol{x}_{(0)})}$ is bounded (A2).

This leads to a zeroth-order convergence result that does not involve derivatives.

Theorem 1 (Zeroth-order MADS)**.** *Let* $\{\boldsymbol{x}_{(k)}\}$ *be the sequence of points generated by* MADS *from an initial point* $\boldsymbol{x}_{(0)} \in \mathcal{X}$ *with* $f\left(\boldsymbol{x}_{(0)}\right) \in \mathbb{R}$ (A1)*. If the level set* $L_{(\boldsymbol{x}_{(0)})}$ *is bounded* (A2)*, then there exists a refining subsequence* $\{\boldsymbol{x}_{(k)}\}_{k \in K}$ *with corresponding refined point* $\boldsymbol{x}_\star \in \mathcal{X}$*.*

As the results are presented, stronger assumptions are introduced and the guarantees on the refined point $\boldsymbol{x}_\star$ are strengthened. Before stating further theoretical results, several definitions from nonsmooth optimization are required. Since the problems of interest are constrained, particular attention is given to directions that point toward feasibility. In nonsmooth optimization, this notion is formalized by the following definition.

Definition 10 (Hypertangent cone [214])**.** *A vector* $\boldsymbol{v} \in \mathbb{R}^{n^{\text{con}}}$ *is said to be a* hypertangent vector *to the set* Ω *at a point* $\boldsymbol{x}_\star \in \Omega$*, if there exists a scalar* $\epsilon > 0$ *such that*

$$\boldsymbol{w} + t\boldsymbol{u} \in \Omega \quad \textit{for all} \quad \boldsymbol{w} \in \mathcal{B}\left(\boldsymbol{x}_\star; \epsilon\right) \cap \ \Omega, \ \ \boldsymbol{u} \in \mathcal{B}\left(\boldsymbol{v}; \epsilon\right) \quad \textit{and} \quad t \in (0, \epsilon).$$

The hypertangent cone $T_\Omega^H(\boldsymbol{x}_\star)$ *to* Ω *at* $\boldsymbol{x}_\star \in \Omega$ *is the set of all hypertangent vectors to* Ω *at* $\boldsymbol{x}_\star$*.*

Figure 2.7 illustrates a hypertangent cone containing directions that locally point toward the feasible region. Generalized directional derivatives along these directions are used in the convergence analysis. These derivatives are defined for Lipschitz continuous functions.

Definition 11 (Lipschitz continuity)**.** *A function* $f : \mathcal{X} \to \mathbb{R}$ *is* Lipschitz continuous *on* $\mathcal{X}$ *if there exists a constant* $\gamma > 0$ *such that, for all* $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X} \subseteq \mathbb{R}^n$*,*

$$|f(\boldsymbol{x}) - f(\boldsymbol{y})| \leq \gamma \|\boldsymbol{x} - \boldsymbol{y}\|.$$

The constant γ *is called a* Lipschitz constant *of* f*.*

Lipschitz continuity is typically not assumed on the whole domain $\mathcal{X}$, since it is only needed locally around points of interest, such as a refined point. In that case, Lipschitz continuity is assumed in a neighborhood of a point $\boldsymbol{x}$ of interest. This assumption is required for the generalized directional derivative defined next.

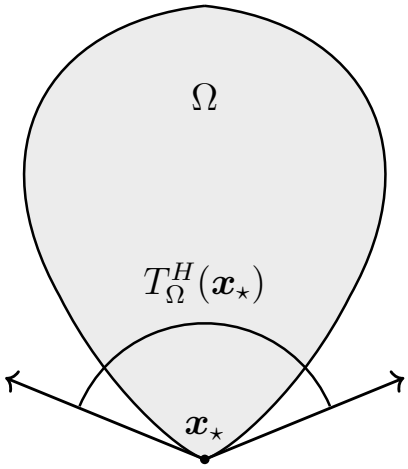


Figure 2.7 Example of a hypertangent cone $T_\Omega^H(\boldsymbol{x}_\star)$ at $\boldsymbol{x}_\star \in \Omega \subset \mathbb{R}^2$.

Definition 12 (Generalized Clarke derivative [82, 137])**.** *Let $f : \mathcal{X} \to \mathbb{R}$ be a Lipschitz continuous function near $x \in \mathcal{X}$. The* generalized directional derivative *of f at $\boldsymbol{x}$ in the direction $\boldsymbol{d} \in \mathbb{R}^n$ is defined by*

$$f^\circ(\boldsymbol{x}; \boldsymbol{d}) \coloneqq \limsup_{\substack{\boldsymbol{y} \to \boldsymbol{x} \\ t \searrow 0}} \frac{f(\boldsymbol{y} + t\boldsymbol{d}) - f(\boldsymbol{y})}{t}. \tag{2.11}$$

Finally, the main convergence result of MADS is presented. It relies on generalized Clarke derivatives at a refined point $\boldsymbol{x}_\star$. The result focuses on the so-called refining directions, which correspond to directions generated by poll steps along a refining subsequence [20, 21].

Theorem 2 (Convergence of MADS)**.** *Let $\{\boldsymbol{x}_{(k)}\}_{k \in K}$ be a refining subsequence with corresponding refined point $\boldsymbol{x}_\star$ produced by* MADS *with the PB. Suppose that the set of refining directions is dense in the unit sphere $\{\boldsymbol{u} \in \mathbb{R}^n : \|\boldsymbol{u}\| = 1\}$.*

i) If $\boldsymbol{x}_\star \in \Omega$ and f is Lipschitz continuous near $\boldsymbol{x}_\star$, then $f^\circ(\boldsymbol{x}_\star; \boldsymbol{d}) \geq 0$ for all $\boldsymbol{d} \in T_\Omega^H(\boldsymbol{x}_\star)$.

ii) If $\boldsymbol{x}_\star \in \mathcal{X} \backslash \Omega$ and h is Lipschitz continuous near $\boldsymbol{x}_\star$, then $h^\circ(\boldsymbol{x}_\star; \boldsymbol{d}) \geq 0$ for all $\boldsymbol{d} \in T_\mathcal{X}^H(\boldsymbol{x}_\star)$.

The result is divided into two cases depending on whether the refined point is feasible or not. If it is feasible, the first condition states that the generalized Clarke directional derivative of the objective function is nonnegative in every hypertangent direction of the feasible region. This corresponds to a first-order necessary optimality condition with respect to the feasible directions of Ω. Otherwise, when the refined point is infeasible, a similar condition holds

for the constraint aggregation function h. This theoretical result is a key property that distinguishes MADS from many other blackbox optimization methods. It establishes a first-order necessary optimality condition at refined points, whereas most derivative-free methods in the literature provide no such guarantee.

2.1.6 Extensions for mixed-variable optimization

This section presents extensions of the MADS algorithm for mixed-variable problems, along with closely related works.

In [33], MADS is equipped with a granular mesh called GMesh. This mesh simultaneously discretizes continuous and granular variables. Granular variables are quantitative variables whose number of decimal places is controlled. In particular, integer variables are granular variables with no decimal places. Thus, MADS with GMesh allows the treatment of constrained mixed-variable problems with continuous and integer variables. The GMesh is a generalization of the mesh Definition 5, where each variable is assigned its own step and frame sizes to account for variable types and scales. The GMesh is defined as

$$M_{(k)} = \left\{ \boldsymbol{x}_{(k)} + \operatorname{diag}\left(\boldsymbol{\delta}_{(k)}\right) \boldsymbol{z} \ : \ \boldsymbol{z} \in \mathbb{Z}^n \right\}, \tag{2.12}$$

where $\boldsymbol{\delta}_{(k)} \in \mathbb{R}^n$ is the vector of step sizes, $\delta_{(k),i}$ is the step size associated with the i-th variable x_i, and $\operatorname{diag}\left(\boldsymbol{\delta}_{(k)}\right)$ is a diagonal matrix.

The GMesh discretization uses the matrix $D = [I \ -I]$, which generates the grids illustrated in Figures Figure 2.2a and Figure 2.3. The poll set for MADS with GMesh is then defined as

$$P_{(k)} = \left\{ \boldsymbol{x}_{(k)} + \operatorname{diag}\left(\boldsymbol{\delta}_{(k)}\right) \boldsymbol{d} \ : \ \boldsymbol{d} \in \mathbb{D}_{(k)} \right\} \subset F_{(k)} \subset M_{(k)}, \tag{2.13}$$

where $\mathbb{D}_{(k)} \subset \mathbb{Z}^n$ is a positive spanning set satisfying $-\boldsymbol{\Delta}_{(k)} \leq \operatorname{diag}\left(\boldsymbol{\delta}_{(k)}\right) \boldsymbol{d} \leq \boldsymbol{\Delta}_{(k)}$ for all $\boldsymbol{d} \in \mathbb{D}_{(k)}$, and $\boldsymbol{\Delta}_{(k)} \in \mathbb{R}^n$ is the vector of poll sizes. The vector $\boldsymbol{\Delta}_{(k)}$ generalizes the frame size of MADS. The frame sizes of an integer and continuous variables respectively take values in the sets $\Delta^{\text{int}}_{(k),i} \in \{a \times 10^b \ : \ a \in \{1, 2, 5\}, \ b \in \mathbb{N}\}$, with $\Delta^{\text{int}}_{(k),i} \geq 1$, for $i \in I^{\text{int}}$, and $\Delta^{\text{con}}_{(k),j} \in \{a \times 10^b \ : \ a \in \{1, 2, 5\}, \ b \in \mathbb{Z}\}$ for $j \in I^{\text{con}}$. The frame and step sizes are decreased after unsuccessful iterations and increased after successful ones. Detailed update rules are given in [33].

A first attempt to optimize mixed blackbox problems with discrete variables was proposed in [19]. The methodology is based on GPS, where a point $\boldsymbol{x} = (\boldsymbol{x}^{\text{dis}}, \boldsymbol{x}^{\text{con}})$ is partitioned into two components: a discrete component $\boldsymbol{x}^{\text{dis}}$, containing the discrete variables, and a

continuous component $\boldsymbol{x}^{\text{con}}$, containing the continuous variables. The discrete component includes both integer and categorical variables. This work introduces two main contributions. First, the continuous variables are optimized using GPS polls performed with a fixed discrete component $\boldsymbol{x}^{\text{dis}}$. Second, the discrete variables are handled using an additional structure called *user-defined neighborhoods*, denoted $\mathcal{N}$. For a given point $\boldsymbol{x} = (\boldsymbol{x}^{\text{dis}}, \boldsymbol{x}^{\text{con}})$, $\mathcal{N}(\boldsymbol{x})$ is a set of neighbors defined by the user. Given an incumbent $\boldsymbol{x}_{(k)}$, the poll is defined as the union of a user-defined neighborhood and the standard GPS poll, *i.e.*, $\mathcal{N}(\boldsymbol{x}_{(k)}) \cup P_{(k)}$.

Several extensions of the methodology in [19] have been proposed. In the thesis [2], concepts from mathematical analysis, including convergent series and the continuity of the additional structure $\mathcal{N}$, are developed rigorously. A thermal insulation optimization problem is addressed and optimized in [155] and a nonlinearly constrained version is studied in [3]. Moreover, the mixed-variable GPS was generalized to a mixed-variable MADS, called MV-MADS, in [4]. As discussed in the introduction, the first project CatMADS detailed in 4 has a strong overlap with MV-MADS, which also extends MADS to mixed-variable optimization through user-defined neighborhoods. However, unlike MV-MADS, CatMADS automatically constructs categorical neighborhoods for the problem at hand, and separates the treatment of integer and categorical variables. This leads to a substantially different convergence analysis, as well as a complete default implementation and numerical experiments. This last work is discussed in detail in the literature review of the second and third article in Chapters 4 and 5.

Dimensional variables were introduced in [166, 167]. These special discrete variables affect the number of variables or constraints. They are a special case of meta variables, discussed in the introduction. Indeed, they are strictly discrete, specifically control the number of variables or constraints, and are limited to a single hierarchical level [25]. A point $\boldsymbol{x} = (\boldsymbol{x}^{\text{dim}}, \boldsymbol{x}^{\text{dis}}, \boldsymbol{x}^{\text{con}})$ is partitioned into three components: the dimensional component $\boldsymbol{x}^{\text{dim}}$, the discrete component $\boldsymbol{x}^{\text{dis}}$, and the continuous component $\boldsymbol{x}^{\text{con}}$. In [19], dimensional variables are implicitly included in the discrete component. Dimensional variables model situations where the number of variables and constraints may vary depending on their values. Similarly to [19], continuous spaces are generated by fixing both the dimensional component $\boldsymbol{x}^{\text{dim}}$ and the discrete component $\boldsymbol{x}^{\text{dis}}$. The optimization process is then carried out in this continuous space with a line-search-based algorithm.

2.2 Bayesian optimization: a surrogate-based approach

In blackbox optimization, a surrogate is a less expensive function that shares similarities with the objective function or with the constraints of the problem [26]. The terms *models* and *surrogates* are sometimes used interchangeably. In this thesis, models and surrogates have

distinct purposes, following the convention in [26].

A model $m : \mathcal{X} \to \mathbb{R}$ is typically constructed to minimize a prediction error. For example, for the objective function f, a model is built such that $f(\boldsymbol{x}) \approx m(\boldsymbol{x})$ for $\boldsymbol{x} \in \mathcal{X}$. In contrast, a surrogate $\tilde{f}(\boldsymbol{x})$ for an objective function focuses on preserving local minima. In theory, a perfect surrogate $\tilde{f}$ perfectly preserves ordering, *i.e.*,

$$f(\boldsymbol{x}) < f(\boldsymbol{y}) \Leftrightarrow \tilde{f}(\boldsymbol{x}) < \tilde{f}(\boldsymbol{y}) \quad \forall (\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{X} \times \mathcal{X}. \tag{2.14}$$

A surrogate $\tilde{c}_j(\boldsymbol{x}) \leq 0$ of a constraint $c_j(\boldsymbol{x}) \leq 0$ ideally preserves its sign [29], such that

$$c_j(\boldsymbol{x}) \leq 0 \Leftrightarrow \tilde{c}_j(\boldsymbol{x}) \leq 0 \quad \forall \boldsymbol{x} \in \mathcal{X}. \tag{2.15}$$

Note that Equations (2.14) and (2.15) represent ideals scenario and that it can not be verified in a blackbox setting. These equations however outlines that the goals of surrogate is used to guide the optimization process without necessarily being an accurate approximation of its corresponding function.

BO is a surrogate-based approach that typically relies on probabilistic surrogate models. A widely used model, especially for mixed-variable problems [194, 201, 216, 224], are the GPs. Other surrogate models can also be used in BO, such as decision trees [122] and random forests [153]. This thesis focuses on BO with GPs, which is introduced in the next section.

2.2.1 Gaussian processes

A GP is a generalization of a multivariate normal distribution to a distribution over a space of functions. In practice, GPs are particularly useful for regression [206]. For the sake of the presentation, GPs are presentend first for the objective function. The constraint functions are added later.

For regression, GPs are used to infer the most likely realization of an underlying stochastic process given observed data, and to predict its values at new points. This setting is sometimes referred to as GP regression. The data is noted $(\mathbb{X}, f(\mathbb{X}))$, where $\mathbb{X}$ is the set of evaluated points and $f(\mathbb{X})$ is the vector of objective values corresponding to the evaluated points $\mathbb{X}$. A GP is said to be probabilistic since the prediction (mean) $\hat{f}(\boldsymbol{x})$ is associated with a measure of uncertainty (standard deviation) $\hat{\sigma}(\boldsymbol{x})$, both provided by the surrogate $\tilde{f}$:

$$\tilde{f}(\boldsymbol{x}) \mid \{(\mathbb{X}, f(\mathbb{X})), \boldsymbol{x}\} \sim N\left(\hat{f}(\boldsymbol{x}), \hat{\sigma}^2(\boldsymbol{x})\right) \tag{2.16}$$

where $\boldsymbol{x} \in \mathcal{X}$ is an unknown point and $N(\cdot, \cdot)$ denotes a normal distribution. Before detailing the prediction $\hat{f}(\boldsymbol{x})$ and the uncertainty $\hat{\sigma}(\boldsymbol{x})$, it is necessary to introduce the kernel that characterizes GPs.

Definition 13 (Kernel)**.** *A kernel $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a similarity measure satisfying the two following properties:*

1. *$\kappa(\boldsymbol{x}, \boldsymbol{y}) = \kappa(\boldsymbol{y}, \boldsymbol{x})$ for any $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$ (symmetric), and*
2. *for any finite set of points $\{\boldsymbol{x}_{(1)}, \boldsymbol{x}_{(2)}, \dots \boldsymbol{x}_{(p)}\}$ with $p \in \mathbb{N}$, the symmetric $p \times p$ matrix $[K]_{i,j} = \kappa\left(\boldsymbol{x}_{(i)}, \boldsymbol{x}_{(j)}\right)$ for $i, j \in \{1, 2, \dots, p\}$ is positive semi-definite.*

Such kernel κ is said to be positive semi-definite.

The kernel κ determines the properties of the GP. For example, a periodic kernel κ induces a periodic regression. From a Bayesian perspective, the kernel κ can be interpreted as prior information. For presentation purposes, the data is assumed to be normalized. Consequently, the prediction $\hat{f}(\boldsymbol{x})$ and the uncertainty $\hat{\sigma}(\boldsymbol{x})$ are entirely characterized by the kernel κ [206]:

$$\begin{cases} \hat{f}(\boldsymbol{x}) = \boldsymbol{\kappa}_{\mathbb{X}}^{\top} \left(K + \sigma_n^2 I\right)^{-1} f(\mathbb{X}) \\ \hat{\sigma}^2(\boldsymbol{x}) = \kappa(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{\kappa}_{\mathbb{X}}^{\top} (K + \sigma_n^2 I)^{-1} \boldsymbol{\kappa}_{\mathbb{X}} \end{cases} \tag{2.17}$$

where

- $\boldsymbol{\kappa}_{\mathbb{X}} = \left(\kappa\left(\boldsymbol{y}_{(1)}, \boldsymbol{x}\right), \kappa\left(\boldsymbol{y}_{(2)}, \boldsymbol{x}\right), \dots, \kappa\left(\boldsymbol{y}_{(|\mathbb{X}|)}, \boldsymbol{x}\right)\right) \in \mathbb{R}^{|\mathbb{X}|}$ is the column vector containing the similarity measures between the evaluated points in $\mathbb{X}$ and the input point $\boldsymbol{x} \in \mathcal{X}$;
- $K \in \mathbb{R}^{|\mathbb{X}| \times |\mathbb{X}|}$ is a positive semi-definite matrix containing all similarity measures between each pair of points in $\mathbb{X}$;
- $\kappa(\boldsymbol{x}, \boldsymbol{x}) \in \mathbb{R}$ is the autocorrelation of the point $\boldsymbol{x} \in \mathcal{X}$;
- $\sigma_n^2 \in \mathbb{R}^+$ is the overall variance of the GP.

Equation (2.17) are obtained by assuming that the vector $\left(f(\mathbb{X}), \hat{f}(\boldsymbol{x})\right)$ follows a conditional multivariate normal distribution, such that

$$\left(f(\mathbb{X}), \hat{f}(\boldsymbol{x})\right) \mid \{\mathbb{X}, \boldsymbol{x}\} \sim N(0, \Sigma) \tag{2.18}$$

where the values $\left(f(\mathbb{X}), \hat{f}(\boldsymbol{x})\right)$ are normalized and Σ is a block covariance matrix given by

$$\Sigma = \begin{bmatrix} K + \sigma_n^2 I & \boldsymbol{\kappa}_{\mathbb{X}} \\ \boldsymbol{\kappa}_{\mathbb{X}}^\top & \kappa(\boldsymbol{x}, \boldsymbol{x}) \end{bmatrix} \in \mathbb{R}^{(|\mathbb{X}|+1)\times(|\mathbb{X}|+1)}. \tag{2.19}$$

The conditional multivariate normal distribution in Equation (2.18) is then further conditioned by $f(\mathbb{X})$ (known values) to determine the posterior distribution $\tilde{f}(\boldsymbol{x}) \mid \{(\mathbb{X}, f(\mathbb{X})), \boldsymbol{x}\}$ expressed in Equation (2.16).

This last conditioning step involves a Schur complement of the covariance matrix in Equation (2.19). This leads to the expressions in Equation (2.17). The equations (2.17) arise from linear algebra operations, yet they still allow the fitting of nonlinear functions over the domain $\mathcal{X}$. Indeed, Figure 2.8 illustrates a GP modeling the function $f(x) = x\sin(x)$ for $x \in [0, 10]$.

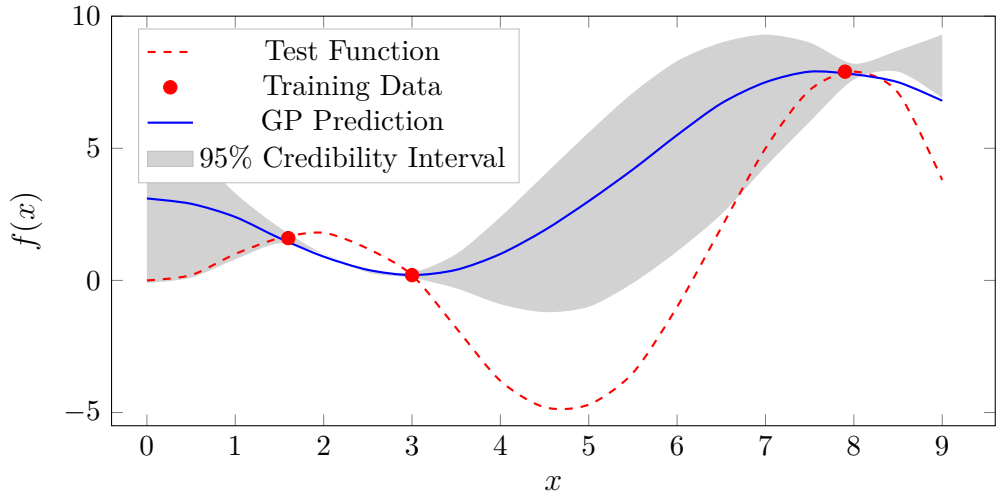


Figure 2.8 GP for the function $f(x) = x\sin(x),\ x \in [0, 10]$.

In Figure 2.8, each point $x \in [0, 10]$ in the domain is associated with a prediction $\hat{f}(x)$ shown in blue and an uncertainty $\hat{\sigma}(x)$ shown in gray. The values of $\hat{f}(x)$ and $\hat{\sigma}(x)$ must be de-normalized to return to the original scale.

This flexibility in fitting the data, illustrated in Figure 2.8, stems from the kernel κ. More precisely, GP regression is a method that uses the kernel trick. The basic idea of this trick is to address nonlinear problems by applying linear methods that are implicitly carried out in a Hilbert space H through a mapping $\phi : \mathcal{X} \to H$. Let $\phi : \mathcal{X} \to H$ be a mapping and let H be a Hilbert space equipped with an inner product $\langle \cdot, \cdot \rangle_H$. The kernel trick states that a

kernel is equivalent to a scalar product induced by ϕ, that is,

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) = \langle \phi(\boldsymbol{x}), \phi(\boldsymbol{y}) \rangle_H \in \mathbb{R}. \tag{2.20}$$

The formulation of a kernel in Equation (2.20) is trivially symmetric. It is also positive semi-definite, since for $\boldsymbol{x}_{(1)}, \boldsymbol{x}_{(2)}, \ldots, \boldsymbol{x}_{(|\mathbb{X}|)} \in \mathbb{X} \cap \mathcal{X}$ (points) and $\alpha = (\alpha_1, \alpha_2, \ldots, \alpha_{|\mathbb{X}|}) \in \mathbb{R}^{|\mathbb{X}|}$ (a vector),

$$\begin{aligned}
\alpha^\top K \alpha = \sum_{i=1}^{|\mathbb{X}|} \sum_{j=1}^{|\mathbb{X}|} \alpha_i \alpha_j \kappa(\boldsymbol{x}_{(i)}, \boldsymbol{x}_{(j)}) &= \sum_{i=1}^{|\mathbb{X}|} \sum_{j=1}^{|\mathbb{X}|} \alpha_i \alpha_j \left\langle \phi(\boldsymbol{x}_{(i)}), \phi(\boldsymbol{x}_{(j)}) \right\rangle_H \\
&= \left\langle \sum_{i=1}^{|\mathbb{X}|} \alpha_i \phi(\boldsymbol{x}_{(i)}), \sum_{j=1}^{|\mathbb{X}|} \alpha_j \phi(\boldsymbol{x}_{(j)}) \right\rangle_H \\
&= \left\| \sum_{i=1}^{|\mathbb{X}|} \alpha_i \phi(\boldsymbol{x}_{(i)}) \right\|^2 \geq 0,
\end{aligned}$$

where K is the kernel matrix defined in Equation (2.19).

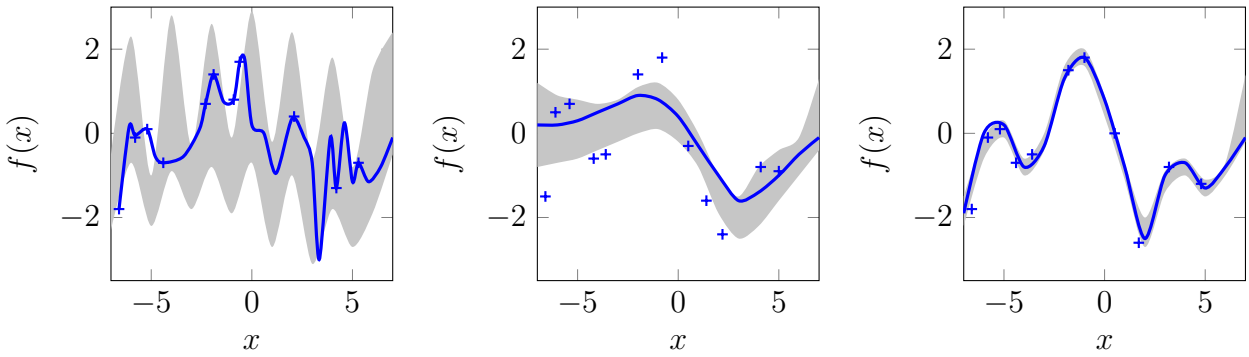
For example, in Figure 2.8, the following one-dimensional Gaussian kernel is used:

$$\kappa(x, y) = \gamma^2 \exp\left(-\frac{|x - y|^2}{l^2} \right), \tag{2.21}$$

where γ controls the variability of the process and l is a hyperparameter controlling the smoothness of the regression. For such a kernel κ, computing $\kappa(\boldsymbol{x}, \boldsymbol{y})$ is equivalent to computing the inner product $\langle \phi(\boldsymbol{x}), \phi(\boldsymbol{y}) \rangle_H$ in an infinite-dimensional space. An inner product $\langle \phi(\boldsymbol{x}), \phi(\boldsymbol{y}) \rangle_H$ in such a space provides a rich similarity measure, but it would be impractical to compute explicitly. However, the quantity $\kappa(\boldsymbol{x}, \boldsymbol{y})$ can be evaluated directly without explicitly constructing the mapping $\phi : \mathcal{X} \to H$. More precisely, the kernel trick consists of applying linear methods that only require inner products in the feature space to solve nonlinear problems. These potentially infinite-dimensional inner products are replaced by evaluations of the kernel κ, so that the mapping ϕ is never explicitly computed.

The kernel is adapted to the data through its hyperparameters, meaning they allow the kernel to become problem-specific. For the kernel in Equation (2.21), the hyperparameter l informally represents the maximum distance over which two points $x, y \in \mathcal{X}$ influence each other, *i.e.*, $|x - y| < l$ indicates that the values $\tilde{f}(x)$ and $\tilde{f}(y)$ are correlated. Figure 2.9 illustrates a one-dimensional GP regression using a Gaussian kernel for different values of the hyperparameter $l \in \{0.3, 1, 3\}$.

In Figure 2.9, smaller values of l produce a more flexible regression, whereas larger values



(a) $\ell = 0.3$ (b) $\ell = 3$ (c) $\ell = 1$

Figure 2.9 GP regression with different kernel length scales (adapted from [206]).

are associated with a smoother regression, whereas Visually, the best regression balancing flexibility and smoothness is obtained with $l = 1$. Figure 2.9 illustrates that the behavior of the regression can be strongly affected by the choice of hyperparameters. These hyperparameters can be adjusted by maxiziming the log marginal likelihood of the observations defined by:

$$\log p\left(f(\mathbb{X}) \mid \mathbb{X}, \theta\right) = -\frac{1}{2} f(\mathbb{X})^{\top} \left(K + \sigma_n^2 I\right)^{-1} f(\mathbb{X}) - \frac{1}{2} \log\left(\det\left(K + \sigma_n^2 I\right)\right) - \frac{|\mathbb{X}|}{2} \log 2\pi, \tag{2.22}$$

where σ_n^2 is the variance of the GP and θ is the vector containing the hyperparameters (including σ_n^2). The hyperparameters associated with the kernel κ are implicitly contained in the kernel matrix K.

Computing the log marginal likelihood in (2.22) involves inverting the matrix $K + \sigma_n^2 I$. The computation of the log marginal likelihood therefore has a complexity of $\mathcal{O}(n^3)$. The matrix $K + \sigma_n^2 I$ is symmetric, which allows the use of a Cholesky decomposition to reduce the computational cost of this operation. Finally, the hyperparameters that best fit the data $(\mathbb{X}, f(\mathbb{X}))$ are determined by maximizing the log marginal likelihood defined in (2.22), such that

$$\theta^* \in \underset{\theta}{\operatorname{argmax}} \ \log p\left(f(\mathbb{X}) \mid \mathbb{X}, \theta\right) \tag{2.23}$$

The optimization of the log marginal likelihood presents several difficulties, including an unstable objective function with multiple local minima. In this context, the GP variance σ_n^2 is a hyperparameter that improves the numerical stability of the inversion of the matrix K, i.e., $K + \sigma_n^2 I$ [206]. Problem (2.23) is typically solved using an L-BFGS (quasi-Newton)

method with multistart [180]. In other words, the L-BFGS procedure is initialized from several starting points. This multistart strategy is commonly performed using a Latin hypercube sampling (LHS) design [180]. In addition, the partial derivatives with respect to each hyperparameter can be obtained using the following expression [206]

$$\frac{\partial}{\partial \theta_j} \log p(f(\mathbb{X}) \mid \mathbb{X}, \theta) = \frac{1}{2} f(\mathbb{X})^\top K^{-1} \frac{\partial K}{\partial \theta_j} K^{-1} f(\mathbb{X}) - \frac{1}{2} \mathrm{tr}\left(K^{-1} \frac{\partial K}{\partial \theta_j} \right).$$

So far, GPR has been presented with 1D figures. This is because the kernel Equation (2.21) used is 1D. This formulation can be generalized to n continuous variables by adapting the kernel itself. A valid kernel must be symmetric and positive semi-definite. Several operations preserve these properties, including addition, multiplication, and tensor products [216]. These operations are commonly used to construct multidimensionnal kernels from one-dimensional ones. For example, the anisotropic Gaussian kernel can be constructed as the product of n one-dimensional Gaussian kernels:

$$\kappa(\boldsymbol{x}, \boldsymbol{y}) = \prod_{i=1}^{n} \kappa(x_i, y_i) = \prod_{i=1}^{n} \gamma_i^2 \exp\left(-\frac{(x_i - y_i)^2}{l_i^2} \right) = \left(\prod_{i=1}^{n} \gamma_i^2 \right) \exp\left(-\sum_{i=1}^{n} \frac{(x_i - y_i)^2}{l_i^2} \right), \tag{2.24}$$

where $\boldsymbol{x} = (x_1, x_2, \ldots, x_n) \in \mathbb{R}^n$, and γ_i and l_i for $i \in \{1, 2, \ldots, n\}$ are the hyperparameters of the GP. The i-th pair of hyperparameters (γ_i, l_i) is associated with the i-th one-dimensional Gaussian kernel defined in (2.21).

2.2.2 Bayesian optimization framework

Now that GPs have been described, BO can be introduced. Recall that BO is a probabilistic surrogate-based approach, and in this thesis surrogates used are GPs.

In principle, BO relies on the trade-off between intensification and exploration. In unconstrained BO, exploration of an objective function consists of evaluating a point in a sparsely sampled region, that is, regions of the domain where the uncertainty of the GP is high. In contrast, intensification consists of evaluating points in promising regions, namely regions where the predictions of the regression appear to be optimal. The next point to evaluate is selected using an acquisition function. The value returned by this function is based on the trade-off between intensification and exploration.

There are many acquisition functions, mostly from the literature in machine learning. In blackbox optimization, a commonly used acquisition function is the *Expected Improvement*

(EI) [145]. When the surrogate is a GP, the EI is expressed as

$$\mathrm{EI}\left(\boldsymbol{x};\tilde{f}\right) = \underbrace{\left(f_{(k)} - \hat{f}(\boldsymbol{x})\right)\Phi\left(\frac{f_{(k)} - \hat{f}(\boldsymbol{x})}{\hat{\sigma}(\boldsymbol{x})}\right)}_{\text{intensification}} + \underbrace{\hat{\sigma}(\boldsymbol{x})\phi\left(\frac{f_{(k)} - \hat{f}(x)}{\hat{\sigma}(\boldsymbol{x})}\right)}_{\text{exploration}}, \tag{2.25}$$

where $f_{(k)}$ is the value of the current best solution, $\tilde{f}$ is the GP with its fitted hyperparameters (the surrogate), and Φ and ϕ denote respectively the cumulative distribution function and the probability density function of the standard normal distribution $\mathcal{N}(0,1)$.

The trade-off between intensification and exploration is represented in (2.25), where the first term corresponds to intensification and the second to exploration. BO based on GPs and the EI is known as *Efficient Global Optimization* (EGO) [145]. EGO determines the next point to evaluate by maximizing the EI, such that

$$\boldsymbol{x}_{\text{next}} \in \underset{\boldsymbol{x}\in\mathcal{X}}{\operatorname{argmax}}\, \mathrm{EI}\left(\boldsymbol{x};\tilde{f}\right).$$

Once the next point has been determined, the objective function f can be evaluated at this point. The data $(\mathbb{X}, f(\mathbb{X}))$ can then be updated by adding the point $\boldsymbol{x}_{\text{next}}$ and its image $f(x_{\text{next}})$. The GP can subsequently be updated with this new pair $(\boldsymbol{x}_{\text{next}}, f(\boldsymbol{x}_{\text{next}}))$. This procedure is computationally expensive because the hyperparameters θ are readjusted, *i.e.*, the log marginal likelihood $p(f(\mathbb{X}) \mid \mathbb{X}, \theta)$ is maximized again. This step constitutes the main bottleneck of BO, since computing the log marginal likelihood $p(f(\mathbb{X}) \mid \mathbb{X}, \theta)$ involves the inversion of a matrix.

A pseudo-code for unconstrained BO (EGO) is presented in Algorithm 3. The convergence stopping criterion is not specified. In contrast to direct search algorithms, BO does not possess an intrinsic convergence criterion. In practice, convergence criteria in BO typically capture a sequence of iterations that intensify a region considered sufficiently small.

In BO, the extension from unconstrained problems to constrained ones can be achieved using surrogates modeling the constraints [249]. More precisely, each left-hand side $c_j : \mathcal{X} \to \bar{\mathbb{R}}$ of a relaxable constraint $c_j(\boldsymbol{x}) \leq 0$ is provided a surrogate $\tilde{c}_j$, where $j \in J$ and J is the set of indices of the relaxable constraints. In this thesis, these surrogates are also GPs. The surrogate $\tilde{c}_j$ of the left-hand side of a relaxable constraint c_j is expressed as

$$\tilde{c}_j(\boldsymbol{x}) \mid \{(\mathbb{X}, c_j(\mathbb{X})), \boldsymbol{x}\} \sim N\left(\hat{c}_j(\boldsymbol{x}), \hat{\omega}_j^2(\boldsymbol{x})\right),$$

where $c_j(\mathbb{X}) \in \mathbb{R}^{|\mathbb{X}|}$ is the vector of evaluated values of the left-hand side of the relaxable

Algorithm 3: Pseudo-code for unconstrained BO.

0. **Initialization**.
   1. Choose a kernel $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$.
   2. Set an evaluation budget $N_{\text{budget}} \in \mathbb{N}$.
   3. Determine an initial dataset $(\mathbb{X}, f(\mathbb{X}))$ using a DoE on $\mathcal{X}$.
   4. Set $k = 0$.

1. **Stopping check**. If $k > N_{\text{budget}}$, then **STOP**

2. **GP construction**. Fit the hyperparameters of the GP model using the current data $(\mathbb{X}, f(\mathbb{X}))$
   - Construct the kernel matrix K:

$$[K]_{i,j} \leftarrow \kappa\big(\boldsymbol{x}_{(i)}, \boldsymbol{x}_{(j)}\big) \qquad \forall \left(\boldsymbol{x}_{(i)}, \boldsymbol{x}_{(j)}\right) \in \mathbb{X} \times \mathbb{X}$$

   - Determine

$$\theta^* \leftarrow \underset{\theta}{\operatorname{argmax}} \ \log p(f(\mathbb{X}) \mid \mathbb{X}, \theta)\,,$$

3. **Selection of the next point**. Determine the next evaluation point by maximizing the expected improvement criterion:

$$\boldsymbol{x}_{\text{next}} \leftarrow \underset{\boldsymbol{x} \in \mathcal{X}}{\operatorname{argmax}} \operatorname{EI}(\boldsymbol{x}; \tilde{f}),$$

4. **Update of the data**. Evaluate $f(x_{\text{next}})$ and update the dataset:

$$\mathbb{X} \leftarrow \mathbb{X} \cup \{x_{\text{next}}\}$$

and

$$f(\mathbb{X}) \leftarrow (f(\mathbb{X}), f(x_{\text{next}}))$$

5. **Iteration**. Set $k \leftarrow k + 1$ and **GO TO** Step 1.

constraint $c_j(\boldsymbol{x}) \leq 0$, and $\hat{c}_j(\boldsymbol{x})$ and $\hat{\omega}_j(\boldsymbol{x})$ are respectively the prediction and the uncertainty of the surrogate c_j at a given point $\boldsymbol{x} \in \mathcal{X}$.

The problem formulation with relaxable constraints can be reused to define a surrogate problem by replacing the objective and constraint functions with their surrogate counterparts, such that

$$\begin{aligned} \min_{\boldsymbol{x} \in \mathcal{X}} \quad & \hat{f}(\boldsymbol{x}) \\ \text{s.t.} \quad & \hat{c}_j(\boldsymbol{x}) \leq 0 \ \forall j \in J. \end{aligned} \tag{2.26}$$

Some formulations transform the constrained surrogate problem into an unconstrained one.

Similarly to the aggregation function h defined in (2.9), the probability that a point $\boldsymbol{x} \in \mathcal{X}$ is feasible [249] is defined as

$$\mathbb{P}\left[\boldsymbol{x} \text{ is feasible}\right] = \prod_{j \in J} \mathbb{P}\left[\tilde{c}_j(\boldsymbol{x}) \leq 0\right].$$

The EI can be generalized to constrained BO through the *Expected Feasible Improvement* (EFI), defined as

$$\mathrm{EFI}\left(\boldsymbol{x}; \tilde{f}\right) = \mathbb{P}\left[\boldsymbol{x} \text{ is feasible}\right] \mathrm{EI}\left(\boldsymbol{x}; \tilde{f}\right).$$

Several other formulations for constrained surrogate problems are presented in [249].

2.2.3 Mixed-variable kernels

No restrictive assumption on the domain $\mathcal{X}$ is required for the construction of a kernel. This property is particularly useful in mixed optimization, where the domain $\mathcal{X}$ contains variables of different types. In essence, BO can be applied on $\mathcal{X}$ as long as an appropriate kernel $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ can be defined. The definition of a kernel allows kernels of various types to be combined with operations that preserve symmetry and positive semi-definiteness. For consistency, the notation of the works presented here has been adjusted and may differ slightly from that used in the original articles.

For mixed-variable problems, a point can be partitioned into components corresponding to the different variable types. In this setting, a point and the domain are denoted

$$\boldsymbol{x} = (\boldsymbol{x}^{\mathrm{cat}}, \boldsymbol{x}^{\mathrm{int}}, \boldsymbol{x}^{\mathrm{con}}) \in \mathcal{X} = \mathcal{X}^{\mathrm{cat}} \times \mathcal{X}^{\mathrm{int}} \times \mathcal{X}^{\mathrm{con}}. \tag{2.27}$$

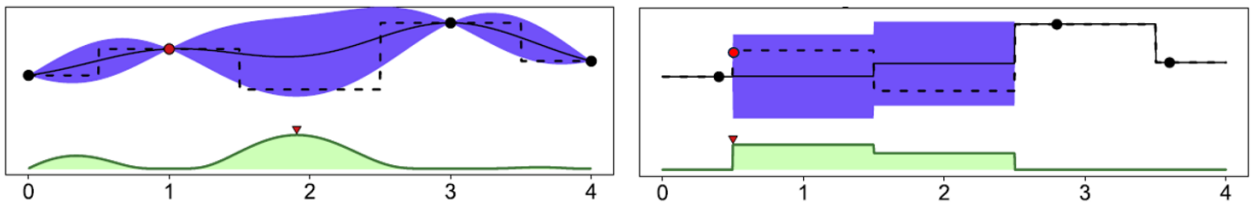
where $\boldsymbol{x}^t \in \mathcal{X}^t$ denotes the component of type $t \in \{\mathrm{cat}, \mathrm{int}, \mathrm{con}\}$. Once appropriate kernels are defined for each component, they can be combined through suitable operations to construct a kernel on the mixed-variable domain $\mathcal{X}$.

For example, in [216], a BO method addresses problems with continuous and categorical variables. Consequently, GPs are characterized by a kernel composed of a categorical kernel κ^{cat} and a continuous kernel κ^{con}. The kernel over the mixed-variable domain can be constructed through different operations preserve symmetry and positive semi-definiteness:

$$\begin{aligned}
\kappa(\boldsymbol{x}, \boldsymbol{y}) &= \kappa^{\mathrm{cat}}(\boldsymbol{x}^{\mathrm{cat}}, \boldsymbol{y}^{\mathrm{cat}}) \kappa^{\mathrm{con}}(\boldsymbol{x}^{\mathrm{con}}, \boldsymbol{y}^{\mathrm{con}}) && \text{(product)}, \\
\kappa(\boldsymbol{x}, \boldsymbol{y}) &= \kappa^{\mathrm{cat}}(\boldsymbol{x}^{\mathrm{cat}}, \boldsymbol{y}^{\mathrm{cat}}) + \kappa^{\mathrm{con}}(\boldsymbol{x}^{\mathrm{con}}, \boldsymbol{y}^{\mathrm{con}}) && \text{(sum)}, \\
\kappa(\boldsymbol{x}, \boldsymbol{y}) &= \left(1 + \kappa^{\mathrm{cat}}(\boldsymbol{x}^{\mathrm{cat}}, \boldsymbol{y}^{\mathrm{cat}})\right) \left(1 + \kappa^{\mathrm{con}}(\boldsymbol{x}^{\mathrm{con}}, \boldsymbol{y}^{\mathrm{con}})\right) && \text{(ANOVA)},
\end{aligned} \tag{2.28}$$

Kernels for continuous variables have already been discussed in Section 2.2.1. The next step is therefore to review approaches from the literature that address the construction of integer and categorical kernels.

Historically, integer variables have often been handled through a continuous relaxation. The relaxed variables selected by maximizing the EI are subsequently rounded to make them compatible with the domain $\mathcal{X}$ of the original problem [112]. This naive approach leads to several issues. In particular, due to the a posteriori rounding, the selected points do not necessarily correspond to the points in the domain $\mathcal{X}$ that are actually evaluated. Figure 2.10a illustrates that the point selected by maximizing the acquisition function (red triangle) does not correspond to the point $\boldsymbol{x}_{\text{next}}$ (red circle) in the original domain [112]. Moreover, the point $\boldsymbol{x}_{\text{next}}$ lies near a local minimum of the acquisition function, which should instead be maximized.



(a) Naive approach with a posteriori rounding after maximizing the acquisition function.

(b) Proposed approach using a transformed continuous kernel.

Figure 2.10 Gaussian process regression for an integer variable. The objective function f is shown as a black dashed curve, the prediction $\hat{f}$ as a black solid line, the uncertainty $\hat{\sigma}$ in purple, and the acquisition function in green (from [112]).

The work of [112] a simple kernel modification to address this issue. Let $\boldsymbol{x}^{\text{rint}} \in \mathbb{R}^{n^{\text{int}}}$ denote the relaxed integer variables. The idea is to use a continuous kernel for the relaxed integer variables and apply the rounding transformation R a priori in the arguments of the kernel, such that

$$\kappa^{\text{int}}(\boldsymbol{x}^{\text{int}}, \boldsymbol{y}^{\text{int}}) = \kappa^{\text{con}}\left(R(\boldsymbol{x}^{\text{rint}}), R(\boldsymbol{y}^{\text{rint}})\right). \tag{2.29}$$

In Figure 2.10b, the regression induced by Equation (2.29) is a stepwise function whose domain is partitioned according to nearest-integer rounding. The transformation R applied to the arguments of the continuous kernel ensures that the regression similarly to the objective function.

The work [194] presents several approaches for constructing categorical kernels κ^{cat} using matrices. The first approach is the full parameterization. This parameterization constructs a categorical kernel κ^{cat} using a single positive semi-definite matrix $T \in \mathbb{R}^{|\mathcal{X}^{\text{cat}}| \times |\mathcal{X}^{\text{cat}}|}$, where

$\mathcal{X}^{\text{cat}}$ is the categorical domain. An entry $[T]_{I(x^{\text{cat}}),I(y^{\text{cat}})}$ of the matrix T represents the similarity (or correlation) between two categorical components $\boldsymbol{x}^{\text{cat}}$ and $\boldsymbol{y}^{\text{cat}}$, which are indexed by $I(\boldsymbol{x}^{\text{cat}})$ and $I(\boldsymbol{y}^{\text{cat}}) \in \{1, 2, \ldots, |\mathcal{X}^{\text{cat}}|\}$. The categorical kernel κ^{cat} is then defined as

$$\kappa^{\text{cat}}(\boldsymbol{x}^{\text{cat}}, \boldsymbol{y}^{\text{cat}}) = \left[T_{I(\boldsymbol{x}^{\text{cat}}),I(\boldsymbol{y}^{\text{cat}})}\right] \in \mathbb{R}.$$

To ensure the positive semi-definiteness of the categorical kernel κ^{cat}, the matrix T is constructed using a Cholesky decomposition [201], such that $T = AA^\top$, where A is a lower triangular matrix with a unit diagonal. The matrix T is therefore positive semi-definite by construction, since

$$v^\top T v = v^\top A A^\top v = \|A^\top v\|^2 \geq 0 \quad \forall v \in \mathbb{R}^{|\mathcal{X}^{\text{cat}}|}.$$

Next, a hyperspherical decomposition is used to construct the lower triangular matrix A. This decomposition represents the elements of the q-th row as a point on a q-dimensional hypersphere. More precisely, the elements of the matrix A are defined as follows [194]:

$$[A]_{q,s} = \begin{cases} 0 & \text{if } s > q, \\ \alpha_{1,0} & \text{if } q = s = 1, \\ \alpha_{q,0} \sin(\alpha_{q,1}) \ldots \sin(\alpha_{q,q-1}) \sin(\alpha_{q,q}) & \text{if } q = s > 1, \\ \alpha_{q,0} \sin(\alpha_{q,1}) \ldots \sin(\alpha_{q,s-1}) \cos(\alpha_{q,s}) & \text{if } s < q, \end{cases} \tag{2.30}$$

where $s \in \{2, 3, \ldots, q-1\}$, $\alpha_{q,0} > 0$ for all $q \in \{1, 2, \ldots, |\mathcal{X}^{\text{cat}}|\}$ (length parameters), and $\alpha_{i,j} \in (0, \pi)$ for all $(i, j) \in \{1, 2, \ldots, |\mathcal{X}^{\text{cat}}|\}^2$ (angle parameters). The length and angle parameters are the hyperparameters of the categorical kernel.

The full parameterization of the matrix T contains $(|\mathcal{X}^{\text{cat}}| + 1)\,|\mathcal{X}^{\text{cat}}|/2$ hyperparameters. For some problems, this parameterization is not practical. Indeed, the number of hyperparameters grows with the number of categorical components, which can increase extremely rapidly. Consequently, fitting the hyperparameters of a GP under a full parameterization may be intractable in practice.

Several alternative parameterizations with fewer hyperparameters have been proposed [194, 201]. The heteroscedastic parameterization operates on a variable-per-variable rather than on full categorical components. Each categorical variable $x_i^{\text{cat}} \in \mathcal{X}_i^{\text{cat}}$ contributes independently to the categorical kernel and is assigned its own positive semidefinite matrix $\mathcal{X}_i^{\text{cat}} \in \mathbb{R}^{|\mathcal{X}_i^{\text{cat}}| \times |\mathcal{X}_i^{\text{cat}}|}$, where $i \in \{1, 2, \ldots, n^{\text{cat}}\}$. An element $[T_i]_{I(x_i^{\text{cat}}),I(y_i^{\text{cat}})}$ represents the

similarity between two categories of the i-th categorical variable, where $I(x_i^{\text{cat}})$ and $I(y_i^{\text{cat}})$ denote the indices of the categories assigned to x_i^{cat} and y_i^{cat}, respectively. In other words, an entry of the matrix T_i^{cat} compares the categories taken by the i-th variable in two categorical components $x^{\text{cat}}, y^{\text{cat}}$, such that

$$\kappa_i^{\text{cat}}(x_i^{\text{cat}}, y_i^{\text{cat}}) = [T_i]_{I(x_i^{\text{cat}}),I(y_i^{\text{cat}})}.$$

Then, the categorical kernel κ^{cat} is defined from the variable-wise kernels as

$$\kappa^{\text{cat}}(\boldsymbol{x}^{\text{cat}}, \boldsymbol{y}^{\text{cat}}) = \prod_{i=1}^{n^{\text{cat}}} \kappa_i^{\text{cat}}(x_i^{\text{cat}}, y_i^{\text{cat}}).$$

Each matrix T_i is constructed following the same procedure as the matrix of the full parameterization, namely through a Cholesky decomposition combined with a hyperspherical parameterization. A matrix T_i is positive semidefinite and contains $(|\mathcal{X}_i^{\text{cat}}| + 1)|\mathcal{X}_i^{\text{cat}}|/2$ hyperparameters, where $i \in \{1, 2, \ldots, n^{\text{cat}}\}$. The heteroscedastic parameterization introduces a total of $\sum_{j=1}^{n^{\text{cat}}} \frac{\left(|\mathcal{X}_j^{\text{cat}}|+1\right)|\mathcal{X}_j^{\text{cat}}|}{2}$ hyperparameters.

The homoscedastic parameterization further reduces the number of hyperparameters by assuming that all matrices T_i^{cat} have a constant diagonal. This corresponds to assuming that all categories of a categorical variable x_i^{cat} share the same variance. Under this assumption, the homoscedastic parameterization introduces a total of $\sum_{j=1}^{n^{\text{cat}}} \frac{(|L_j|-1)|L_j|}{2}$ hyperparameters.

The parsimonious parameterization is the most compact parameterization, introducing the smallest number of hyperparameters. The similarity between two categories depends only on whether they are identical or different. An entry of the matrix is defined as

$$[T^{\text{cat}}]_{L(x_i^{\text{cat}}),L(y_i^{\text{cat}})} = \begin{cases} v_i & \text{if } x_i^{\text{cat}} = y_i^{\text{cat}}, \\ c_i & \text{if } x_i^{\text{cat}} \neq y_i^{\text{cat}}, \end{cases}$$

where v_i denotes the auto-correlation of the i-th categorical variable and c_i measures the similarity between two distinct categories of that variable. To ensure that the matrices T_i^{cat} remain positive semidefinite, the constraint $-\left(|L_i| + 1\right)^{-1} v_i < c_i < v_i$ is imposed for each $i \in \{1, 2, \ldots, n^{\text{cat}}\}$.

The different parameterizations mainly reflect a trade-off between flexibility and computational costs. The full parameterization is highly flexible since it models all correlations between categorical components, but the large number of hyperparameters may imply intractability. Conversely, parsimonious parameterizations are more scalable but rely on stronger

modeling assumptions that may introduce significant modeling error when they are not valid.

The work of [270] proposes a categorical kernel based on a continuous kernel where each categorical variable x_i^{cat} is mapped to a latent space through an encoding $\Phi_i^{\text{cat}} : \mathcal{X}_i^{\text{cat}} \to \mathcal{L}_i \subseteq \mathbb{R}^2$. The categorical kernel is then defined as

$$\kappa^{\text{cat}}(\boldsymbol{x}^{\text{cat}}, \boldsymbol{y}^{\text{cat}}) = \prod_{i=1}^{n^{\text{cat}}} \kappa_i^{\text{con}}\left(\Phi_i^{\text{cat}}(x_i^{\text{cat}}), \Phi_i^{\text{cat}}(y_i^{\text{cat}})\right),$$

where $\kappa_i^{\text{con}} : \mathcal{L}_i \times \mathcal{L}_i \to \mathbb{R}$. The coordinates of the encoded vectors are not important themselves. Indeed, it is the relative distances in the latent space that determine the similarity between categories. The (relative) positions of the categories in $\mathcal{L}_i$ are obtained by maximizing the log marginal likelihood (2.22), and can therefore be interpreted as hyperparameters of the GP. In [90], the GPs are used for BO. The maximization of the EI is done in the latent spaces $\mathcal{L}_i$. The EI problem is formulated using an augmented Lagrangian with pre-image constraints, which allow to recover categorical variables from encoded ones.

Hierarchical kernels are not discussed in this section, as they are extensively detailed in the article 4 of Chapter 7. That work provides an extensive review of hierarchical modeling approaches. The present project can be viewed as a unification of several hierarchical modeling approaches.

2.3 Metaheuristics

Metaheuristic methods are sophisticated sampling techniques typically inspired by biological processes. This thesis does not focus on metaheuristics and uses these methods only for benchmarking. Nevertheless, they are commonly employed for mixed-variable black-box optimization and are therefore worth discussing in the literature review. The review of metaheuristics is less detailed than those of the direct search and BO approaches. In particular, only genetic algorithms and CMA-ES are discussed, as they are the metaheuristics for which literature exists on the problems of interest. For an exhaustive survey on metaheuristic methods for blackbox optimization, see [248].

2.3.1 Genetic algorithms

A very popular type of metaheuristic is the genetic algorithm [95]. These algorithms are inspired by the theory of evolution. Candidate solutions are encoded as *chromosomes* and modified through operators such as *mutation* and *crossover*. For example, the crossover operation exchanges parts of promising chromosomes to generate new candidate points. The

mutation randomly changes parts of the chromosomes to create new points outside of the pool gene. At each iteration, the method maintains a *population* of fixed size, representing a set of candidate solutions (chromosomes). The population typically starts with a randomly generated set of points. Then, at each iteration, a new *generation* of candidate points, called *offsprings*, is produced from the current population. The individuals in the population are ranked according to their *fitness*, which represents how promising they are. In the unconstrained case, the fitness is typically defined by the objective value. The generation of new candidate points favors operations applied to the most *fit* individuals in the population. This generation is integrated into the population based on the fitness of the individuals.

The operations to generate new points can be extended directly to integer and categorical variables. These variables are encoded into binary vectors and then the operations are directly applied on these vectors. The following figure illustrates the generation of two chromosomes. Figure 2.11 illustrates the application of the crossover and mutation operators on two binary

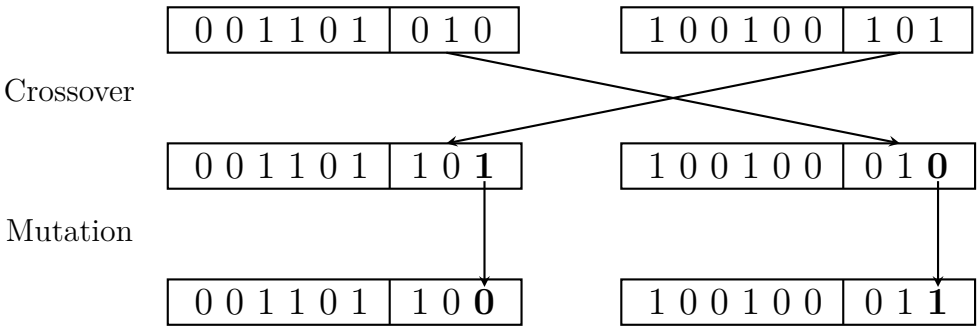


Figure 2.11 Illustration of crossover and mutation operators applied to two binary chromosomes (inspired by [84]).

chromosomes. A crossover is first performed on the last three binary variables, exchanging these segments between the two chromosomes and producing two offspring. A mutation is then applied by flipping the last binary variable of each offspring.

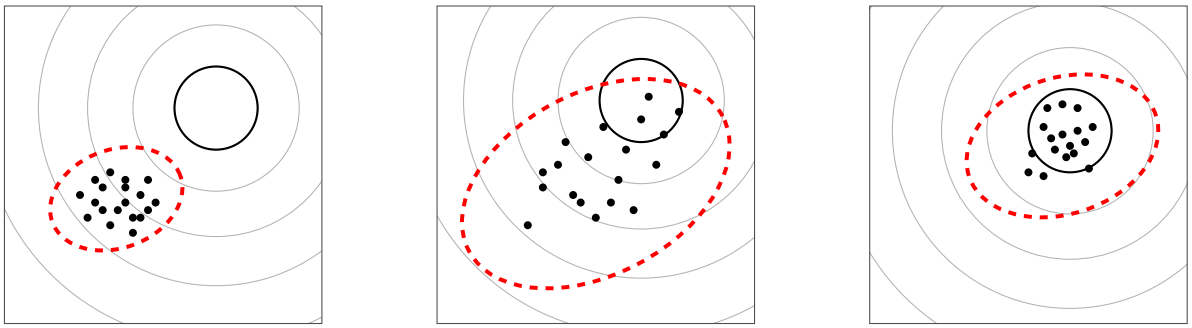
In constrained problems, constraint violations are usually incorporated into the fitness function through penalty terms. By consequence, infeasible solutions receive lower fitness and are less likely to be selected.

The Non-dominated Sorting Genetic Algorithm-II (NSGA-2) [95] is spreadly used for single objective problems with quantitative and categorical variables, even though it was initially designed for multi-objective optimization.

2.3.2 Covariance Matrix Adaptation Evolution Strategy

The Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [129] is another metaheuristic that is based on the theory of evolution. It is another method that uses iteratively updates a population of fixed size, and generates candidate solutions by sampling this population. In the continuous setting, the population is modeled as a multivariate Gaussian distribution. More precisely, given a population, a mean and covariance matrix are determined with a maximum likelihood procedure. This finds the most likely multivariate Gaussian distribution that generates the population.

Once the distribution is set, candidate solutions are sampled from it. The population is updated similarly as the genetic algorithm. At a given iteration, the sampled points that improve the objective and/or the constraint functions replace less promising points. The mean and covariance matrix are then estimated with the new population, which moves the distribution towards better regions and reshapes to favor sampling in promising directions. This iterative procedure is illustrated in the following figure.



(a) Initial population and sampling distribution.

(b) Displacement and reshaping of the distribution.

(c) Concentration of the population around the optimum.

Figure 2.12 Illustration of the evolution CMA-ES for an unconstrained continuous problem. Black points represent sampled candidate solutions and the red dashed ellipse represents the sampling distribution.

Figure 2.12 illustrates the evolution of the sampling distribution over successive generations. Initially, the candidate solutions are concentrated far from the optimum. As the generations progress, the distribution is displaced and reshaped toward more promising regions, before becoming more concentrated around the optimum.

CatCMA extends the covariance matrix adaptation evolution strategy (CMA-ES) to problems involving both categorical and continuous variables [128]. In this approach, categorical variables are modeled by probability vectors defined over their respective categories, while

continuous variables are modeled as in classic CMA-ES. These probabilistic models are jointly updated during the optimization process based on the performance of the sampled candidate solutions. At each iteration, new candidate solutions are generated by sampling from this joint probabilistic distribution. The algorithm explores categorical variables and continuous variables simultaneously.

CHAPTER 3 ORGANIZATION OF THE THESIS

Up on a hill, is where we begin
This little story, a long time ago

— The Strokes, *The Modern Age*

The main contributions of the thesis are the next four chapters, each consisting of a a paper that is either accepted, in revision or submitted to a scientific journal.

- Chapter 4 presents CatMADS, an extension of the MADS algorithm for categorical variables. The MADS algorithm is extensively discussed in Chapter 2.
- Next, Chapter 5 develops novel categorical neighborhoods that are constructed from GPs. These neighborhoods incorporate information from both the objective and constraints functions.
- Afterwards, Chapter 6 proposes a hierarchical distance that can compare any points that do not share the same variables or subjected to the same bounds. This project is more related to machine learning than optimization.
- Finally, Chapter 7 generalizes the hierarchical distance in Chapter 6 and unifies different hierarchical frameworks from the literature. Hierarchical BO is also developed from the hierarchical distance Chapter 6.

Secondary contributions that are not part of a publishable manuscript are described in Chapter 8. Finally, the main contributions of the thesis are synthesized in Chapter 9. The limitations of these contributions are also highlighted, and possible improvements or extensions for future work are discussed.

CHAPTER 4 ARTICLE 1: CATMADS: MESH ADAPTIVE DIRECT SEARCH FOR CONSTRAINED BLACKBOX OPTIMIZATION WITH CATEGORICAL VARIABLES

There are two colours in my head
What, what is that you tried to say?
What, what was that you tried to say?

— Radiohead, *Everything in Its Right Place*

Authors. Charles Audet, Youssef Diouane, Edward Hallé-Hannan, Sébastien Le Digabel and Christophe Tribes

Submitted on June 12, 2025; under review at *SIAM Journal on Optimization*
(see arXiv:2506.06937)

Abstract. Solving optimization problems in which functions are blackboxes and variables involve different types poses significant theoretical and algorithmic challenges. Nevertheless, such settings frequently occur in simulation-based engineering design and machine learning. This paper extends the Mesh Adaptive Direct Search (MADS) algorithm to address mixed-variable problems with categorical, integer and continuous variables. MADS is a robust derivative-free optimization framework with a well-established convergence analysis for constrained quantitative problems. CatMADS generalizes MADS by incorporating categorical variables through distance-induced neighborhoods. A detailed convergence analysis of CatMADS is provided, with flexible choices balancing computational cost and local optimality strength. Four types of mixed-variable local minima are introduced, corresponding to progressively stronger notions of local optimality. CatMADS integrates the progressive barrier strategy for handling constraints, and ensures Clarke stationarity. An instance of CatMADS employs cross-validation to construct problem-specific categorical distances. This instance is compared to state-of-the-art solvers on 60 mixed-variable problems from the Cat-suite collection. Half of these problems are constrained. Data profiles show that CatMADS achieves the best results, demonstrating that the framework is empirically efficient in addition to having strong theoretical foundations.

Keywords. Blackbox optimization, derivative-free optimization, mixed-variable, categorical variables, mesh adaptive direct search.

4.1 Introduction

Many engineering and machine learning problems involve functions without explicit expressions, known as *blackbox* functions. These functions often include *categorical* variables that take qualitative values. In this context, traditional gradient-based or relaxation-based methods can not be directly employed. This work proposes CatMADS, an extension of the Mesh Adaptive Direct Search (MADS) algorithm [21], that addresses constrained blackbox problems with quantitative and categorical variables. CatMADS manages categorical variables with problem-specific distances.

This work covers many real-life applications. Machine learning (ML) models are characterized by hyperparameters that can be optimized to improve performance. Hyperparameter tuning, involves optimizing a time-consuming mixed-variable blackbox that trains the model and returns a test score [25, 125]. In [13], a simulation estimates the energy produced by a solar powerplant for given design variables, such as dimensions of pieces or turbine types. An active research field in engineering is the simulation-based design of aircraft [63, 222]. These simulation input geometry and materials, and output energy-based scores via numerical methods [171].

Notation

Vectors are in bold font. The letters $\boldsymbol{x}$ and $\boldsymbol{y}$ are reserved for points. Scalars are in normal font. Subscripts without parenthesis denote variables, *e.g.*, x_1^{cat} is the first categorical variable. Subscripts with parentheses are reserved for indexing iterations, *e.g.*, $\boldsymbol{x}_{(k)}$, or for listing points, *e.g.*, $\boldsymbol{x}_{(1)}, \boldsymbol{x}_{(2)}, \ldots, \boldsymbol{x}_{(s)}$. For functions, subscripts represent the parameters on which they depend. Superscripts are reserved for types, with small exceptions in the convergence analysis. $\mathbb{R}$, $\mathbb{Z}$ and $\mathbb{N}$ denote the set of real numbers, integers and nonnegative integers, respectively. Other sets denoted using capital letters with either normal or calligraphic font, *e.g.*, A or $\mathcal{A}$, except for the set of known points noted $\mathbb{X}$. For a neighborhood, the semicolon distinguishes its center from the parameter controlling its size, *e.g*, an open ball of radius ϵ centered at $\boldsymbol{u}$ is noted $\mathcal{B}(\boldsymbol{u}; \epsilon)$.

4.1.1 Scope

This work considers inequality constrained mixed-variable blackbox optimization problems of the form

$$\min_{\boldsymbol{x} \in \Omega} f(\boldsymbol{x}), \tag{4.1}$$

where $f : \mathcal{X} \to \overline{\mathbb{R}}$ (with $\overline{\mathbb{R}} = \mathbb{R} \cup \{+\infty\}$) is the objective function, $\mathcal{X}$ is the domain of the objective and constraint functions, $\Omega := \{\boldsymbol{x} \in \mathcal{X} \ : \ g_j(\boldsymbol{x}) \leq 0, \ j \in J\}$ is the feasible set, $g_j : \mathcal{X} \to \overline{\mathbb{R}}$ is the j-th constraint function of the problem, with $j \in J$, and $|J|$ denotes the number of constraints.

The objective and the constraint functions are assumed to be provided by blackboxes, defined as in [26]: "*any process that when provided an input, returns an output, but the inner working of the process is not analytically available*". In this context, derivatives are unavailable and the function evaluations are performed by executing a potentially time-consuming process. For example, the estimation of an aircraft drag could be done using a finite element simulation that outputs a drag value, provided a geometry and materials used.

The execution of a blackbox can crash or fail to return a valid value because of *hidden constraints* [79], *e.g.*, an unexpected division by zero. In that case, the point $\boldsymbol{x} \in \mathcal{X}$ is assigned $f(\boldsymbol{x}) = +\infty$. A point $\boldsymbol{x} \notin \mathcal{X}$ can also be assigned $f(\boldsymbol{x}) = +\infty$ to outline that an *unrelaxable* constraint, which must be satisfied to have a proper blackbox execution, is not satisfied. In contrast, the constraint functions characterizing the feasible set Ω are said to be *relaxable and quantifiable*, since they can be evaluated and return a proper value without being satisfied [162].

The target optimization problems are mixed-variable, meaning that a variable can be of any type among *continuous* (con), *integer* (int) and *categorical* (cat). Variables of the same type are regrouped into a column vector called a *component* and expressed as

$$\boldsymbol{x}^t := (x_1^t, x_2^t, \ldots, x_{n^t}^t) \in \mathcal{X}^t := \mathcal{X}_1^t \times \mathcal{X}_2^t \times \ldots \times \mathcal{X}_{n^t}^t, \tag{4.2}$$

where $t \in \{\text{cat}, \text{int}, \text{con}\}$ refers to the type, $n^t \in \mathbb{N}$ is the number of variables of type t, $\mathcal{X}^t$ is the set of type t, $x_i^t \in \mathcal{X}_i^t$ denotes the i-th variable of type t, and $\mathcal{X}_i^t$ is the set of values of x_i^t. For convenience, the indices of the variables of type t are regrouped in the set $I^t := \{1, 2, \ldots, n^t\}$.

The continuous $\boldsymbol{x}^{\text{con}} \in \mathcal{X}^{\text{con}} \subseteq \mathbb{R}^{n^{\text{con}}}$ and integer $\boldsymbol{x}^{\text{int}} \in \mathcal{X}^{\text{int}} \subseteq \mathbb{Z}^{n^{\text{int}}}$ components are composed of *quantitative* variables that belong to ordered sets with well-defined metrics. For readability, the integer and continuous components can be grouped into the quantitative component $\boldsymbol{x}^{\text{qnt}} := (\boldsymbol{x}^{\text{int}}, \boldsymbol{x}^{\text{con}}) \in \mathcal{X}^{\text{qnt}} := \mathcal{X}^{\text{int}} \times \mathcal{X}^{\text{con}}$. The integer $\boldsymbol{x}^{\text{int}}$ and categorical $\boldsymbol{x}^{\text{cat}}$ components both contain discrete variables. However, categorical variables are not quantitative and do not admit numerical bounds. A categorical variable takes qualitative values, called categories, in a set that is supposed to be finite. For $i \in I^{\text{cat}}$, the i-th categorical variable $x_i^{\text{cat}} \in \mathcal{X}_i^{\text{cat}} := \{c_1, c_2, \ldots, c_{\ell_i}\}$, where $\ell_i \in \mathbb{N}$ is a finite number of categories and c_j is a category

for $j \in \{1, 2, \ldots, \ell_i\}$, $i \in I^{\text{cat}}$. For example, the blood type of a person takes a category in the set $\{\text{O-}, \text{O+}, \ldots, \text{AB+}\}$. The usual distance functions are not suitable for categorical variables, which makes them difficult to treat [25, 130]. The categorical set $\mathcal{X}^{\text{cat}}$ is finite and its cardinality is $|\mathcal{X}^{\text{cat}}| = \prod_{i=1}^{n^{\text{cat}}} \ell_i$.

Finally, a point $\boldsymbol{x} \in \mathcal{X}$ is partitioned into component by types, such that

$$\boldsymbol{x} \coloneqq \left(\boldsymbol{x}^{\text{cat}}, \boldsymbol{x}^{\text{int}}, \boldsymbol{x}^{\text{con}}\right) \in \mathcal{X}^{\text{cat}} \times \mathcal{X}^{\text{int}} \times \mathcal{X}^{\text{con}} \tag{4.3}$$

where $\mathcal{X} \coloneqq \mathcal{X}^{\text{cat}} \times \mathcal{X}^{\text{int}} \times \mathcal{X}^{\text{con}}$ is the domain that consists of Cartesian products of the categorical set $\mathcal{X}^{\text{cat}}$, the integer set $\mathcal{X}^{\text{int}}$ and the continuous set $\mathcal{X}^{\text{con}}$.

Note that this work does not cover problems with *meta variables* that influence the inclusion or bounds of other variables [25, 125]. The dimensions and bounds of the problems are fixed.

4.1.2 Objectives, contributions and organization

MADS is a direct search algorithm that provides an exhaustive convergence theory for constrained quantitative problems [21, 33]. Nevertheless, the direct search approach remains relatively underdeveloped for mixed-variable problems with categorical variables, especially in comparison to Bayesian optimization (BO) [112, 194, 216] and metaheuristics [95] approaches. The mixed-variable MADS (MV-MADS), an extension of MADS for discrete variables, has several convergence guarantees [4]. However, MV-MADS requires the implementation of explicit *user-defined neighborhoods* that define discrete variable neighborhoods. Such neighborhoods can be tricky or arbitrary to define in a blackbox optimization context with limited information, yet they greatly impact performance and theoretical results. In practice, user-defined neighborhoods are not user-friendly and discourage practitioners from using this method. MV-MADS treats integer variables as categorical variables with these discrete neighborhoods, which prevents it from fully capitalizing on their properties and sacrifices potential efficiency gains. For these reasons, MV-MADS is rarely considered a state-of-the-art option for the class of problems studied.

CatMADS generalizes MADS and improves, both theoretically and practically, the management of categorical variables of MV-MADS. CatMADS is off-the-shelf, flexible and offers local convergence guarantees, which BO or metaheuristic methods lack. The research gap addressed is the absence of a rigorous and automatic optimization method for mixed-variables problems with constraints. This work introduces and adapts different notions of local optimality to develop the theoretical results of CatMADS. In doing so, several characterizations of local minima are proposed and multiple key definitions, such as hypertangent cones, are

adapted to the mixed-variable setting. The four main contributions of this work are:

1. develop interpretable and systematic neighborhoods that structure categorical variables automatically through distances. This structure is a central mechanism of CatMADS and directly influences the quality of the guarantees with respect to categorical variables.

2. integrate an efficient mechanism for treating quantitative variables with convergence guarantees. This is achieved by generalizing the version of MADS that uses a granular mesh [33], referred to as G-MADS, so that it tackles integer and continuous variables in a more general mixed-variable setting with categorical variables.

3. handle constraints with state-of-the-art techniques supported by convergence guarantees. CatMADS extends both the progressive barrier [22] and the extended poll [19], and unifies these mechanisms for efficient constraint handling. Together, they allow exploration of promising categorical variables that may not satisfy relaxable constraints and they influence the convergence guarantees. By contrast, MV-MADS requires an initial feasible point, whereas CatMADS does not, and employs the simpler extreme barrier strategy that simply rejects infeasible points [21].

4. establish a comprehensive convergence analysis that fully characterizes the guarantees of CatMADS and integrates the contributions of the previous objectives into a unified framework. In this analysis, several types of local minima are introduced to offer compromises between the strength of local optimality and the evaluation cost of CatMADS.

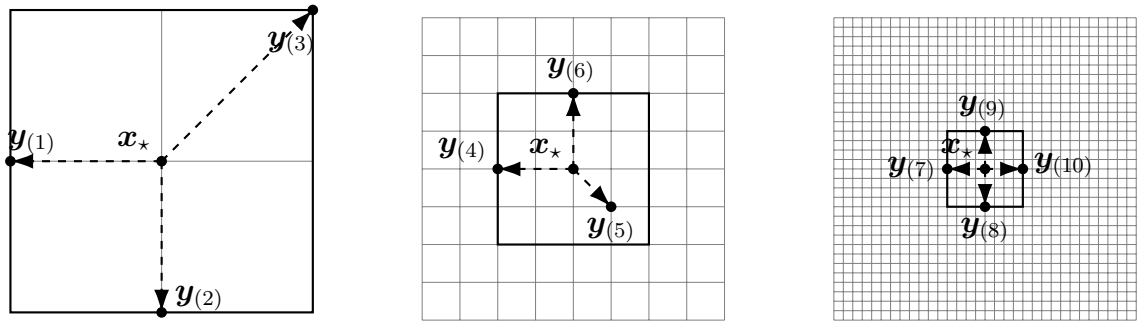
Some contributions of the work have a broader scope than the work. The new neighborhoods provide a principled way to structure and model categorical variables, in other areas of optimization including combinatorial optimization. Similarly, the different types of local minima and the adaptations of nonsmooth optimization definitions, such as the hypertangent cone, can be used by any mixed-variable optimization method. Finally, in the implementation, the problem-specific categorical distance constructed from problem data is relevant to both optimization and machine learning models involving categorical variables.

The rest of the document is organized as follows. Essential elements of MADS are discussed in Section 4.1.3 and related work is covered in Section 4.1.4. The general framework CatMADS is presented in Section 4.2. Section 4.3 presents the theoretical results of CatMADS, along with adapted definitions from nonsmooth optimization and new types of local minima tailored for our setting. Finally, computational experiments with state-of-the-art solvers are described in Section 4.4.

4.1.3 Essential concepts of MADS

MADS is a direct search method that iteratively seeks candidate points with a strict decrease of the objective function and/or constraint functions [21]. The candidate points lie on a mesh that discretizes the space with step sizes. The MADS algorithm has two main steps. The *search*, an optional step, evaluates points on the mesh using flexible strategies. Although optional, the search allows to incorporate efficient optimization strategies that can significantly improve the global solution quality. Notable empirical improvements are often achieved with search steps based on quadratic model optimization [85]. The key step, called the *poll*, evaluates candidates points around the *incumbent solution*, which is the best solution found since the algorithm was launched. This is the step that provides convergence guarantees. If a better solution is found during the search or the poll, then it becomes the incumbent solution and the iteration is said to be successful. Otherwise, the iteration is unsuccessful, the incumbent solution is unchanged and the mesh is refined to discretize the space more finely.

For the continuous case, the mesh has a single step size and the poll is constructed with a set of directions positively spanning the space. The poll converges with a succession of unsuccessful iterations, such that candidate points are evaluated in an increasingly denser region around a minimum: this is schematized for $\mathbb{R}^2$ in Figure 4.1.



(a) (b) (c)

Figure 4.1 Three consecutive polls of MADS converging to a local minimum $\boldsymbol{x}_\star \in \mathbb{R}^2$.

In Figure 4.1, the poll generates candidate points with positive basis that are represented by dotted arrows starting from $\boldsymbol{x}_\star$. These candidate points $\boldsymbol{y}_{(1)}, \boldsymbol{y}_{(2)}, \ldots, \boldsymbol{y}_{(10)}$ must lie on the mesh (grey grid) and within the frame (black square). The frame represents the delimitation in which the poll is performed and it is greater or equal to the mesh size. As MADS converges, the mesh and frame sizes shrink toward zero, and the step size shrinks more aggressively to force polls in denser regions. This increases the number of admissible candidates because

more mesh points lie inside the frame. In Figure 4.1a, there are $3 \times 3 = 9$ grid intersections lie in the frame, hence 8 potential candidates since the center is already taken by $x_\star$. In Figure 4.1b, the mesh shrinks more rapidly than the frame, yielding 25 grid intersections in the frame and therefore 24 admissible points after excluding the center. The convergence analysis of MADS relies on sets of candidate points becoming increasingly dense. For more details on the MADS algorithm, see Chapter 8 of [26]. The state-of-the art strategy to handle constraints is the progressive barrier, which is detailed in Section 4.2.3.

4.1.4 Related work

The recent literature on mixed-variable blackbox problems mostly comes from BO with GPs, focusing on similarity measures, referred to as *kernels*, that characterize GPs. For continuous and integer variables, squared-exponential kernels can be used straightforwardly [112]. However, categorical variables require additional work. In [112], categorical variables are one-hot encoded into binary vectors with one binary variable per category. These binary variables are then continuously relaxed, allowing the squared-exponential kernels to be used. Relaxed categorical variables are transformed into their original form with a transformation that, for each categorical variable, sets the highest-valued category to one, and all its other categories to zero.

In [270], each categorical variable is provided with a 2D continuous space for encoding its categories. The categories are mapped with a maximum likelihood procedure that positions correlated categories closely and uncorrelated categories farther apart. The authors of [90] propose a pre-image problem that reconstructs the encoded categorical variables. This is useful in BO, since the GPs are used to form a subproblem that determines the next point to evaluate. If categorical variables are encoded, then the solution is also encoded.

An alternative approach to binary vectors employs matrices [194, 201, 216]. Each categorical variable is assigned a matrix whose elements are hyperparameters representing correlations or anti-correlations between its categories. The hyperparameters are constructed with hypersphere decompositions that parametrize the matrices [201]. A parametrization with more hyperparameters offers greater modeling flexibility, but adjusting its hyperparameters becomes more costly. See [224] for in-depth presentation of different parametrizations. This matrix approach implies that the BO subproblem is also mixed-variable. In [194], the subproblem is optimized with a genetic algorithm inspired from [244].

Genetic algorithms are also commonly employed in blackbox optimization, such as the Nondominated Sorting Genetic Algorithm [95] (NSGA-2). These algorithms encode points into *chromosomes* that are subject to different operations inspired from the theory of evolution.

For instance, the crossover operation exchange parts of promising chromosomes to heuristically propose new solutions. Many other metaheuristic methods exist for the problems of interest, such as the mixed-variable Particle Swarm Optimization [258]. For a comprehensive survey, see [248].

Finally, the work [167] shares many similarities with MV-MADS, and by consequence CatMADS. In [167], the discrete variables are handled with user-defined neighborhoods, as in MV-MADS. This works capitalizes on the flexibility of user-defined neighborhoods to treat a general class of problems with discrete variables, called *dimensional variables* [167], that influence the dimension of the problem. However, the continuous variables are optimized by solving, for each fixed configuration of the discrete variables, a continuous subproblem handled with a local line-search method under linear constraints. The approach proposed in [167] shares essentially the same limitations as MV-MADS: integer variables are handled as categorical variables, user-defined neighborhoods are difficult to specify and to interpret in blackbox settings, an initial feasible point is required, and the theory focuses on a single notion of local optimality, without offering trade-offs between evaluation cost and strength.

4.2 The CatMADS general framework

CatMADS follows the paradigm of direct search methods: an initialization step is done at the beginning, then search and poll steps are iteratively performed until a stopping or convergence criteria is met. At each iteration, the algorithm seeks to improves an incumbent solution $\boldsymbol{x}_{(k)} \in \mathcal{X}$. The framework is presented in its general form in Algorithm 4. Initially, CatMADS is discussed without constraints; they are addressed in Section 4.2.3, after the poll step is detailed.

Algorithm 4: The CatMADS framework.

0. **Initialization**. Initialize poll parameters, perform an initial Design of Experiment (DoE) and set $k = 0$
1. **Opportunistic search** (finitely many mesh points, possibly none).
2. **Opportunistic poll**. Perform polling over candidates points in $P_{(k)} = P_{(k)}^{\text{cat}} \cup P_{(k)}^{\text{qnt}}$
3. **Opportunistic extended poll** (optional). If Steps 1. and 2. are unsuccessful, perform quantitative polls
   centered at points in $P_{(k)}^{\text{cat}}$ with objective values sufficiently close to $f\left(\boldsymbol{x}_{(k)}\right)$
4. **Update**. Update poll parameters, check stopping criterion and set $k \leftarrow k + 1$
   If the iteration is successful, increase mesh and frame parameters and **GO TO** Step 1.
   Else, decrease mesh and frame parameters, and if the mesh parameters are at their minimum sizes, then **STOP**

All the parameters of CatMADS are described below. A DoE is done at the initialization,

allocating an initial budget of evaluations to explore the domain $\mathcal{X}$ before any optimization. The DoE must contain at least one initial point $\boldsymbol{x}_{(0)} \in \mathcal{X}$, but possibly outside Ω.

As with MADS, an iteration of CatMADS is said to be successful if, at any step, a new incumbent solution $\boldsymbol{x}_{(k+1)} \in \mathcal{X}$ is found. In that case, CatMADS proceeds directly to the update step. This early stopping strategy is said *opportunistic*, and it reduces the number of evaluations required in practice [26]. An iteration is said to be unsuccessful if no better solution is found.

In CatMADS, the search has the same purpose than MADS described in Section 4.1.3. The poll $P_{(k)}$ is the union of the categorical poll $P_{(k)}^{\text{cat}}$ and quantitative poll $P_{(k)}^{\text{qnt}}$. The categorical poll $P_{(k)}^{\text{cat}}$ replaces the discrete poll of MV-MADS, and it uses more flexible neighborhoods based on a distance defined in Section 4.2.1. Furthermore, in comparison to MV-MADS, the integer variables are not grouped with the categorical variables, but rather with the continuous variables to form quantitative variables. This modification allows to use the granular mesh of G-MADS [33] and efficiently treat integer variables. The quantitative poll $P_{(k)}^{\text{qnt}}$, detailed in Section 4.2.2, is an adaptation of the G-MADS algorithm for this work.

If both search and poll steps are unsuccessful, then the extended poll, adapted from [19], is deployed. This step performs quantitative polls around points in the categorical poll $P_{(k)}^{\text{cat}}$ whose objective value is worse than the incumbent value $f(\boldsymbol{x}_{(k)})$ (unsuccessful), but still considered sufficiently close to $f(\boldsymbol{x}_{(k)})$. This assessment of whether a point is sufficiently close to $f(\boldsymbol{x}_{(k)})$ is described in Section 4.2.4. The test is flexible, allowing the user to control its selectivity, including the option to reject all points. The extended poll is optional. CatMADS is designed so that its baseline convergence properties do not rely on this optional step. When the extended poll is activated, the theoretical guarantees are strengthened, leading to stronger notions of local optimality. This creates a hierarchy of convergence results that depends on the extended poll. The impact of the extended poll on convergence is clarified in Section 4.3.

4.2.1 Distanced-based categorical poll

The categorical poll $P_{(k)}^{\text{cat}}$ is characterized by two elements: (i) a categorical distance $d^{\text{cat}} : \mathcal{X}^{\text{cat}} \times \mathcal{X}^{\text{cat}} \to \mathbb{R}_+$ establishing the notion of proximity between categorical components, and (ii) a parameter $m_{(k)} \in \{1, 2, \ldots, |\mathcal{X}^{\text{cat}}|\}$, called the *number of neighbors*, determining how many categorical components are polled.

The quality of a categorical poll depends greatly on the distance employed. Basic categorical distances, such as Hamming distances, only compute mismatches between categories.

CatMADS does not prevent using these distances, but they may fail to provide meaningful comparisons. For example, if $x^{\text{cat}} \in \{\text{Red (R), Blue (B), Green (G)}\}$, then $d^{\text{cat}}(\text{R}, \text{B}) = 1 = d^{\text{cat}}(\text{R}, \text{G})$. With such distances, it is not always possible to determine which category is the closest to R. In CatMADS, effort needs to be spent on constructing problem-specific categorical distances that allow useful comparisons. In the computational experiments in Section 4.4, the categorical distance is built automatically by fitting a mixed-variable interpolation of the objective function f. When some knowledge of the problem is available, a user-defined distance assigning smaller values to similar categories may be appropriate.

A value of $m_{(k)} = 1$ implies the categorical poll is empty, since the neighborhood only contains the current categorical component. At the other opposite, $m_{(k)} = |\mathcal{X}^{\text{cat}}|$ signifies all categorical components, except the current solution $x_{(k)}$, belong to the categorical poll. Higher values allow more categorical component to be evaluated in the categorical poll, but at greater computational costs. The number of neighbors can be based by default on the number of categorical components in the problem, *e.g.*, $m_{(k)} = \lfloor |\mathcal{X}^{\text{cat}}| / 2 \rfloor + 1$, to scale with the problem. This is ultimately a user choice that should take into account the categorical distance. For instance, if mismatching-based distances are employed, higher values of $m_{(k)}$ may be preferable, as each neighbor is selected with limited information. In contrast, when more sophisticated problem-specific distances are used, then each neighbor is chosen in more informed manner, which may justify using lower values of $m_{(k)}$.

The following definition introduces the structure establishing neighborhoods for categorical variables.

Definition 14 (Distance-induced neighborhood)**.** *For a given $m \in \{1, 2, \ldots, |\mathcal{X}^{\text{cat}}|\}$, the* distance-induced neighborhood $\mathcal{N}(\boldsymbol{x}^{\text{cat}}; m) \subseteq \mathcal{X}^{\text{cat}}$ *is the set of m closest categorical components to the categorical component $\boldsymbol{x}^{\text{cat}} \in \mathcal{X}^{\text{cat}}$ with respect to a categorical distance $d^{\text{cat}} : \mathcal{X}^{\text{cat}} \times \mathcal{X}^{\text{cat}} \to \mathbb{R}_+$.*

Formally, $|\mathcal{N}(\boldsymbol{x}^{\text{cat}}; m)| = m$, and for any pair $\boldsymbol{u}, \boldsymbol{v} \in \mathcal{X}^{\text{cat}}$ such that $\boldsymbol{u} \in \mathcal{N}(\boldsymbol{x}^{\text{cat}}; m)$ and $\boldsymbol{v} \notin \mathcal{N}(\boldsymbol{x}^{\text{cat}}; m)$, the distance d^{cat} satisfies $d^{\text{cat}}(\boldsymbol{x}^{\text{cat}}, \boldsymbol{u}) \leq d^{\text{cat}}(\boldsymbol{x}^{\text{cat}}, \boldsymbol{v})$.

By definition, a categorical component $\boldsymbol{x}^{\text{cat}} \in \mathcal{X}^{\text{cat}}$ belongs to its own distance-induced neighborhood: $\boldsymbol{x}^{\text{cat}} \in \mathcal{N}(\boldsymbol{x}^{\text{cat}}; m)$ for any $m \in \{1, 2, \ldots, |\mathcal{X}^{\text{cat}}|\}$. The definition generalizes user-defined neighborhoods from [4], since it suffices to construct an appropriate categorical distance inducing the user-defined neighborhoods. The categorical poll is defined from distance-induced neighborhoods.

Definition 15 (Categorical poll)**.** *For a given incumbent solution $\boldsymbol{x}_{(k)} = \left(\boldsymbol{x}_{(k)}^{\text{cat}}, \boldsymbol{x}_{(k)}^{\text{qnt}}\right) \in \mathcal{X}$, the categorical poll $P_{(k)}^{\text{cat}} \subset \mathcal{X}$ contains points whose categorical components belong to the*

distance-induced neighborhood $\mathcal{N}\left(\boldsymbol{x}_{(k)}^{\text{cat}}; m_{(k)}\right)$*, such that*

$$P_{(k)}^{\text{cat}} \;:=\; \left(\mathcal{N}\left(\boldsymbol{x}_{(k)}^{\text{cat}}; m_{(k)}\right) \setminus \{\boldsymbol{x}_{(k)}^{\text{cat}}\}\right) \times \{\boldsymbol{x}_{(k)}^{\text{qnt}}\} \;\subset\; \mathcal{X}$$

where $\left|P_{(k)}^{\text{cat}}\right| = m_{(k)} - 1$.

In Definition 15, the categorical poll contains points of the domain, however the incumbent quantitative component $\boldsymbol{x}_{(k)}^{\text{qnt}} \in \mathcal{X}^{\text{qnt}}$ is fixed so that categorical polls effectively modify categorical components of incumbent solutions. The categorical poll satisfies $\left|P_{(k)}^{\text{cat}}\right| \in \{0, 1, \ldots, \mathcal{X}^{\text{cat}} - 1\}$ because the incumbent categorical component $\boldsymbol{x}_{(k)}^{\text{cat}}$ is removed from the distance-induced neighborhood to avoid reevaluation.

4.2.2 Quantitative poll and mesh

In G-MADS, the mesh discretizes the quantitative space, determines the potential points that can be generated by the quantitative poll. The mesh also controls the spacing between the incumbent solution and the points in these polls. A step size is assigned to each variable in order to control its granularity. For this work, the granular mesh allows to tackle simultaneously continuous and integer variables, referred to as the quantitative ones. The following redefines the granular mesh in [33].

Definition 16 (Quantitative mesh [33])**.** *For a given iteration* $k \geq 0$*, the* quantitative mesh*, centered at the quantitative component* $\boldsymbol{x}_{(k)}^{\text{qnt}} \in \mathcal{X}^{\text{qnt}}$*, discretizes the quantitative set* $\mathcal{X}^{\text{qnt}}$*, as follows*

$$M_{(k)}^{\text{qnt}} \;:=\; \left\{\boldsymbol{x}_{(k)}^{\text{qnt}} + \text{diag}\left(\boldsymbol{\delta}_{(k)}\right)\boldsymbol{z} \;:\; \boldsymbol{z} \in \mathbb{Z}^{n^{\text{qnt}}}\right\} \;\subset\; \mathcal{X}^{\text{qnt}} \tag{4.4}$$

where $\boldsymbol{\delta}_{(k)} := \left(\delta_{(k),1}, \delta_{(k),2}, \ldots, \delta_{(k),n^{\text{qnt}}}\right) \in \mathbb{R}_+^{n^{\text{qnt}}}$ *is the mesh size vector and* $\delta_{(k),i} \in \mathbb{R}_+$ *is the mesh size parameter of the* i*-th quantitative variable* $x_{(k),i}^{\text{qnt}} \in \mathcal{X}_i^{\text{qnt}}$.

In Definition 16, the step size of an integer variable belongs more precisely to set of integers $\mathbb{Z}$ and its minimal value is one. For clarity, Equation (4.4) from Definition 16 may be expanded as follows to distinguish integer and continuous variables

$$M_{(k)}^{\text{qnt}} := \left\{\boldsymbol{y}^{\text{qnt}} = \left(\boldsymbol{x}_{(k)}^{\text{int}} + \text{diag}\left(\boldsymbol{\delta}_{(k)}^{\text{int}}\right)\boldsymbol{z}^{\text{int}},\; \boldsymbol{x}_{(k)}^{\text{con}} + \text{diag}\left(\boldsymbol{\delta}_{(k)}^{\text{con}}\right)\boldsymbol{z}^{\text{con}}\right) \;:\; \boldsymbol{z}^{\text{int}} \in \mathbb{Z}^{n^{\text{int}}},\; \boldsymbol{z}^{\text{con}} \in \mathbb{Z}^{n^{\text{con}}}\right\}$$

where $\boldsymbol{\delta}_{(k)}^{\text{int}} \in \mathbb{N}^{n^{\text{int}}}$ and $\boldsymbol{\delta}_{(k)}^{\text{con}} \in \mathbb{R}_+^{n^{\text{con}}}$ are respectively the integer and continuous mesh size vectors.

In Definition 16, the quantitative mesh $M_{(k)}^{\text{qnt}}$ is a subset of the quantitative set $\mathcal{X}^{\text{qnt}}$, as it represents a discretization of the quantitative variables. Similarly to the categorical poll, the

quantitative poll contains point in the domain. Hence, to formulate the quantitative poll on points, this next definition extends Definition 16 to discretize the domain $\mathcal{X}$.

Definition 17 (Mesh [4]). *For a given iteration $k \geq 0$, the* mesh *that discretizes the domain $\mathcal{X}$ is formed as Cartesian product of the categorical set and the quantitative mesh, such that $M_{(k)} \coloneqq \mathcal{X}^{\text{cat}} \times M_{(k)}^{\text{qnt}} \subset \mathcal{X}$.*

Next, the quantitative poll is defined from the mesh. This poll uses the standard mechanisms of MADS. Mathematically, candidate points, which lie on the mesh, are constructed by translating a quantitative incumbent solution $\boldsymbol{x}_{(k)}^{\text{qnt}}$ with directions from a positive basis.

Definition 18 (Quantitative poll [4]). *For a given incumbent solution $\boldsymbol{x}_{(k)} = \left(\boldsymbol{x}_{(k)}^{\text{cat}}, \boldsymbol{x}_{(k)}^{\text{qnt}}\right) \in \mathcal{X}$, the* quantitative poll $P_{(k)}^{\text{qnt}} \subset \mathcal{X}$ *contains points whose quantitative components are generated from a positive spanning set of directions $\mathcal{D}_{(k)}$, such that*

$$P_{(k)}^{\text{qnt}} \coloneqq \left\{\left(\boldsymbol{x}^{\text{cat}}, \boldsymbol{x}_{(k)}^{\text{qnt}} + \operatorname{diag}\left(\boldsymbol{\delta}_{(k)}\right) \boldsymbol{d}\right) \in M_{(k)} \ : \ \boldsymbol{d} \in \mathcal{D}_{(k)}\right\} \subset \mathcal{X}.$$

Each direction $\boldsymbol{d} \in \mathcal{D}_{(k)}$, scaled by the mesh size vector $\boldsymbol{\delta}_{(k)}$, must be within the geometry of the frame, characterized by frame size vector $\boldsymbol{\Delta}_{(k)} \geq \boldsymbol{\delta}_{(k)}$, such that

$$-\boldsymbol{\Delta}_{(k)} \leq \operatorname{diag}\left(\boldsymbol{\delta}_{(k)}\right) \boldsymbol{d} \leq \boldsymbol{\Delta}_{(k)}, \tag{4.5}$$

where $\boldsymbol{\Delta}_{(k)} \coloneqq \left(\Delta_{(k),1}, \Delta_{(k),2}, \ldots, \Delta_{(k),n^{\text{qnt}}}\right) \in \mathbb{R}_+^{n^{\text{qnt}}}$ and $\Delta_{(k),i}$ is the frame size parameter of the i-th quantitative variable $x_{(k),i}^{\text{qnt}} \in \mathcal{X}_i^{\text{qnt}}$ for $i \in I^{\text{qnt}}$.

For the quantitative variables, the management of parameters is done variable-wise in order to consider the variable type and the scale of the variables. The frame sizes of an integer and continuous variables are respectively restricted to the following discrete sets $\Delta_{(k),i}^{\text{int}} \in \{a \times 10^b \ : \ a \in \{1, 2, 5\}, \ b \in \mathbb{N}\}$, with $\Delta_{(k),i}^{\text{int}} \geq 1$, for $i \in I^{\text{int}}$, and $\Delta_{(k),j}^{\text{con}} \in \{a \times 10^b \ : \ a \in \{1, 2, 5\}, \ b \in \mathbb{Z}\}$ for $j \in I^{\text{con}}$. A step size is always lower than or equal to its corresponding frame size, *i.e.*, $\delta_{(k),i}^t \leq \Delta_{(k),i}^t$. The frame and step sizes parameters, as well as their updating rules, are detailed in [33]. The frame and step sizes are decreased on unsuccessful iterations, and increased on successful ones. A step size must decrease toward zero more rapidly than its frame size to ensure convergence: typically, it decreases toward zero with an order of the square of its frame size [26, 33]. The initialization of the frame and step sizes considers the bounds and the initial values of $\boldsymbol{x}_{(0)}$ or a DoE: see [32, 33] for more details.

By default, G-MADS uses Householder transformations to construct positive spanning set of directions $\mathcal{D}_{(k)}$. At each iteration k, a random and normalized vector $\boldsymbol{v}_{(k)} \in \mathbb{R}^{n^{\text{qnt}}}$ is

generated. From $\boldsymbol{v}_{(k)}$, the Householder transformation $I - 2\boldsymbol{v}_{(k)}\boldsymbol{v}_{(k)}^\top$ is done. Afterwards, each column of the matrix is used to create two opposing directions. These directions are rescaled with the frame and step sizes parameters to ensure that the inequality in (4.5) is respected. This ensures that the poll points belong to the mesh and reside within the frame, as in Figure 4.1.

4.2.3 Domination and constraint handling with the progressive barrier

The strategy to handle the constraints, called the progressive barrier (PB) [22], is now described. The PB works with two incumbent solutions, instead of a single one $\boldsymbol{x}_{(k)} \in \mathcal{X}$ as previously: the feasible incumbent solution $\boldsymbol{x}_{\text{FEA}} \in \Omega$ and the infeasible incumbent solution $\boldsymbol{x}_{\text{INF}} \in \mathcal{X} \setminus \Omega$. For readability, the subscript k is discarded when subscripts FEA and INF are used. The key idea of the PB is to perform two polls. The *feasible poll* is centered at $\boldsymbol{x}_{\text{FEA}}$ and is done as described in previous sections. The *infeasible poll* is centered at $\boldsymbol{x}_{\text{INF}}$, and this solution is updated to progressively approach feasibility throughout the iterations.

In Problem (4.1), recall that a point $\boldsymbol{x} \in \mathcal{X}$ is feasible if and only if all the constraints are respected. Hence, a point $\boldsymbol{y} \in \mathcal{X}$ is infeasible if at least one constraint is not respected, *i.e.*, there exists $j \in J$ with $g_j(\boldsymbol{x}) > 0$. The infeasibility of a point does not necessarily imply that it is not promising. In fact, optimal solutions often lie on the boundary of the feasible set. In order to quantify the infeasibility of a point, the constraint violation function $h : \mathcal{X} \to \overline{\mathbb{R}}$ is introduced as follows

$$h(\boldsymbol{x}) \coloneqq \begin{cases} \sum\limits_{j \in J} \left(\max\{0, g_j(\boldsymbol{x})\}\right)^2 & \text{if } \boldsymbol{x} \in \mathcal{X}, \\ \infty & \text{otherwise} \end{cases} \tag{4.6}$$

with $h(\boldsymbol{x}) = 0 \Leftrightarrow \boldsymbol{x} \in \Omega$ (feasible) and $h(\boldsymbol{x}) > 0 \Leftrightarrow \boldsymbol{x} \in \mathcal{X} \setminus \Omega$ (infeasible). A point $\boldsymbol{x} \notin \mathcal{X}$ is valued as $h(\boldsymbol{x}) = \infty$, which conveniently outlines that it is not a solution considered.

The PB introduces a threshold parameter $h_{(k)}^{\max}$, called the *barrier*. This parameter controls the acceptable infeasibility of $\boldsymbol{x}_{\text{INF}}$. More precisely, the feasible and infeasible solutions are determined with the following expressions:

$$\boldsymbol{x}_{\text{FEA}} \in \operatorname{argmin}\left\{f(\boldsymbol{x}) \,:\, \boldsymbol{x} \in \Omega \cap \mathbb{X}\right\} \quad \text{and} \quad \boldsymbol{x}_{\text{INF}} \in \operatorname{argmin}\left\{f(\boldsymbol{x}) \,:\, 0 < h(\boldsymbol{x}) \leq h_{(k)}^{\max},\, \boldsymbol{x} \in \mathbb{X}\right\} \tag{4.7}$$

where $\mathbb{X} \subset \mathcal{X}$, called the *set of known points*, is the set of points whose objective and constraint functions were evaluated since the algorithm was launched. The barrier is initialized at $h_{(0)}^{\max} = \infty$ and it is reduced iteratively toward zero.

The update of the barrier and the mesh parameters is done with notions of dominance [26].

Definition 19 (Feasible and infeasible dominance)**.** *The point* $\boldsymbol{x} \in \Omega$ *is said to* dominate $\boldsymbol{y} \in \Omega$*, denoted* $\boldsymbol{x} \prec_f \boldsymbol{y}$*, if* $f(\boldsymbol{x}) < f(\boldsymbol{y})$*. The point* $\boldsymbol{x} \in \mathcal{X} \setminus \Omega$ *is said to dominate* $\boldsymbol{y} \in \mathcal{X} \setminus \Omega$*, denoted* $\boldsymbol{x} \prec_h \boldsymbol{y}$*, if* $f(\boldsymbol{x}) \leq f(\boldsymbol{y})$ *and* $h(\boldsymbol{x}) \leq h(\boldsymbol{y})$ *with a least one strict inequality.*

In the previous sections without constraints, an iteration is either successful or unsuccessful. In the constrained case, a successful iteration is divided in two cases based on Definition 19:

1. an iteration is *dominating* if either a candidate point $\boldsymbol{y} \in \Omega$ with $\boldsymbol{y} \prec_f \boldsymbol{x}_{\text{FEA}}$ or a candidate point $\boldsymbol{y} \in \mathcal{X} \setminus \Omega$ with $\boldsymbol{y} \prec_h \boldsymbol{x}_{\text{INF}}$ is found at any step. The barrier is updated as $h^{\max}_{(k+1)} := h(\boldsymbol{x}_{\text{INF}})$, and the mesh and frame sizes parameters are increased.
2. an iteration is *improving* if it is not dominating and if a candidate point $\boldsymbol{y} \in \mathcal{X} \setminus \Omega$ that improves the constraint violation function, such that $0 < h(\boldsymbol{y}) < h(\boldsymbol{x}_{\text{INF}})$ is found at the end of an iteration. In this case, the mesh and frame parameters are unchanged, and the barrier is updated after going through all the steps with $h_{(k+1)} := \max\{h(\boldsymbol{v}) \ : \ h(\boldsymbol{v}) < h(\boldsymbol{x}_{\text{INF}}),\ \boldsymbol{v} \in \mathbb{X}\}$.

The opportunistic strategy is reused with the PB, however it can only be applied to dominating iterations. An iteration is unsuccessful if it is not dominating or improving. In that case, the parameters of G-MADS are decreased and the barrier is unchanged.

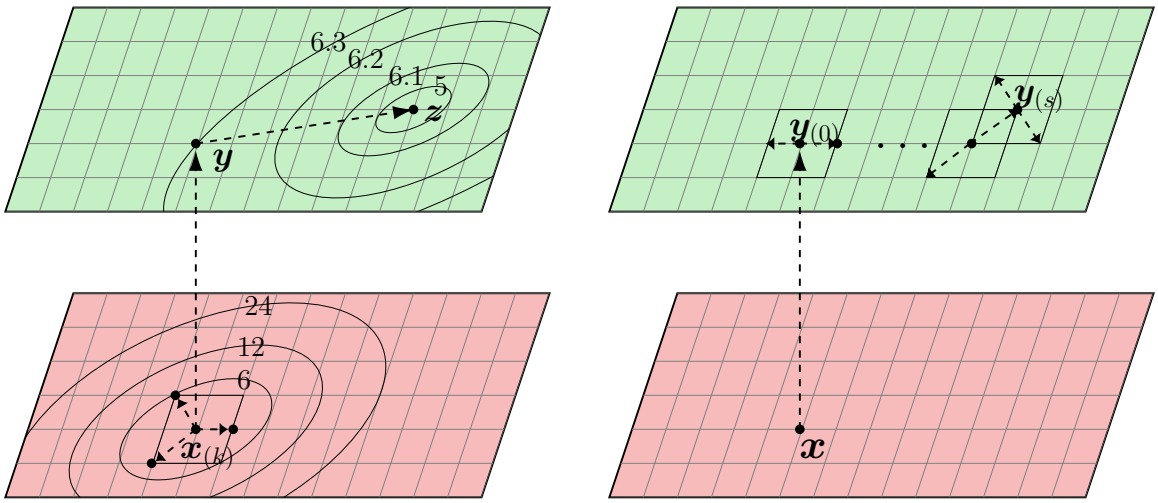
With the PB, the poll set $P_{(k)}$ from Algorithm 4 becomes $P_{\text{FEA}} \cup P_{\text{INF}}$, where the feasible poll set P_{FEA} and infeasible poll set P_{INF} are respectively centered at the feasible and infeasible solutions. The infeasible and feasible polls both have their own quantitative and categorical polls. The feasible and infeasible categorical polls have their own number of neighbors. Similarly, the feasible and infeasible quantitative polls have their own set of directions.

The integration of the PB within CatMADS is achieved by 1) considering improving iterations, 2) performing two polls instead of one and 3) generalizing the extended poll from [19] to use the PB instead of the extreme barrier, as detailed in the next section.

4.2.4 Extended poll

The extended poll is optionally performed following unsuccessful search and poll steps. Figure 4.2a illustrates the extended poll on the RG example in which the red and green planes represent two quantitative variables in $\mathbb{R}^2$ corresponding to the two colors (categories).

In the figure, the incumbent solution $\boldsymbol{x}_{(k)}$ is in the red plane. Both the quantitative poll and categorical poll at $\boldsymbol{y} \in P^{\text{cat}}_{(k)}$ in the green plane fail to improve $\boldsymbol{x}_{(k)}$. The extended poll



(a) Extended poll initiated at $\boldsymbol{y} \in P_{(k)}^{\text{cat}}$, successfully terminating at $\boldsymbol{z}$ where $4 = f(\boldsymbol{z}) < f(\boldsymbol{x}_{(k)}) = 5$.

(b) The sequence of extended poll points initiated at $y = \boldsymbol{y}_{(0)}$, and terminating at $\boldsymbol{z} = \boldsymbol{y}_{(s)}$.

Figure 4.2 The RG example for the extended poll with the mesh in gray.

initiated at $\boldsymbol{y} \in P_{(k)}^{\text{cat}}$ generates a series of quantitative polls in the green plane, successfully terminating at $\boldsymbol{z}$ with $4 = f(\boldsymbol{z}) < f(\boldsymbol{x}_{(k)}) = 5$.

An important drawback of the extended poll is its potential high costs, especially when dealing with many variables. To mitigate these costs, points from the categorical poll that undergo quantitative polls are selected based on a test. The test accepts points whose objective value is worse, but sufficiently close to an incumbent value.

Definition 20 (ξ-extended poll trigger)**.** *For a given parameter $\xi \in \overline{\mathbb{R}}$, the ξ-extended poll trigger function $t_\xi : \mathcal{X} \times \mathcal{X} \to \{0, 1\}$ at the unsuccessful poll point $\boldsymbol{y} \in P_{(k)}^{\text{cat}} = P_{FEA}^{\text{cat}} \cup P_{INF}^{\text{cat}}$ is defined as*

$$t_\xi\left(\boldsymbol{x}_{(k)}, \boldsymbol{y}\right) := \begin{cases} 1 & \text{if } 0 \leq f\left(\boldsymbol{y}\right) - f\left(\boldsymbol{x}_{(k)}\right) \leq \xi \left| f\left(\boldsymbol{x}_{(k)}\right)\right|, \\ 0 & \text{otherwise.} \end{cases} \tag{4.8}$$

The parameter $\xi \in \overline{\mathbb{R}}$ has the following impact:

- if $\xi < 0$, then no point passes the test, since the two inequalities are mutually exclusive;
- if $\xi = \infty$, then all points trivially pass the test;
- if $\xi \in \mathbb{R}_+$, the test checks if the relative deterioration is within a ratio of ξ.

In CatMADS, the extended poll can be deactivated by setting $\xi < 0$. In MV-MADS, it is always activated and the default implementation proposed is $\xi = 0.05$, meaning that a deterioration of 5% on the best feasible value is accepted.

In this work, the extended poll is generalized with the PB containing two incumbent solutions. To take into account the PB, the ξ-extended poll triggers are performed by considering wether the unsuccessful points are feasible or not. If an unsuccessful point $\boldsymbol{y} \in P_{(k)}^{\text{cat}}$ is feasible, then the ξ-extended poll trigger is performed with the feasible incumbent $\boldsymbol{x}_{\text{FEA}}$. Otherwise, it is performed with the infeasible solution $\boldsymbol{x}_{\text{INF}}$, and in that case, the point must also respect the barrier to be selected. Mathematically, the set of selected points undergoing extended polls is expressed as

$$\mathcal{E}_{(k)} \coloneqq \left\{ \boldsymbol{y} \in P_{(k)}^{\text{cat}} \ : \ \left(t_\xi\left(\boldsymbol{x}_{\text{FEA}}, \boldsymbol{y}\right) = 1 \text{ and } h\left(\boldsymbol{y}\right) = 0\right) \text{ or } \left(t_\xi\left(\boldsymbol{x}_{\text{INF}}, \boldsymbol{y}\right) = 1 \text{ and } 0 < h\left(\boldsymbol{y}\right) \leq h_{(k)}^{\max}\right) \right\}. \tag{4.9}$$

where $h(\boldsymbol{y}) = 0$ signifies that $\boldsymbol{y}$ is feasible.

Once selected for the extended poll, a point $\boldsymbol{y} \in \mathcal{E}_{(k)}$ undergoes a finite sequence of quantitative polls initiated around it. A quantitative poll is performed until a new incumbent solution is opportunistically found, or until it does not produce a dominating point. This is formalized in the next definition.

Definition 21 (Extended poll initiated at $\boldsymbol{y}$ [19])**.** *For a given iteration $k > 0$, the* extended poll initiated at $\boldsymbol{y} \coloneqq \boldsymbol{y}_{(0)} \in \mathcal{E}_{(k)}$ *produces a finite sequence of iterates $\boldsymbol{y}_{(0)}, \boldsymbol{y}_{(1)}, \boldsymbol{y}_{(2)}, \ldots, \boldsymbol{y}_{(s)}$, such that $\boldsymbol{y}_{(j)} \prec \boldsymbol{y}_{(j-1)}$ for all $j \in S \coloneqq \{1, 2, \ldots, s\}$ and $\boldsymbol{x}_{(k)} \prec \boldsymbol{y}$. The operator $\prec$ and the incumbent solution $\boldsymbol{x}_{(k)}$ are either:*

- *the feasible dominance $\prec_f$ and the feasible incumbent solution $\boldsymbol{x}_{FEA} \in \Omega$, if $\boldsymbol{y} \in \Omega$;*
- *or the infeasible dominance $\prec_h$ and the infeasible incumbent solution $\boldsymbol{x}_{INF} \in \mathcal{X} \setminus \Omega$, if $\boldsymbol{y} \in \mathcal{X} \setminus \Omega$.*

The point $\boldsymbol{z} \coloneqq \boldsymbol{y}_{(s)} \in \mathcal{X}$, called an endpoint*, is either a new incumbent solution or the quantitative poll at $\boldsymbol{z}$ fails to produce a dominating point, such that $\boldsymbol{w} \not\prec \boldsymbol{z}$ for all $\boldsymbol{w} \in \left\{\left(\boldsymbol{y}^{\text{cat}}, \boldsymbol{z}^{\text{qnt}} + diag\left(\boldsymbol{\delta}_{(k)}\right)\boldsymbol{d}\right) \ : \ \boldsymbol{d} \in \mathcal{D}\left(\boldsymbol{z}\right)\right\}$, where $\mathcal{D}\left(\boldsymbol{z}\right)$ is the set of directions used for $\boldsymbol{z}$.*

From Definition 21, it follows that, for a given iteration $k > 0$, an extended poll initiated at $\boldsymbol{y} \in \mathcal{E}_{(k)}$ generates an union of quantitative polls performed at $\boldsymbol{y}_{(0)}, \boldsymbol{y}_{(1)}, \ldots, \boldsymbol{y}_{(s)}$, denoted

$\mathcal{E}_{(k)}(\boldsymbol{y})$ and expressed as

$$\mathcal{E}_{(k)}(\boldsymbol{y}) \coloneqq \bigcup_{j\in S} \Big\{ \left(\boldsymbol{y}^{\text{cat}}, \boldsymbol{y}^{\text{qnt}}_{(j)} + \text{diag}\left(\boldsymbol{\delta}_{(k)}\right)\boldsymbol{d}\right) \in M_{(k)} : \boldsymbol{d} \in \mathcal{D}\left(\boldsymbol{y}_{(j)}\right), \\ \boldsymbol{y}^{\text{qnt}}_{(j)} = \boldsymbol{y}^{\text{qnt}}_{(j-1)} + \text{diag}\left(\boldsymbol{\delta}_{(k)}\right)\bar{\boldsymbol{d}} \text{ for some } \bar{\boldsymbol{d}} \in \mathcal{D}\left(\boldsymbol{y}_{(j-1)}\right), \\ \boldsymbol{y}_{(j)} \prec \boldsymbol{y}_{(j-1)} \Big\} \subset \mathcal{X}.$$

In Figure 4.2b, the RG example is reused to illustrate Definition 21. The sequence of polls is initiated at $\boldsymbol{y} \in P^{\text{cat}}_{(k)}$ and yields improvements until the quantitative polls produces a new incumbent solution $\boldsymbol{z}$. The quantitative polls have their own sets of two directions. In comparison to quantitative polls in Definition 18, extended polls do not require a positively spanning set of directions. This allows using fewer directions, thereby reducing their cost. In the example, the mesh is not increased with improvements of the extended poll. In theory, it could be increased, as long as the extended poll ends with the initial mesh and frame sizes.

From Definition 21 and the sets of selected points in (4.9), the extended poll $\bar{P}_{(k)}$ at iteration k is defined as the union of extended polls initiated at the selected points, such that

$$\bar{P}_{(k)} \coloneqq \bigcup_{\boldsymbol{y}\in\mathcal{E}_{(k)}} \mathcal{E}_{(k)}(\boldsymbol{y}) \tag{4.10}$$

where $\mathcal{E}_{(k)}(\boldsymbol{y})$ in Definition 21 adjusts its operator $\prec$ to either $\prec_f$ or $\prec_h$, depending on whether $\boldsymbol{y}$ is feasible or not.

The extended poll can be viewed as a mixed-variable poll since it proposes points that modify both the categorical and quantitative variables of incumbent solutions. In CatMADS, these polls offer additional optimization mechanisms that balance evaluation costs and local solution quality. This becomes apparent in the next section, as they improve convergence guarantees at some additional cost.

4.3 Convergence analysis

The convergence analysis of CatMADS follows the structure used in [4, 21]. It relies on standard assumptions: an initial point $\boldsymbol{x}_{(0)} \in \mathcal{X}$ with $f\left(\boldsymbol{x}_{(0)}\right) \in \mathbb{R}$ is available, and all iterates $\{\boldsymbol{x}_{(k)}\}$ lie in a compact set. The results concern local optimality, as CatMADS is a local method. The strongest theorems provide necessary optimality conditions, and nothing about global optimality is claimed. The convergence analysis is divided into four parts. First, different types of local minima of varying strengths are presented. Second, Section 4.3.2 redefines and shows the existence of a refining subsequence. Third, convergence results with respect to the continuous variables are derived in Section 4.3.3, using notions of nonsmooth

calculus. Finally, Section 4.3.4 strengthens the convergence results by studying the extended poll.

4.3.1 Types of mixed-variable local minima

In the following, an open ball for the continuous variables is denoted $\mathcal{B}(\boldsymbol{u};\epsilon) \coloneqq \{\boldsymbol{v} \in \mathbb{R}^{n^{\text{con}}} : \|\boldsymbol{u}-\boldsymbol{v}\|_2 < \epsilon\}$, and a neighborhood for integer variables is defined as $\mathcal{I}(\boldsymbol{u};\eta) \coloneqq \{\boldsymbol{v} \in \mathbb{Z}^{n^{\text{int}}} : \|\boldsymbol{u}-\boldsymbol{v}\|_1 \leq \eta\}$. In this work, a local minimum adapts the definitions from [19,167], originally stated for continuous variables and discrete variables with user-defined neighborhoods. The new definition is formulated for categorical, integer and continuous variables and relies on distance-induced neighborhoods, as follows.

Definition 22 (Local minimum)**.** *A point* $\boldsymbol{x} = (\boldsymbol{x}^{\text{cat}}, \boldsymbol{x}^{\text{int}}, \boldsymbol{x}^{\text{con}}) \in \Omega$ *is a* mixed-variable local minimum *if there exists a triplet* $m \in \mathbb{N}^*$, $\eta \in \mathbb{N}^*$, $\epsilon \in \mathbb{R}_+^*$, *such that*

$$f(\boldsymbol{x}) \leq f(\boldsymbol{y}), \quad \textit{for all } \boldsymbol{y} \in (\mathcal{N}(\boldsymbol{x}^{\text{cat}};m) \times \mathcal{I}(\boldsymbol{x}^{\text{int}};\eta) \times \mathcal{B}(\boldsymbol{x}^{\text{con}};\epsilon)) \cap \Omega.$$

In Figure 4.3, a local minimum is schematized for the *RBG example* containing a single categorical variable $x^{\text{cat}} \in \{\text{Red}, \text{Green}, \text{Blue}\}$, a single integer variable $x^{\text{int}} \in \{0,1,2\}$ and no relaxable constraint, *i.e.*, $\Omega = \mathcal{X}$. The planes represent the set $\mathcal{X}^{\text{con}} \subset \mathbb{R}^2$ for fixed categorical and integer variables. The gray zones depicts continuous balls (unconstrained) and dotted arrows represent the couples of categorical and integer components involve in the minimum.

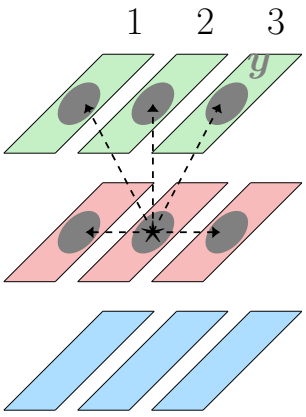


Figure 4.3 A local minimum $\star = (\text{Red}, 2, \boldsymbol{x}_\star^{\text{con}})$ for the RBG example, where $\boldsymbol{x}^{\text{con}} \in \mathcal{X}^{\text{con}} \subset \mathbb{R}^2$, $m = 1$ and $\eta = 1$.

The local minimum, as described in Definition 22, is not covered by CatMADS. In practice, the evaluations required for reaching optimality could be substantial, especially in a context

with expensive-to-evaluate blackboxes. Indeed, the Cartesian product of the distance-induced neighborhood and integer neighborhood in Definition 22 yields $(2n^{\text{int}}+1)\times m$ combinations of categorical and integer components, and each combination would require a local optimization with respect to the continuous variables. In Figure 4.3, this corresponds to optimizing continuous variables in each grey ellipse, *i.e.*, for each pair of categorical and integer values in their respective neighborhoods. In contexts with few integer and categorical variables, and where function evaluations are not expensive, a subproblem approach that performs continuous optimization for each pair of integer and categorical components could be suitable. This approach could lead to convergence guarantees targeting Figure 4.3. That said, CatMADS is not designed for this specific setting. To balance computational costs and solution quality, three weaker types of local minima are proposed.

The new types of local minima are named with a specific nomenclature in which the types of variables are combined by a hyphen or separated by a comma. A hyphen symbolizes that the types of variables are jointly interacting. For instance, the nomenclature of the local minimum in Definition 22 is $[\text{cat}-\text{int}-\text{con}]$, since all types of variables are jointly interacting via Cartesian products. The weaker types of local minima discard some or all of these joint interactions, meaning commas appear in their nomenclature. Notably, the weakest type of local minima is named [cat, int, con], which signifies that all types of variables are separated. Inspired by [252], the $[\text{cat, int, con}]$ minimum is defined below.

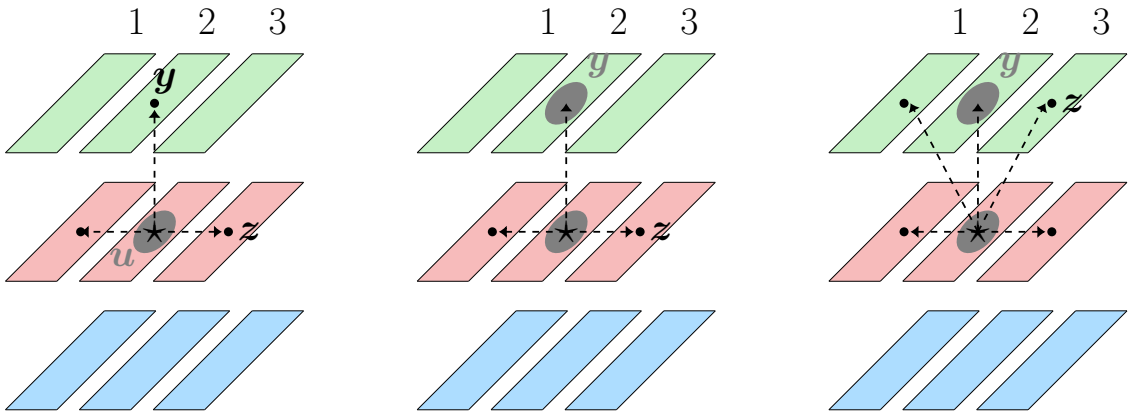
Definition 23 ($[\text{cat, int, con}]$ local minimum)**.** *A point* $\boldsymbol{x}=(\boldsymbol{x}^{\text{cat}},\boldsymbol{x}^{\text{int}},\boldsymbol{x}^{\text{con}})\in\Omega$ *is said to be a* $[\text{cat, int, con}]$ local minimum *if there exists a triplet* $m\in\mathbb{N}^*$, $\eta\in\mathbb{N}^*$, $\epsilon\in\mathbb{R}_+^*$, *such that*

$$
\begin{aligned}
&f(\boldsymbol{x})\leq f(\boldsymbol{y}), \quad \textit{for all } \boldsymbol{y}^{\text{cat}}\in\mathcal{N}\left(\boldsymbol{x}^{\text{cat}};m\right) \quad \textit{with } \boldsymbol{y}=(\boldsymbol{y}^{\text{cat}},\boldsymbol{x}^{\text{int}},\boldsymbol{x}^{\text{con}})\in\Omega,\\
&f(\boldsymbol{x})\leq f(\boldsymbol{z}), \quad \textit{for all } \boldsymbol{z}^{\text{int}}\in\mathcal{I}\left(\boldsymbol{x}^{\text{int}};\eta\right) \quad \textit{with } \boldsymbol{z}=(\boldsymbol{x}^{\text{cat}},\boldsymbol{z}^{\text{int}},\boldsymbol{x}^{\text{con}})\in\Omega,\\
&f(\boldsymbol{x})\leq f(\boldsymbol{u}), \quad \textit{for all } \boldsymbol{u}^{\text{con}}\in\mathcal{B}\left(\boldsymbol{x}^{\text{con}};\epsilon\right) \quad \textit{with } \boldsymbol{u}=(\boldsymbol{x}^{\text{cat}},\boldsymbol{x}^{\text{int}},\boldsymbol{u}^{\text{con}})\in\Omega.
\end{aligned}
$$

In Definition 23, each inequality is associated with a specific type of variables, meaning that there is no interaction between different types of variables. The $[\text{cat, int, con}]$ minimum is visualized for the RBG example in Figure 4.4a. For this type of local minimum, the vertical dotted line represents the point in the distance-induced neighborhood, whereas the horizontal dotted lines represent the integer neighborhood.

The next type of local minimum is $[\text{cat}-\text{con, int}]$ with joint interactions between categorical and continuous variables.

Definition 24 ($[\text{cat}-\text{con, int}]$ local minimum)**.** *A point* $\boldsymbol{x}=(\boldsymbol{x}^{\text{cat}},\boldsymbol{x}^{\text{int}},\boldsymbol{x}^{\text{con}})\in\Omega$ *is said to be a* $[\text{cat}-\text{con, int}]$ local minimum *if there exists a triplet* $m\in\mathbb{N}^*$, $\eta\in\mathbb{N}^*$, $\epsilon\in\mathbb{R}_+^*$, *such*



(a) [cat, int, con] local minimum. (b) [cat – con, int] local minimum. (c) [cat – con, cat – int] local minimum.

Figure 4.4 Three new types of local minima at $\star = (\text{Red}, 2, \boldsymbol{x}^{\text{con}}_{\star})$ for the RBG example, where $\boldsymbol{x}^{\text{con}} \in \mathcal{X}^{\text{con}} \subset \mathbb{R}^2$, $m = 1$ and $\eta = 1$.

that

$$
\begin{array}{llll}
f(\boldsymbol{x}) \leq f(\boldsymbol{y}), & \textit{for all } \left(\boldsymbol{y}^{\text{cat}}, \boldsymbol{y}^{\text{con}}\right) \in \mathcal{N}\left(\boldsymbol{x}^{\text{cat}}; m\right) \times \mathcal{B}\left(\boldsymbol{x}^{\text{con}}; \epsilon\right) & \textit{with } \boldsymbol{y} = \left(\boldsymbol{y}^{\text{cat}}, \boldsymbol{x}^{\text{int}}, \boldsymbol{y}^{\text{con}}\right) \in \Omega, \\
f(\boldsymbol{x}) \leq f(\boldsymbol{z}), & \textit{for all } \quad \boldsymbol{z}^{\text{int}} \quad \in \mathcal{I}\left(\boldsymbol{x}^{\text{int}}; \eta\right) & \textit{with } \boldsymbol{z} = \left(\boldsymbol{x}^{\text{cat}}, \boldsymbol{z}^{\text{int}}, \boldsymbol{x}^{\text{con}}\right) \in \Omega.
\end{array}
$$

The [cat – con, int] local minimum type is schematized for the RBG example in Figure 4.4b. In comparison to the [cat, int, con] local minimum type, the [cat – con, int] type strengthens optimality with a joint interaction in the first inequality. In Figure 4.4b, this interaction is visualized at $\boldsymbol{y}$, which has a continuous ball around it.

The last type of local minimum is [cat – con, cat – int] with interactions between the categorical and continuous variables, and between the categorical and integer variables.

Definition 25 ([cat – con, cat – int] local minimum)**.** *A point* $\boldsymbol{x} = (\boldsymbol{x}^{\text{cat}}, \boldsymbol{x}^{\text{int}}, \boldsymbol{x}^{\text{con}}) \in \Omega$ *is said to be a* [cat – con, cat – int] local minimum *if there exists a triplet* $m \in \mathbb{N}^*$, $\eta \in \mathbb{N}^*$, $\epsilon \in \mathbb{R}^*_+$, *such that*

$$
\begin{array}{llll}
f(\boldsymbol{x}) \leq f(\boldsymbol{y}), & \textit{for all } \left(\boldsymbol{y}^{\text{cat}}, \boldsymbol{y}^{\text{con}}\right) \in \mathcal{N}\left(\boldsymbol{x}^{\text{cat}}; m\right) \times \mathcal{B}\left(\boldsymbol{x}^{\text{con}}; \epsilon\right) & \textit{with } \boldsymbol{y} = \left(\boldsymbol{y}^{\text{cat}}, \boldsymbol{x}^{\text{int}}, \boldsymbol{y}^{\text{con}}\right) \in \Omega, \\
f(\boldsymbol{x}) \leq f(\boldsymbol{z}), & \textit{for all } \left(\boldsymbol{z}^{\text{cat}}, \boldsymbol{z}^{\text{int}}\right) \in \mathcal{N}\left(\boldsymbol{x}^{\text{cat}}; m\right) \times \mathcal{I}\left(\boldsymbol{x}^{\text{int}}; \eta\right) & \textit{with } \boldsymbol{z} = \left(\boldsymbol{z}^{\text{cat}}, \boldsymbol{z}^{\text{int}}, \boldsymbol{x}^{\text{con}}\right) \in \Omega.
\end{array}
$$

The [cat – con, cat – int] minimum further strengthens the optimality of the [cat – con, int] type by adding interactions between categorical and integer variables. In Figure 4.4c,

this additional interaction is represented at $\boldsymbol{z}$, which modifies the categorical and integer variables of $\boldsymbol{x}_\star$.

This subsection concludes with two remarks regarding the local minima. First, in the three new types of local minima, only the integer component of the minimum interacts with the continuous variables. In other words, all integer components $\boldsymbol{y}^{\text{int}} \in \mathcal{I}(\boldsymbol{x}^{\text{int}}; \eta) \setminus \{\boldsymbol{x}^{\text{int}}\}$ do not interact with the continuous variables: in Figure 4.4, these components are represented as dark points without continuous balls. This choice reduces the evaluation cost considerably, while still exploiting the ordered and quantitative nature of integer variables when characterizing the minima.

Second, if a user wishes to obtain local convergence guarantees with continuous balls around all discrete components in the neighborhood, it suffices to model the integer variables as categorical variables. The resulting local minimum is equivalent to the one used in MV-MADS or in [167] for mixed-variable problems, except that discrete components are selected via a categorical distance and the neighborhood size is controlled by m. With this approach, integer variables become part of the categorical neighborhood, and the choice of m becomes more delicate, since it determines how many integer and categorical components are included in a minimum. This can be problematic, because small values of m may ignore many integer components, whereas large values of m can substantially increase the evaluation cost. From an optimization standpoint, this modeling choice is generally not recommended by the authors as it treats integer variables as categorical variables.

4.3.2 Refining subsequence

This part of the convergence analysis studies the behavior of sequences of iterates produced by CatMADS. As for G-MADS [33], convergence is established via unsuccessful iteration subsequences in which the step sizes of continuous variables get infinitely fine, and step sizes of integer variables reduce to the value one. For CatMADS, these subsequences must also possess convergent distance-induced neighborhoods. As a generalization of G-MADS, the convergence of the step and frame size parameters is directly inherited from G-MADS. This leads to this key definition of the convergence analysis.

Definition 26 (Refining subsequence and refined point)**.** *Let U be the set of indices of the unsuccessful iterations generated by an instance of* CatMADS*. If K is an infinite subset of U*

such that

$$\begin{cases} \lim_{k\in K} \boldsymbol{x}_{(k)} & = \boldsymbol{x}_\star & \text{for some } \boldsymbol{x}_\star \in \mathcal{X}, \\ \lim_{k\in K} \delta^{\text{con}}_{(k),i} & = 0 & \text{for all } i \in I^{\text{con}}, \\ \lim_{k\in K} \delta^{\text{int}}_{(k),j} & = 1 & \text{for all } j \in I^{\text{int}}, \\ \lim_{k\in K} m_{(k)} & = m_\star & \\ \lim_{k\in K} \mathcal{N}\left(\boldsymbol{x}^{\text{cat}}_{(k)}; m_{(k)}\right) & = \mathcal{N}(\boldsymbol{x}^{\text{cat}}_\star; m_\star) & \end{cases}$$

then the subsequence $\{\boldsymbol{x}_{(k)}\}_{k\in K}$ *of poll centers with* $\boldsymbol{x}_{(k)} \in M_{(k)}$ *is said to be a* refining subsequence*, and* $\boldsymbol{x}_\star \in \mathcal{X}$ *is said to be a* refined point*.*

In Definition 26, the limits $\lim_{k\in K} \delta^{\text{con}}_{(k),i} = 0$ for all $i \in I^{\text{con}}$ and $\lim_{k\in K} \delta^{\text{int}}_{(k),j} = 1$ for all $j \in I^{\text{int}}$ signify, respectively, that the discretization of continuous variables asymptotically vanishes and integer variables reach unit granularity. The limit $\lim_{k\in K} m_{(k)} = m_\star$ means the size of the distance-induced neighborhoods converges. Since $m_{(k)} \in \{1, 2, 3 \ldots, \mathcal{X}^{\text{cat}}\}$ is a user-controlled parameter, this must be ensured by design. A simple approach is to fix $m_{(k)}$ for some iteration $k > 0$. More sophisticated mechanisms can be implemented, such as a monotone schedule that starts large and decreases with the remaining evaluation budget. Note that if $m_\star = 1$, only the categorical component of the refined point $\boldsymbol{x}_\star$ is considered, so no guarantees for the categorical variables follow. Choosing $m_\star > 1$ ensures that a refined point $\boldsymbol{x}_\star$ has guarantees for the categorical variables: this choice should be based on the budget of evaluation available. The last limit concerns the convergence of distance-induced neighborhoods. Once $\lim_{k\in K} m_{(k)} = m_\star$ is ensured, then the limit $\lim_{k\in K} \mathcal{N}\left(\boldsymbol{x}^{\text{cat}}_{(k)}; m_{(k)}\right) = \mathcal{N}(\boldsymbol{x}^{\text{cat}}_\star; m_\star)$ follows directly. Finally, it is the PB that determines if a refining subsequence converges to a feasible refined point or not.

From Definition 26, the next *zeroth-order* result of Theorem 3 is proposed with only the two assumptions: an initial point $\boldsymbol{x}_{(0)} \in \mathcal{X}$ with $f\left(\boldsymbol{x}_{(0)}\right) \in \mathbb{R}$ is available and all iterates $\{\boldsymbol{x}_{(k)}\}$ lie on a compact set $\mathcal{X}$. Since CatMADS is built upon G-MADS, the proof of the next theorem is partly given by construction. The rest of the proof sequentially chooses infinite subsequence that satisfy the requirements of a refining subsequence [21].

Theorem 3 (Zeroth-order)**.** *Let* $\{\boldsymbol{x}_{(k)}\}$ *be the sequence of points generated by* CatMADS *from an initial point* $\boldsymbol{x}_{(0)} \in \mathcal{X}$ *with* $f\left(\boldsymbol{x}_{(0)}\right) \in \mathbb{R}$*. If all iterates* $\{\boldsymbol{x}_{(k)}\}$ *belong to a compact set, then there exists a refining subsequence* $\{\boldsymbol{x}_{(k)}\}_{k\in K}$ *with corresponding refined point* $\boldsymbol{x}_\star \in \mathcal{X}$*.*

Proof. Let U be the set of indices of unsuccessful iterations generated by CatMADS. For all $k > 0$, the intersection of a mesh $M_{(k)}$ and the compact domain $\mathcal{X}$ is finite. Thus, relative to G-MADS, the extended poll introduces only a finite sequence of quantitative polls per iteration, and the categorical poll introduces only a finite number of categorical modifications possible

per iteration. Therefore, CatMADS inherits the following result of G-MADS: a sequence of iterates $\{x_{(k)}\}$ generated in a compact set is such that $\liminf_{k\to\infty} \delta^{\text{con}}_{(k),i} = 0$ for each $i \in I^{\text{con}}$ and $\liminf_{k\to\infty} \delta^{\text{int}}_{(k),j} = 0$ for each $j \in I^{\text{int}}$ (see Theorems 3.1 and 3.2 in [33]). Similarly, since the set is compact, there exists an infinite subset of indices $K_2 \subseteq K_1$ such that $\lim_{k\in K_2} \boldsymbol{x}^{\text{con}}_{(k)} = \boldsymbol{x}^{\text{con}}_{\star}$. Additionally, since the categorical variables, integer variables and number of neighbors belong to finite sets, infinite subsets of indices can be chosen successively as $K_5 \subseteq K_4 \subseteq K_3$, such that $\lim_{k\in K_3} \boldsymbol{x}^{\text{int}}_{(k)} = \boldsymbol{x}^{\text{int}}_{\star}$, $\lim_{k\in K_4} \boldsymbol{x}^{\text{cat}}_{(k)} = \boldsymbol{x}^{\text{cat}}_{\star}$ and $\lim_{k\in K_5} m_{(k)} = m_{\star}$. Consequently, $\lim_{k\in K_5} \mathcal{N}\left(\boldsymbol{x}^{\text{cat}}_{(k)}; m_{(k)}\right) = \mathcal{N}\left(\boldsymbol{x}^{\text{cat}}_{\star}; m_{\star}\right)$. Finally, the infinite subset of indices $K = K_5$ satisfies Definition 26. □

Although the directions for the quantitative polls are generated randomly at each iteration k, Theorem 3 is a deterministic result. Let $\mathbb{S}^{\text{con}} := \{\boldsymbol{u} \in \mathbb{R}^{n^{\text{con}}} : \|\boldsymbol{u}\| = 1\}$ be the unit sphere in $\mathbb{R}^{n^{\text{con}}}$. Along a refining subsequence in a compact set, the categorical and integer components eventually remain fixed. Then, as iterations go to infinity, the normalized continuous polling directions generated by G-MADS grow dense on the unit sphere $\mathbb{S}^{\text{con}}$ [21]. By consequence, any direction on $\mathbb{S}^{\text{con}}$ is arbitrarily close to a direction generated by G-MADS. More formally, for all $\epsilon > 0$ and for any direction $s^{\text{con}} \in \mathbb{S}^{\text{con}}$, there exists an iteration k that generates a direction $v^{\text{con}}_{(k)}$, such that $\left\| \frac{v^{\text{con}}_{(k)}}{\|v^{\text{con}}_{(k)}\|} - s^{\text{con}} \right\| < \epsilon$.

Note that implementation-wise, converging distance-induced neighborhoods in a refining subsequence can be easily ensured by fixing both the number of neighbors $m_{(k)}$ and the categorical distance d^{cat} after a given iteration $k > 0$.

4.3.3 Nonsmooth calculus for CatMADS

The second part of the convergence analysis involves the continuous variables for refined points. For fixed categorical and integer variables, the analysis involves the Clarke generalized derivative [82, 137] and the hypertangent cone [214], restricted to the continuous variables. The vectors $\boldsymbol{u}, \boldsymbol{v}$ and $\boldsymbol{w}$ are used next to denote continuous vectors. The *continuous feasible set at the refined point* $\boldsymbol{x}_{\star}$, which contains the continuous components that are feasible given the categorical and integer components of $\boldsymbol{x}_{\star}$, is defined as

$$\Lambda_{\star} := \left\{ \boldsymbol{u} \in \mathcal{X}^{\text{con}} : \left(\boldsymbol{x}^{\text{cat}}_{\star}, \boldsymbol{x}^{\text{int}}_{\star}, \boldsymbol{u}\right) \in \Omega \right\}. \tag{4.11}$$

For a given refined point $\boldsymbol{x}_{\star}$, the following definition is such that the hypertangent cone contains vectors in $\mathbb{R}^{n^{\text{con}}}$ and rooted at $\boldsymbol{x}^{\text{con}}_{\star}$.

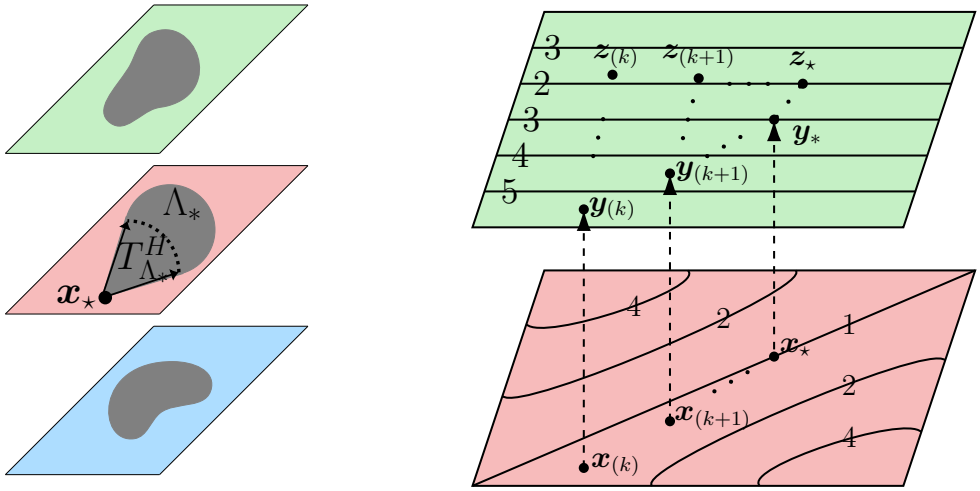
Definition 27 (Hypertangent cone [214])**.** *A vector* $\boldsymbol{v} \in \mathbb{R}^{n^{\text{con}}}$ *is said to be a* hypertangent vector *to the set* $\Lambda_{\star}$ *at a point* $\boldsymbol{x}_{\star} = (\boldsymbol{x}^{\text{cat}}_{\star}, \boldsymbol{x}^{\text{int}}_{\star}, \boldsymbol{x}^{\text{con}}_{\star}) \in \Omega$*, if there exists a scalar* $\epsilon > 0$ *such*

that

$$\left(\boldsymbol{x}^{\text{cat}}, \boldsymbol{x}^{\text{int}}, \boldsymbol{w} + t\boldsymbol{u}\right) \in \Omega \quad \textit{for all} \quad \boldsymbol{w} \in \mathcal{B}\left(\boldsymbol{x}_\star^{\text{con}}; \epsilon\right) \cap \Lambda_\star, \quad \boldsymbol{u} \in \mathcal{B}\left(\boldsymbol{v}; \epsilon\right) \quad \textit{and} \quad t \in (0, \epsilon).$$

The hypertangent cone $T^H_{\Lambda_\star}(\boldsymbol{x}_\star)$ *to* $\Lambda_\star$ *at* $\boldsymbol{x}_\star \in \Omega$ *is the set of all hypertangent vectors to* $\Lambda_\star$ *at* $x_\star$.

In Figure 4.5a, the continuous feasible set $\Lambda_\star$ and the hypertangent cone $T^H_{\Lambda_\star}$ at a refined point $\boldsymbol{x}_\star$ are illustrated for the RBG example, where $x^{\text{int}}_\star = 2$. In the figure, the feasible



(a) Continuous feasible set and hypertangent cone at a refined point $\boldsymbol{x}_\star$, where gray areas represent feasible regions and $x^{\text{int}}_\star = 2$.

(b) Refining, neighbor and endpoint subsequences for two categories example.

Figure 4.5 Illustrative examples for the convergence analysis.

regions are represented in gray. The shape depends on the RBG category. The hypertangent cone $T^H_{\Lambda_\star} \subset \mathbb{R}^2$ is represented with a dotted curve and contains the continuous directions rooted at $\boldsymbol{x}_\star$ that point inside $\Lambda_\star$. The next definition formalizes a subset of these continuous directions.

Definition 28 (Refined direction [21])**.** *Let* $\{\boldsymbol{x}_{(k)}\}_{k\in K}$ *be a refining subsequence with corresponding refined point* $\boldsymbol{x}_\star$*. A vector* $\boldsymbol{v}_\star \in \mathbb{R}^{n^{\text{con}}}$ *is said to be a* refining direction *if there exists an infinite subsequence* $L \subseteq K$ *with poll directions* $\boldsymbol{d}_{(k)} = \left(\boldsymbol{d}^{\text{int}}_{(k)}, \boldsymbol{d}^{\text{con}}_{(k)}\right) \in \mathcal{D}_{(k)}$ *such that* $\boldsymbol{d}^{\text{int}}_{(k)} = \boldsymbol{0}$ *and* $\lim_{k\in L} \frac{\boldsymbol{v}_{(k)}}{\|\boldsymbol{v}_{(k)}\|} = \frac{\boldsymbol{v}_\star}{\|\boldsymbol{v}_\star\|}$ *with* $\boldsymbol{v}_{(k)} = \text{diag}\left(\boldsymbol{\delta}^{\text{con}}_{(k)}\right) \boldsymbol{d}^{\text{con}}$.

To employ generalized directional derivatives [82], the objective function f must be locally Lipschitz continuous with respect to the continuous variables. For a given refined point $x_\star$,

the restriction of the objective function in $\mathcal{X}^{\text{con}}$ is noted $f_\star : \mathcal{X}^{\text{con}} \to \overline{\mathbb{R}}$ and formulated as

$$f_\star(\boldsymbol{u}) \coloneqq f\left(\boldsymbol{x}_\star^{\text{cat}}, \boldsymbol{x}_\star^{\text{int}}, \boldsymbol{u}\right). \tag{4.12}$$

With this function, the generalized directional derivative can be redefined for this work.

Definition 29 (Generalized directional derivative [82, 137])**.** *For a refined point $\boldsymbol{x}_\star \in \Omega$, let $f_\star : \mathcal{X}^{\text{con}} \to \overline{\mathbb{R}}$ be a Lipschitz continuous function near $\boldsymbol{u} \in \Lambda_\star$. The* generalized directional derivative *of the restriction $f_\star$ at $\boldsymbol{u}$ in the direction $\boldsymbol{v} \in \mathbb{R}^{n^{\text{con}}}$ is defined by*

$$f_\star^\circ(\boldsymbol{u}; \boldsymbol{v}) \coloneqq \limsup_{\substack{\boldsymbol{w} \to \boldsymbol{u},\, t \searrow 0, \\ \boldsymbol{w} \in \Lambda_\star,\, \boldsymbol{w} + t\boldsymbol{v} \in \Lambda_\star}} \frac{f_\star(\boldsymbol{w} + t\boldsymbol{v}) - f_\star(\boldsymbol{w})}{t}. \tag{4.13}$$

Under a local Lipschitz continuity assumption, the following theorem strengthens the convergence properties of a refined point $\boldsymbol{x}_\star$ with respect to its continuous variables. The result extends a strictly continuous result from previous work [32] to mixed-variable settings through the definitions of hypertangent cone, refined direction and generalized directional derivative defined above. The proof follows by the same arguments with the straightforward adaptations.

Theorem 4 (Generalized directional derivative at a refined point)**.** *Let $\boldsymbol{x}_\star \in \Omega$ be a refined point generated by* CatMADS *such that $f_\star$ is Lipschitz continuous near $\boldsymbol{x}_\star^{\text{con}}$. If $\boldsymbol{v}_\star \in T_{\Lambda_\star}^H(\boldsymbol{x}_\star)$ is a refining direction, then $f_\star^\circ(\boldsymbol{x}_\star^{\text{con}}; \boldsymbol{v}_\star) \geq 0$.*

Proof. Let $\{\boldsymbol{x}_{(k)}\}_{k \in L}$ be a refining subsequence with corresponding refined point $\boldsymbol{x}_\star \in \Omega$, and let $\boldsymbol{v}_\star \in T_{\Lambda_\star}^H(\boldsymbol{x}_\star)$ be a refining direction. From Definition 26, the categorical and integer components are fixed, such that $\boldsymbol{x}_{(k)}^{\text{cat}} = \boldsymbol{x}_\star^{\text{cat}}$ and $\boldsymbol{x}_{(k)}^{\text{int}} = \boldsymbol{x}_\star^{\text{int}}$, ensuring that the continuous feasible set $\Omega_\star$ does not change through the iterations. Definition 16 ensures the existence of bounds $\boldsymbol{\Delta}_{(k)}^{\text{con}} \geq \boldsymbol{0}$ such that the continuous directions satisfy $-\boldsymbol{\Delta}_{(k)}^{\text{con}} \leq \boldsymbol{v}_{(k)} = \operatorname{diag}\left(\boldsymbol{\delta}_{(k)}^{\text{con}}\right) \boldsymbol{d}^{\text{con}} \leq \boldsymbol{\Delta}_{(k)}^{\text{con}}$ for all $k \in L$. Theorem 3 ensures that $\lim_{k \in L} \|\boldsymbol{\Delta}_{(k)}^{\text{con}}\| = 0$, and therefore, $\lim_{k \in L} \|\boldsymbol{v}_{(k)}\| = 0$. By Definition 28, the refining direction $\boldsymbol{v}_\star \in T_{\Lambda_\star}^H(\boldsymbol{x}_\star)$ is such that $\lim_{k \in L} \frac{\boldsymbol{v}_{(k)}}{\|\boldsymbol{v}_{(k)}\|} = \frac{\boldsymbol{v}_\star}{\|\boldsymbol{v}_\star\|}$. Thus, it follows that $\boldsymbol{x}_{(k)}^{\text{con}} + \boldsymbol{v}_{(k)} \in \Lambda_\star$ for all $k \in L$ sufficiently large, by the definitions of the hypertangent cone and the refining direction, and knowing thats $\lim_{k \in L} \|\boldsymbol{v}_{(k)}\| = 0$. It follows that $f_\star^\circ(\boldsymbol{x}_\star^{\text{con}}; \boldsymbol{v}_\star) \geq 0$ can be developed exactly as in Theorem 6.4 of [32], since $\boldsymbol{x}_{(k)}^{\text{con}} + \boldsymbol{v}_{(k)}$ are unsuccessful iterations with $\boldsymbol{d}_{(k)}^{\text{int}} = \boldsymbol{0}$ for all $k \in L$. □

The next result provides a necessary optimality condition with generalized directional derivatives at a refined point, provided that the function $f_\star$ is locally Lipschitz continuous. The

result depends on whether the PB produces a feasible or infeasible refined point. If the refined point is feasible, then the result involves the function $f_\star$ and the hypertangent cone $T^H_{\Lambda_\star}(\boldsymbol{x}_\star)$, as Theorem 4. Otherwise, it involves the function $h_\star$ (with the same expression of $f_\star$ in Equation (4.12) but for the aggregation function h) and the continuous set $\mathcal{X}^{\text{con}} \subseteq \Lambda_\star$: in this case, the optimization problem of interest is $\min_{\boldsymbol{x}\in\mathcal{X}} h(\boldsymbol{x})$.

Theorem 5 (Convergence of CatMADS)**.** *Let $\{\boldsymbol{x}_{(k)}\}_{k\in K}$ be a refining subsequence with corresponding refined point $\boldsymbol{x}_\star$ produced by* CatMADS *with the PB. Suppose that the set of refining directions is dense in the unit sphere $\{\boldsymbol{u} \in \mathbb{R}^{n^{\text{con}}} : \|\boldsymbol{u}\| = 1\}$.*

i) If $\boldsymbol{x}_\star \in \Omega$ and $f_\star$ is Lipschitz continuous near $\boldsymbol{x}^{\text{con}}_\star$, then $f^\circ_\star(\boldsymbol{x}^{\text{con}}_\star; \boldsymbol{v}) \geq 0$ for all $\boldsymbol{v} \in T^H_{\Lambda_\star}(\boldsymbol{x}_\star)$.
ii) If $\boldsymbol{x}_\star \in \mathcal{X} \setminus \Omega$ and $h_\star$ is Lipschitz continuous near $\boldsymbol{x}^{\text{con}}_\star$, then $h^\circ_\star(\boldsymbol{x}^{\text{con}}_\star; \boldsymbol{v}) \geq 0$ for all $\boldsymbol{v} \in T^H_{\mathcal{X}^{\text{con}}}(\boldsymbol{x}_\star)$.

Proof. Since the refining subsequence $\{\boldsymbol{x}_{(k)}\}_{k\in K}$ is such that $\boldsymbol{x}^{\text{cat}}_{(k)} = \boldsymbol{x}^{\text{cat}}_\star$ and $\boldsymbol{x}^{\text{int}}_{(k)} = \boldsymbol{x}^{\text{int}}_\star$ for all $k \in K$, the proof is a direct consequence of Theorem 4, as well as Corollary 3.4, Theorem 3.5. and Corollary 3.6 from [22] (PB for the continuous case). □

Theorem 5 is a fundamental theoretical result of CatMADS and represents a necessary condition to achieve a $\big[$cat, int, con$\big]$ minimum with $\eta = 1$ and $m = m_\star \in \mathbb{N}$.

4.3.4 Stronger local minima

Definition 26 implies that a refining subsequence $\{\boldsymbol{x}_{(k)}\}_{k\in K}$ has an associated subsequence $\{\boldsymbol{y}_{(k)}\}_{k\in K}$, called the *neighbor subsequence*, whose points have categorical components in the distance-induced neighborhoods of $\{\boldsymbol{x}_{(k)}\}_{k\in K}$, *i.e.*, $\boldsymbol{y}_{(k)} = \left(\boldsymbol{y}^{\text{cat}}_{(k)}, \boldsymbol{x}^{\text{qnt}}_{(k)}\right) \in \mathcal{N}\left(\boldsymbol{x}^{\text{cat}}_{(k)}; m_{(k)}\right) \times \mathcal{X}^{\text{qnt}}$. Since the refining subsequence converges to a refined point $\boldsymbol{x}_\star$, the neighbor subsequence converges to a point $\boldsymbol{y}_\star$, referred to as a *refined neighbor point*. The next theorem, adapted from MV-MADS [4], ensures that a refined point $\boldsymbol{x}_\star$ has an objective value that is lower or equal than a neighbor refined point $\boldsymbol{y}_*$, under continuity hypothesis.

Theorem 6. *If the restriction of the objective function in $\mathcal{X}^{\text{con}}$ is lower semi-continuous at a refined point $\boldsymbol{x}_\star$ produced by* CatMADS*, and upper semi-continuous at the refined neighbor point $\boldsymbol{y}_\star$, then $f(\boldsymbol{x}_\star) \leq f(\boldsymbol{y}_\star)$.*

Proof. Let $\{\boldsymbol{x}_{(k)}\}_{k\in K}$ be a refining subsequence converging to the refined point $\boldsymbol{x}_\star$, and let $\{\boldsymbol{y}_{(k)}\}_{k\in K}$ be a neighbor subsequence converging to the neighbor refined point $\boldsymbol{y}_\star$. By definition of the refining subsequence, it is ensured that $f(\boldsymbol{x}_{(k)}) \leq f\left(\boldsymbol{y}_{(k)}\right)$ for all $k \in K$ since

$\boldsymbol{x}_{(k)}$ yields an unsuccessful iteration and $\boldsymbol{y}_{(k)} \in P_{(k)}^{\text{cat}}$. By the semi-continuity hypothesis it follows that $f(\boldsymbol{x}_\star) \leq \lim_{k\in K} f(\boldsymbol{x}_{(k)}) \leq \lim_{k\in K} f(\boldsymbol{y}_{(k)}) \leq f(\boldsymbol{y}_\star)$. □

Next, when the extended poll is deployed, a neighbor subsequence $\{\boldsymbol{y}_{(k)}\}_{k\in K}$ has an associated subsequence $\{\boldsymbol{z}_{(k)}\}_{k\in K}$, called the *endpoint subsequence*, containing endpoints with $\boldsymbol{z}_{(k)} \in \mathcal{E}_{(k)}\left(\boldsymbol{y}_{(k)}\right)$ as in Definition 21. In Figure 4.5b, the different subsequences are illustrated on a similar example as the RG one in Figure 4.2a, but with different level curves. The refining subsequence $\{\boldsymbol{x}_{(k)}\}_{k\in K}$ converges in the red plane; the neighbor and endpoints subsequences reside in the green plane consisting of a piecewise linear function in two parts.

Finally, the convergence analysis concludes with a theorem regarding the two stronger types of minima, obtained by incorporating the extended poll and using additional hypotheses. In the following theorem, it is first assumed that $\xi \geq 0$. This yields interactions between categorical and continuous variables and leads to the [cat−con, int] minimum in Definition 24. This interaction is represented by $\boldsymbol{y}$ in Figure 4.4b. Then, it is assumed that $\xi = \infty$. This activates the extended poll at every iteration and adds interactions between categorical and integer variables, which correspond to the [cat − con, cat − int] minimum from Definition 25. This additional interaction is schematized by $\boldsymbol{z}$ in Figure 4.4c.

Theorem 7 (Stronger results of CatMADS from the extended poll)**.** *Suppose that the hypotheses of Theorems 5 and 6 hold. Let $\{\boldsymbol{x}_{(k)}\}_{k\in K}$ be a refining subsequence with corresponding refined point $\boldsymbol{x}_\star \in \Omega$ produced by* CatMADS *as in the first case of Theorem 5. Assume that $\xi \geq 0$ and that the refined neighbor $\boldsymbol{y}_\star$ belongs to Ω. Then there exists $\epsilon > 0$ such that*

$$f(\boldsymbol{x}_\star) \leq f(\boldsymbol{y}) \text{ for any } (\boldsymbol{y}^{\text{cat}}, \boldsymbol{y}^{\text{con}}) \in \mathcal{N}(\boldsymbol{x}_\star^{\text{cat}}; m_\star) \times \mathcal{B}(\boldsymbol{x}_\star^{\text{con}}; \epsilon) \text{ with } \boldsymbol{y} = \left(\boldsymbol{y}^{\text{cat}}, \boldsymbol{x}_\star^{\text{int}}, \boldsymbol{y}^{\text{con}}\right) \in \Omega.$$

In addition, if $\xi = \infty$, then

$$f(\boldsymbol{x}_\star) \leq f(\boldsymbol{z}) \text{ for any } \left(\boldsymbol{z}^{\text{cat}}, \boldsymbol{z}^{\text{int}}\right) \in \mathcal{N}\left(\boldsymbol{x}_\star^{\text{cat}}; m_\star\right) \times \mathcal{I}\left(\boldsymbol{x}_\star^{\text{int}}; 1\right) \text{ with } \boldsymbol{z} = \left(\boldsymbol{z}^{\text{cat}}, \boldsymbol{z}^{\text{int}}, \boldsymbol{x}_\star^{\text{con}}\right) \in \Omega.$$

Proof. Let $\boldsymbol{y}_\star \in \Omega$ be a refined neighbor point with $\boldsymbol{y}_\star^{\text{cat}} \in \mathcal{N}(\boldsymbol{x}_\star^{\text{cat}}; m_\star)$, and let $\boldsymbol{z}_\star \in \mathcal{E}(\boldsymbol{y}_\star)$ be a corresponding endpoint subsequence as illustrated in Figure 4.5b. By definition, $\boldsymbol{z}_\star \in \Omega$. The first part of the proof with $\xi > 0$ is shown directly with two cases:

- if $f(\boldsymbol{x}_\star) < f(\boldsymbol{y}_\star)$, then by the semi-continuity hypothesis of Theorem 6, there exists $\epsilon > 0$ such that $f(\boldsymbol{x}_\star) \leq f(\boldsymbol{y})$ for any $(\boldsymbol{y}^{\text{cat}}, \boldsymbol{y}^{\text{con}}) \in \mathcal{N}(\boldsymbol{x}_\star^{\text{cat}}; m_\star) \times \mathcal{B}(\boldsymbol{x}_\star^{\text{con}}; \epsilon)$ with $\boldsymbol{y} = (\boldsymbol{y}^{\text{cat}}, \boldsymbol{x}_\star^{\text{int}}, \boldsymbol{y}^{\text{con}}) \in \Omega$.
- otherwise, $f(\boldsymbol{x}_\star) = f(\boldsymbol{y}_\star)$, and it follows that $f(\boldsymbol{x}_\star) = f(\boldsymbol{y}_\star) = f(\boldsymbol{z}_\star)$, since $f(\boldsymbol{x}_{(k)}) \leq f(\boldsymbol{z}_{(k)}) \leq f(\boldsymbol{y}_{(k)})$ for all $k \in K$, by definition. Therefore, the extended poll must have

been performed around $\boldsymbol{y}_{(k)} \in P^{\text{cat}}_{(k)}$ for infinitely many $k \in K$ sufficiently large [4], which guarantees $f(\boldsymbol{x}_\star) \leq f(\boldsymbol{y})$ for any $(\boldsymbol{y}^{\text{cat}}, \boldsymbol{y}^{\text{con}}) \in \mathcal{N}(\boldsymbol{x}^{\text{cat}}_\star; m_\star) \times \mathcal{B}(\boldsymbol{x}^{\text{con}}_\star; \epsilon)$ with $\boldsymbol{y} = (\boldsymbol{y}^{\text{cat}}, \boldsymbol{x}^{\text{int}}_\star, \boldsymbol{y}^{\text{con}}) \in \Omega$.

Suppose furthermore that $\xi = \infty$. By consequence, the extended poll must have been performed around $\boldsymbol{y}_{(k)}$ for infinitely many $k \in K$ sufficiently large [4]. It follows that $\boldsymbol{z}_{(k)} \in \mathcal{E}(\boldsymbol{y}_{(k)})$ with $(\boldsymbol{y}^{\text{cat}}_{(k)}, \boldsymbol{z}^{\text{int}}_{(k)}) \in \mathcal{N}(\boldsymbol{x}^{\text{cat}}_\star; m_\star) \times \mathcal{I}(\boldsymbol{x}^{\text{int}}_\star; 1)$ (issued from the extended poll modifying integer variables) must have been evaluated for infinitely many $k' \in K' \subseteq K$ sufficiently large, which guarantees that $f(\boldsymbol{x}_\star) \leq f(\boldsymbol{z})$ for any $(\boldsymbol{z}^{\text{cat}}, \boldsymbol{z}^{\text{int}}) \in \mathcal{N}(\boldsymbol{x}^{\text{cat}}_\star; m_\star) \times \mathcal{I}(\boldsymbol{x}^{\text{int}}_\star; 1)$ with $\boldsymbol{z} = (\boldsymbol{z}^{\text{cat}}, \boldsymbol{z}^{\text{int}}, \boldsymbol{x}^{\text{con}}_\star) \in \Omega$. □

Note that, if the set of refining directions is dense in the unit sphere at $\boldsymbol{z}^{\text{con}}_\star$, then Theorem 5 can be adapted to the endpoint subsequence $\{\boldsymbol{z}_{(k)}\}_{k \in K}$. This provides a necessary condition for the continuous variables of $\boldsymbol{z}_\star$ with generalized directional derivatives. Moreover, the parameter $\xi \in \overline{\mathbb{R}}$ in the extended poll controls which type of local minimum is considered.

- If $\xi < 0$, the extended poll is never executed and the corresponding local minimum is of type $\big[$cat, int, con$\big]$.
- If $0 \leq \xi < \infty$, the extended poll is triggered whenever the relative deterioration is at most ξ, and this choice leads to a local minimum of type $\big[$cat-con, int$\big]$.
- If $\xi = \infty$, the extended poll is always executed and this setting aims at a local minimum of type $[\text{cat} - \text{con}, \text{cat} - \text{int}]$.

4.4 Computational experiments

This section compares a simple instance of CatMADS on analytical problems with openly available state-of-the-art solvers. Implementation details of the instance are provided in Section 4.4.1. The solvers used for benchmarking are discussed in Section 4.4.2. Finally, Section 4.4.3 details data profiles that measure the relative performance of the solvers.

Mixed-variable problems with categorical variables that are not simply discretized continuous variables are relatively sparse in the literature. Prior to this work, analytical mixed-variable problems were gathered and created for benchmarking this work with data profiles. These problems are summarized in Tables 4.1 and 4.2. They are either taken from the literature, modifications of existing ones from the literature, or adaptations of classic continuous optimization problems, such as the Rosenbrock and Rastrigin problems. Techniques employed to

create mixed-variable problems include replacing continuous variables with functions depending on both continuous and categorical variables, and transforming a function into another one with cases, each assigned to a category.

The collection currently contains 60 problems, half unconstrained and half constrained. The numbers of categorical, integer, and continuous variables are chosen at the level of the collection rather than problem by problem, with the aim of covering a broad range of mixed-variable settings. Most continuous variables come from the original problems. Some continuous variables are discretized into integer variables, and additional integer variables are introduced as coefficients or terms. The number of categorical combinations (components) varies as follows: low-combinatorial problems have only a few combinations, moderate ones have around 10 to 25, and highly combinatorial problems can reach up to about 100 combinations. The constraints are either inherited from the original problems or manufactured using the same transformations employed to obtain mixed-variable versions of classic problems. They are constructed so that the feasible set is non-empty.

The complete presentation of the problems is contained in Cat-Suite, a new collection of mixed-variable problems described in this technical report [126]. The implementation of CatMADS is publicly available at https://github.com/bbopt/CatMADS_prototype.

4.4.1 Implementation details

The prototype instance of CatMADS is presented in Algorithm 5 and it is intentionally simple. The implementation is built on top of the NOMAD software [30], a C++ implementation of the MADS algorithm with many extensions such as G-MADS. The following steps are adapted from NOMAD: the quantitative poll (G-MADS), the initial DoE with a Latin hypercube sampling, the PB, the update, and the search steps.

Algorithm 5 starts by initializing the static and dynamic parameters. The dynamic parameters are initially indexed by $k = 0$. The DoE must be performed during the initialization, since at least two points are required to construct the categorical distance $d^{\text{cat}} : \mathcal{X}^{\text{cat}} \times \mathcal{X}^{\text{cat}} \to \mathbb{R}_+$. This distance uses the *one-hot encoding technique* that assigns a unique binary variable to each category [224]. The categorical component $\boldsymbol{x}^{\text{cat}} \in \mathcal{X}^{\text{cat}}$ is represented by a total of $\ell := \sum_{i=1}^{n^{\text{cat}}} \ell_i$ binary variables, where $\ell_i \in \mathbb{N}$ is the number of categories of the i-th categorical variable x_i^{cat}. For each categorical variable, exactly one of its binary variables is assigned the value 1, and the others are set to 0. For example, let $x_1^{\text{cat}} \in \{\text{Red (R)}, \text{Blue (B)}, \text{Green (G)}\}$ and $x_2^{\text{cat}} \in \{\text{True (T)}, \text{False (F)}\}$. Then, $\boldsymbol{x}^{\text{cat}} = (\text{R}, \text{F})$ is represented by $((1, 0, 0), (0, 1))$. With this approach, each binary variable can be assigned a parameter that is adjusted to data of the problem. Each parameter takes a value in $[0, 1]$. The categorical distance is

Table 4.1 Unconstrained test problems in Cat-Suite.

Name	n^{cat}	ℓ	n^{int}	n^{con}	Smooth	Ref.	Original problem
Cat-1	2	9	2	$\star$	No	[138]	Ackley
Cat-2	2	9	2	3	No	[138]	Beale
Cat-3	2	4	2	2	Yes	[194]	Augmented-Branin
Cat-4	2	4	2	4	No	[138]	Bukin-6
Cat-5	1	6	1	3	Yes	[168]	EVD-52
Cat-6	2	9	0	2	Yes	[194]	Goldstein
Cat-7	3	16	3	2	No	[138]	Goldstein-Price
Cat-8	1	4	1	5	Yes	[168]	HS78
Cat-9	2	9	2	$\star$	No	[138]	Rastrigin
Cat-10	2	6	2	$\star$	No	[138]	Rosenbrock
Cat-11	1	4	1	4	Yes	[168]	Rosen-Suzuki
Cat-12	1	5	$\star$	$\star$	No	[138]	Styblinski-Tang
Cat-13	1	10	0	4	Yes	[180]	Toy
Cat-14	1	10	0	8	No	[180]	Toy
Cat-15	1	5	3	4	Yes	[168]	Wong-1
Cat-16	2	9	2	$\star$	No	[138]	Zakharov
Cat-17	2	49	1	7	No	[246]	Ishigami
Cat-18	2	100	1	7	No	[246]	Hartmann
Cat-19	2	64	0	$\star$	No	[246]	Levy
Cat-20	3	80	2	4	No	[246]	Camel
Cat-21	1	61	0	4	No	[168]	Gamma
Cat-22	1	51	0	6	Yes	[168]	EVD-61
Cat-23	3	12	0	5	No	[127]	Hal-04
Cat-24	1	21	2	3	Yes	[168]	OET5
Cat-25	1	18	10	10	Yes	[168]	Wong 3
Cat-26	1	10	3	5	Yes	[216]	Roustant
Cat-27	2	100	1	4	Yes	[178]	Kowalik-Osborne
Cat-28	2	36	0	2	Yes	[138]	Three-Hump
Cat-29	2	64	3	4	Yes	[138]	McCormick
Cat-30	4	81	4	6	Yes	[138]	Shekel

developed with sums of weighted Euclidean norms, one per categorical variable, such that

$$d^{\text{cat}}\left(\boldsymbol{x}^{\text{cat}}, \boldsymbol{y}^{\text{cat}}\right) := \left(\sum_{i \in I^{\text{cat}}} \left\| \boldsymbol{E}_i\left(x_i^{\text{cat}}\right) - \boldsymbol{E}_i\left(y_i^{\text{cat}}\right) \right\|_{\boldsymbol{\theta}_i^{\text{cat}}}^2 \right)^{1/2} \tag{4.14}$$

where $\boldsymbol{E}_i\left(x_i^{\text{cat}}\right) \in \left\{ \boldsymbol{z} \in \{0,1\}^{\ell_i} : \mathbf{1}^\top z = 1 \right\}$ is the *one-hot* representation of the i-th categorical variable x_i^{cat}, $\boldsymbol{\theta}_i^{\text{cat}} \in [0,1]^{\ell_i}$ is the vector of parameters of x_i^{cat}, and $\|\boldsymbol{u} - \boldsymbol{v}\|_{\boldsymbol{w}}^2 := (\boldsymbol{u} - \boldsymbol{v})^\top \operatorname{diag}(\boldsymbol{w}) (\boldsymbol{u} - \boldsymbol{v})$ is the weighted Euclidean norm.

Table 4.2 Constrained test problems in Cat-Suite.

Name	n^{cat}	ℓ	n^{int}	n^{con}	m	Smooth	Ref.	Original problem
Cat-cstrs-1	2	9	2	3	3	No	[138]	Beale
Cat-cstrs-2	2	4	2	2	1	Yes	[194]	Augmented-Branin
Cat-cstrs-3	2	9	2	4	2	No	[138]	Bukin-6
Cat-cstrs-4	1	4	4	4	3	Yes	[168]	Dembo-5
Cat-cstrs-5	1	6	1	3	1	No	[168]	EVD-52
Cat-cstrs-6	2	9	2	3	4	Yes	[89]	G-09
Cat-cstrs-7	2	9	0	2	1	Yes	[194]	Goldstein
Cat-cstrs-8	2	25	2	2	2	Yes	[138]	Himmelblau
Cat-cstrs-9	2	4	3	5	4	Yes	[168]	HS-114
Cat-cstrs-10	1	3	2	4	6	Yes	[168]	Pentagon
Cat-cstrs-11	1	8	2	2	3	Yes	[89]	Pressure-Vessel
Cat-cstrs-12	2	25	1	2	2	Yes	[89]	Reinforced-Concrete
Cat-cstrs-13	2	6	2	$\star$	1	No	[138]	Rosenbrock
Cat-cstrs-14	1	5	$\star$	$\star$	2	No	[138]	Styblinski–Tang
Cat-cstrs-15	1	10	0	4	2	Yes	[180]	Toy
Cat-cstrs-16	1	6	4	6	3	Yes	[168]	Wong-2
Cat-cstrs-17	2	64	0	6	11	No	[89]	Speed-reducer
Cat-cstrs-18	1	50	2	2	5	No	[89]	Spring
Cat-cstrs-19	6	64	2	2	8	No	[89]	G07
Cat-cstrs-20	2	100	2	7	10	Yes	[89]	Car-side-impact
Cat-cstrs-21	1	18	0	16	7	No	[168]	Dembo 7
Cat-cstrs-22	1	12	0	12	3	No	[168]	MAD
Cat-cstrs-23	1	13	10	10	4	No	[168]	Wong 3
Cat-cstrs-24	2	100	2	4	5	No	[207]	Welded-beam
Cat-cstrs-25	1	10	0	2	3	No	[207]	Three-bar truss
Cat-cstrs-26	2	36	2	2	6	No	[138]	Three-hump
Cat-cstrs-27	2	64	3	4	3	No	[138]	McCormick
Cat-cstrs-28	2	100	2	2	5	No	[117]	G06
Cat-cstrs-29	4	81	4	6	3	No	[138]	Shekel
Cat-cstrs-30	2	49	1	7	3	No	[246]	Ishigami

The categorical distance in Equation (4.14) can be expanded and expressed more explicitly. For $i \in I^{\text{cat}}$, let $\theta_i\left(x_i^{\text{cat}}\right) \in [0,1]$ and $\theta_i\left(y_i^{\text{cat}}\right) \in [0,1]$ be respectively the parameters associated to the categories taken by x_i^{cat} and y_i^{cat}, then

$$d^{\text{cat}}\left(\boldsymbol{x}^{\text{cat}}, \boldsymbol{y}^{\text{cat}}\right) = \left(\sum_{i \in I^{\text{cat}}} \mathbb{I}\left[x_i^{\text{cat}} \neq y_i^{\text{cat}}\right] \left(\theta_i\left(x_i^{\text{cat}}\right) + \theta_i\left(y_i^{\text{cat}}\right)\right) \right)^{1/2} \tag{4.15}$$

where $\mathbb{I}$ is the indicator function. This expression shows that for each categorical variable, if

Algorithm 5: A simple instance of CatMADS based on cross-validation.

Given $f : \mathcal{X} \to \overline{\mathbb{R}}$, a domain $\mathcal{X}$ and constraint functions $g_j : \mathcal{X} \to \overline{\mathbb{R}}$ with $j \in J$.

0. Initialization

$k \leftarrow 0$	iteration counter
$h^{\max}_{(0)} = \infty$	initial barrier parameter
$m \in \mathbb{N}$	number of points in categorical polls
$\boldsymbol{\Delta}_{(0)} \in \mathbb{R}_+^{n^{\text{qnt}}}$	initial frame size parameters (G-MADS)
$\boldsymbol{\delta}_{(0)} \in \mathbb{R}_+^{n^{\text{qnt}}}$ with $\boldsymbol{\delta}_{(0)}^{n^{\text{qnt}}} \leq \boldsymbol{\Delta}_{(0)}^{n^{\text{qnt}}}$	initial delta size parameters (G-MADS)
$\mathbb{X} \leftarrow \{\boldsymbol{x}_{(1)}, \boldsymbol{x}_{(2)}, \ldots, \boldsymbol{x}_{(p)}\}$	initial DoE of $p \geq 2$ points
Set $\boldsymbol{x}_{\text{FEA}}$ and $\boldsymbol{x}_{\text{INF}}$ with (4.7)	initial solutions (only one might exist)
$\xi \in \overline{\mathbb{R}}$	parameter controlling ξ-extended poll triggers
Construct d^{cat}	problem-specific categorical distance

While *no stopping criteria is met* **do**

1. Opportunistic search $S_{(k)} = S^{\text{SPC}}_{(k)} \cup S^{\text{QUAD}}_{(k)}$

If quantitative poll $P^{\text{qnt}}_{(k-1)}$ or speculative search $S^{\text{SPC}}_{(k-1)}$ is successful

1.1. Speculative search $S^{\text{SPC}}_{(k)}$ on $\mathcal{X}^{\text{qnt}}$

1.2. Quadratic search $S^{\text{QUAD}}_{(k)}$ on $\mathcal{X}^{\text{qnt}}$

2. Opportunistic poll $P_{(k)} = P^{\text{cat}}_{\text{FEA}} \cup P^{\text{qnt}}_{\text{FEA}} \cup P^{\text{cat}}_{\text{INF}} \cup P^{\text{qnt}}_{\text{INF}}$

3. Opportunistic extended poll $\bar{P}_{(k)}$

4. Update

4.1. Update incumbent solutions $\boldsymbol{x}_{\text{FEA}}$ and $\boldsymbol{x}_{\text{INF}}$ with (4.7)

4.2. G-MADS and barrier update:

If $\boldsymbol{y} \prec_f \boldsymbol{x}_{\text{FEA}}$ or $\boldsymbol{y} \prec_h \boldsymbol{x}_{\text{INF}}$ for some $\boldsymbol{y} \in S_{(k)} \cup P_{(k)}$ (dominating iteration)

Increase $\boldsymbol{\Delta_{(k+1)}}$ and $\boldsymbol{\delta_{(k+1)}}$, and set $h^{\max}_{(k+1)} = h(\boldsymbol{x}_{\text{INF}})$

Else if $0 < h(\boldsymbol{y}) < h(\boldsymbol{x}_{\text{INF}})$ for some $\boldsymbol{y} \in S_{(k)} \cup P_{(k)}$ (improving)

Set $h^{\max}_{(k+1)} = \max\{h(\boldsymbol{v}) : h(\boldsymbol{v}) < h(\boldsymbol{x}_{\text{INF}}), \boldsymbol{v} \in \mathbb{X} \cup S_{(k)} \cup P_{(k)}\}$

Otherwise (unsuccessful)

Decrease $\boldsymbol{\Delta}_{(k+1)}$ and $\boldsymbol{\delta}_{(k+1)}$, and set $h^{\max}_{(k+1)} = h(\boldsymbol{x}_{\text{INF}})$

5. Termination

If stopping criteria is met then **stop**

Else $\mathbb{X} \leftarrow \mathbb{X} \cup S_{(k)} \cup P_{(k)}$ and $k \leftarrow k + 1$

end

there is a mismatch, the parameters of the two categories contribute to the distance.

The weights reflect the importance of each binary variable assigned to a specific category and allows to construct problem-specific categorical distances. These weights are determined by minimizing the *Root Mean Square Error* (RMSE) of an *inverse distance weighting* interpolation, where quantitative variables are normalized and categorical are one-hot encoded. The RMSE is evaluated using 3-fold cross-validation on the DoE. Integer and continuous

variables have no weights, since these variables are not considered for the categorical distance. In Section 4.2, recall that a mismatching-based distance on the RBG example yields $d^{\text{cat}}(\text{R, B}) = 0 = d^{\text{cat}}(\text{R, G})$. Now, with the proposed distance, the distances would be $d^{\text{cat}}(\text{R,B}) = \theta_R + \theta_B$ and $d^{\text{cat}}(\text{R, G}) = \theta_R + \theta_G$, where the real parameters θ_R, θ_B and θ_G are tuned by regression. This distance distinguishes categories more finely than mismatches. In other words, the neighbors in distance-induced neighborhoods are selected in a meaningful manner.

The initial DoE is allocated 20% of the total budget of evaluations. This percentage was determined empirically through preliminary tests. It represents a considerable number of evaluations. However, the categorical distance is constructed only once at the initialization, the DoE must be sufficiently large to ensure that the categorical distance adequately models the mixed-variable objective function f.

The stopping criteria include a budget limit on the number of evaluations and a convergence criterion triggered when the mesh size parameters reach their lower bounds. The trigger parameter ξ of the extended poll dictates which type of local minimum is associated to the convergence condition: if $\xi < 0$, then the convergence condition represents a necessary condition for a $\left[\text{cat, con, int}\right]$ minimum; else if $\xi \in \mathbb{R}_+$, then it represents a $\left[\text{cat} - \text{con, int}\right]$ minimum, and otherwise $\xi = \infty$ represents a $\left[\text{cat} - \text{con, cat} - \text{int}\right]$ minimum. Note that obtaining the strongest type $\left[\text{cat} - \text{con, cat} - \text{int}\right]$ with a finite value $\xi > 0$ would require global information on bounds for f that ensure the extended poll is always triggered. In the blackbox setting considered, such information is not assumed, so $\xi = \infty$ is used to obtain these stronger guarantees.

The search is performed opportunistically. If the previous iteration had a successful quantitative poll (feasible or infeasible), then the speculative search of NOMAD is executed on the quantitative space. The speculative (or dynamic) search generates candidate points along the last successful direction of G-MADS [21, 161]. To capitalize on promising directions, the speculative search is repeated as long as it remains successful.

The default quadratic search of NOMAD [85] is also used on the quantitative variables. At each iteration, quadratic models for the objective and constraint functions are constructed around incumbent solutions, with categorical variables fixed. For a given incumbent solution, the proposed point minimizes its corresponding model of f, while either satisfying or respecting the barrier of its model constraints, for feasible and infeasible incumbents, respectively. For more details on the speculative and quadratic searches, see [30].

The number of elements m in distance-induced neighborhoods is constant and common for the feasible and infeasible categorical polls, each of which contains $m - 1$ points. For a given

problem, m is set by the formula $m = \max\left(3, \sqrt{|\mathcal{X}^{\text{cat}}|}\right)$, which scales with the total number of categories $|\mathcal{X}^{\text{cat}}| = \prod_{i=1}^{n^{\text{cat}}} \ell_i$. The square root balances the scaling of the product. The formula ensures $m - 1 \geq 2$, to avoid cycling in the categorical polls. Alternatives formulas were tested, notably linear ones, *e.g.*, $m - 1 = 0.5\,|\mathcal{X}^{\text{cat}}|$, but none performed better than the square-root formula in preliminary tests.

For the poll, the evaluation order prioritizes points generated from quantitative polls over those from categorical polls. Moreover, points from quantitative polls are ranked using quadratic models as in [30], while points from categorical polls follow their generation order.

4.4.2 Comparison solvers

G-MADS and MV-MADS are not included among the solvers. G-MADS is simply not suitable for the class of problems, since it does not treat categorical variables. Using arbitrary or random label encodings to treat categorical variables as integers would break the underlying logic of G-MADS, in particular by inducing a mesh with no meaningful interpretation for categorical variables. MV-MADS requires manually specified user-defined neighborhoods for each problem. Most of the problems do not posses intuitive neighborhoods, and defining 60 specific neighborhoods would be a monumental task beyond the scope of the work. Again, these user-defined neighborhoods highlights a major practical limitation of MV-MADS.

CatMADS is benchmarked with the following state-of-the-art solvers or libraries, used in various fields involving blackboxes with categorical variables [142, 200, 242]:

- Pymoo, a Python-based library of optimization solvers. The solver used for benchmarking is the mixed-variable and single objective implementation of a genetic algorithm [49];
- Optuna, a covariance matrix adaptation evolution strategy (CMA-ES) for categorical variables [8];
- SMT, a surrogate modeling toolbox implementing several regression models, including GPs with a BO solver [226].

The solvers from the libraries Pymoo and Optuna are used off-the-shelf, with only evaluation budget specified and launched using the same DoE as CatMADS. Since the benchmarking targets global performance over a collection of problems, they are not tuned problem-by-problem. Moreover, tuning these solvers on a collection of problems would be simply too time-consuming and lies outside the scope of the work. Accordingly, the recommended default parameter are used for these solvers.

SMT requires several user-defined choices. The implementation used for benchmarking is intentionally basic, since BO is a computationally demanding approach. For instance, running the unconstrained Cat-1 problem with half of the evaluation budget using the implementation provided in the guide took more than 12 hours with an 11th-generation Intel i7-11800H (2.30 GHz) CPU. For SMT, fully evaluating the budget of $250n$ evaluations is not suitable for benchmarking across several instances of the problems. To mitigate computational time, the basic Gower kernel is used for categorical variables, as in the guide, and two additional stopping criteria are introduced: the solver is terminated either when the acquisition value falls below 10^{-14} for three consecutive iterations, or when 33% of the evaluation budget is reached.

The current implementation of BO in SMT is limited to unconstrained problems. However, constrained problems can be reformulated as unconstrained ones. This is done using a probability of feasibility reformulation [249], which incorporates the likelihood of feasibility into the objective function. The acquisition subproblem is then optimized using the qEI method with $200n$ trials.

Both BoTorch and SMT were tested as potential benchmarking solvers. Only SMT was retained, as it showed better performance and was sufficient to represent BO methods in the comparisons.

For benchmarking, each optimization problem is attributed a budget of evaluation of $250n$, where $n = n^{\text{cat}} + n^{\text{int}} + n^{\text{con}}$ is the total number of variables.

4.4.3 Data profiles

All solvers are subject to some randomness. Hence, each problem is run with three random seeds to ensure fairness. An instance $p \in \mathcal{P}$ refers to a problem with a specific seed, where $\mathcal{P}$ is the set of instances. There is a total of 160 instances, half constrained and half unconstrained.

For a given instance, let $f_\star$ be the smallest feasible objective function value and f_0 be an initial feasible function value. Let $\mathcal{S}$ be a set of solvers considered for benchmarking. A solver $s \in \mathcal{S}$ is said to have τ-solved an instance $p \in \mathcal{P}$ when it has produced a feasible incumbent solution $\boldsymbol{x}_{\text{FEA}} \in \Omega$ with a reduction $f_0 - f(\boldsymbol{x}_{\text{FEA}})$ that is sufficient relatively to the smallest reduction $f_0 - f_\star$ obtained by any solver on this instance [179]. The convergence test can be expressed as

$$f_0 - f(\boldsymbol{x}_{\text{FEA}}) \geq (1 - \tau)(f_0 - f_\star) \tag{4.16}$$

where $\tau \in [0, 1]$ is a given tolerance and f_0 is either

- the smallest function value in the common DoE for unconstrained problems [28],
- or the largest function value among the first feasible solution found by the solvers otherwise [28]. For the unconstrained problems, this approach allows to compare solvers on problems without feasible solutions in the DoE.

A data profile [179] represents the portion of τ-solved instances by a solver $s \in \mathcal{S}$, such that

$$\text{data}_s(\kappa) \coloneqq \frac{1}{|\mathcal{P}|} \left| \left\{ p \in \mathcal{P} \, : \, \frac{k_{p,s}}{n_p + 1} \leq \kappa \right\} \right| \in [0, 1] \tag{4.17}$$

where $k_{p,s} \geq 0$ is the number of evaluations required for the solver $s \in \mathcal{S}$ to τ-solve an instance $p \in \mathcal{P}$ and n_p is the number of variables in the instance $p \in \mathcal{P}$. In the following plots, the horizontal axis κ represents groups of $(n_p + 1)$ evaluations.

Results for the extended poll

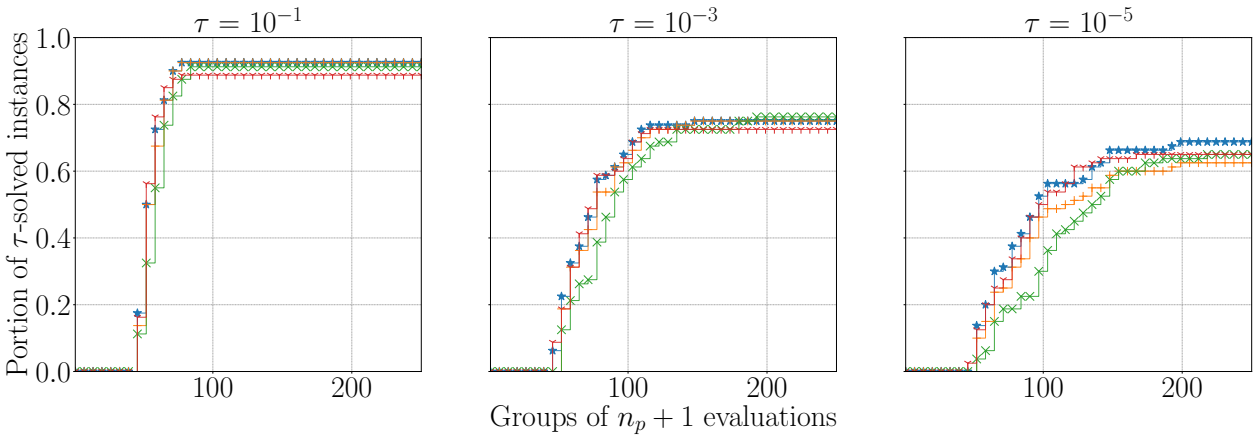
Before comparing CatMADS with the other solvers, ξ is chosen with data profiles showcasing the relative performance of CatMADS with different values of ξ. These data profiles are computed on the first sixteen unconstrained problems and the first sixteen constrained problems, with five random seeds for each, as shown in Figures 4.6a and 4.6b.

The value 0.05, associated to a $[\text{cat}-\text{con}, \text{int}]$ local minimum, yielded the best performance on the two smallest tolerances 10^{-3} and 10^{-5} for both unconstrained and constrained instances. The value $\xi = \infty$ under-performs for the smallest tolerance 10^{-5}. Although $\xi = \infty$ is associated to the stronger $[\text{cat}-\text{con}, \text{cat}-\text{int}]$ local minimum its costs degrades performance. Based on these data profiles, the parameter ξ is set to 0.05.

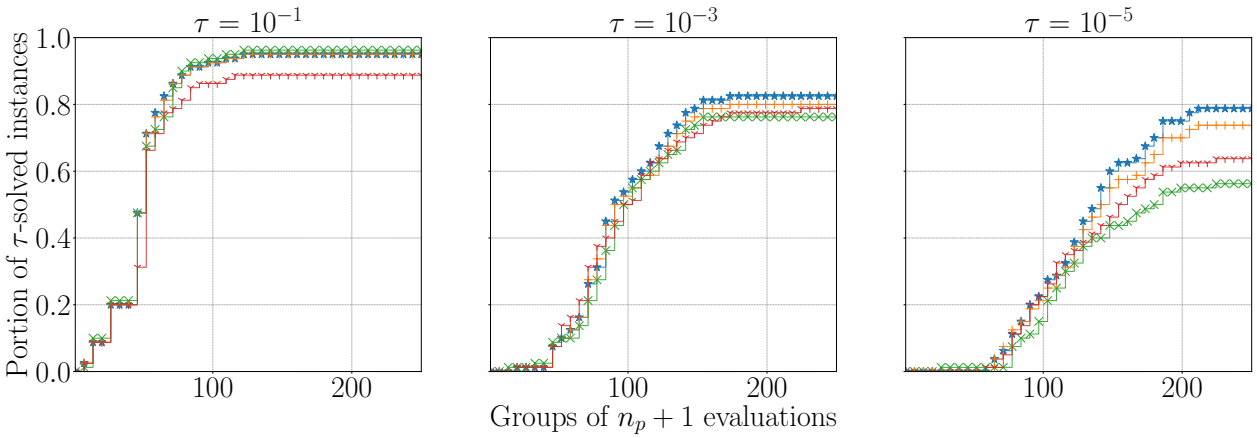
Results with existing solvers

For the next data profiles, the solvers and CatMADS are contained in the set of solvers $\mathcal{S} = \{\text{CatMADS}, \text{Pymoo}, \text{Optuna}, \text{SMT}\}$. The data profiles comparing CatMADS with state-of-the-art solvers on the unconstrained and constrained instances are presented in Figures 4.7a and 4.7b. The data profiles showcase that CatMADS strongly dominates the other solvers. For instance, the data profiles in Figure 4.7a with $\tau = 10^{-5}$ shows that CatMADS τ-solves approximately 70% of the problems against 30% for Pymoo. CatMADS performs particularly well on the smallest tolerance $\tau = 10^{-5}$ and the constrained problems.

4.5 Discussion



(a) Unconstrained test problems.



(b) Constrained test problems.

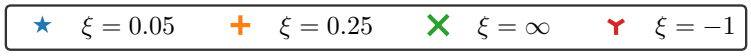
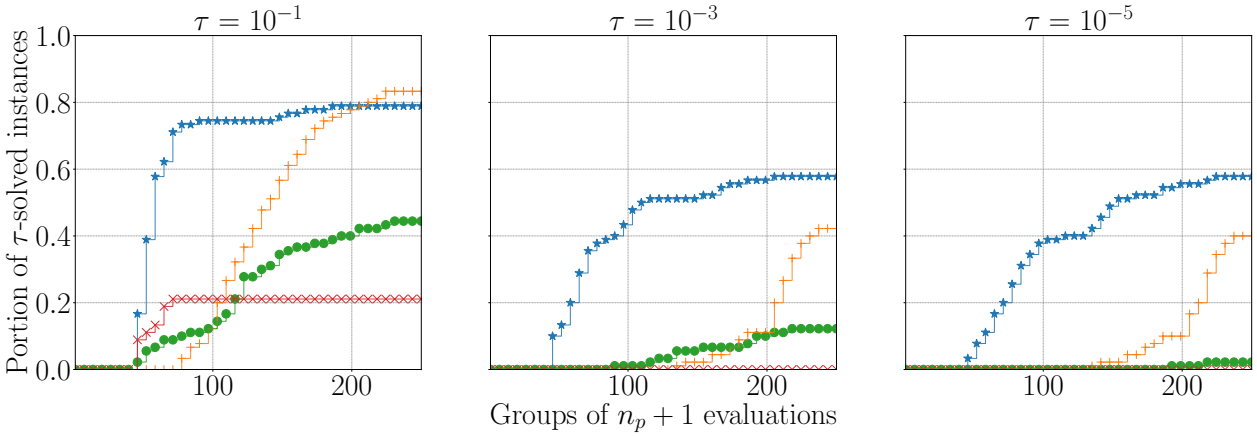


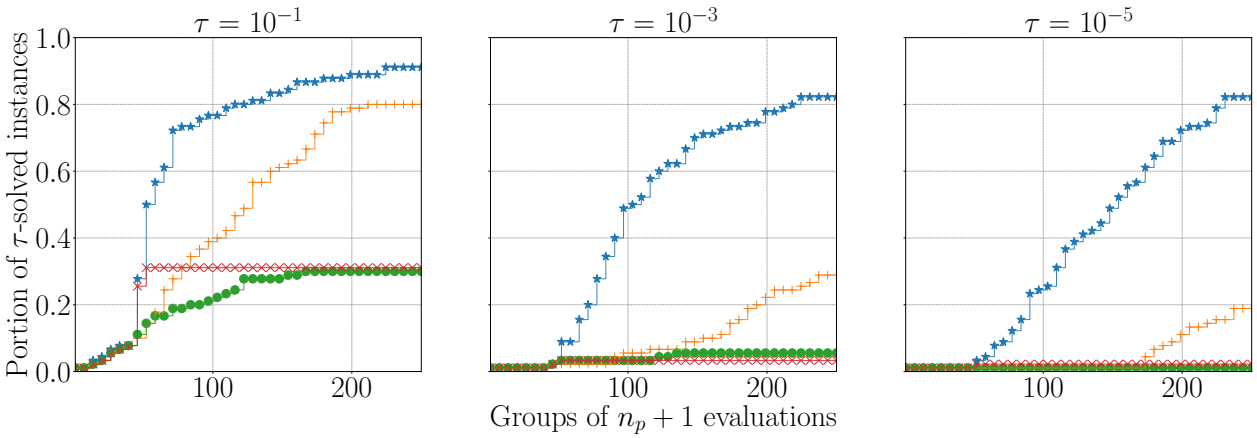
Figure 4.6 Data profiles for different values of ξ.

The present work introduces CatMADS, a generalization of MADS for constrained mixed-variable blackbox optimization with categorical, integer and continuous variables. CatMADS combines three main mechanisms. First, categorical variables are structured through distance-based neighborhoods that are constructed automatically with categorical distances. Second, integer and continuous variables are handled efficiently via a G-MADS adapted to the more general mixed-variable setting. Third, constraints are treated with a unified progressive barrier [22] and extended poll [19], which together promote exploration of promising but currently infeasible candidates and affect the strength of the guarantees.

CatMADS can be employed off-the-shelf on blackbox optimization problems. Its convergence analysis establishes guarantees under standard assumptions and is organized as a hierarchy of



(a) Unconstrained test problems.



(b) Constrained test problems.

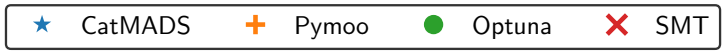


Figure 4.7 Comparison results on the test problems.

three mixed-variable local minima of increasing strength. In doing so, the work also adapts fundamental notions from nonsmooth analysis, such as generalized directional derivatives and hypertangent cones, to a mixed-variable setting with distance-based neighborhoods.

A prototype implementation of CatMADS is compared to state-of-the-art solvers on 60 mixed-variable problems. Even if basic, the implementation is automatic and outperforms Pymoo, Optuna and SMT. As such, CatMADS addresses the research gap of an efficient mixed-variable constrained blackbox method with convergence guarantees.

The implementation in this work is intentionally simple. In future work, CatMADS will be hybridized with Bayesian optimization to produce an even more efficient solver. Search steps

focused on exploration will also be developed to balance the local nature of CatMADS. Moreover, CatMADS will be applied on practical applications such as hyperparameter optimization of deep models or metaheuristic methods.

Data availability statement

Scripts and data are publicly available at https://github.com/bbopt/CatMADS_prototype.

Conflict of interest statement

The authors state that there are no conflicts of interest.

CHAPTER 5 ARTICLE 2: SURROGATE-BASED CATEGORICAL NEIGHBORHOODS FOR MIXED-VARIABLE BLACKBOX OPTIMIZATION

I know the pieces fit
Cause I watched them fall away
Mildewed and smoldering
Fundamental differing

— Tool, *Schism*

Authors. Charles Audet, Youssef Diouane, Edward Hallé-Hannan, Sébastien Le Digabel and Christophe Tribes

Submitted on March 22, 2026 to *Journal of Global Optimization*
(see arXiv:2603.27839)

Abstract. In simulation-based engineering, design choices are often obtained following the optimization of complex blackbox models. These models frequently involve mixed-variable domains with quantitative and categorical variables. Unlike quantitative variables, categorical variables lack an inherent structure, which makes them difficult to handle, especially in the presence of constraints. This work proposes a systematic approach to structure and model categorical variables in constrained mixed-variable blackbox optimization. Surrogate models of the objective and constraint functions are used to induce problem-specific categorical distances. From these distances, surrogate-based neighborhoods are constructed using notions of dominance from bi-objective optimization, jointly accounting for information from both the objective and the constraint functions. This study addresses the lack of automatic and constraint-aware categorical neighborhood construction in mixed-variable blackbox optimization. As a proof of concept, these neighborhoods are employed within CatMADS, where the problem-specific distances are derived from Gaussian processes. The resulting method is benchmarked on the Cat-Suite collection of 60 mixed-variable optimization problems and compared against state-of-the-art solvers. Data profiles indicate competitive performance of the proposed method. Finally, its performance is demonstrated on a real-world aircraft design application.

Keywords. Blackbox optimization, derivative-free optimization, mixed-variable problems, constrained optimization, categorical variables

5.1 Introduction

In most cases, design choices in engineering and machine learning imply the constrained optimization of complex computer simulations that are *expensive-to-evaluate* blackboxes with no analytical expressions. In addition, these blackboxes may involve different types of variables, such as continuous, integer and categorical variables. The latter are qualitative and lack structure. For example, in aerospace engineering, optimal aircraft design involves simulations with choices of materials or propulsion systems [63, 222]. Deep learning models are characterized by hyperparameters, such as the activation function or the choice of optimizer, whose values affect the training and validation pipeline [25, 125, 158]. Standard optimization methods using gradients and relaxations in the presence blackboxes and categorical variables are not suited for this context.

The motivation of this work is to improve the optimization of categorical variables in a constrained blackbox settings where information is limited and constraints may be difficult to handle. This is done by proposing a systematic approach to structure categorical variables with neighborhoods constructed using surrogate models of the objective and constraint functions. These neighborhoods can be directly incorporated into mixed-variable methods using categorical neighborhoods. As a proof-of-concept, they are are employed with CatMADS [23], a mixed-variable extension of the Mesh Adaptive Direct Search (MADS) algorithm [21] for constrained blackbox optimization.

Notation

The notation used follows [23]. Vectors are in bold, and scalars are in normal font. Superscripts are reserved for types of variables. Subscripts without parentheses index variables, *e.g.*, x_1^{cat} is the first categorical variable. Subscripts with parentheses are reserved for indexing an iteration k, *e.g.* $\boldsymbol{x}_{(k)}$, or for listing points, *e.g.* $\boldsymbol{x}_{(1)}, \boldsymbol{x}_{(2)}, \ldots, \boldsymbol{x}_{(s)}$.

Sets are denoted using capital letters with either normal or calligraphic font, *e.g.* A or $\mathcal{A}$, except for the set of known points noted $\mathbb{X}$.

5.1.1 Constrained mixed-variable blackbox optimization

This study tackles inequality constrained mixed-variable blackbox optimization problems formulated as

$$\min_{\boldsymbol{x} \in \Omega} f(\boldsymbol{x}), \tag{5.1}$$

where $f : \mathcal{X} \to \overline{\mathbb{R}}$ is the objective function with $\overline{\mathbb{R}} = \mathbb{R} \cup \{+\infty\}$, $\mathcal{X}$ is the domain of the objective and constraint functions, $\boldsymbol{x} \in \mathcal{X}$ is a point, $\Omega \coloneqq \{\boldsymbol{x} \in \mathcal{X} \ : \ g_j(\boldsymbol{x}) \leq 0, \ j \in J\}$ is the feasible set defined by the constraints, $g_j : \mathcal{X} \to \overline{\mathbb{R}}$ is the j-th constraint function of the problem with $j \in J$ and $|J| \in \mathbb{N}$ is the number of constraints.

The constraints defining the feasible set Ω are supposed to be *relaxable* and *quantifiable*, meaning that when they are evaluated, they can return a proper value without being satisfied [162]. An *unrelaxable* constraint, which must be satisfied to have a proper blackbox execution, is considered out of the domain, such that if $\boldsymbol{x} \notin \mathcal{X}$, then it is assigned $f(\boldsymbol{x}) = +\infty$. Similarly, if a point $\boldsymbol{x} \in \mathcal{X}$ hits a *hidden constraint* that crashes or invalidates the output values, then it is simply assigned $f(\boldsymbol{x}) = +\infty$.

The functions involved in the main optimization problem are supposed to be *blackboxes*, which are defined as follows in [26]: "*any process that when provided an input, returns an output, but the inner working of the process is not analytically available*". For example, the strain experienced by a beam can be estimated using a finite element simulation that outputs strain values for a given geometry, material and loading conditions. The lack of an analytical expression implies that derivatives are unavailable with respect to the continuous variables. The function evaluations are determined by executing a process that is possibly time-consuming.

The problems considered are said to be mixed-variable. Each variable is either *continuous* (con), *integer* (int) or *categorical* (cat). The variables of the same type are contained in a corresponding column vector called a *component*. For $t \in \{\text{cat}, \text{int}, \text{con}\}$, the component of type t is expressed as

$$\boldsymbol{x}^t \coloneqq \left(x_1^t, x_2^t, \ldots, x_{n^t}^t\right) \in \mathcal{X}^t \coloneqq \mathcal{X}_1^t \times \mathcal{X}_2^t \times \ldots \times \mathcal{X}_{n^t}^t, \tag{5.2}$$

where $x_i^t \in \mathcal{X}_i^t$ represents the i-th variable of type t, and $\mathcal{X}_i^t$ denotes its bounds. The variable index i ranges over the set $I^t \coloneqq \{1, 2, \ldots, n^t\}$. The number of variables of type t is noted $n^t \in \mathbb{N}$. The set of type t is noted $\mathcal{X}^t$ and it is such that $\boldsymbol{x}^t \in \mathcal{X}^t$.

Following the components, a point $\boldsymbol{x} \in \mathcal{X}$ is defined by a partition by types, and the domain is expressed as Cartesian products of the sets by types, such that

$$\boldsymbol{x} \coloneqq \left(\boldsymbol{x}^{\text{cat}}, \boldsymbol{x}^{\text{int}}, \boldsymbol{x}^{\text{con}}\right) \in \mathcal{X} \coloneqq \mathcal{X}^{\text{cat}} \times \mathcal{X}^{\text{int}} \times \mathcal{X}^{\text{con}}. \tag{5.3}$$

The categorical variables take *categories* representing qualitative values. For $i \in I^{\text{cat}}$, the i-th categorical variable x_i^{cat} takes values in the set $\mathcal{X}_i^{\text{cat}} \coloneqq \{c_1, c_2, \ldots, c_{\ell_i}\}$, where c_j is a category

with $j \in \{1, 2, \ldots, \ell_i\}$ and ℓ_i is the number of categories of the variable. For example, the color for an object could be taking a category in the set {RED, BLUE, GREEN}. These variables are inherently difficult to treat because their sets are not ordered and usual distance functions are not suitable [25, 130]. The categorical set $\mathcal{X}^{\text{cat}}$ is assumed to be finite. The total number of categorical components is $|\mathcal{X}^{\text{cat}}| = \prod_{i=1}^{n^{\text{cat}}} \ell_i$.

The integer and continuous variables take quantitative values and they belong to ordered sets in which standard distance functions are well-defined. These variables are typically much easier to optimize than the categorical ones. For convenience, the integer and continuous variables may be regrouped in a quantitative component such that $\boldsymbol{x}^{\text{qnt}} := (\boldsymbol{x}^{\text{int}}, \boldsymbol{x}^{\text{con}}) \in \mathcal{X}^{\text{qnt}} := \mathcal{X}^{\text{int}} \times \mathcal{X}^{\text{con}}$.

Note that this study does not consider *meta variables*, which influence the inclusion or bounds of other variables [25, 125]. The problems have fixed dimensions and bounds.

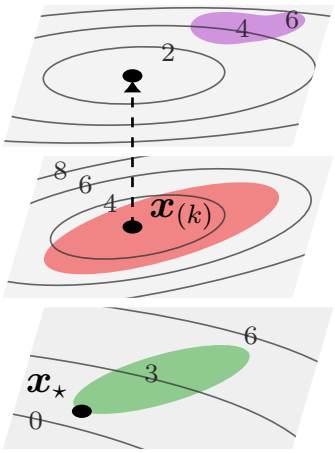
5.1.2 Research gap, objectives and contributions

Bayesian optimization (BO) frameworks have been successfully applied to mixed-variable blackbox problems [194, 201, 224, 270]. However, BO does not provide local optimization mechanisms, convergence guarantees, or principled stopping criteria. Several metaheuristics rely on operations based on Gower-type distances that reduce categorical differences to mismatches between categories. Such generic measures are not problem-informed and can lead to a large number of unnecessary evaluations.

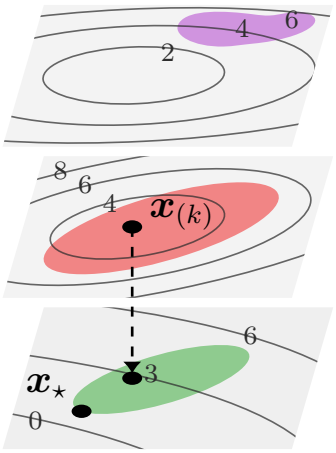
There are other mixed-variable methods, such as CatMADS and MV-MADS, that handle categorical variables with neighborhoods [4, 23, 167]. In these methods, neighborhoods are used to generate candidate solutions by modifying the variables of an incumbent. The main advantage of these neighborhoods is that they introduce a local mechanism that leads to theoretical guarantees for discrete variables. The performance of neighborhood-based approaches depends highly on the quality of the neighborhoods. In MV-MADS, neighborhoods are constructed with user-defined rules, which can be difficult to establish in a blackbox context. In practice, these neighborhoods are primarily intended to handle discrete variables, but they may also modify the continuous variables of an incumbent. The theoretical results of MV-MADS assume that the points in the user-defined neighborhoods belong to the feasible set Ω. This constitutes an important methodological restriction for handling constraints and does not capitalize on information from relaxable constraints. CatMADS focuses instead on neighborhoods strictly for categorical variables and constructs them using categorical distances defined solely from the objective function: The constraint functions are thus not considered. This can be problematic for problems in which the behavior of the constraint

functions changes drastically with respect to the categorical variables.

Figure 5.1 illustrates a simple example with one categorical variable $x^{\text{cat}} \in \{\text{PURPLE}, \text{RED}, \text{GREEN}\}$ and two continuous variables. Each plane corresponds to a category, and the colored regions denote feasibility. Each plane also shows some level curves of the objective function.



(a) Categorical neighborhood relying only on objective information.



(b) Categorical neighborhood integrating objective and constraints information.

Figure 5.1 A motivating example where the incumbent $\boldsymbol{x}_{(k)}$ has objective value $f(\boldsymbol{x}_{(k)}) = 3$, while the global minimum $\boldsymbol{x}_\star$ has $f(\boldsymbol{x}_\star) = 1$. The neighborhood on the left selects PURPLE BASED STRICTLY ON PROXIMITY WITH THE OBJECTIVE. THE NEIGHBORHOOD ON THE RIGHT SELECTS GREEN BASED ON PROXIMITY OF BOTH THE CONSTRAINTS AND THE OBJECTIVE.

In Figure 5.1, the current solution $\boldsymbol{x}_{(k)}$ lies in the continuous subspace associated with the categorical component $\boldsymbol{x}^{\text{cat}} = \text{RED}$. It corresponds to a local minimum with respect to the continuous variables. However, the global minimum $\boldsymbol{x}_\star$ resides in the continuous subspace associated with $x^{\text{cat}} = \text{GREEN}$. In the setting, reaching this global minimum from $\boldsymbol{x}_{(k)}$ can be difficult when using a neighborhood-based approach. From $\boldsymbol{x}_{(k)}$ in the subspace of RED, objective-based or user-defined neighborhoods would likely prioritize the category PURPLE. The restriction of the objective function in the RED and PURPLE subspaces appears more similar. Consequently, an objective-based neighborhood may focus on the PURPLE category, although: 1) its associated continuous subspace is highly infeasible and yields poorer feasible values, and 2) the feasible regions in the continuous subspaces of RED and GREEN are much more similar. This can degrade the performance of neighborhood-based methods and the quality of the local minima with respect to the categorical variables. To mitigate this issue, one could rely on global mechanisms such as BO or design better-informed neighborhoods. In the presence of noise or nonsmoothness, BO may fail to properly tackle

the continuous variables and by consequence the mixed-variable problem itself. More local and robust mechanisms may therefore be needed to ensure incremental progress.

This motivates the development of local neighborhoods that are better-informed and robust. The research gap addressed is the automatic construction of problem-specific categorical neighborhoods that incorporate information from both the objective and the constraint functions in a blackbox setting. Note that, hereafter, categorical neighborhoods are referred to simply as neighborhoods. The work does not aim to introduce a new optimization method, but rather to provide additional structural information that facilitates handling categorical variables. These better-informed neighborhoods provide a systematic structure for categorical variables, improve interpretability, and are particularly well suited for blackbox constrained optimization. They enhance the quality of local minima with respect to categorical variables and improve the efficiency neighborhood-based methods. The number of evaluations for optimizing categorical variables may decrease, since each categorical neighbor is selected in a more informed manner.

The present work is structured through four objectives, representing the main contributions of the work. The first objective is to construct a problem-specific categorical distance using kernel-based surrogate models, such as Gaussian Processes (GPs), so that categorical components with similar predicted objective values are considered closer. The second objective is to define a categorical distance that captures similarity with respect to the constraint functions by aggregating surrogate models of the constraints. Building on these two distances, the third objective is to construct categorical neighborhoods balancing similarity between the objective and the constraints. The first three objectives are developed independently of any specific optimization framework and for purely categorical domains. The fourth objective is to generalize the proposed neighborhoods to mixed-variable domains. As a proof of concept for the fourth objective, the neighborhoods are used in CatMADS.

5.1.3 Related work

In [23], categorical distance is constructed using a basic interpolation of the objective function. Neighborhoods are characterized with this single distance function and they are constructed without accounting the constraint functions, which may lead to neighborhoods containing elements that are highly dissimilar in terms of feasibility. In [4, 167], categorical neighborhoods are specified manually by the user, which requires prior knowledge about the problem structure, which may be ambiguous in a blackbox context. The primary focus of CatMADS and MV-MADS lies in the optimization process, while neighborhood design remains secondary. In contrast, the present work emphasizes the systematic construction of categorical

neighborhoods, independently of any particular optimization framework.

Several metaheuristic methods have been proposed to address categorical variables in mixed-variable optimization. These methods construct candidate points with operations transforming categorical variables of existing solutions. Such operations have similar roles to categorical neighborhoods. CatCMA extends the covariance matrix adaptation evolution strategy (CMA-ES) to problems involving both categorical and continuous variables [128]. In this approach, categorical variables are modeled by probability vectors over their respective categories, while continuous variables are modeled by a multivariate normal distribution. A joint probabilistic distribution over categorical and continuous variables is introduced to guide the optimization in mixed-variable domains. The Non-dominated Sorting Genetic Algorithm-II (NSGA-2) [95], although originally developed for multi-objective optimization, has also been extended to mixed-variable problems involving categorical variables. [49] provides implementations with crossover and mutation operators tailored to each variable type, from which candidate points are generated over a population of solutions. Candidate points are constructed with these operators applied to a population of solutions. For more details, the survey [248] provides an exhaustive survey for mixed-variable metaheuristics.

In Bayesian optimization, categorical variables are structured with similarity measures, called *kernels*. In optimization, kernels are problem-specific similarity measures characterizing surrogate models that guide the optimization. In the literature, various techniques exist for constructing categorical kernels, most notably *one-hot encoding* and matrix-based approaches. One-hot encoding assigns a unique binary variable to each category, and they are assigned their corresponding hyperparameter weighting the importance of each category in the kernel [112]. Matrix-based approaches have shown great results in BO [194, 201, 216, 224]. The elements of these matrices are hyperparameters that model correlations and anti-correlations between different categories [201]. The similarities between categories are directly encoded in matrices. Other techniques for encoding categorical variables include continuous latent spaces. In [270], each categorical variable is assigned a 2D continuous space, in which its categories are mapped in a way that correlated categories are close, and uncorrelated ones are far. The coordinates of the categories can be viewed as hyperparameters representing correlations.

The rest of this document is organized as follows. An illustrative optimization problem is introduced in Section 5.2 to guide the presentation. The essential concepts of kernels and GPs are presented in Section 5.3. In Section 5.4, the categorical distances and neighborhoods are developed. The novel neighborhoods are adapted and used within the CatMADS framework in Section 5.5 to provide proof of concept. Lastly, some numerical experiments benchmarking

CatMADS using the new neighborhoods with other solvers are presented in Section 5.6.

5.2 An illustrative mechanical-part design problem

To outline the proposed contributions, a mechanical-part design problem is used as an illustrative example in the following sections. The goal is to minimize the physical strain of a structural piece subject to a budget constraint and an ecological score constraint. A point represents a design, and it is composed of three categorical variables and two continuous variables. The categorical variables correspond to the choice of supplier $u \in \{\text{A}, \text{B}\}$, the material $a \in \{\text{ALUM}, \text{STEEL}, \text{COMP}[1], \text{WOOD}\}$, and the shape $s \in \{\text{SQUARE}, \text{CIRCLE}, \text{ELLIPSE}\}$. The continuous variables are a length $l \in [5, 10]$ and a ratio of recycle material $r \in [0, 1]$. The domain and a point are defined as

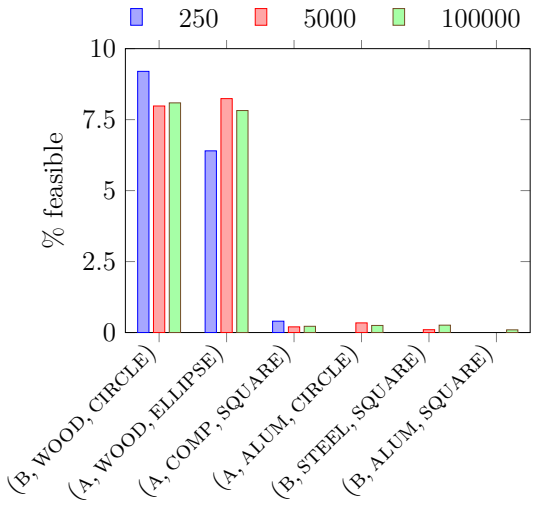
$$\boldsymbol{x} = (u, a, s, l, r) \in \mathcal{X} = \mathcal{X}^{\text{cat}} \times \mathcal{X}^{\text{con}} \tag{5.4}$$

where $\mathcal{X}^{\text{cat}} = \{\text{A}, \text{B}\} \times \{\text{ALUM}, \text{STEEL}, \text{COMP}, \text{WOOD}\} \times \{\text{SQUARE}, \text{CIRCLE}, \text{ELLIPSE}\}$, and $\mathcal{X}^{\text{con}} = [5, 10] \times [0, 1]$. The categorical set $\mathcal{X}^{\text{cat}}$ contains $2 \times 4 \times 3 = 24$ categorical components, and the feasible region Ω is unknown.

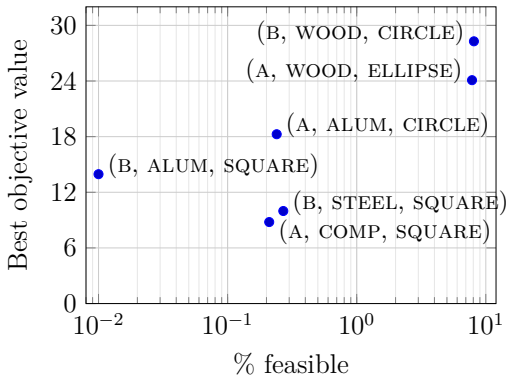
The problem is intentionally constructed as a blackbox in accordance with the scope of this work. The functions are evaluated through numerical iterative routines to a prescribed accuracy. By design, the problem features strong interactions between the categorical variables and constraint feasibility. Each categorical component $\boldsymbol{x}^{\text{cat}} = (u, a, s) \in \mathcal{X}^{\text{cat}}$ induces a different feasible region over the continuous subspace $\mathcal{X}^{\text{con}} = [5, 10] \times [0, 1]$. Some categories may appear intuitively similar, *e.g.*, the circle and ellipse categories for the cross-section shape. However, the objective function, and, in particular, the constraint functions, behave very differently across categories. As a result, constructing categorical neighborhoods without incorporating information from both the objective and the constraints can lead to misleading similarity measures and poor optimization performance. The difficulty of the problem is outlined in Figure 5.2 that describes three Latin Hypercube Sampling (LHS) experiments and a graph.

In the histogram of Figure 5.2a, LHS is performed independently within each categorical component. Each bar represents the percentage of feasible points obtained for a given number of samples. Only the categorical component for which at least one feasible point was sampled is presented. As the number of points per categorical component increases, more components with feasible points are discovered. The categorical component (A, ALUM, CIRCLE) does not

[1] COMP for "composite"



(a) Feasibility rate per categorical component.



(b) Best objective function value vs. % feasible with 100 000 points per categorical component.

Figure 5.2 Statistical analysis of feasibility per categorical component in the piece machining problem.

have any feasible point with 250 samples (there is no blue bin), but it does with 5 000 and 100 000 samples. The histogram suggests that only six categorical components may be capable of achieving feasibility.

The graph in Figure 5.2b shows the best feasible objective value obtained by performing an LHS with 100 000 points for each categorical component. Overall, best objective values are associated with smaller percentages of feasibility.

5.3 Mixed-variable kernels

This study uses similarity measures derived from available data to induce categorical distances including kernel-based measures. A similarity measure quantifies how close or related two data points are in the decision space. A kernel is a particular type of similarity measure. The data points of Problem (5.1) lie in the domain $\mathcal{X}$, and the kernel is defined on this domain. From such a kernel, a categorical kernel is extracted and used to construct categorical neighborhoods. In practice, kernels can compare mixed-variable points, enabling the construction of interpolation models that rely exclusively on similarity measures, such as mixed-variable Gaussian processes.

Formally, a kernel $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is a symmetric similarity measure, such that

1. $\kappa(\boldsymbol{x}, \boldsymbol{y}) = \kappa(\boldsymbol{y}, \boldsymbol{x})$ for any $\boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$, and

2. for any finite set of points $\{\boldsymbol{x}_{(1)}, \boldsymbol{x}_{(2)}, \ldots, \boldsymbol{x}_{(p)}\}$ with $p \in \mathbb{N}$, the symmetric $p \times p$ matrix $[K]_{i,j} = \kappa\left(\boldsymbol{x}_{(i)}, \boldsymbol{x}_{(j)}\right)$ for $i, j \in \{1, 2, \ldots, p\}$ is positive semi-definite.

Such kernel κ is said to be positive semi-definite. The kernel is constructed following the approach of the classic textbook [206]. For $t \in \{\text{cat}, \text{qnt}\}$ and $i \in I^t$, each variable $x_i^t \in \mathcal{X}_i^t$ is assigned a one-dimensional kernel $\kappa_i^t : \mathcal{X}_i^t \times \mathcal{X}_i^t \to \mathbb{R}$ of the appropriate variable type. The one-dimensional kernels are combined with multiplications that conserve symmetry and semi-positive definiteness.

Common kernels for quantitative variables include polynomial, Matern or Gaussian kernels, *e.g.*, see [206]. Typically, the quantitative kernel can be built directly as product of one-dimensional Gaussian kernels, *i.e.*,

$$\kappa^{\text{qnt}}\left(\boldsymbol{x}^{\text{qnt}}, \boldsymbol{y}^{\text{qnt}}\right) := \prod_{i=1}^{n^{\text{qnt}}} \exp\left(-\theta_i^{\text{qnt}}\left(x_i^{\text{qnt}} - y_i^{\text{qnt}}\right)^2\right), \tag{5.5}$$

where $\boldsymbol{\theta}^{\text{qnt}} := \left(\theta_1^{\text{qnt}}, \theta_2^{\text{qnt}}, \ldots, \theta_{n^{\text{qnt}}}^{\text{qnt}}\right) \in \mathbb{R}_+^{n^{\text{qnt}}}$ is the vector of hyperparameters of the quantitative kernel. The quantitative kernel and the following ones are all parameterized by hyperparameters. For readability, hyperparameters are not explicitly noted in the arguments of kernels.

5.3.1 Categorical kernels

As mentioned in Section 5.1.3, there are currently two main state-of-the-art approaches for constructing categorical kernels: one-hot encoding and matrix-based methods. Again, the matrix-based approach encodes similarity measures between categories directly into matrices. More precisely, each categorical variable $x_i^{\text{cat}} \in \mathcal{X}_i^{\text{cat}}$ is assigned a positive semi-definite matrix $T_i \in \mathbb{R}^{\ell_i^{\text{cat}} \times \ell_i^{\text{cat}}}$ where ℓ_i^{cat} is the number of categories of x_i^{cat}. A matrix $T_i^{\text{cat}} := L_i L_i^\top$ is built as a symmetric and positive semi-definite matrix where L_i is a lower-triangular matrix parameterized by a hypersphere decomposition (lengths and angles) [55, 136]. An element of a matrix L_i contains hyperparameters of the categorical kernel, whereas an element of a matrix T_i^{cat} is a correlation measure between two categories of the corresponding categorical variable. For example, in the machining piece problem, the similarity matrix associated with

the material choice $a \in \{\text{ALUM (A)}, \text{STEEL (S)}, \text{COMP (C)}, \text{WOOD (W)}\}$ can be expressed as

$$T_a^{\text{cat}} = \begin{bmatrix} \vartheta_{\text{A,A}} & & & \\ \vartheta_{\text{S,A}} & \vartheta_{\text{S,S}} & & \\ \vartheta_{\text{C,A}} & \vartheta_{\text{C,S}} & \vartheta_{\text{C,C}} & \\ \vartheta_{\text{W,A}} & \vartheta_{\text{W,S}} & \vartheta_{\text{W,C}} & \vartheta_{\text{W,W}} \end{bmatrix} \tag{5.6}$$

where $\vartheta_{i,j} \in [-1, 1]$ denotes a similarity measure between categories i and j, with $i, j \in \{\text{A}, \text{S}, \text{C}, \text{W}\}$, and T_a^{cat} is symmetric by construction. The values of $\vartheta_{i,j}$ are implicitly determined through the hyperparameters of the lower-triangular matrix L_a in the hypersphere decomposition, such that $T_a^{\text{cat}} = L_a L_a^\top$. For similarities between identical categories (autocorrelations), the homoscedastic parametrization fixes a constant value across categories, typically $\vartheta_{i,i} = 1$, whereas the heteroscedastic parametrization allows $\vartheta_{i,i} \in [-1, 1]$ to be learned as adjustable hyperparameters.

For the matrix-based approach, the categorical kernel is derived from the categorical matrices $T_1^{\text{cat}}, T_2^{\text{cat}}, \ldots, T_{n^{\text{cat}}}^{\text{cat}}$. Essentially, two points $\boldsymbol{x}$ and $\boldsymbol{y}$ are compared variable-wise, with each matrix T_i^{cat} providing the correlation parameter comparing categories x_i^{cat} and y_i^{cat}. These correlation parameters are then multiplied. The number of hyperparameters associated to the categorical variable x_i^{cat} depends on the number of categories ℓ_i and the chosen parametrization of L_i^{cat}. While more hyperparameters increase modeling flexibility, they also imply greater computational costs: see [224] for a detailed presentation of parametrizations and the hypersphere decomposition.

The matrix-based approach typically involves many hyperparameters. in BO. Alternative approaches with fewer hyperparameters are available. The common *one-hot encoding* technique assigns a unique binary variable to each category. The categorical component $\boldsymbol{x}^{\text{cat}} \in \mathcal{X}^{\text{cat}}$ is represented by a total of $\sum_{i=1}^{n^{\text{cat}}} \ell_i$ binary variables, where $\ell_i \in \mathbb{N}$ is the number of categories of the i-th categorical variable x_i^{cat}. For each categorical variable, exactly one of its binary variables is assigned the value 1, and the others are set to 0. In the machining problem, recall that the categorical variables are $u \in \{\text{A}, \text{B}\}$, $a \in \{\text{ALUM}, \text{STEEL}, \text{COMP}, \text{WOOD}\}$ and $s \in \{\text{SQUARE}, \text{CIRCLE}, \text{ELLIPSE}\}$. The categorical component (B, WOOD, CIRCLE) would be represented by $((0,1), (0,0,0,1), (0,1,0))$, whereas (A, STEEL, ELLIPSE) would be represented by $((1,0), (0,1,0,0), (0,0,1))$. With this approach, each binary variable can be assigned a hyperparameter. Then, a categorical kernel can be defined via Gaussian kernels as follows

$$\kappa^{\text{cat}}\left(\boldsymbol{x}^{\text{cat}}, \boldsymbol{y}^{\text{cat}}\right) \coloneqq \prod_{i=1}^{n^{\text{cat}}} \exp\left(-\left\|\boldsymbol{E}_i\left(x_i^{\text{cat}}\right) - \boldsymbol{E}_i\left(y_i^{\text{cat}}\right)\right\|_{\boldsymbol{\theta}_i^{\text{cat}}}^2\right) \tag{5.7}$$

where $\boldsymbol{E}_i\left(x_i^{\text{cat}}\right) \in \left\{\boldsymbol{z} \in \{0,1\}^{\ell_i} \,:\, \mathbf{1}^\top z = 1\right\}$ is the *one-hot* representation of the i-th categorical variable x_i^{cat}, and $\boldsymbol{\theta}_i^{\text{cat}} \in \mathbb{R}^{\ell_i}$ is the vector of hyperparameters of x_i^{cat}, and $\|\boldsymbol{u}-\boldsymbol{v}\|_{\boldsymbol{w}}^2 := (\boldsymbol{u}-\boldsymbol{v})^\top \text{diag}\,(\boldsymbol{w})\,(\boldsymbol{u}-\boldsymbol{v})$ is the weighted Euclidean norm. In (5.7), the norm represents distances between binary vectors, where the weights are the hyperparameters of the binary variables.

5.3.2 Adjusting the kernels with mixed-variable GPs

Once appropriate quantitative and categorical kernels are defined, the kernel $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ can be constructed with the product

$$\kappa\left(\boldsymbol{x}, \boldsymbol{y}\right) := \kappa^{\text{cat}}\left(\boldsymbol{x}^{\text{cat}}, \boldsymbol{y}^{\text{cat}}\right) \kappa^{\text{qnt}}\left(\boldsymbol{x}^{\text{qnt}}, \boldsymbol{y}^{\text{qnt}}\right), \tag{5.8}$$

which is parameterized by a vector of hyperparameters $\boldsymbol{\theta} := (\boldsymbol{\theta}^{\text{cat}}, \boldsymbol{\theta}^{\text{qnt}})$.

To adjust the hyperparameters of the kernel, a regression model linking input data points to their corresponding images is required. In BO, GPs are commonly used as surrogate models for objective and constraint functions, as they can represent a wide variety of functions, including mixed-variable ones.

For readability, the construction of GP surrogates is presented for the objective function. This surrogate is noted $\tilde{f}$ and it is constructed with a set of points $\mathbb{X} := \{\boldsymbol{x}_{(1)}, \boldsymbol{x}_{(2)}, \ldots, \boldsymbol{x}_{(p)}\}$ and a vector of corresponding images $\boldsymbol{f} = (f(\boldsymbol{x}_{(1)}), f(\boldsymbol{x}_{(2)}), \ldots, f(\boldsymbol{x}_{(p)}))$. This procedure yields a prediction function $\hat{f} : \mathcal{X} \to \mathbb{R}$ and a variance function $\hat{\sigma}^2 : \mathcal{X} \to \mathbb{R}_+$ that, in the noiseless setting, can be expressed as in [206]:

$$\begin{aligned} \hat{f}(\boldsymbol{x}) &:= \boldsymbol{\kappa}(\boldsymbol{x})^\top K^{-1} \boldsymbol{f}, \\ \hat{\sigma}^2(\boldsymbol{x}) &:= \kappa(\boldsymbol{x}, \boldsymbol{x}) - \boldsymbol{\kappa}(\boldsymbol{x})^\top K^{-1} \boldsymbol{\kappa}(\boldsymbol{x}) \geq 0, \end{aligned} \tag{5.9}$$

where $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ is the kernel, $\boldsymbol{\kappa}(\boldsymbol{x}) := \left(\kappa(\boldsymbol{x}, \boldsymbol{x}_{(1)}), \kappa(\boldsymbol{x}, \boldsymbol{x}_{(2)}), \ldots, \kappa(\boldsymbol{x}, \boldsymbol{x}_{(p)})\right) \in \mathbb{R}^p$ is the kernel vector comparing an input point $\boldsymbol{x} \in \mathcal{X}$ and the points in $\mathbb{X}$, and $K \in \mathbb{R}^{p \times p}$ is the kernel matrix that compares all points in $\mathbb{X}$ such that $[K]_{i,j} := \kappa\left(\boldsymbol{x}_{(i)}, \boldsymbol{x}_{(j)}\right)$ for $i, j \in \{1, 2, \ldots, p\}$.

The kernel characterizes the GP by quantifying how correlated or similar input points are, *i.e.*, how close their output should be. Concretely, it controls the general behavior of the fit, *e.g.*, the smoothness in a continuous setting. In the mixed-variable setting, defining an appropriate mixed-variable kernel is key to constructing a GP surrogate that can model the functions of interest. In fact, the GP that best fits the available data is found by adjusting the hyperparameters of the kernel. They are typically obtained by maximizing the marginal

likelihood of the GP [206]:

$$\theta_\star \in \operatorname*{argmax}_\theta \log \mathbb{P}\left[\boldsymbol{f} \mid \mathbb{X}, \boldsymbol{\theta}\right]$$
$$\text{where} \quad \log \mathbb{P}\left[\boldsymbol{f} \mid \mathbb{X}, \boldsymbol{\theta}\right] = -\frac{1}{2}\boldsymbol{f}^\top K^{-1}\boldsymbol{f} - \frac{1}{2}\log\det\left(K\right) - \frac{p}{2}\log\left(2\pi\right). \tag{5.10}$$

The marginal likelihood is continuously differentiable. Hence, it can be optimized with standard continuous solvers. After this step, the GP is properly constructed and the categorical kernel is finely adjusted to the data problem. The categorical kernel is used in the next section.

The GP surrogates of the constraint functions are constructed using the same procedure as for the objective function. Specifically, for $j \in J$, each constraint function g_j is provided a probabilistic surrogate $\tilde{g}_j$ with a predictive mean function $\hat{g}_j : \mathcal{X} \to \mathbb{R}$ and variance function $\hat{\sigma}_j^2 : \mathcal{X} \to \mathbb{R}_+$, as well as its own kernel and hyperparameters.

5.4 Surrogate-based categorical neighborhoods

Kernels and surrogate models are used to construct neighborhoods that structure categorical variables. The proposed neighborhoods are defined using two categorical distances, one associated with the objective function and another one associated with the constraint functions. Proximity between categorical variables simultaneously considers similarities in the objective and constraint functions.

This section focuses exclusively on categorical variables. Quantitative variables are temporarily omitted to simplify the presentation. The surrogate models $\tilde{f}$ and $\tilde{g}$, as well as the kernel, are defined on the categorical set $\mathcal{X}^{\text{cat}}$. A neighborhood constructed at an incumbent $\boldsymbol{u} \in \mathcal{X}^{\text{cat}}$ is denoted $\mathcal{N}(\boldsymbol{u}; m)$ where $m \in \{1, 2, \ldots, |\mathcal{X}^{\text{cat}}|\}$ is the number of neighbors. This notation is formalized at the end of the section.

Section 5.4.1 presents categorical distances that establish proximity with respect to the objective function. Afterwards, Section 5.4.2 proposes a categorical distance for the constraint functions. Based on these categorical distances, surrogate-based neighborhoods are developed in Section 5.4.3.

5.4.1 Categorical distance for the objective function

Kernels are equivalent to scalar products in Hilbert spaces [175]. This property allows categorical kernels to be transformed into categorical distances. Let $\boldsymbol{\phi} : \mathcal{X}^{\text{cat}} \to \mathcal{H}$ be a mapping

that maps the categorical components of the categorical set $\mathcal{X}^{\text{cat}}$ into a Hilbert space $\mathcal{H}$ equipped with a scalar product. Then, the categorical kernel is uniquely defined according to the Moore-Aronszajn Theorem [16] and satisfies $\kappa^{\text{cat}}(\boldsymbol{u}, \boldsymbol{v}) = \boldsymbol{\phi}(\boldsymbol{u})^\top \boldsymbol{\phi}(\boldsymbol{v})$. Note that the categorical kernel κ^{cat} is still computed as described in Section 5.3, *i.e.*, with Gaussian kernels or matrices. The categorical distance $d_f : \mathcal{X}^{\text{cat}} \times \mathcal{X}^{\text{cat}} \to \mathbb{R}_+$ is defined with the categorical kernel [119], and it is ensured to be a metric via the implicit mapping ϕ:

$$\begin{aligned} d_f(\boldsymbol{u}, \boldsymbol{v}) &:= \kappa^{\text{cat}}(\boldsymbol{u}, \boldsymbol{u}) + \kappa^{\text{cat}}(\boldsymbol{v}, \boldsymbol{v}) - 2\kappa^{\text{cat}}(\boldsymbol{u}, \boldsymbol{v}) \\ &= \boldsymbol{\phi}(\boldsymbol{u})^\top \boldsymbol{\phi}(\boldsymbol{u}) + \boldsymbol{\phi}(\boldsymbol{v})^\top \boldsymbol{\phi}(\boldsymbol{v}) - 2\boldsymbol{\phi}(\boldsymbol{u})^\top \boldsymbol{\phi}(\boldsymbol{v}) \\ &= \|\boldsymbol{\phi}(\boldsymbol{u}) - \boldsymbol{\phi}(\boldsymbol{v})\|^2. \end{aligned} \tag{5.11}$$

Although the mapping ϕ and the Hilbert space $\mathcal{H}$ are not explicitly known, (5.11) shows that the categorical distance can be equivalently interpreted in a Hilbert space with $d_f(\boldsymbol{u}, \boldsymbol{v}) = \|\boldsymbol{\phi}(\boldsymbol{u}) - \boldsymbol{\phi}(\boldsymbol{v})\|^2$. Without constraint functions, categorical neighborhoods are characterized only by the categorical distance. Hence, unconstrained neighborhoods can be viewed as being implicitly constructed in a Hilbert space, where two categorical components $\boldsymbol{u}, \boldsymbol{v} \in \mathcal{X}^{\text{cat}}$ that are more similar correspond to a smaller distance $\|\boldsymbol{\phi}(\boldsymbol{u}) - \boldsymbol{\phi}(\boldsymbol{v})\|^2$. Figure 5.3 illustrates this Hilbert space as a two-dimensional continuous space, where categorical components are mapped into continuous vectors. Note that this is only an equivalent representation. In practice, the distance is computed as $d_f(\boldsymbol{u}, \boldsymbol{v}) = \kappa^{\text{cat}}(\boldsymbol{u}, \boldsymbol{u}) + \kappa^{\text{cat}}(\boldsymbol{v}, \boldsymbol{v}) - 2\kappa^{\text{cat}}(\boldsymbol{u}, \boldsymbol{v})$.

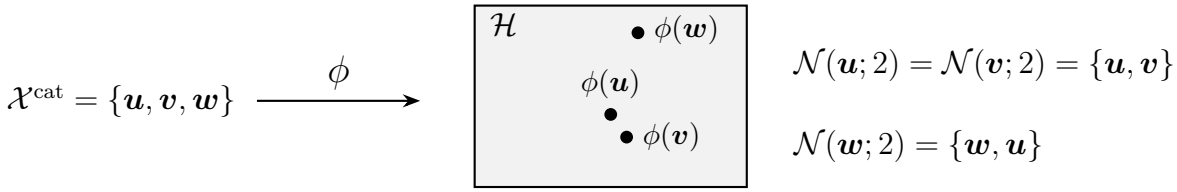


Figure 5.3 Visualization of a unconstrained neighborhood equivalently constructed in a Hilbert space.

In Figure 5.3, the categorical components are mapped in a Hilbert space via an implicit mapping ϕ. Schematically, neighborhoods containing two elements can be viewed as follows: the neighborhoods constructed at $\boldsymbol{u}$ and $\boldsymbol{v}$ coincide and contain the components $\{\boldsymbol{u}, \boldsymbol{v}\}$, while the neighborhood constructed at $\boldsymbol{w}$ contains $\{\boldsymbol{w}, \boldsymbol{u}\}$.

On the one hand, the construction of the categorical distance d_f requires non-negligible computational effort, notably because of the maximization of the marginal likelihood of the GP in (5.10). On the other hand, it is induced by a finely-tuned kernel. The categorical distance

in (5.11) is constructed from the problem data through the kernel, so that correlated categorical components are embedded closer to each other in a Hilbert space, while uncorrelated components are placed farther apart. This behavior is particularly useful for constructing categorical neighborhoods. This distance is finer than binary categorical distances, such as the Hamming distance, which treats all mismatches equally and does not use information from the problem. As an example, let $\mathcal{X}^{\text{cat}} = \{\text{R}, \text{B}, \text{G}\}$ and d_{HM} denote the Hamming distance. Then $d_{\text{HM}}(\text{R}, \text{G}) = d_{\text{HM}}(\text{R}, \text{B}) = 1$, implying that it is not possible to determine whether green or blue is closer to red.

If a surrogate model is available, but no kernel is defined, the categorical pseudo-distance can be approximated by a pseudo-distance using prediction values, such that $d_f(\boldsymbol{u}, \boldsymbol{v}) \approx \left|\hat{f}(\boldsymbol{u}) - \hat{f}(\boldsymbol{v})\right|$. This pseudo-distance trivially satisfies symmetry and triangular inequality properties, but not the identity of indiscernibles property since $d_f(\boldsymbol{u}, \boldsymbol{v}) = 0 \not\Leftrightarrow \boldsymbol{u} = \boldsymbol{v}$ when $\hat{f}$ is not injective. Hence, categorical components with similar prediction values are considered close. Conceptually, this mimics the behavior of a kernel inducing an interpolation model, such as GPs. Although simple, this pseudo-distance captures proximity with respect to the objective function f. That said, when a kernel with adjusted hyperparameters is available, the categorical distance of (5.11) is generally preferable, as it corresponds to measuring similarity in an inner-product Hilbert space.

5.4.2 Categorical pseudo-distance for constraint functions

As discussed previously, the main goal is to construct neighborhoods that incorporate information from both the objective and constraint functions. This section develops a pseudo-distance that establishes proximity with respect to the constraint functions.

When a neighborhood is constructed at a feasible component, it should first maintain feasibility and then promote improvement in the objective function, *i.e.*, similarity with respect to the objective. In this setting, differences in the degree of feasibility between categorical components that are both predicted to be feasible are of limited importance. From an optimization perspective, once feasibility is predicted, the focus should instead be on improving the objective function.

In particular, large distances between a "highly feasible" component and a "marginally feasible" component should be avoided, as both satisfy the constraints and should be compared primarily regarding the objective. For this reason, the proposed pseudo-distance is constructed so that it evaluates to zero when both compared components are predicted to be feasible.

The next functions are used to construct the pseudo-distance. For each $j \in J$, a function $\hat{g}_j^+ : \mathcal{X}^{\text{cat}} \to \mathbb{R}$ is defined from the predictive model $\hat{g}_j : \mathcal{X}^{\text{cat}} \to \mathbb{R}$ and its associated uncertainty model $\hat{\sigma}_j : \mathcal{X}^{\text{cat}} \to \mathbb{R}+$, both derived from the surrogate model $\tilde{g}_j$ of the j-th constraint. Let $\lambda \geq 0$ be a relaxation parameter used to relax a predicted constraint value $\hat{g}_j(\boldsymbol{u})$ using its associated uncertainty $\hat{\sigma}_j(\boldsymbol{u}) \geq 0$. The function $\hat{g}_j^+$ sets to zero any predicted constraint value $\hat{g}_j(\boldsymbol{u})$ that is feasible after relaxation by $\lambda\hat{\sigma}_j(\boldsymbol{u})$ and normalizes the remaining values, such that

$$\hat{g}_j^+(\boldsymbol{u}) \coloneqq \begin{cases} 0 & \text{if } \hat{g}_j(\boldsymbol{u}) - \lambda\hat{\sigma}_j(\boldsymbol{u}) \leq 0, \\ \psi\left(\hat{g}_j(\boldsymbol{u})\right) & \text{otherwise,} \end{cases} \tag{5.12}$$

where ψ is a normalization function ensuring that $\hat{g}_j^+(\boldsymbol{u}) \in [0,1]$ holds for any $\boldsymbol{u} \in \mathcal{X}^{\text{cat}}$. For example, ψ may apply a min-max normalization on the bounds of $\hat{g}_j(\boldsymbol{u})$ when $\hat{g}_j(\boldsymbol{u}) - \lambda\hat{\sigma}_j(\boldsymbol{u}) > 0$ is considered. Note that if the surrogate model does not provide an uncertainty measure, then λ can be set to zero, meaning that no uncertainty is taken into account. For readability, (5.12) may be expressed in a vector form $\hat{\boldsymbol{g}}^+ : \mathcal{X}^{\text{cat}} \to \mathbb{R}_+^{|J|}$, considering all indices $j \in J$ compactly.

The pseudo-distance function associated with the constraint functions is constructed from the vector function $\hat{\boldsymbol{g}}^+$, as follows :

$$d_{\boldsymbol{g}}(\boldsymbol{u}, \boldsymbol{v}) \coloneqq \|\hat{\boldsymbol{g}}^+(\boldsymbol{u}) - \hat{\boldsymbol{g}}^+(\boldsymbol{v})\|_p, \tag{5.13}$$

where $p \geq 1$ and $d_{\boldsymbol{g}}(\boldsymbol{u}, \boldsymbol{v}) = 0$ are ensured when both arguments are predicted to be feasible after relaxation, *i.e.*, when $\hat{\boldsymbol{g}}(\boldsymbol{u}) - \lambda\hat{\boldsymbol{\sigma}}(\boldsymbol{u}) \leq \boldsymbol{0}$ and $\hat{\boldsymbol{g}}(\boldsymbol{v}) - \lambda\hat{\boldsymbol{\sigma}}(\boldsymbol{v}) \leq \boldsymbol{0}$. The normalization function ψ ensures that the pseudo-distance $d_{\boldsymbol{g}}$ is not biased by the relative scales of the constraint function values. The symmetry and triangular inequality properties are trivially satisfied by virtue of the p-norm. However, the identity of indiscernibles is not respected, *i.e.*, $d_{\boldsymbol{g}}(\boldsymbol{u}, \boldsymbol{v}) = 0 \not\Leftrightarrow \boldsymbol{u} = \boldsymbol{v}$ since the vector function $\hat{\boldsymbol{g}}^+$ is not injective by design. This last unsatisfied property makes $\hat{\boldsymbol{g}}^+$ a pseudo-distance function instead of a distance function.

5.4.3 Surrogate-based neighborhoods

The novel categorical neighborhoods are developed in this section. A neighborhood is constructed at a component $\boldsymbol{u} \in \mathcal{X}^{\text{cat}}$. The other components in the neighborhood are selected based on trade-offs between distances concerning the objective and the constraint functions. This selection is formalized using notions of dominance inspired by bi-objective optimization. In the following, two ranking functions are derived from the distance functions introduced in

previous sections: a primary and a secondary ranking function, both parameterized by a component $\boldsymbol{u} \in \mathcal{X}^{\text{cat}}$ at which the neighborhood is constructed. Conceptually, the first selected components correspond to Pareto-optimal solutions concerning the two ranking functions, while subsequent components prioritize the primary ranking function over the secondary one.

The primary and secondary ranking functions are defined in two cases depending on the component $\boldsymbol{u} \in \mathcal{X}^{\text{cat}}$. If $\boldsymbol{u}$ is feasible, then the primary ranking function is derived from $d_{\boldsymbol{g}}$ (constraints), since the priority is to construct neighborhoods with components that maintain feasibility. Otherwise, $\boldsymbol{u}$ is infeasible, and the primary ranking function is based on d_f (objective), allowing greater flexibility with respect to feasibility. This may allow finding promising solutions that are difficult to reach when imposing feasibility at all times.

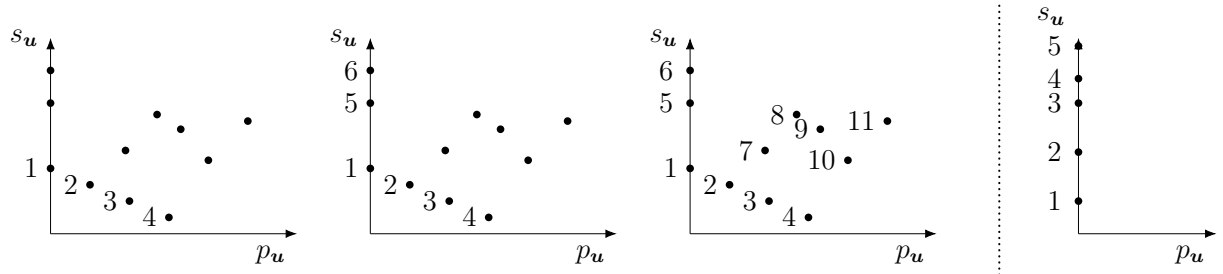
The primary and secondary ranking functions, noted $p_u : \mathcal{X}^{\text{cat}} \to \mathbb{R}_+$ and $s_u : \mathcal{X}^{\text{cat}} \to \mathbb{R}$ respectively, are parameterized by $\boldsymbol{u} \in \mathcal{X}^{\text{cat}}$ and defined as follows

$$\begin{aligned} p_{\boldsymbol{u}}(\boldsymbol{v}) &\coloneqq d_{\boldsymbol{g}}(\boldsymbol{u}, \boldsymbol{v}) \quad \text{and} \quad s_{\boldsymbol{u}}(\boldsymbol{v}) \coloneqq d_f(\boldsymbol{u}, \boldsymbol{v}) \quad \text{if } \boldsymbol{u} \in \Omega, \\ p_{\boldsymbol{u}}(\boldsymbol{v}) &\coloneqq d_f(\boldsymbol{u}, \boldsymbol{v}) \quad \text{and} \quad s_{\boldsymbol{u}}(\boldsymbol{v}) \coloneqq d_{\boldsymbol{g}}(\boldsymbol{u}, \boldsymbol{v}) \quad \text{if } \boldsymbol{u} \notin \Omega. \end{aligned} \tag{5.14}$$

An illustrative example is presented before introducing the formal ordering rules that determine the components selected in a neighborhood. These rules are induced by the ranking functions. In this example, there are twelve categorical components. Hence, eleven components must be ordered, since the neighborhood stems from $\boldsymbol{u} \in \mathcal{X}^{\text{cat}}$, which is naturally ordered first. Visually, the ordering of the components is performed in the space of images $(s_{\boldsymbol{u}}, p_{\boldsymbol{u}})$ across three steps, each corresponding to a sub-figure from left to right in Figure 5.4.

The first four components are the Pareto solutions, *i.e.*, the nondominated solutions, in the space of images. The Pareto components are ordered among themselves solely with the primary ranking function $p_{\boldsymbol{u}}$. The components 5 and 6 are not Pareto, but they have a primary ranking function value of zero with the component $\boldsymbol{u}$ at which the neighborhood is constructed. This situation may most notably occur when the primary ranking function is based on the pseudo-distance $d_{\boldsymbol{g}}$: recall that when both components are predicted to be feasible, this pseudo-distance returns zero. Among themselves, components 5 and 6 are ordered using the secondary ranking function. Finally, the remaining components 7 to 11 are directly ordered with the primary ranking function.

Note that when the problem is unconstrained, the component $\boldsymbol{u} \in \Omega$ and, consequently, the primary ranking function is based on $d_{\boldsymbol{g}}$. However, in this case, the pseudo-distance function $d_{\boldsymbol{g}}$ is always set to zero by convention, since its components are always feasible. Therefore,



(a) Pareto components.

(b) Non-Pareto components with $p_{\boldsymbol{u}}(\boldsymbol{v}) = 0$.

(c) Remaining components ordered with the primary ranking function.

(d) Unconstrained problem.

Figure 5.4 Ordering components with ranking functions. (a)-(c) three steps for constrained problems. (d) single step for unconstrained problems. The component $\boldsymbol{u}$ is located at the origin and is not displayed.

only the distance function of the objective d_f is considered. An unconstrained example is illustrated with six components (including $\boldsymbol{u}$ that is not displayed) in Figure 5.4d, where the ordering is done directly based on their height.

The relation establishing a partial order on the categorical set $\mathcal{X}^{\text{cat}}$, given a component $\boldsymbol{u} \in \mathcal{X}^{\text{cat}}$ and ranking functions, is now formally introduced.

Definition 30 (Ordering relation)**.** *Given a component $\boldsymbol{u} \in \mathcal{X}^{\text{cat}}$, and ranking functions $p_{\boldsymbol{u}} : \mathcal{X}^{\text{cat}} \to \mathbb{R}_+$ and $s_{\boldsymbol{u}} : \mathcal{X}^{\text{cat}} \to \mathbb{R}_+$, the ordering relation $\preceq$ defined on the categorical set $\mathcal{X}^{\text{cat}}$ is such that for any pair $\boldsymbol{v}, \boldsymbol{w} \in \mathcal{X}^{\text{cat}}$, $\boldsymbol{v}$ is ordered before or tied to $\boldsymbol{w}$, noted $\boldsymbol{v} \preceq \boldsymbol{w}$, when any of the following conditions hold:*

1. $p_{\boldsymbol{u}}(\boldsymbol{v}) = p_{\boldsymbol{u}}(\boldsymbol{w}) = 0$ *and* $s_{\boldsymbol{u}}(\boldsymbol{v}) \leq s_{\boldsymbol{u}}(\boldsymbol{w})$,
2. $p_{\boldsymbol{u}}(\boldsymbol{v}) > 0$, $p_{\boldsymbol{u}}(\boldsymbol{w}) > 0$ *and* $p_{\boldsymbol{u}}(\boldsymbol{v}) \leq p_{\boldsymbol{u}}(\boldsymbol{w})$,
3. $\boldsymbol{v}$ *is not dominated by any component of* $\mathcal{X}^{\text{cat}} \setminus \{\boldsymbol{u}\}$ *in the space of images* $(p_{\boldsymbol{u}}, s_{\boldsymbol{u}})$, *but* $\boldsymbol{w}$ *is dominated by at least one such component,*
4. $p_{\boldsymbol{u}}(\boldsymbol{v}) \leq p_{\boldsymbol{u}}(\boldsymbol{w})$.

The ordering relation in Definition 30 does not establish a total order, because there can be ties, notably when $\boldsymbol{v}$ and $\boldsymbol{w}$ are such that $p_{\boldsymbol{u}}(\boldsymbol{v}) = p_{\boldsymbol{u}}(\boldsymbol{w})$ and $s_{\boldsymbol{u}}(\boldsymbol{v}) = s_{\boldsymbol{u}}(\boldsymbol{w})$. Nevertheless, it provides a systematic way to identify promising categorical components with information from both the objective and constraint functions. If the problem is unconstrained, then only the first condition is used since all components are feasible, as illustrated in Figure 5.4d.

Now that the ordering relation is formally defined, the neighborhoods based on surrogate models are finally introduced.

Definition 31 (Surrogate-based neighborhood)**.** *For a given component* $\boldsymbol{u} \in \mathcal{X}^{\text{cat}}$ *and integer* $m \in \{1, 2, \ldots, |\mathcal{X}^{\text{cat}}|\}$*, the* surrogate-based neighborhood $\mathcal{N}(\boldsymbol{u}; m) \subseteq \mathcal{X}^{\text{cat}}$ *is the set containing* m *categorical components of lowest ordering with respect to an ordering relation* $\preceq$ *defined on* $\mathcal{X}^{\text{cat}}$*. The relation* $\boldsymbol{v} \preceq \boldsymbol{w}$ *is satisfied whenever* $\boldsymbol{v} \in \mathcal{N}(\boldsymbol{u}; m)$ *and* $\boldsymbol{w} \notin \mathcal{N}(\boldsymbol{u}; m)$*.*

Definition 31 generalizes distance-induced neighborhoods in [23] by using relations instead of a single distance for selecting components. In fact, when a problem is unconstrained, a surrogate-based neighborhood is equivalent to a distance-induced neighborhood using d_f. That said, the implementation in [23] uses a basic mixed-variable interpolation to construct the categorical distance. The kernel-induced categorical distance d_f introduced in Section 5.4.1 provides a more structured similarity measure, as it is induced by an inner product in a Hilbert space through well-defined categorical kernels. The improvements does not only concerns constrained problems.

5.4.4 Illustrative case study of the mechanical-part problem

The mechanical-part design problem from Section 5.2 is used here to develop a case study with categorical neighborhoods. A single Latin Hypercube Sampling (LHS) of 96 points is performed on the mixed-variable domain described in Section 5.2, in which three points are generated for each of the 32 categorical components. Only one point sample is feasible, that is, $\boldsymbol{x}_\star = (\text{B}, \text{WOOD}, \text{CIRCLE}, 5.5, 1)$ with objective function value $f(\boldsymbol{x}_\star) = 28.55$ is known. In study, the goal is to find another triplet of categorical variables maintaining feasibility and improving the objective value without modifying the continuous variables. The continuous variables are fixed, and the current categorical component is noted $\boldsymbol{u} = (\text{B}, \text{WOOD}, \text{CIRCLE}) \in \mathcal{X}^{\text{cat}}$. The neighborhood resulting of the Gower distance has exactly six categorical components, at distance one from $\boldsymbol{u}$. Table 5.1 compares three neighborhood-generation strategies producing exactly six points. Each point has same continuous variables of $\boldsymbol{x}_\star$ and is different than $\boldsymbol{u}$. The first and simplest strategy uses the Gower distance. The second strategy uses an objective-based categorical distance [23], induced by a mixed-variable Inverse Distance Weighting (IDW) regression model. The third strategy employs the surrogate-based neighborhoods from Definition 31, using GPs as surrogate models.

The results highlight clear differences between the three strategies. The Gower distance and the objective-based neighborhood fail to identify any feasible point, whereas the surrogate-

Table 5.1 Neighbors of $\boldsymbol{u} = (\text{B}, \text{WOOD}, \text{CIRCLE})$ proposed by three strategies.

Strategy	Neighbor	Supplier	Material	Shape	f	g_1	g_2	Feasible
Gower distance	1	B	ALUM	CIRCLE	18.4928	-0.1300	0.0150	No
	2	B	STEEL	CIRCLE	14.5128	-0.0225	0.0625	No
	3	B	COMP	CIRCLE	9.5329	-0.0800	0.0575	No
	4	B	WOOD	SQUARE	24.0428	0.0050	0.0650	No
	5	B	WOOD	ELLIPSE	24.5628	-0.1075	0.0125	No
	6	A	WOOD	CIRCLE	28.3468	0.3450	0.5600	No
Objective-based [23]	1	A	WOOD	CIRCLE	28.3468	0.3450	0.5600	No
	2	B	WOOD	SQUARE	24.0428	0.0050	0.0650	No
	3	B	ALUM	CIRCLE	18.4928	-0.1300	0.0150	No
	4	B	WOOD	ELLIPSE	24.5628	-0.1075	0.0125	No
	5	B	STEEL	CIRCLE	14.5128	-0.0225	0.0625	No
	6	B	COMP	CIRCLE	9.5329	-0.0800	0.0575	No
Surrogate-based	**1**	A	ALUM	CIRCLE	**18.2868**	**-0.0225**	**-0.0451**	**Yes**
	2	B	ALUM	CIRCLE	18.4928	-0.1300	0.0150	No
	3	A	WOOD	CIRCLE	28.3468	0.3450	0.5600	No
	4	A	WOOD	ELLIPSE	**24.3568**	**-0.6900**	**-0.4715**	**Yes**
	5	B	STEEL	SQUARE	**10.0028**	**-0.0225**	**-0.0451**	**Yes**
	6	A	COMP	SQUARE	**8.8169**	**-0.0225**	**-0.0451**	**Yes**

based neighborhood proposed identifies four feasible points. Recall that Figure 5.2b shows the results of a Latin Hypercube Sampling with 100 000 samples per categorical component, suggesting that only six categorical components are likely to admit feasible continuous regions. The proposed method identifies four of these six components within the six evaluations, while the incumbent component accounts for another one. The only remaining feasible component is (B, ALUM, SQUARE), which appears to be extremely unlikely to yield feasible points, with only 0.01% feasibility observed in a LHS of 100 000 points in Figure 5.2b.

Only the surrogate-based strategy improves the objective value while maintaining feasibility. The best improvement is obtained with the categorical component (A, COMP, SQUARE), yielding an objective value of 8.8169. This component also corresponds to that of the best solution identified in Section 5.2 using an LHS with 100, 000 points per component.

The ordering and selection of categorical components in the surrogate-based neighborhood are presented in Figure 5.5. The figure plots the primary and secondary ranking functions $p_{\boldsymbol{u}}$ and $s_{\boldsymbol{u}}$ introduced in Section 5.4.3, with logarithmic scales for readability. The selected components are shown in blue, together with their categorical component and a number indicating their ranking. On the horizontal axis $p_{\boldsymbol{u}}$, an axis break is introduced to display components with distance zero on a logarithmic scale. In the figure, the feasible neighbors all

have a distance of zero with respect to $p_{\boldsymbol{u}}$: they lie on the vertical axis. The objective-based approach identifies (B, STEEL, SQUARE), which corresponds to the component ranked fifth by the surrogate-based approach.

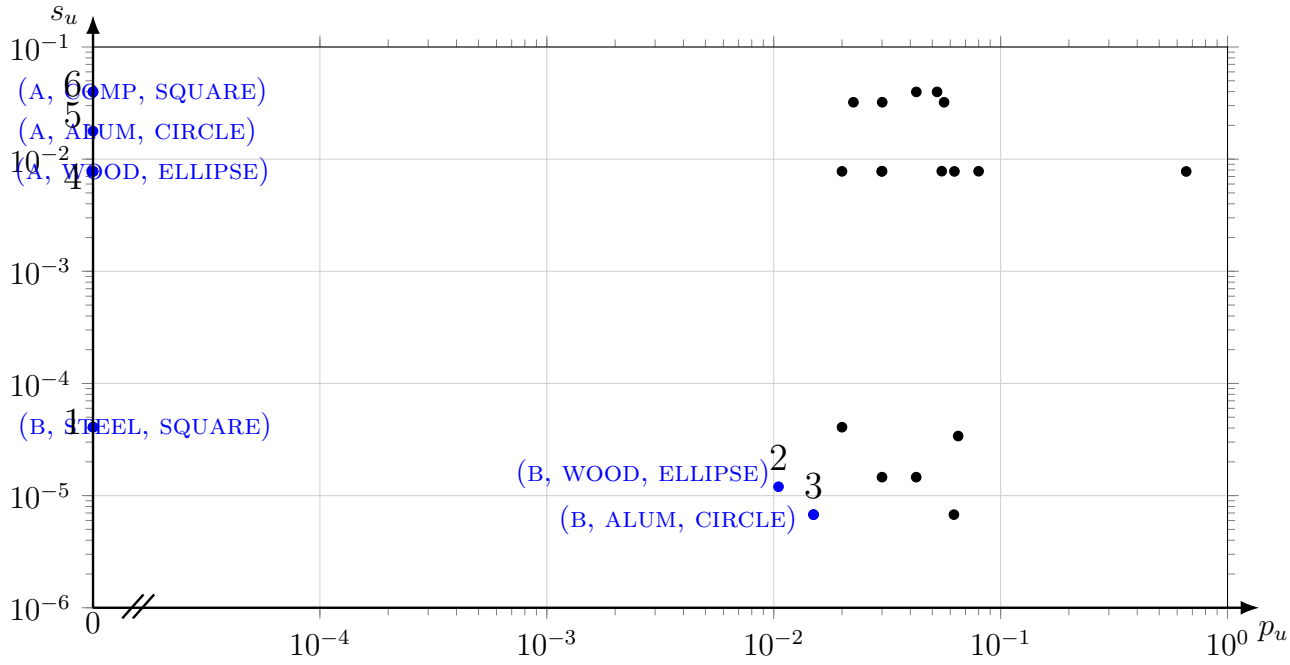


Figure 5.5 Ordering of categorical components with the surrogate-based neighborhood in the mechanical-part design problem. The incumbent $\boldsymbol{u} = (\text{B}, \text{WOOD}, \text{CIRCLE})$ is at the origin and it is not displayed.

5.5 Adapting surrogate-based neighborhoods for mixed-variable direct search

The categorical neighborhoods developed in Section 5.4 are now used for optimization in this section, which also reintroduces the integer and continuous variables discarded previously.

5.5.1 Adapting the surrogate-neighborhoods for mixed-variables domains

The pseudo-distance and distance functions from Section 5.4 require minor adaptations now that the surrogate models are defined on the domain $\mathcal{X}$ rather than on the categorical set $\mathcal{X}^{\text{cat}}$. They are now computed between points in the domain, while the quantitative variables are held fixed so that comparisons only concern the categorical variables. More precisely, for a given and fixed quantitative component $\boldsymbol{x}^{\text{qnt}} \in \mathcal{X}$, the distances introduced in Section 5.4

are computed on the set

$$(\mathcal{X} \times \mathcal{X})_{\boldsymbol{x}^{\text{qnt}}} \coloneqq \left\{(\boldsymbol{x}, \boldsymbol{y}) \in \mathcal{X} \times \mathcal{X} : \boldsymbol{x}^{\text{qnt}} = \boldsymbol{y}^{\text{qnt}}\right\}. \tag{5.15}$$

The distance and pseudo-distance functions are said to be parameterized by the quantitative component $\boldsymbol{x}^{\text{qnt}}$, since it influences their behavior without appearing as an explicit argument.

Consequently, the primary and secondary ranking functions $p_{\boldsymbol{x}} : \mathcal{X} \to \mathbb{R}_+$ and $s_{\boldsymbol{x}} : \mathcal{X} \to \mathbb{R}$ are now defined on the domain $\mathcal{X}$ and parameterized at a point $\boldsymbol{x} \in \mathcal{X}$, which can be either feasible or infeasible. In this context, an argument of the ranking functions shares the same quantitative component as the incumbent solution, that is $\boldsymbol{x}^{\text{qnt}}$. The ordering relation $\preceq$ thus compares points of the domain sharing the same quantitative component, making the comparisons effectively on categorical components.

Finally, surrogate-based neighborhoods must be generalized to the mixed-variable setting. They are redefined by considering the ordering relation on the full domain and by choosing that the neighborhoods themselves contain categorical components only. This choice ensures consistency with both the CatMADS framework and the notion of local minima discussed in the next section. The following definition adapts Definition 31 in the more general mixed-variable case.

Definition 32 (Mixed-variable surrogate-based neighborhood)**.** *For a given point $\boldsymbol{x} \in \mathcal{X}$ and integer $m \in \{1, 2, \ldots, |\mathcal{X}^{\text{cat}}|\}$, the* surrogate-based neighborhood $\mathcal{N}(\boldsymbol{x}; m) \subseteq \mathcal{X}^{\text{cat}}$ *is the set containing m categorical components of the lowest ordering with respect to an ordering relation $\preceq$ defined on $\mathcal{X}$. The relation $\boldsymbol{y} \preceq \boldsymbol{z}$ is satisfied whenever*

$$\begin{aligned} &\boldsymbol{y} = (\boldsymbol{y}^{\text{cat}}, \boldsymbol{x}^{\text{qnt}}) \in \mathcal{X} \quad \textit{where} \quad \boldsymbol{y}^{\text{cat}} \in \mathcal{X}^{\text{cat}} \cap \mathcal{N}(\boldsymbol{x}; m) \\ \textit{and} \quad &\boldsymbol{z} = (\boldsymbol{z}^{\text{cat}}, \boldsymbol{x}^{\text{qnt}}) \in \mathcal{X} \quad \textit{where} \quad \boldsymbol{z}^{\text{cat}} \in \mathcal{X}^{\text{cat}} \setminus \mathcal{N}(\boldsymbol{x}; m). \end{aligned}$$

Note that Definition 32 reduces to Definition 31 when restricted to categorical components. Moreover, a consequence of this definition is that for any $\boldsymbol{x} \in \mathcal{X}$ and $m \in \{1, 2, \ldots, |\mathcal{X}^{\text{cat}}|\}$,

$$\mathcal{N}(\boldsymbol{x}; m) = \left\{\boldsymbol{y}^{\text{cat}}_{(1)}, \boldsymbol{y}^{\text{cat}}_{(2)}, \ldots, \boldsymbol{y}^{\text{cat}}_{(m)}\right\} \subseteq \mathcal{X}^{\text{cat}}$$

in which $\boldsymbol{y}^{\text{cat}}_{(1)} = \boldsymbol{x}^{\text{cat}}$ and where $\left(\boldsymbol{y}^{\text{cat}}_{(1)}, \boldsymbol{x}^{\text{qnt}}\right) \preceq \left(\boldsymbol{y}^{\text{cat}}_{(2)}, \boldsymbol{x}^{\text{qnt}}\right) \preceq \ldots \preceq \left(\boldsymbol{y}^{\text{cat}}_{(m)}, \boldsymbol{x}^{\text{qnt}}\right)$. In the next sections, the surrogate-based neighborhoods from Definition 32 are used to handle the categorical variables.

5.5.2 Background on CatMADS

CatMADS is a direct search framework that iteratively seeks points yielding a strict decrease of the objective function and/or the constraints [23]. It generalizes MADS by tackling categorical variables with neighborhoods induced by data from the problem. The quantitative variables are essentially treated as they are in MADS. For simplicity, CatMADS is presented in Algorithm 6 without considering constraints.

Algorithm 6: The CatMADS framework for unconstrained problems (adapted from [23]).

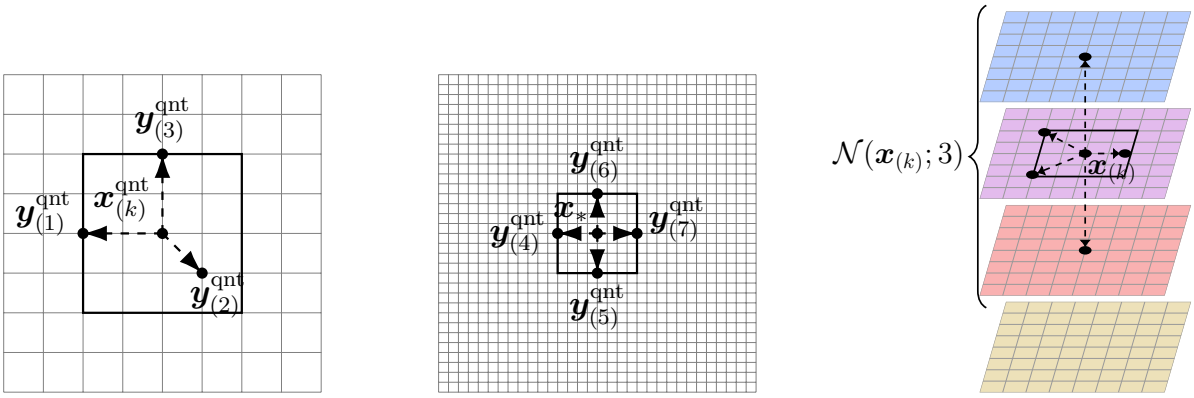
0. **Initialization**. Set $k = 0$, perform an Design of Experiment (DoE) and define an initial mesh
1. **Opportunistic search** (optional).
2. **Opportunistic poll**. Perform polling around the incumbent solution
3. **Opportunistic extended poll** (optional). If Steps 1 & 2 are unsuccessful, perform quantitative polls
   around points in $P_{(k)}^{\text{cat}}$ with objective function values sufficiently close to $f(\boldsymbol{x}_{(k)})$
4. **Update**. Set $k \leftarrow k + 1$, update mesh and check stopping criterion
   If the iteration is successful, increase mesh size and **GO TO** Step 1.
   Else, decrease mesh size, and if the mesh is at its minimum size, then **STOP**

Before any optimization is done, a DoE samples points in the domain using a portion of the budget of evaluations. The DoE is used to gather information on the problem and construct categorical neighborhoods, as well as for determining an initial solution.

The main steps of the CatMADS algorithm are Steps 1, 2, and 3. If a solution with strictly lower objective function value is found during these steps, it becomes the incumbent solution and the iteration is deemed *successful*. Otherwise, the iteration is *unsuccessful*. The *search* is an optional step allowing flexible evaluation of points at the intersection of the mesh and the domain. The poll is the core mechanism of CatMADS. At iteration k, trial points are generated around the incumbent $\boldsymbol{x}_{(k)}$ through two polls, a quantitative one and a categorical one. The quantitative poll fixes the categorical component and performs a standard MADS poll on the quantitative variables. The categorical poll fixes the quantitative component and explores neighboring categorical components selected through a surrogate-based ordering relation. Two consecutive quantitative polls on $\mathbb{R}^2$ are illustrated in Figure 5.6. Figure 5.6c illustrates both poll types.

Arrows emerging from an incumbent solution represent a positive basis. Quantitative components produced are on the *mesh* in gray and within the black square, called the *frame*. The frame delimits the region in which a quantitative poll can produce quantitative components.

In the example in Figure 5.6c, $x_{(k)}^{\text{cat}} = \text{RED}$ and the number of components in the neigh-



(a) Quantitative poll $P^{\text{qnt}}_{(k)}$. (b) Quantitative poll $P^{\text{qnt}}_{(k+1)}$. (c) CatMADS (unconstrained) $P^{\text{cat}}_{(k)}$.

Figure 5.6 (a)–(b) MADS (quantitative) polls at iteration k and $k+1$ where $\boldsymbol{x}_{(k)} = \boldsymbol{x}_{(k+1)}$, and (c) a CatMADS poll.

borhood is three. The neighborhood contains the RED (incumbent), YELLOW, and PURPLE categories. Each colored plane represents the quantitative domain associated with one categorical component. Vertical arrows correspond to the categorical poll, while the quantitative poll is performed within the RED plane.

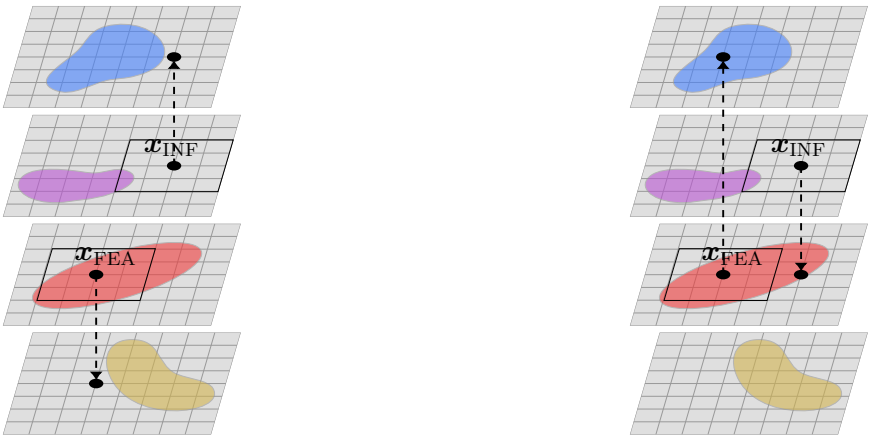
If both the search and poll steps fail to improve the objective, the optional step, called the *extended poll*, may be launched at Step 3. This step revisits points in the categorical poll $P^{\text{cat}}_{(k)}$ that almost improved the incumbent value $f(\boldsymbol{x}_{(k)})$ by calibrating their quantitative variables with additional evaluations.

On an unsuccessful iteration, the mesh and frame are further discretized. CatMADS terminates when the mesh and frame have reached their minimum allowable discretization.

In CatMADS, the constraints are handled with the progressive barrier strategy (PB) [22]. The PB works with two incumbent solutions, each with their own independent poll, as described above. The *feasible poll* is performed around the *feasible incumbent solution* $\boldsymbol{x}_{\text{FEA}}$, and this solution is considered feasible when it strictly improves the objective function. The *infeasible poll* is performed at the *infeasible incumbent solution* $\boldsymbol{x}_{\text{INF}}$, and this solution is forced to become more feasible through the iterations. For more details on the update with the PB, see [26, Chapter 12].

5.5.3 Improvements of CatMADS with surrogate-based neighborhoods

In [23], categorical neighborhoods are constructed using information from the objective function only. Moreover, the implementation in [23] relies on a simple mixed-variable interpolation. The surrogate-based neighborhoods introduced here do not affect the theoretical convergence results, as the proposed method remains an instance of the CatMADS framework. However, the resulting categorical neighborhoods incorporate information from both the objective and the constraint functions and rely on an objective distance induced by kernel-based similarities. As a result, the categorical polls should be guided more effectively toward feasible regions than in the original implementation. Again, this has no impact on the theoretical results, but it is expected to improve empirical performance and accelerate convergence. The improved categorical poll with the surrogate-based neighborhoods is illustrated in Figure 5.7.



(a) Basic categorical poll in [23].

(b) Poll with surrogate-based neighborhoods.

Figure 5.7 The CatMADS framework in the presence of constraints.

In Figure 5.7a, the infeasible regions are in gray and the feasible regions are colored. The feasible categorical poll starts from the red category and selects the yellow category, which is the most similar with respect to the objective function. The generated trial point is infeasible.

In contrast, in Figure 5.7b, the generated trial point lies in the blue category.

A similar behavior is observed for the infeasible categorical poll. In the basic implementation, the generated trial point comes from the category that is closest with respect to the objective function, which again leads to an infeasible trial point in Figure 5.7a. In comparison, the surrogate-based implementation selects a category that is less similar in terms of the objective but considers the similarities of both the objective and the constraint functions. This results

in a feasible trial point in the red category in Figure 5.7b.

5.6 Computational experiments

This section describes the benchmarking of an instance of CatMADS using the novel neighborhoods, as described in Section 5.5. This is a proof of concept demonstrating the utility of surrogate-based neighborhoods in mixed-variable blackbox optimization. Since the surrogate models are GPs with one-hot encoding of the categorical variables, the new resulting method is referred to as $\text{CatMADS}_{\text{GP}}$.

The implementation of $\text{CatMADS}_{\text{GP}}$ is identical to that of CatMADS, except for the construction of the neighborhoods and a BO search step strategically reusing the surrogates. Moreover, in CatMADS, IDW interpolation is constructed following the DoE, whereas $\text{CatMADS}_{\text{GP}}$ also updates the GPs as long as a BO search step is performed. To mitigate computational costs, the BO search step and update of GPs are stopped whenever the number of evaluations exceeds 500 or 33% of the budget of evaluations. The GPs are implemented with the SMT Python library [226]. For a given problem, the number of neighbors is given by $m = \max(3, |\mathcal{X}^{\text{cat}}|^{1/2})$ [23]. This value scales with the total number of categories $|\mathcal{X}^{\text{cat}}|$, and ensures there are at least two different categorical components other than the incumbent one. For the detailed implementation, see [23, Section 4.1].

The benchmarking includes state of the art solvers used in different mixed-variable blackbox optimization applications [142, 200, 242]:

- Pymoo, a Python library containing a collection of optimization algorithms. Specifically, the mixed-variable implementation of a genetic algorithm is used [49];
- Optuna, a mixed-variable solver extending the metaheuristic covariance matrix adaptation evolution strategy (CMA-ES) to categorical variables [8]. The solver is also available as a Python implementation;
- CatMADS, a prototype implementation built on the NOMAD software. It serves as the reference implementation of the original CatMADS framework, from which $\text{CatMADS}_{\text{GP}}$ is derived.

5.6.1 Optimization results for the mechanical-part problem

The mechanical-part design problem from Section 5.2 is now treated as a mixed-variable blackbox optimization problem. To emulate a realistic blackbox setting in which function

evaluations are expensive, a small budget of only 200 evaluations is allowed. All solvers are initialized with the same DoE comprising 40 evaluations, corresponding to 20% of the budget, without any feasible point. As shown in Figure 5.2, an important difficulty of this problem is the identification of a feasible solution using a limited number of evaluations.

Convergence graphs are presented in Figure 5.8. The plot on the left shows the progress of the constraint aggregation function, and the one on the right shows the progress of the best feasible objective function. Feasibility is declared when $h(\boldsymbol{x}_{\text{FEA}}) \leq 10^{-8}$, as represented by the horizontal dashed line. On the left plot, the curves are interrupted as soon as a feasible solution is generated. The plots on the right start at this value.

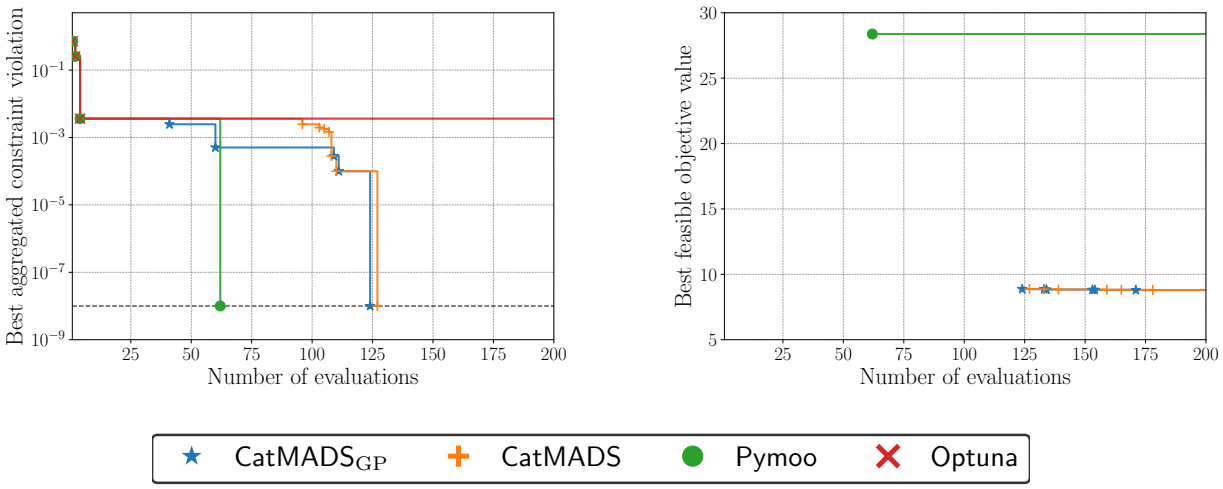


Figure 5.8 Convergence graphs for the mechanical-part design problem with a budget of 200.

Pymoo is the first method to reach feasibility, as shown on the left of Figure 5.8. However, the corresponding objective function value is high and remains unchanged throughout the budget. This indicates that Pymoo is stuck in a suboptimal categorical component. In contrast, CatMADS$_{\text{GP}}$ and CatMADS reach feasibility later, but with lower objective values. Both methods identify significantly better solutions, suggesting that they reach more favorable categorical components. Finally, Optuna fails to produce a feasible point.

The mechanical-part problem was run with multiple seeds. The seed affects both the DoE and the stochastic components of the solvers, such as sampling procedures or the generation of directions. The results lead to similar conclusions as those observed in Figure 5.8. Pymoo reaches feasibility in most runs, and when it does, it is usually the first one. However, in approximately 75% of the runs, the resulting objective function value remains high, as observed in Figure 5.8. CatMADS$_{\text{GP}}$ and CatMADS consistently achieve the best objective values, and CatMADS$_{\text{GP}}$ reaches feasibility faster than CatMADS. Optuna rarely finds a feasible point.

5.6.2 Data profiles with existing solvers

In this section, CatMADS$_{\mathsf{GP}}$ is tested on the 30 unconstrained and 30 constrained mixed-variable problems of the Cat-Suite collection [126]. As in CatMADS [23], the evaluation budget per problem is set to $250n$, where n is the number of variables, and the initial DoE consists of 20% of this budget.

Each problem is instantiated with three different seeds, resulting in a total of 180 instances. An instance, denoted $p \in \mathcal{P}$, corresponds to a problem with a specific seed. The set of all instances is denoted $\mathcal{P}$. The convergence test for an instance $p \in \mathcal{P}$ depends on an initial objective value $f_0 \in \mathbb{R}$. This value is defined as

- the least objective function value in the common DoE, for unconstrained problems [28];
- the smallest objective function value among the first feasible solutions found by the solvers, for constrained problems [28].

This definition enables the comparison of solvers even when the DoE fails to produce a feasible solution.

The set of solvers in the comparison is $\mathcal{S} = \{\text{CatMADS}_{\mathsf{GP}}, \text{CatMADS}, \text{Pymoo}, \text{Optuna}\}$. A solver $s \in \mathcal{S}$ τ-solves an instance $p \in \mathcal{P}$ if it produces a feasible incumbent solution $\boldsymbol{x}_{\text{FEA}} \in \Omega$ such that the reduction $f_0 - f(\boldsymbol{x}_{\text{FEA}})$ is within τ of the smallest reduction $f_0 - f_\star$ obtained by any solver on this instance [179]. More precisely, the τ-convergence test is defined as

$$f_0 - f(\boldsymbol{x}_{\text{FEA}}) \geq (1-\tau)(f_0 - f_\star), \tag{5.16}$$

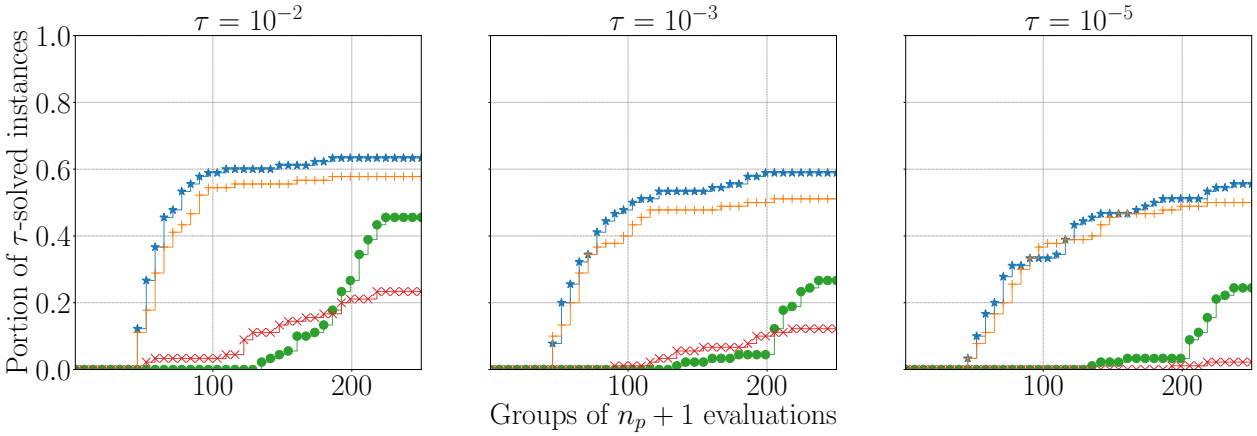
where $\tau \in [0,1]$ is a given tolerance.

A data profile [179] represents the fraction of instances τ-solved by a solver $s \in \mathcal{S}$ as a function of the computational budget. It is defined as

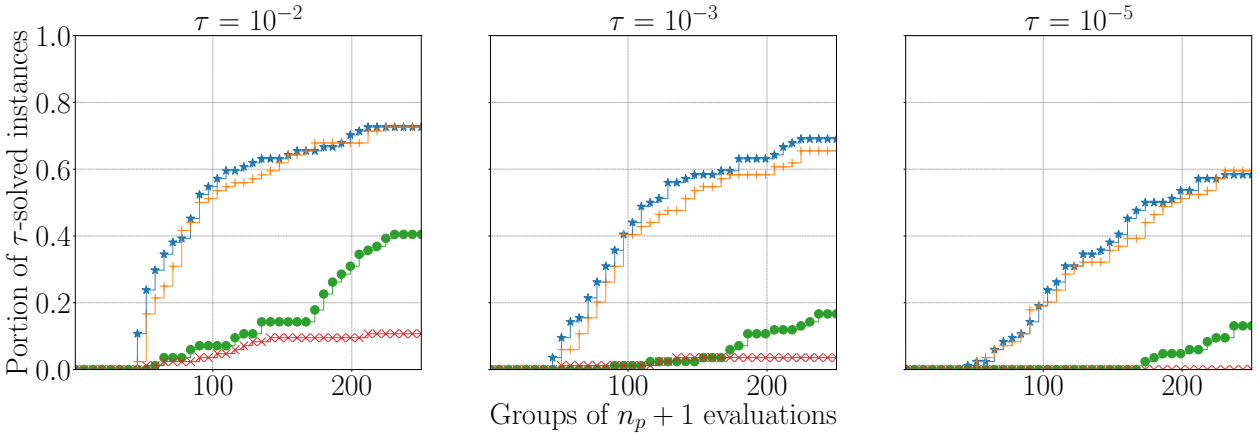
$$\text{data}_s(\kappa) \coloneqq \frac{1}{|\mathcal{P}|} \left| \left\{ p \in \mathcal{P} \ : \ \frac{k_{p,s}}{n_p + 1} \leq \kappa \right\} \right| \in [0,1], \tag{5.17}$$

where $k_{p,s} \geq 0$ is the number of evaluations required by solver s to τ-solve instance p, and n_p is the number of variables in p. In the following plots, κ represents budgets measured in multiples of $(n_p + 1)$ evaluations. The data profiles comparing CatMADS$_{\mathsf{GP}}$ with the solvers from the literature on the unconstrained and constrained instances are presented in Figures 4.7a and 4.7b.

The data profiles for unconstrained problems in Figures 4.7a and 4.7b show that CatMADS$_{\mathsf{GP}}$



(a) Unconstrained test problems.

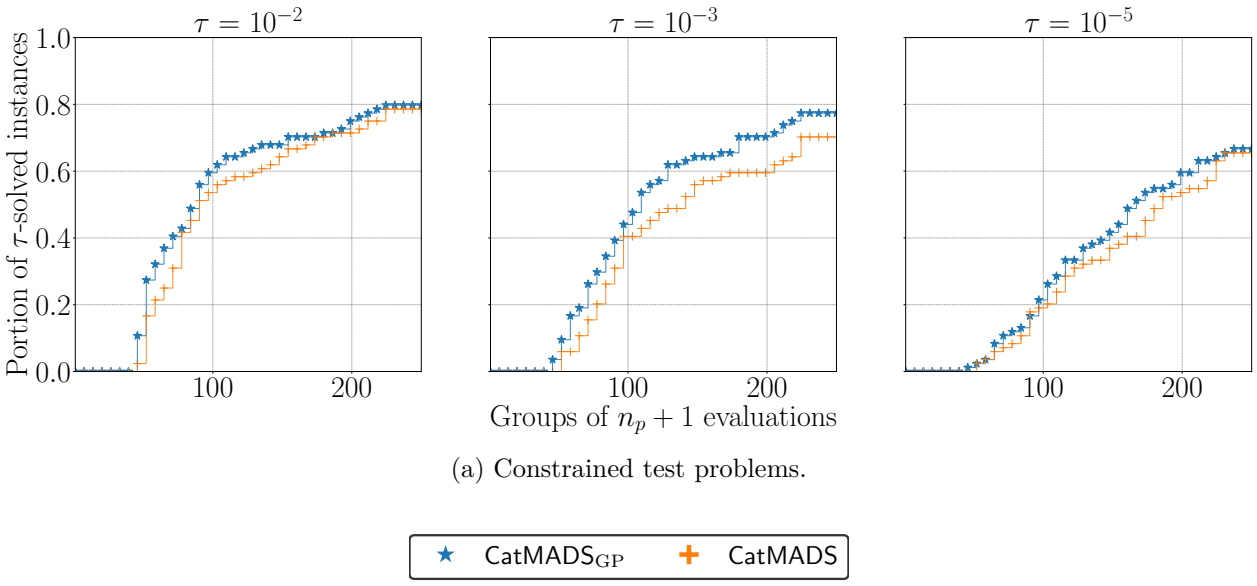


(b) Constrained test problems.

Figure 5.9 Comparison results on the test problems.

performs best across all tolerances, followed by CatMADS. The performances of the three other methods significantly deteriorate as the tolerance τ gets smaller. For instance, at $\tau = 10^{-2}$ and $250(n_p + 1)$ evaluations, CatMADS$_{GP}$ τ-solves approximately 65% of the problems, compared to 60% for CatMADS, 45% for Pymoo and 25% for Optuna.

For the constrained case, the profiles in Figure 5.9b show that both CatMADS variants clearly outperform Pymoo and Optuna over all tolerances. To better analyze the relative performance of the two variants, Figure 5.10a shows the performance profiles constructed using only the two CatMADS variants. CatMADS$_{GP}$ dominates CatMADS for the three tolerances on the constrained problems. The most significant improvement is at $\tau = 10^{-3}$.



(a) Constrained test problems.

Figure 5.10 Comparison results on the constrained test problems between the two CatMADS variants.

5.7 Discussion

This work introduces a novel approach to structure categorical variables through neighborhoods constructed from surrogate models. These neighborhoods incorporate information from both the objective and the constraint functions, making them suitable for constrained mixed-variable optimization. To balance similarity with respect to the objective and constraint functions, the construction relies on dominance relations.

As a proof of concept, the surrogate-based neighborhoods are first illustrated on the mechanical-part design problem. A distinctive feature of this problem is the difficulty of identifying feasible points among the categorical components. The proposed neighborhoods navigate this challenge more effectively than alternative neighborhood strategies.

The surrogate-based neighborhoods are then integrated into the CatMADS algorithm, resulting in the new method CatMADS$_{\text{GP}}$, that is tested against several state-of-the-art solvers. Data profiles indicate that CatMADS$_{\text{GP}}$ consistently achieves better performance on the CatSuite collection. These results suggest that surrogate-based neighborhoods provide a promising mechanism for constrained mixed-variable blackbox optimization.

Further improvements could extend this study. The current implementation of CatMADS$_{\text{GP}}$ uses a fixed number of neighbors throughout the iterations, and more sophisticated strategies could dynamically adjust the number of neighbors. Future extensions would define and

analyze neighborhoods involving meta variables [25], *i.e.*, variables whose values dictate the number of variables and/or or constraints present in the optimization problem.

Data availability statement

Scripts and data are publicly available at https://github.com/bbopt/surrogate_based_neighborhoods.

Conflict of interest statement

The authors state that there are no conflicts of interest.

Acknowledgments

We gratefully acknowledge Dr. Paul Saves for his help with the implementation of SMT and the kernel.

CHAPTER 6 ARTICLE 3: A DISTANCE FOR MIXED-VARIABLE AND HIERARCHICAL DOMAINS WITH META VARIABLES

Authors. Edward Hallé-Hannan, Charles Audet, Youssef Diouane, Sébastien Le Digabel and Paul Saves

And the seven trumpets blowing sweet rock and roll
Gonna blow right down inside your soul
Pythagoras with the looking glass reflects the full moon
In blood, he's writing the lyrics of a brand new tune

— Genesis, *Supper's Ready*

Submitted on August 20, 2024; accepted on August 12, 2025 in *Neurocomputing*
(see doi:10.1016/j.neucom.2025.131208)

Abstract. Heterogeneous datasets emerge in various machine learning and optimization applications that feature different input sources, types or formats. Most models or methods do not natively tackle heterogeneity. Hence, such datasets are often partitioned into smaller and simpler ones, which may limit the generalizability or performance, especially when data is limited. The first main contribution of this work is a modeling framework that generalizes hierarchical, tree-structured, variable-size or conditional search frameworks. The framework models mixed-variable and hierarchical domains in which variables may be continuous, integer, or categorical, with some identified as meta when they influence the structure of the problem. The second main contribution is a novel distance that compares any pair of mixed-variable points that do not share the same variables, allowing to use whole heterogeneous datasets that reside in mixed-variable and hierarchical domains with meta variables. The contributions are verified empirically through regression and classification experiments using simple distance-based models applied to datasets of hyperparameters with corresponding performance scores.

Keywords. Machine learning, optimization, hierarchical, mixed-variable, meta variables

6.1 Introduction

Machine learning tasks or optimization problems with heterogeneous datasets face inherent challenges that arise from several reasons, such as the generation of data from different sources

and the presence of various types of variables. Although recent models, such as large language models, can natively process heterogeneous objects [208], these models require massive data and computational resources. For this reason, simpler and less costly models, that do not inherently handle heterogeneity, are often employed in practice. Heterogeneous datasets are often partitioned into homogeneous data subsets of similar types that are easier to tackle: this approach is undesirable when data is limited or *expensive-to-generate* [11,141,185], since the performance of methods and generalizability of models can be negatively impacted.

This work has two main contributions: a comprehensive modeling framework and a novel distance function between any pair of points in a heterogeneous dataset. These contributions allow to use whole mixed-variable datasets, in which variables vary from one data point to another. The motivation is to allow affordable models and data-driven optimization methods to handle such data heterogeneity, especially when data is limited or expensive. The work opens up the possibility for research on enhancing models like K-nearest neighbors (KNN) or Gaussian processes, and optimization methods for expensive-to-evaluate functions.

6.1.1 Scope of the work

The present work focuses on modeling a heterogeneous domain $\mathcal{X}$ with two fundamental characteristics. First, $\mathcal{X}$ is *mixed-variable*, meaning that a point is composed of finitely many variables from any type amongst categorical (cat), integer (int) or continuous (con). Second, $\mathcal{X}$ is *hierarchical*, signifying that variables have different degrees of importance and influence between each other: this work studies domains and datasets in which two mixed-variable points $x, y \in \mathcal{X}$ do not necessarily share the same variables and/or are not necessarily subject to same bounds. The following example illustrates a domain defining admissible hyperparameters of a Multi-Layer Perceptron (MLP).

HP	Bounds	Type
Learning rate (r)	$]0,1[$	cont
Optimizer (o)	$\{\texttt{ASGD}, \texttt{ADAM}\}$	cat

if $o = \texttt{ASGD}$

HP	Bounds	Type
Update (α)	$]0,1[$	cont
# layers (l)	$\{0,1\}$	int
# units (u_i)	$U_{\texttt{ASGD}}$	int

if $o = \texttt{ADAM}$

HP	Bounds	Type
Average (β)	$]0,1[$	cont
# layers (l)	$\{0,1,2\}$	int
# units (u_i)	$U_{\texttt{ADAM}}$	int

Figure 6.1 Hyperparameters of a MLP involving a hierarchical and mixed-variable domain.

The mixed-variable domain from Figure 6.1 is hierarchical for the following reasons. First, the optimizer variable o controls the bounds on the number of layers and units, and both optimizers `ASGD` and `ADAM` possess their own hyperparameter, α and β, respectively. Second, the number of layers l controls how many units u_i are present in each layer, since $0 < i \leq l$.

The following terminology is used throughout the document. A *point* x refers to a generic element of the domain $\mathcal{X}$, whereas a *data point* $x_{(i)}$, with $i \in \{1, 2, \ldots, N\}$, refers to a specific element of $\mathcal{X}$ from a dataset of N points. For example, $x \in \mathcal{X}$ is a mixed-variable vector containing hyperparameters, and $x_{(1)}$ is a known point where $r = 0.5, o =$ `ADAM`, $\beta = 0.6$ and $l = 0$. The datasets are always related to a domain $\mathcal{X}$. In supervised learning or data-driven optimization, a target function $f : \mathcal{X} \to \mathbb{R}$ is modeled or optimized with a dataset $\{(x_{(i)}, f(x_{(i)}))\}_{i=1}^{N}$ of $N \in \mathbb{N}$ data pairs. In unsupervised learning, the dataset $\{x_{(i)}\}_{i=1}^{M}$ consists of $M \in \mathbb{N}$ data points.

In this work, a hierarchical domain is modeled with a graph structure that is specific to the domain. However, the proposed distance, called the *meta distance*, is not defined for graphs. The following table distinguishes the meta distance from distances defined for graphs. Table 6.1 outlines the main differences between the meta and graph distances. The meta

Table 6.1 Comparison between the meta distance and graph distances.

Meta distance (present work)	Graph distances [44, 204, 219, 268]
Uses a single graph structure to treat a hierarchical domain.	Multiple graphs, each with its own structure and architecture.
Compare points.	Compare graphs with nodes and edges.
Uses standard distances, such as Euclidean or Hamming.	Uses graph operations, such as node or edge insertions, structure comparisons or paths.
Useful for ML or optimization involving mixed-variable points with different variables.	Useful for graph-based ML, such as graph matching, graph similarity or computer vision.

distance is specifically tailored for mixed-variable and hierarchical domains. Graph distances can also be useful for such domains. For instance, [268] proposes a graph distance comparing model architectures via embeddings and representations, which are learned from graph neural networks. Architectures are represented as nodes, and they are connected by edges if they are similar.

The present paper does not only compare architectures or structures representing hierarchies, but also compares variables of different types taking values or categories, as in Figure 6.1. This study focuses on variables, and graph distances are outside the scope.

Data fusion techniques integrating multiple data subsets into a single heterogeneous dataset [80,

269] is related, but also considered outside the scope. The modeling framework developed in this manuscript implicitly performs data fusion on data subsets that do not share the same variables.

This research is motivated by real-life applications. In deep learning, hyperparameter optimization seeks an optimal configuration of a neural network with respect to its hyperparameters [104, 159, 264, 266]. This problem is typically mixed-variable and hierarchical. In [166], the optimization of a magnetic resonance device includes a variable that determines the number of magnets, and each additional magnet involves new design variables. In [63], an aircraft engine is designed through a heterogeneous dataset, in which a part of the data includes a fan and another part does not. More applications from various fields are also covered, including software architecture design [10], statistical medical research [133], drug discovery in heterogeneous datasets [260] and, multiple vehicle routing problems [132].

6.1.2 Objectives, contributions and organization of the work

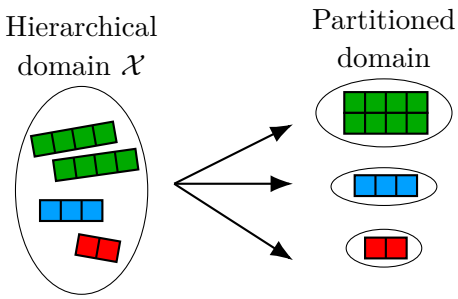
The overall objective is to develop an interpretable and constant-time distance function for mixed-variable and hierarchical domains in which two points do not necessarily share the same dimension, bounds or variables. Similar distances exist in the literature, but have at least one of the following issues: reliance on randomness (non-interpretable), lacking properties of a metric, such as the identity of indiscernibles, or complexity depending on datasets or solving an optimization subproblem.

The research gap addressed is the lack of mixed-variable distance functions defined on hierarchical domains without these issues. The motivation is to use whole datasets with mixed-variable and hierarchical domains, in order to improve generalization of models and to provide data-efficient optimization in the context of poor data availability.

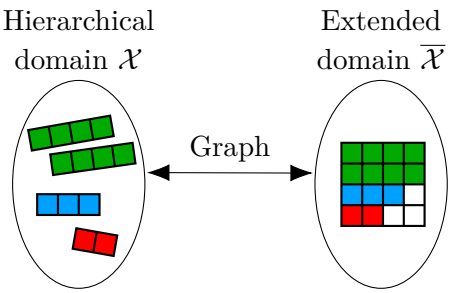
To achieve the overall objective, two steps, representing the main contributions, are distinguished. The first formalizes a modeling framework that thoroughly models a hierarchical domain with a graph structure specific to the problem. This graph encompasses the information related to the hierarchy of variables and the hierarchical interrelationships between variables. For example, in Figure 6.1, the optimizer has the highest hierarchy since it influences bounds and inclusion of variables, but it is not influenced by other variables. In the literature, variants of hierarchical domains are referred to as tree-structured [45], variable-size [196] as well as conditional search space or neural architecture search [143]. The modeling framework developed in this study generalizes and unifies all these variants, as well as the framework in [135], which introduced the term hierarchical in the context of this work.

The second step constructs a distance function based on the proposed modeling framework. The distance is defined on the *extended domain* $\overline{\mathcal{X}}$, rather than directly on its corresponding domain $\mathcal{X}$. The extended domain is an extension of the domain that involves all included and excluded variables. Excluded variables are those that are excluded for a given point $x \in \mathcal{X}$, but present in another point $y \in \mathcal{X}$. For example, in Figure 6.1, the number of units in the second layer is excluded, when there is only one layer. Excluded variables are considered by the distance, since they simultaneously provide valuable information and facilitate the comparison of two points that do not share the same variables.

The motivation and the objectives are illustrated in Figure 6.2. The colors represent datasets. Each group of four green blocks corresponds to a dataset, where each data point has four variables. The blue and red blocks correspond to datasets with three and two variables, respectively.



(a) Motivation: avoid partitioning a hierarchical domain and splitting heteregeneous data.



(b) Objectives: define a distance on the extended domain, in which data is aggregated.

Figure 6.2 Global overview of the work.

Figure 6.2a illustrates a commonly used approach in which the heteregenous dataset is split into simpler homogeneous datasets, each containing data points with the same variables. Figure 6.2b schematizes the approach employed in this work. The extended domain $\overline{\mathcal{X}}$ aggregates datasets, even if they have different variables. The relations between the variables and the meta distance are established in the extended domain. Finally, a graph structure establishes a direct correspondence between the two domains $\mathcal{X}$ and $\overline{\mathcal{X}}$.

The rest of the document is organized as follows. A literature review is detailed in Section 6.2. Next, the extended point $\overline{x}$ and the extended domain $\overline{\mathcal{X}}$ are thoroughly defined in Section 6.3. Afterwards, in Section 6.4, the meta distance is defined on the extended domain $\overline{\mathcal{X}}$, which induces a distance on the domain $\mathcal{X}$. Finally, computational experiments are carried out in Section 6.5.

6.2 Literature review

This section presents the relevant literature in three parts. Section 6.2.1 details background work that this study relies on. Afterward, Section 6.2.2 discusses related work within the scope of this work. Finally, background and related work are compared with this study in Section 6.2.3.

6.2.1 Background work

In this work, the particularity that two points in a hierarchical domain $\mathcal{X}$ do not share the same variables, dimension or bounds is a consequence of the so-called *meta* variables introduced in the previous work [25]. These meta variables determine if other variable(s), called *decreed*, are excluded or included in a point of the domain $\mathcal{X}$, and control their bounds. In addition to their type, each variable is assigned a role, such as meta or decreed, that reflects either how it influences the dimension, structure or bounds of the domain $\mathcal{X}$, or how it is subject to the influence of other variables. Variables that neither influence nor are influenced by other variables are assigned the *neutral* role, and they are always included in a point. The framework in [25], tackling only a single level of hierarchy, is generalized in the present paper. Notably, roles are generalized and a graph structure is introduced to tackle an arbitrary (finite) level of hierarchy.

An important reference for this work is the technical report [135], which proposes a mixed-variable kernel function for hierarchical spaces, each paired with a directed acyclic graph, where the nodes are the variables. Variables with child nodes are required to be categorical. The kernel is constructed from one-dimensional kernels for which pseudodistances take into account whether the variables are included or excluded. The inclusion of a variable is determined by a Kronecker delta function that takes the values of its ancestors as arguments.

6.2.2 Related work

Most of the literature on distance or similarity measures for heterogeneous datasets treats the simpler case where heterogeneity originates solely from the variety of variable types. In classification, variants of the K-nearest neighbors, based on distances (or similarity measures) built with combinations of continuous, integer or categorical distances, are commonly studied [7, 12, 197]. Decision trees or random forests are also used for classification [238], and even regression [153]. In regression, many kernel functions (similarity measures) have been recently developed for constructing Gaussian processes (GPs) [206] over heterogeneous datasets with mixed-variables. Kernel methods are well adapted for mixed-variable problems, since

mixed kernels can be directly constructed with products or additions of well-documented continuous [206], integer [112] or categorical kernels [194, 201, 224, 270].

In [111], a novel similarity measure, called the Earth mover's intersection (EMI), compares sets of different sizes with an Earth mover's distance (EMD). EMD measures the minimum cost to transform one distribution into another by solving an optimal transport problem. Such distances are also known as *Wasserstein* distances [240].

In [196], GPs are constructed on said variable-design spaces, which contain dimensional variables [167] that are essentially discrete meta variables controlling the inclusion or exclusion of other variables. The variable-wise decomposition kernel handles the hierarchical domain by comparing only the shared variables between two points based on their respective dimensional variables.

In [43], feature models manage and capture heterogeneity across data points through tree-structured models consisting of features nodes and relationships arcs, that represent parent-child dependencies or integrity constraints [17]. Recent advances in features models addressed complex dependencies and constraints in large-scale heterogeneous datasets [41] with semantic logic.

6.2.3 Comparison of background and related work

The following table compares approaches and frameworks, from background and related work, with similar objectives to the current work. The first column indicates whether the approach or framework tackles mixed-variable domains. The second one shows if variables that are not shared between points are considered in distance computations. The third column verifies if the distance is deterministic, *i.e.*, interpretable. The last two columns show how the number of parameters and the cost of the distance scales with the number of variables n or the number of data points N. The other columns are self-explanatory.

In the table, "Subproblems" refers to the approach that simply partitions hierarchical domains, as in Figure 6.2a. This approach addresses mixed-variable domains, but does not handle hierarchical domains.

The meta framework from previous work [25] does not provides a distance and only tackles a single layer of hierarchy. This framework is proposed specifically for unifying specific optimization approaches in context where derivatives are not available.

The variable-size framework proposes a mixed-variable kernel that considers only shared variables between points and supports only a single layer of hierarchy.

The EMI is defined over sets or distributions of varying sizes, inherently tackling arbi-

Table 6.2 Comparison of frameworks based on key features, where ✓signifies true, $\sim$ means partially true, ✗ represents false, n is the number of variables and N is the number of points in a dataset.

Approach/ framework	Mixed-var.	Comparisons different var.	Deterministic/ interpretable	Layers of hierarchy	Type of proximity	# of param.	Cost of $d(x, y)$
Subproblems	✓	✗	✓	0	Distance	0	$\mathcal{O}(1)$
Meta [25]	✓	✗	✗	1	None	✗	✗
Variable-size [196]	✓	✗	✓	1	Kernel	$\mathcal{O}(n)$	$\mathcal{O}(1)$
Hierarchical [135]	$\sim$	✓	$\sim$	m	Kernel	$\mathcal{O}(n)$	$\mathcal{O}(1)$
EMI [111]	✓	✓	✓	m	Distance	$\mathcal{O}(N^2)$	$\mathcal{O}(N^3 \log N)$
Current work	✓	✓	✓	m	Distance	$\mathcal{O}(n)$	$\mathcal{O}(1)$

trary level of hierarchy noted as $m \in \mathbb{N}$ [111]. The drawback is the computational cost of $d(x, y)$, which involves solving an optimal transport problem with a worst-case complexity of $\mathcal{O}(N^3 \log N)$. This is impractical for machine learning or optimization with distance-based models.

The hierarchical framework models multiple layers of hierarchy. The variables of higher hierarchy influencing other variables are restricted to the categorical type. The kernel function relies on pseudo-distances without the identity of indiscernibles, *i.e.*, $d(x, y) = 0 \not\Rightarrow x = y$. This is problematic, particularly in optimization, as distinct points with a distance of zero can lead to convergence issues or unreliable exploration. In supervised learning, this may cause different inputs to be treated as identical, resulting in poorer generalization and potential numerical instability.

The framework of the current work fully supports mixed-variable domains, considers all variables between two points, even if not shared, and handles an arbitrary level of hierarchies. It generalizes the previous work [25] in multiple aspects, as presented in Table 6.2, and satisfies the overall objective introduced in Section 6.1.2.

6.3 Modeling framework for hierarchical domains based on meta variables

In this section, hierarchical domains that generate heterogeneous datasets are formalized. In Section 6.3.1, the roles of variables are explicitly introduced. Then, excluded variables and extended point, containing all variables whether they are excluded or not, are defined in Section 6.3.2. In Section 6.3.3, notions of graph theory are adapted for this work. Afterwards, the restricted sets, in which variables of the extended point belong, are detailed in Section 6.3.4. Subsequently, the extended domain $\overline{\mathcal{X}}$ is introduced in Section 6.3.5.

6.3.1 Roles of variables

The roles of variables are established from the the decree property that is generalized from [25]. The following definition allows a variable to simultaneously have the decree property, and have its inclusion or admissible values determined by a decree dependency (by other variables with the decree property): this is not allowed in [25].

Definition 33 (Decree property and decree dependency)**.** *The decree property is attributed to variables whose values determine if other variables are included or excluded from a point $x \in \mathcal{X}$, or whose values determine the admissible values (or bounds) of other variables.*

A decree dependency refers to the inclusion or admissible values dependency of a variable with respect to an another variable with the decree property. Variables can have multiple decree dependencies with different variables.

In Section 6.3.3, decree dependencies are viewed as parent-children dependencies, where the values of a parent variable determines the inclusion or admissible values of its children variables. In Figure 6.1, the optimizer o has the decree property, since it determines the inclusion of, among others, the update α, which the update α is included when $o =$ `ASGD` and excluded otherwise, hence it has has a decree dependency with the optimizer o (parent).

Definition 33 on the decree property and decree dependency establishes four possible cases, which are formalized as the roles of variables in the following definition.

Definition 34 (Roles of variables)**.** *The role of a variable represents its relation to the decree property. A variable is assigned one of the following roles:*

1. *meta* (m)*, if it has the decree property, and has no decree dependency;*
2. *meta-decreed* (md)*, if it has the decree property, and has at least one decree dependency;*
3. *decreed* (dec)*, if it does not have the decree property, but has at least one decree dependency;*
4. *neutral* (neu)*, if it does not have the decree property nor decree dependency.*

Recall that the role of a variable must not be confused with its variable type. Each variable has its own variable type and is assigned its own role. Figure 6.3 further details the MLP example by adding roles and schematizing decree dependencies.

Figure 6.3b illustrates the decree dependencies between hyperparameters of the MLP example, using solid lines for inclusion dependencies and dotted lines for admissible values

HP	Bounds	Type	Role
Learning rate (r)	$]0,1[$	Cont	Neutral
Optimizer (o)	$\{\texttt{ASGD}, \texttt{ADAM}\}$	Cat	Meta

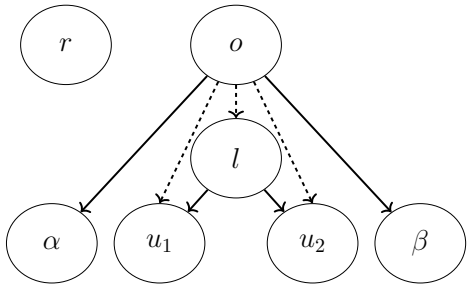
if $o = \texttt{ASGD}$

HP	Bounds	Type	Role
Update (α)	$]0,1[$	Cont	Decreed
# layers (l)	$\{0,1\}$	Int	Meta-dec.
# units (u_i)	$U_{\texttt{ASGD}}$	Int	Decreed

if $o = \texttt{ADAM}$

HP	Bounds	Type	Role
Average (β)	$]0,1[$	Cont	Decreed
# layers (l)	$\{0,1,2\}$	Int	Meta-dec.
# units (u_i)	$U_{\texttt{ADAM}}$	Int	Decreed

(a) Types and roles of the hyperparameters.



(b) Decree dependencies as arcs, solid for inclusion and dotted for values.

Figure 6.3 Roles of the variables and decree dependencies in the MLP example.

dependencies. In the MLP example, the optimizer $o \in \{\texttt{ADAM}, \texttt{ASGD}\}$ is a categorical variable that is assigned the meta role, since it determines the inclusion and the admissible values of other variables (decree propriety), and neither its inclusion nor its admissible values are determined by other variables (no decree dependency). For convenience, a variable and its role are referred similarly as a variable and its type, *e.g.*, the optimizer is referred as a meta categorical variable. The number of layers $l \in L_o$ is a meta-decreed integer variable, since its admissible values are determined by the optimizer o (decree dependency), and it determines the inclusion of the number of units $u_1, u_2 \in U_o$, where

$$L_o \coloneqq \begin{cases} \{0,1\} & \text{if } o = \texttt{ASGD}, \\ \{0,1,2\} & \text{if } o = \texttt{ADAM}, \end{cases} \quad \text{and} \quad U_o \coloneqq \begin{cases} U_{\texttt{ASGD}} & \text{if } o = \texttt{ASGD}, \\ U_{\texttt{ADAM}} & \text{if } o = \texttt{ADAM}. \end{cases}$$

The update α is a decreed continuous variable, since it does not have the decree property, and its inclusion is determined by the optimizer o. Finally, the learning rate $r \in]0,1[$ is a neutral categorical variable, as it does not have the decree property, and it has no decree dependency.

6.3.2 Excluded variables and extended point

In previous work [25], a point contains only variables that are included for the given values of variables with the decree property. In this work, variables that are excluded are also

considered, since 1) it provides useful information for computing distances between two points of the domain $\mathcal{X}$ that do not share the same variables, and 2) it facilitates the computations themselves. This last remark leads to the following definition.

Definition 35 (Excluded variable)**.** *An excluded variable is a meta-decreed or decreed variable, that, for the given values of the variables associated to its decree dependencies, is not included in the given point $x \in \mathcal{X}$, but is included in at least one other point $y \in \mathcal{X}$. An excluded variable is assigned the special value* `EXC` *and its variable type is conserved.*

In the MLP example, the update α is a decreed continuous variable, which is excluded when the optimizer $o =$ `ADAM`, whereas $\alpha \in]0, 1[$ (included) when $o =$ `ASGD`. Definition 35 is introduced to allow the meta distance to consider every variable that is included in at least one point of the domain, collectively referred as all the included and excluded variables. The definition of an extended point formalizes the previous sentence.

Definition 36 (An extended point)**.** *An extended point $\overline{x}$ contains all the included and excluded variables of a corresponding point $x \in \mathcal{X}$. For $r \in R := \{\mathrm{m}, \mathrm{md}, \mathrm{dec}, \mathrm{neu}\}$ and $i \in I^r := \{1, 2, \ldots, n^r\}$, the i-th variable assigned to the role r is noted $\overline{x}_i^r$, where $n^r \in \mathbb{N}$ is the number of variables assigned to the role r.*

The bar notation of an extended point $\overline{x}$ outlines that a corresponding point x is conceptually extended to incorporate excluded variables. This notation is inspired by the extended set of real numbers $\overline{\mathbb{R}} = \mathbb{R} \cup \pm\{\infty\}$. A point $x \in \mathcal{X}$ contains only included variables, but its variables can be attributed roles, similarly as an extended point $\overline{x}$.

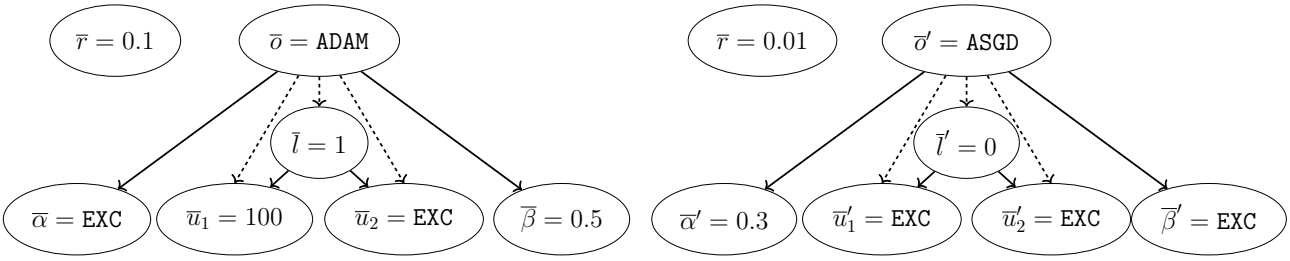
Second, meta and neutral variables are always included, hence there is no distinction between these variables whether they are part of an extended point $\overline{x}$ or of a point x, *i.e.*, $\overline{x}_i^r = x_i^r$ for $r \in \{\mathrm{m}, \mathrm{neu}\}$ and $i \in I^r$. The admissible values of meta and neutral variables are also fixed. In contrast, meta-decreed $\overline{x}_i^{\mathrm{md}}$ and decreed $\overline{x}_j^{\mathrm{dec}}$ variables of an extended point $\overline{x}$ may be excluded from a point x, and/or their admissible values may differ between points.

Third, the framework proposed in [25] is a special case of the one developed in this work, as it includes meta variables but lacks meta-decreed variables. The special case can be recovered easily by removing the meta-decreed variables: this is convenient as the special case provides a specialized framework for problems of great interests, such as most hyperparameter optimization problems.

Fourth, meta-decreed and decreed variables can be seen as bounds-dependent variables and, in that case, the bounds of an excluded variable are restricted to the empty set.

Fifth, both the roles and types of a variables carry important information for properly computing distances between variables in Section 6.4. To avoid a cumbersome notation, variable types are not explicit, but they are implicitly considered in the computation of distances in Section 6.4.

The following figures schematize two extended points of the MLP example. Each figure builds upon Figure 6.3b by assigning values to the variables.



(a) Extended point $\overline{x}$ with ADAM and one layer.

(b) Extended point $\overline{x}'$ with ASGD and no layer.

Figure 6.4 Two different extended points in the MLP example.

In Figure 6.4, the two extended points contain all included and excluded variables of the domain, facilitating variable comparison. For instance, the update $\overline{\alpha}$ remains comparable, while being excluded in the extended point $\overline{x}$ on the left. Both extended points share the same arc representations. In the next section, elements of graph theory formalize a common graph structure shared by all extended points of a problem.

6.3.3 Notions from graph theory

At this stage of the work, Definition 33 allows meta-decreed variables to impact each others through decree dependencies. Consequently, decree dependencies between meta-decreed variables can lead to circular reasoning or contradiction. Indeed, this can be shown with a simple example with only two meta-decreed binary variables:

$$x_1^{\text{md}} = \begin{cases} 0 & \text{if } x_2^{\text{md}} = 1, \\ 1 & \text{if } x_2^{\text{md}} = 0, \end{cases} \qquad x_2^{\text{md}} = \begin{cases} 0 & \text{if } x_1^{\text{md}} = 0, \\ 1 & \text{if } x_1^{\text{md}} = 1, \end{cases}$$

where $x_1^{\text{md}} = 0 \Rightarrow x_2^{\text{md}} = 0 \Rightarrow x_1^{\text{md}} = 1 \neq 0$ (contradiction). To avoid these problematic, and uncommon situations, an assumption about the decree dependencies must be introduced. Beforehand, the role graph, which among other things allows to formulate the assumption, is defined below.

Definition 37 (Role graph)**.** *The role graph $G = (V, A)$ is a graph structure, where*

- *V is the set of variables that contains all the included and excluded variables, represented as nodes,*
- *A is the set of decree dependencies that contains references for all inclusion-exclusion and admissible values dependencies between all the included and excluded variables, represented as arcs.*

An arc $a \in A$, which refers to a decree dependency, connects a parent (variable) to a child (variable), whose inclusion or admissible values are influenced by the parent. A parent is either a meta or meta-decreed $\overline{x}_i^r$, and a child is either meta-decreed or decreed, such that $a = (\overline{x}_i^r, \overline{x}_j^{r'})$, where $r \in \{\mathrm{m}, \mathrm{md}\}$ and $r' \in \{\mathrm{md}, \mathrm{dec}\}$, with $i \neq j$ when $r = r' = \mathrm{md}$. A child can have multiple parents, and vice versa.

In the MLP example, the set of nodes correspond to all included and excluded variables, *i.e.*, $V = \left\{\overline{o}, \overline{l}, \overline{\alpha}, \overline{\beta}, \overline{u}_1, \overline{u}_2, \overline{r}\right\}$. The arcs of the graph represent pairs of parent-child variables, in which a child has a decree dependency with its parent. From Figure 6.4, the set of decree dependencies in the example is $A = \left\{\left(\overline{o}, \overline{l}\right), \left(\overline{o}, \overline{\alpha}\right), \left(\overline{o}, \overline{\beta}\right), \left(\overline{o}, \overline{u}_1\right), \left(\overline{o}, \overline{u}_2\right), \left(\overline{l}, \overline{u}_1\right), \left(\overline{l}, \overline{u}_2\right)\right\}$. The influence of parents on the inclusion and/or admissible values of their children is formalized with these arcs in the next section.

Now that the role graph G is properly defined, the assumption discarding situations with circular reasoning or contradiction is introduced.

Assumption 1. *The role graph G is a Directed Acyclic Graph (DAG).*

Assumption 1 ensures that the nodes in the role graph G are (partially) ordered. Circular decree dependencies, as presented in the example with two meta-decreed variables, are forbidden. To determine such a partial order, it suffices to apply a topological sort on G.

Under Assumption 1, the role graph G is a data structure that: 1) contains all the included and excluded variables in the set of variables V, 2) contains references for all inclusion-exclusion or admissible values dependencies in the set of decree dependencies A, and 3) establishes the roles of variables according to the positions of nodes in the DAG:

- a meta variable $\overline{x}_i^{\mathrm{m}}$ is a root node;
- a meta-decreed variable $\overline{x}_i^{\mathrm{md}}$ is an internal node with at least one parent and one child;
- a decreed variable $\overline{x}_i^{\mathrm{dec}}$ is a leaf node with at least one parent and no child;

- a neutral variable $\overline{x}_i^{\text{neu}}$ is an isolated node.

Figure 6.5 schematizes the positions of variables based on their role. In this simple example,

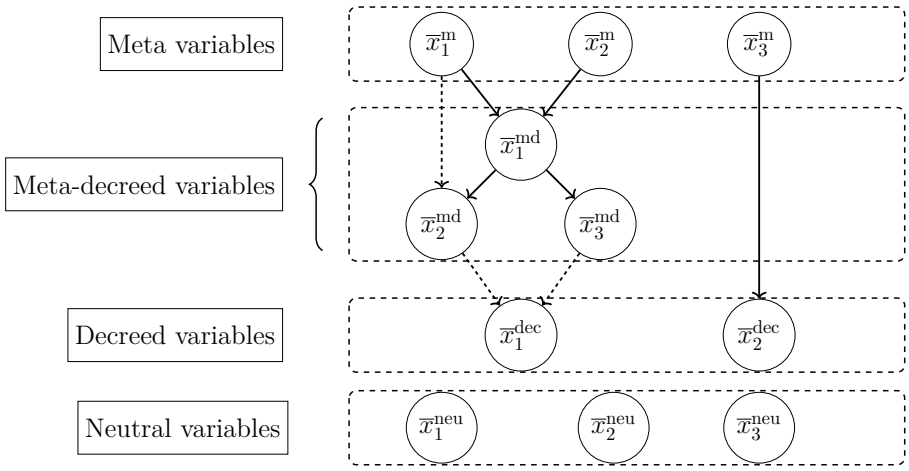


Figure 6.5 Example of a role graph showing positions of variables based on their role.

there are three levels of hierarchy, as the longest path contains three edges. The graph is acyclic, ensuring no circular dependencies are present. The example contains two levels of meta-decreed variables. In general, there can be an arbitrary number of levels. Disconnected trees can exist, meaning not all variables must be part of a single connected structure. Finally, decree dependencies (arcs) do not need to follow a strict layer-by-layer structure. For instance, x_1^{m} has arcs connecting to both x_1^{md} and x_2^{md}, which are on different levels.

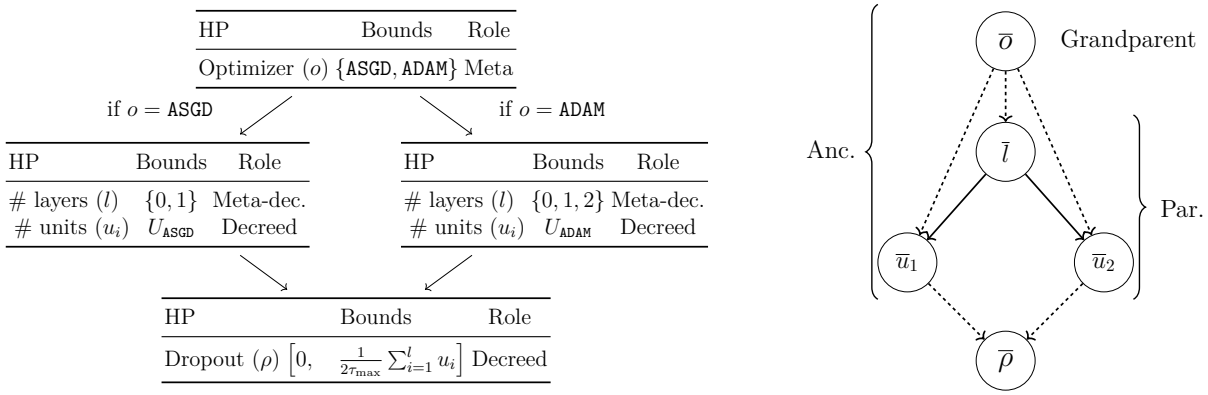
Recall that meta-decreed and decreed variables are child variables whose inclusions or admissible values are determined by the values of their parents composing their decree dependencies. The role graph G outlines that the inclusion and/or admissible values of a variable can be determined by multiple different decree dependencies, that is, from multiple parents. Hence, to determine the inclusion and/or the admissible values of a variable, it is necessary to consider simultaneously all the values of its parents. In the MLP example, the number of units $\overline{u}_1$ must consider simultaneously the values of its parents the optimizer $\overline{o}$, for its admissible values, and the number of layers $\overline{l}$, for its inclusion. The following definition introduces formally the notion of the parents in our context.

Definition 38 (Parents)**.** *For $r \in R$ and $i \in I^r$, the parents $\overline{\text{par}}_i^r$ of the variable $\overline{x}_i^r$ is the subset of variables for which there exists an arc from those variables to $\overline{x}_i^r$, i.e.,*

$$\overline{\text{par}}_i^r := \{\overline{v} \in V \ : \ (\overline{v}, \overline{x}_i^r) \in A\}\,. \tag{6.1}$$

The parents are used to handle the inclusion-exclusion or the admissible values of meta-decreed and decreed variables in the next section. Note that although meta and neutral variables have no parents, they are still defined (as empty sets) for these roles. This is allows to propose a concise expression for the extended domain $\mathcal{X}$ in Section 6.3.5.

For a given variable, the inclusion or admissible values of its parents can be determined by their own parents (*i.e.*, grandparents). In next section, the ancestors of a variable, representing possible multiple generations of parents and grandparents, will be used to determine all the values that such variable can take across all possible extended points. The next example, illustrated in Figure 6.6, removes some hyperparameters and adds the dropout ρ to the MLP example. This variant justifies the need to consider ancestors when determining all the possible values.



(a) Bounds and role for the variant.

(b) Dropout and its ancestors.

Figure 6.6 A variant of the MLP example with the dropout.

In the MLP variant, the admissible values (the bounds) of the dropout ρ are determined by the values of its parents, that is, the number of layers l and the number of units u_i with $0 < i \leq l$. The constant $\tau_{\max} \coloneqq \max\left\{\sum_{i=1}^{l} u_i \ : \ o \in \{\texttt{ASGD}, \texttt{ADAM}\},\ l \in L_o,\ u_i \in U_o \text{ for } 1 \leq i \leq l\right\}$ implies that the optimizer o (grandparent) must also be considered to determine all the possible values ρ can take. However, note that the bounds of the dropout ρ do not have an explicit dependency with the values of the optimizer o since, for a given MLP problem, the constant $\tau_{\max}$ is fixed.

The ancestors can be defined recursively by starting at the parents, then passing by the parents of the parents, and so on, until the roots are reached.

Definition 39 (Ancestors). *For $r \in R$ and $i \in I^r$, the ancestors of the variable $\overline{x}_i^r$, noted $\overline{\text{anc}}_i^r$, is the subset of variables that are either parents or recursively ancestors of parents of*

$\overline{x}_i^r$, *i.e.,*

$$\overline{\text{anc}}_i^r := \overline{\text{par}}_i^r \cup \left(\bigcup_{\overline{x}_j^{r'} \in \overline{\text{par}}_i^r} \overline{\text{anc}}_j^{r'} \right) \tag{6.2}$$

where $\overline{\text{anc}}_j^{r'}$ *denotes the ancestors of the parent variable* $\overline{x}_j^{r'}$*. The recursion in* (6.2) *stops at the root nodes, i.e., with* $\overline{\text{par}}_j^{\text{m}} = \emptyset, \forall j \in I^{\text{m}}$.

Now that several notions of graph theory have been adapted to this work, the definition of a hierarchical domain is formally established.

Definition 40 (Hierarchical domain)**.** *A hierarchical domain is a domain with at least one variable with the decree property, i.e., at least one meta variable.*

Definition 40 is one specific approach to formalize a hierarchical domain, yet there exist several equivalent statements.

Theorem 8 (Hierarchical domain equivalences)**.** *Let* $\mathcal{X} \neq \emptyset$*, with* $G = (V, A)$ *as its corresponding role graph. Then the following statements are equivalent:*

1. $\mathcal{X}$ *is a hierarchical domain.*
2. *The set of decree dependencies is non-empty, i.e.* $A \neq \emptyset$.
3. *There is a least one point containing a variable with a least one parent.*

Proof. The theorem is a direct consequence of Definitions 37 and 39. □

Theorem 8 emphasizes that the different bounds or inclusion-exclusion of variables within a hierarchical domain $\mathcal{X}$ is a consequence of interrelationships between variables, that is, its decree dependencies (Definition 33). Note that a variable with missing entries can be modeled as a decreed variable whose inclusion is determined by an additional binary meta variable.

6.3.4 Universal sets and restricted sets

As mentioned in Section 6.3.3, meta-decreed and decreed variables are subjected to the values of their parents, since they must respect their decree dependencies for the given values of their parents. In this section, the dependencies of a variable with respect to its parents are modeled through its restricted set, that is obtained by conditioning its universal set with the values of its parents. The universal set is defined next.

Definition 41 (Universal set)**.** *For $r \in R$ and $i \in I^r$, the universal set $\overline{\mathcal{X}}_i^r$ of the variable $\overline{x}_i^r$ is the set that contains all possible values that the variable can take by considering all possible values assigned to its ancestors $\overline{\text{anc}}_i^r$.*

In the MLP example, the universal set of the number of units $\overline{u}_1$ is $\overline{U} = U_{\texttt{ASGD}} \cup U_{\texttt{ADAM}} \cup \{\texttt{EXC}\}$. The number of units $\overline{u}_1$ can be excluded, depending on the number of layers $\overline{l}$, hence its universal set must contain the special value `EXC`.

As discussed in the previous section, the values of the ancestors must be considered, in addition to those of the parents, to determine a universal set. In the MLP example, the universal set of the dropout ρ reduces to $\overline{P} = [0, 0.5]$, *i.e.*, it is obtained when $\sum_{i=1}^{l} u_i = \tau_{\max}$. The bounds of the dropout $\left[0, \frac{1}{2\tau_{\max}} \sum_{i=1}^{l} u_i\right]$ depends on its parents the number of layers l and the number of units u_i, where $0 < i \leq l$. However, to determine its universal set $\overline{P}$, the constant $\tau_{\max} = \max\left\{\sum_{i=1}^{l} u_i \ : \ o \in \{\texttt{ASGD}, \texttt{ADAM}\},\ l \in L_o,\ u_i \in U_o \text{ for } 1 \leq i \leq l\right\}$ must be determined by considering the optimizer o (grandparent). Again, τ is a constant, hence the dropout ρ has no decree dependency with the optimizer o.

The next step is to develop the restricted set of a variable. The restricted set of the variable $\overline{x}_i^r$ is the subset of the universal set $\overline{\mathcal{X}}_i^r$ such that $\overline{x}_i^r$ respects the decree dependencies for the given values of its parents $\overline{\text{par}}_i^r$, which are identified by arcs of the role graph G. The formal definition of the restricted set is presented below.

Definition 42 (Restricted set)**.** *For $r \in R$ and $i \in I^r$, and a given role graph $G = (V, A)$, the restricted set of the variable $\overline{x}_i^r$ is the universal set $\overline{\mathcal{X}}_i^r$ conditioned by the values of its parents $\overline{\text{par}}_i^r$, expressed as*

$$\overline{\mathcal{X}}_i^r/_{\overline{\text{par}}_i^r} := \left\{\overline{x}_i^r \in \overline{\mathcal{X}}_i^r \ : \ \forall\ (\overline{v}, \overline{x}_i^r) \in A,\ \overline{x}_i^r \textit{ respects the decree dependencies for the values of } \overline{v}\right\}.$$

The notation $/_{\overline{\text{par}}_i^r}$ for a restricted set $\overline{\mathcal{X}}_i^r/_{\overline{\text{par}}_i^r}$ means "*given*" the values of its parents. The parents are placed as subscripts to outline the dependency. Meta and neutral variables are always included and have no parent, hence their restricted set is simply their universal set: $\overline{\mathcal{X}}_i^{\text{m}}/_{\overline{\text{par}}_i^{\text{m}}} = \overline{\mathcal{X}}_i^{\text{m}}\ \forall i \in I^{\text{m}}$ and $\overline{\mathcal{X}}_j^{\text{neu}}/_{\overline{\text{par}}_j^{\text{neu}}} = \overline{\mathcal{X}}_j^{\text{neu}}\ \forall j \in I^{\text{neu}}$. The restricted set of a meta-decreed or decreed variable requires the values of its parents for determining both its inclusion and its admissible values when it is included. If the decree dependencies, given the values of parent variables, dictate that a child is excluded, then its restricted set is $\{\texttt{EXC}\}$. In the

MLP example used in this work, the restricted set of the number of units $\overline{u}_i$ is

$$\overline{U}_i/_{\overline{o},\overline{l}} = \begin{cases} U_{\texttt{ASGD}} & \text{if } \overline{o} = \texttt{ASGD} \text{ and } 1 \leq i \leq \overline{l}, \\ U_{\texttt{ADAM}} & \text{if } \overline{o} = \texttt{ADAM} \text{ and } 1 \leq i \leq \overline{l}, \\ \{\texttt{EXC}\} & \text{otherwise,} \end{cases}$$

where $i \in \{1, 2\}$ and $\overline{l} \in \{0, 1, 2\}$.

6.3.5 Extended domain and transfer mapping

From the restricted sets, the extended domain $\overline{\mathcal{X}}$ is formally introduced.

Definition 43 (Extended domain)**.** *The extended domain $\overline{\mathcal{X}}$ is a hierarchical domain constructed from a domain $\mathcal{X}$, its corresponding role graph $G = (V, A)$ and from the restricted sets of all included and excluded variables. The extended domain $\overline{\mathcal{X}}$ is expressed as*

$$\overline{\mathcal{X}} := \left\{ \ \overline{x} \ : \ \overline{x}_i^r \in \overline{\mathcal{X}}_i^r/_{\overline{\text{par}}_i^r}, \ \forall r \in R, \ \forall i \in I^r \ \right\}.$$

Definition 43 expresses that an extended point $\overline{x} \in \overline{\mathcal{X}}$ must respect all the inclusion-exclusion or admissible values dependencies, *i.e.*, decree dependencies, between its variables. Recall that the main objective is to equip the domain $\mathcal{X}$ with a distance $\text{dist}_p : \mathcal{X} \times \mathcal{X} \to \overline{\mathbb{R}}^+$ to facilitate optimization or machine learning tasks on domains that involve heterogeneous dataset. This will be done by introducing the meta distance $\overline{\text{dist}}_p : \overline{\mathcal{X}} \times \overline{\mathcal{X}} \to \overline{\mathbb{R}}^+$ on the extended domain $\overline{\mathcal{X}}$, and then by inducing a distance on the domain via the bijective mapping defined in the following theorem.

Theorem 9 (One-to-one correspondence)**.** *The transfer mapping $T_G : \mathcal{X} \to \overline{\mathcal{X}}$, which assigns the extended point $T_G(x) \in \overline{\mathcal{X}}$ to any point $x \in \mathcal{X}$ by adding its excluded variables determined by the role graph G, is a bijection.*

Proof. Injectivity. Let $x, y \in \mathcal{X}$ be such that $x \neq y$. By definition of T_G, there is a least one variable whose value differ between the two extended points $\overline{x} = T_G(x)$ and $\overline{y} = T_G(y)$, since 1) there is a least one included variable between x and y that does not share the same value; or 2) there is a least one variable that is strictly excluded for one point between x and y. Therefore, there is a least one variable that also does not share the same value in the extended points $\overline{x}$ and $\overline{y}$. This show that T_G is injective since $x \neq y \Rightarrow T_G(x) \neq T_G(y)$.

Surjectivity. Let $\overline{x} \in \overline{\mathcal{X}}$ be an extended point. Then, let $G' = (V', E')$ be the subgraph of the role graph G obtained by removing the nodes and arcs corresponding to the excluded

variables that take the value `EXC` in $\overline{x}$. The set V' is nonempty since either $\overline{x}$ has no meta variables and thus $V' = V$, or $\overline{x}$ has at least one meta variable $\overline{x}_i^{\mathrm{m}} = x_i^{\mathrm{m}} \in V'$. Thus, V' is the set of variables that are included in $\overline{x}$, and E' is the set of decree dependencies between the included variables of $\overline{x}$. By construction, each variable in V' respects the decree dependencies between the other variables in the set V'. Let x be the point that contains only the included variables of the extended point $\overline{x}$, *i.e.*, the variables in the set V'. The point x necessarily belongs to domain $\mathcal{X}$, since it only contains included variables that respect the decree dependencies between each others. Indeed, otherwise, if the decree dependencies would not be respected, then x would 1) contain a variable that should not be included; or 2) not contain a variable that should be included; or 3) contain an included variable that would take a value that is not allowed by the decree dependencies. Then, by definition of the mapping T_G, $T_G(x) = \overline{x}$, since $x \in \mathcal{X}$ contains all the included variables of the extended point $\overline{x} \in \overline{\mathcal{X}}$. This shows that T_G is thus surjective, since $\forall \overline{x} \in \overline{\mathcal{X}},\ \exists x \in \mathcal{X}$, such that $T_G(x) = \overline{x}$.

The transfer mapping T_G is both injective and surjective, thus bijective. □

A consequence of Theorem 9 is that if a distance $\overline{\mathrm{dist}}_p : \overline{\mathcal{X}} \times \overline{\mathcal{X}} \to \overline{\mathbb{R}}^+$ is well-defined on the extended domain $\overline{\mathcal{X}}$, then a distance $\mathrm{dist}_p : \mathcal{X} \times \mathcal{X} \to \overline{\mathbb{R}}^+$ can be induced on the domain $\mathcal{X}$ with the bijective transfer mapping $T_G : \mathcal{X} \to \overline{\mathcal{X}}$. This is done in Section 6.4.

6.4 Distance for hierarchical domains

In this section, the meta distance $\overline{\mathrm{dist}}_p : \overline{\mathcal{X}} \times \overline{\mathcal{X}} \to \overline{\mathbb{R}}^+$ is defined. Section 6.4.1 presents the included-excluded distance function that can compute distances for variables that can be included or excluded. Then, Section 6.4.2 presents the meta distance that is constructed with included-excluded distances, one per variable.

6.4.1 Included-excluded distances

For two extended points $\overline{x}, \overline{y} \in \overline{\mathcal{X}}$, the distance regarding the i-th variable assigned to role r, respectively $\overline{x}_i^r$ and $\overline{y}_i^r$, is computed through three cases:

1. both $\overline{x}_i^r$ and $\overline{y}_i^r$ are excluded, hence the distance is set to zero;
2. exactly one variable $\overline{x}_i^r$ or $\overline{y}_i^r$ is excluded, hence the distance is set to a parameter that models a distance between a variable that is included for one extended point, and excluded for the other extended point;
3. both $\overline{x}_i^r$ and $\overline{y}_i^r$ are included, hence an one-dimensional distance function d is used, *e.g.*, the Euclidean distance [240].

Recall that a meta or neutral variable $\overline{x}_i^r \in \overline{\mathcal{X}}_i^r$ is always included, hence for $r \in \{\mathrm{m}, \mathrm{neu}\}$, the distance between $\overline{x}_i^r$ and $\overline{y}_i^r$ is always computed in the third case. For a meta-decreed or decreed variable, the restricted sets of $\overline{x}_i^r$ and $\overline{y}_i^r$ may differ, hence its corresponding included-excluded distance must be defined on its universal set $\overline{\mathcal{X}}_i^r$ in order to allow comparisons of any pairs $\overline{x}_i^r, \overline{y}_i^r$ with different restricted sets. In the MLP example, the universal set of the number of units $\overline{u}_1$ is $\overline{U} = U_{\texttt{ASGD}} \cup U_{\texttt{ADAM}} \cup \{\texttt{EXC}\}$. Hence, to compare the number of units $\overline{u}_i$ from any two pairs of extended points, the corresponding included-excluded distance of $\overline{u}_i$ must be defined on its universal set $\overline{U}$. The following theorem formalizes the discussion above on the three cases and the universal set by introducing a novel distance based on distances proposed in [221, 226].

Theorem 10 (Included-excluded distance)**.** *Let $\overline{\mathcal{X}}_i^r$ be the universal set of the i-th variable assigned to the role $r \in R$, noted $\overline{x}_i^r$, and define $\overline{\mathcal{Y}}_i^r = \overline{\mathcal{X}}_i^r \setminus \{\texttt{EXC}\}$. Consider $d : \overline{\mathcal{Y}}_i^r \times \overline{\mathcal{Y}}_i^r \to \overline{\mathbb{R}}^+$, a one-dimensional extended real-valued distance for the variable $\overline{x}_i^r$ when it is included, and $\theta_i^r \in \overline{\mathbb{R}}^+$ a parameter greater than or equal to $\sup\{d(\mu, \nu) : \mu, \nu \in \overline{\mathcal{Y}}_i^r\}/2$. Then, for $u, v \in \overline{\mathcal{X}}_i^r$, the function $d_i^r : \overline{\mathcal{X}}_i^r \times \overline{\mathcal{X}}_i^r \to \overline{\mathbb{R}}^+$ defined by*

$$d_i^r(u, v) := \begin{cases} d(u, v) & \text{if } u \neq \texttt{EXC} \neq v \text{ (both included)}, \\ 0 & \text{if } u = \texttt{EXC} = v \text{ (both excluded)}, \\ \theta_i^r & \text{otherwise} \quad \text{(one excluded)}, \end{cases} \tag{6.3}$$

is a one-dimensional extended real-valued distance function.

Proof. Let $r \in R$ and $i \in I^r$. The identity of indiscernibles, nonnegativity and symmetry of d_i^r are trivially proven since θ_i^r is strictly positive and since d is a distance function. The rest of the proof consists of proving that d_i^r satisfies the triangle inequality. Let $u, v, z \in \overline{\mathcal{X}}_i^r$:

Case 1 (both variables are excluded): if $u = \texttt{EXC} = v$, then

$$d_i^r(u, v) = 0 \ \leq \ d_i^r(u, z) \ + \ d_i^r(z, v), \text{ by nonnegativity of } d_i^r.$$

Case 2 (only one variable is excluded, by symmetry v): if $u \neq \texttt{EXC} = v$ and

- if $z \neq \texttt{EXC}$, then

$$d_i^r(u, v) = \theta_i^r \ \leq \ d(u, z) \ + \ \theta_i^r \ = \ d_i^r(u, z) \ + \ d_i^r(z, v), \text{ by nonnegativity of } d.$$

- if $z = \texttt{EXC}$, then

$$d_i^r(u,v) = \theta_i^r \;\leq\; \theta_i^r \;+\; 0 \;=\; d_i^r(u,z) \;+\; d_i^r(z,v).$$

Case 3 (both variables are included): if $u \neq \texttt{EXC}$ and $v \neq \texttt{EXC}$ and

- if $z \neq \texttt{EXC}$, then

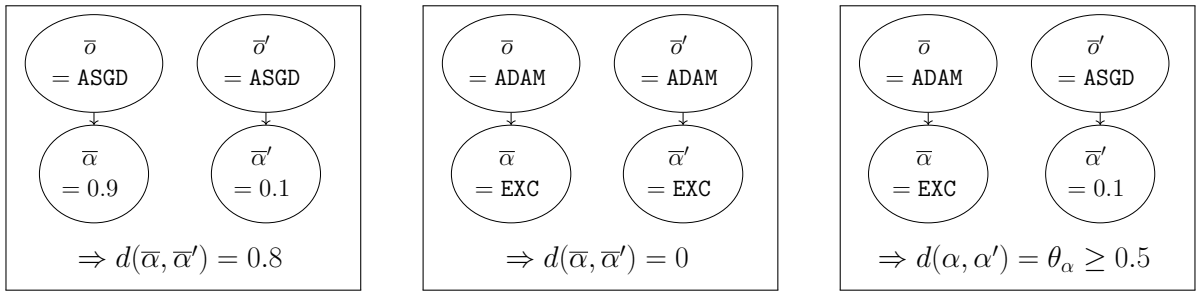
$$d_i^r(u,v) \;=\; d(u,v) \;\leq\; d(u,z) \;+\; d(z,v) \;=\; d_i^r(u,z) \;+\; d_i^r(z,v).$$

- if $z = \texttt{EXC}$, then

$$d_i^r(u,v) \;=\; d(u,v) \;\leq\; \sup\{d(\mu,\nu) : \mu,\nu \in \overline{\mathcal{Y}}_i^r\} \;\leq\; 2\theta_i^r = d_i^r(u,z) \;+\; d_i^r(z,v).$$

□

The following figure schematizes the included-excluded distance on a variant of the MLP example with only two variables. Each graph represents an extended point.



(a) Both included. (b) Both excluded. (c) One excluded.

Figure 6.7 Included-excluded distance for α, whose inclusion is controlled by the optimizer.

Figure 6.7 presents the three cases of the included-excluded distance for the update α, whose inclusion is controlled by the optimizer. In Figure 6.7a, the update is included in both extended points. In Figure 6.7b, it is excluded in both extended points, hence the distance is zero. Finally, in Figure 6.7c, the update is excluded in only one extended point. In this case, the distance is set to a parameter θ_α bounded below half of the largest possible distance achievable for two included updates, *i.e.*, $\sup\{d(\alpha,\alpha') : \alpha,\alpha' \in]0,1[\}/2 = 1/2$.

Next, multiple comments on the included-excluded distance are provided. First, the included-excluded distance is precisely useful for meta-decreed and decreed variables that can have

different restricted sets, *i.e.* that can be included or excluded, and/or have different admissible values. As mentioned previously, for variables that are always included, such as meta and neutral variables, included-excluded distances are always computed with the both included case. The included-excluded distance is nevertheless defined for these variables in order to obtain a concise formulation of the meta distance in Section 6.4.2.

Second, the included-excluded distance is compatible with any variable type. Indeed, in (6.3), the two cases both excluded and one excluded does not regard the variable type, and the case both included allows to use any distance function d.

Third, the universal set $\overline{\mathcal{X}}_i^r$ allows to compare any pair of variables $\overline{x}_i^r, \overline{y}_i^r$. Moreover, it is also necessary to establish a lower bound on the parameter θ_i^r, which ensures the triangular inequality in the last case of the proof. The inequality $\theta_i^r \geq \sup\{d(\mu,\nu) : \mu,\nu \in \overline{\mathcal{X}}_i^r \setminus \{\texttt{EXC}\}\}/2$ implies that the distance $d_i^r(\overline{x}_i^r, \overline{y}_i^r) = \theta_i^r$ (one excluded case) must be at least half the largest distance between any pairs of included variables $\overline{x}_i^r, \overline{y}_i^r$, with possibly different restricted sets. Apart from its lower bound, the parameter θ_i^r is flexible.

Fourth, the included-excluded distance is more formally an extended real-valued one [42], since it is allowed to take the infinite value. The infinity value allows to consider meta-decreed or decreed variables with unbounded restricted sets. For example, let $r \in \{\mathrm{md}, \mathrm{dec}\}$ and $i \in I^r$, such that $\overline{\mathcal{X}}_i^r / \overline{\mathrm{par}}_i^r = [0, \infty[$ when it is included, and $\overline{\mathcal{X}}_i^r / \overline{\mathrm{par}}_i^r = \{\texttt{EXC}\}$ when it is excluded. In this example, $\overline{\mathcal{X}}_i^r = [0, \infty[\cup \{\texttt{EXC}\}$, therefore $\sup\{d(\mu,\nu) : \mu,\nu \in \overline{\mathcal{X}}_i^r \setminus \{\texttt{EXC}\}\} = \infty$, hence the parameter θ_i^r must be set to infinity to guaranty the triangular inequality which is a behaviour one would generally avoid. If the restricted sets are always bounded, then included-excluded distance becomes a standard distance that maps into $\mathbb{R}$, instead of $\overline{\mathbb{R}}$.

6.4.2 Meta distance and induced distance

Now that the included-excluded distance has been detailed, the following theorem formally introduces the meta distance.

Theorem 11 (Meta distance)**.** *For any $p \geq 1$, the function $\overline{\mathrm{dist}}_p : \overline{\mathcal{X}} \times \overline{\mathcal{X}} \to \overline{\mathbb{R}}^+$ defined by*

$$\overline{\mathrm{dist}}_p(\overline{x}, \overline{y}) := \left(\sum_{r \in R} \sum_{i \in I^r} d_i^r \left(\overline{x}_i^r, \overline{y}_i^r\right)^p \right)^{1/p}, \tag{6.4}$$

is an extended real-valued distance function, where $R = \{\mathrm{m}, \mathrm{md}, \mathrm{dec}, \mathrm{neu}\}$, $i \in I^r = \{1, 2, \ldots, n^r\}$ and $n^r \in \mathbb{N}$ is the number of variables assigned to the role r.

Proof. The identity of indiscernibles, nonnegativity and symmetry of $\overline{\mathrm{dist}}_p$ are trivially proven

since the operations of summation and exponentiation with $p \geq 1$ on the included-excluded distances in (6.4) conserve these properties. The rest of the proof consists of showing the triangular inequality is respected by demonstrating that $\overline{\text{dist}}_p$ is equivalent to a p-norm, that respects the triangular inequality. Let $K = \{1, 2, \ldots, n^{\text{m}} + n^{\text{md}} + n^{\text{dec}} + n^{\text{neu}}\}$ be a set of indices that reorders the indices $r \in \{\text{m}, \text{md}, \text{dec}, \text{neu}\}$ and $i \in \{1, 2, \ldots, n^r\}$:

- $a_k := d_i^{\text{m}}\left(\overline{x}_i^{\text{m}}, \overline{y}_i^{\text{m}}\right)$, for $k = i$ with $i \in \{1, 2, \ldots, n^{\text{m}}\}$,
- $a_k := d_j^{\text{md}}\left(\overline{x}_j^{\text{md}}, \overline{y}_j^{\text{md}}\right)$, for $k = n^{\text{m}} + j$ with $j \in \{1, 2, \ldots, n^{\text{md}}\}$,
- $a_k := d_l^{\text{dec}}\left(\overline{x}_l^{\text{dec}}, \overline{y}_l^{\text{dec}}\right)$, for $k = n^{\text{m}} + n^{\text{md}} + l$ with $l \in \{1, 2, \ldots, n^{\text{dec}}\}$,
- $a_k := d_v^{\text{neu}}\left(\overline{x}_v^{\text{neu}}, \overline{y}_v^{\text{neu}}\right)$, for $k = n^{\text{m}} + n^{\text{md}} + n^{\text{dec}} + v$ with $v \in \{1, 2, \ldots, n^{\text{neu}}\}$.

Finally, let $a = \left(a_1, a_2, \ldots, a_{|K|}\right)$, then

$$\|a\|_p = \left(\sum_{k=1}^{|K|} |a_k|^p\right)^{1/p} = \left(\sum_{r \in R} \sum_{i \in I^r} d_i^r\left(\overline{x}_i^r, \overline{y}_i^r\right)^p\right)^{1/p} = \overline{\text{dist}}_p\left(\overline{x}, \overline{y}\right)$$

□

For $p \to \infty$, the meta distance $\overline{\text{dist}}_\infty : \overline{\mathcal{X}} \times \overline{\mathcal{X}} \to \overline{\mathbb{R}}^+$ converge towards the supremum distance, *i.e.*,

$$\overline{\text{dist}}_\infty\left(\overline{x}, \overline{y}\right) := \max\left\{\ d_i^r\left(\overline{x}_i^r, \overline{y}_i^r\right)\ :\ r \in R,\ i \in I^r\right\}, \tag{6.5}$$

which is trivially a distance function by virtue of the max function.

Note that, in practice, variables often require scaling to improve the conditioning and eliminate biases related to variable scales. In the context of the work, scaling categorical variables and excluded variables is ambiguous. Fortunately, in our proposed distance, an included-excluded distance d_i^r is defined with a flexible distance d for its both included case. Therefore, the distance d can be defined using a scaling parameter, such that $d(a, b) = \omega_i^r d'(a, b)$, where $\omega_i^r > 0$ is a weight parameter related to the variable $\overline{x}_i^r$ and d' is a one-dimensional distance of appropriate variable type. The weight parameter ω_i^r can be used to automatically scale the lower bounds of the parameter θ_i^r, since the lower bound of θ_i^r is defined with the one-dimensional distance d. In Section 6.5, scaling parameters will be used to better adjust our proposed distance to the datasets; this helps to remove biases that are related to variable scales.

Theorem 9, which establishes the bijection between the domain $\mathcal{X}$ and the extended domain $\overline{\mathcal{X}}$, implies that a distance $\text{dist}_p : \mathcal{X} \times \mathcal{X} \to \overline{\mathbb{R}}^+$ can be induced from the meta distance $\overline{\text{dist}}_p : \overline{\mathcal{X}} \times \overline{\mathcal{X}} \to \overline{\mathbb{R}}^+$. The following corollary is a direct consequence of Theorems 9 and 11.

Corollary 1 (Induced distance)**.** *For $p \geq 1$, the induced function $\mathrm{dist}_p : \mathcal{X} \times \mathcal{X} \to \overline{\mathbb{R}}^+$ defined by*

$$\mathrm{dist}_p(x, y) := \overline{\mathrm{dist}}_p\left(T_G(x), T_G(y)\right) = \overline{\mathrm{dist}}_p\left(\overline{x}, \overline{y}\right), \tag{6.6}$$

is an extended real-valued distance, where $\overline{\mathrm{dist}}_p : \overline{\mathcal{X}} \times \overline{\mathcal{X}} \to \overline{\mathbb{R}}^+$ is a meta distance and $T_G : \mathcal{X} \to \overline{\mathcal{X}}$ is the bijective transfer mapping.

Corollary 1 unpacks most of the contributions. To arrive at Corollary 1, it was necessary to: 1) define an extended point $\overline{x}$, restricted sets and the extended domain $\overline{\mathcal{X}}$ using notions from graph theory; 2) define the transfer mapping $T_G : \mathcal{X} \to \overline{\mathcal{X}}$, and prove that it is bijective, 3) define the included-excluded distances on the universal set for tackling variables that can either be included or excluded, or with different admissible values, and 4) define the meta distance $\overline{\mathrm{dist}}_p : \overline{\mathcal{X}} \times \overline{\mathcal{X}} \to \overline{\mathbb{R}}^+$ based on the contributions mentioned above.

6.5 Computational experiments on mixed-variable and hierarchical problems

This section compares three approaches on mixed-variable and hierarchical regression and classification problems with simple distance-based models. The first approach, named Sub, divides a problem into subproblems, each assigned to a portion of the domain in which the included variables are fixed. In the experiments, Sub constructs one model per subproblem, and each model uses an Euclidean distance defined on their corresponding subdomain.

The second approach, called Meta, constructs a single model with the induced distance from (6.6), where $p = 2$. This approach directly tackles a hierarchical problem and aggregates data across the subproblems.

The third approach, named Hybrid, employs the variable-size framework [196] in which one layer of hierarchy is tackled and only shared included variable are compared. Hybrid constructs a single model for the first two variants, and two models for the other variants.

The experiments focus on comparing approaches and their implementation of a distance. For this reason, simple distance-based models are employed: Inverse Distance Weighting (IDW) models for regression, and K-nearest neighbors models for classification. Earth mover's (or Wasserstein) distances are not considered [111, 240], since their computations require solving an optimal transport problem. Distances are evaluated repeatedly in the experiments, and they are unsuitable in this setting.

6.5.1 Description of the hyperparameters domain and its datasets

In the experiments, data points are vector of hyperparameters. In order to restrict the number of variables, few important hyperparameters are intentionally discarded, such as the momentum, the activation function and the batch size. The hyperparameters domain (HPD) of below, is used to generate datasets in hierarchical and mixed-variable domains.

HP	Variable	Bounds	Role
Learning rate	r	$]0,1[$	Neutral
Optimizer	o	$\{\texttt{ASGD}, \texttt{ADAM}\}$	Meta

if $o = \texttt{ASGD}$

HP	Variable	Bounds	Role
Decay	α_1	$]0,1[$	Decreed
Power update	α_2	$]0,1[$	Decreed
Average start	α_3	$]0,1[$	Decreed
# of hidden layers	l	$L_{\texttt{ASGD}}$	Meta-dec.
# of units layer i	u_i	$U_{\texttt{ASGD}}$	Decreed

if $o = \texttt{ADAM}$

HP	Variable	Bounds	Role
Running average 1	β_1	$]0,1[$	Decreed
Running average 2	β_2	$]0,1[$	Decreed
Numerical stability	β_3	$]0,1[$	Decreed
# of hidden layers	l	$L_{\texttt{ADAM}}$	Meta-dec.
# of units layer i	u_i	$U_{\texttt{ADAM}}$	Decreed

HP	Variable	Bounds	Role
Dropout	ρ	$\left[0, \frac{1}{2\tau_{\max}} \sum_{i=1}^{l} u_i\right]$	Decreed

Figure 6.8 Hierarchical and mixed-variable domain of the HPD.

In Figure 6.8, the bounds of the hyperparameters α_1, α_2, α_3, β_1, β_2 and β_3 are normalized. The HPD has two levels of hierarchy, it contains meta and meta-decreed variables influencing inclusions or bounds.

To perform regression or classification, data points must be associated with corresponding images. The image $f(x) \in [0, 100]$ of a vector of hyperparameters $x \in \mathcal{X}$ consists of a performance test score between 0 and 100, as illustrated below.

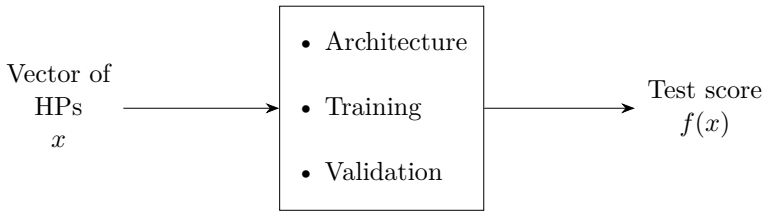


Figure 6.9 A data point $x \in \mathcal{X}$ and its image $f(x) \in [0, 100]$ in the computational experiments.

Figure 6.9 illustrates that computing the image $f(x)$ of a vector of hyperparameters x involves training, validating and testing a deep model. The architecture of this deep model is not

treated as a variable, but rather a fixed parameter influencing the behavior of f.

In the experiments, there are 40 independent datasets. A dataset is identified with the nomenclature HPD-**variant-size-architecture**, where the three elements in bold are:

- a **variant** of the HPD, in which some hyperparameters are fixed. The variants range from #1 to #5, representing increasing difficulties. They are detailed in A.
- a **size**, amongst Very Small (VS), Small (S), Medium (M) and Large (L), representing the amount of data available. The sizes represent a second lever of difficulty, as smaller datasets are typically more difficult to tackle. The datasets are independent across sizes, *i.e.*, there are no subsets of datasets. Their sizes are detailed in B.
- a type of **architecture**, either MLP or Convolutional Neural Network (CNN), is used to structure the hyperparameters and compute f at a point $x \in \mathcal{X}$. The type of architecture introduces diversity into the data, as discussed previously.

For the CNN architecture, units represent channels. The MLP architecture uses Fashion-MNIST [265], whereas CNN employs CIFAR10 [156]. Fashion-MNIST and CIFAR10 are strictly used for generating data points, as in Figure 6.9: they must not be confused with the actual datasets used in the computational experiments. Details on the generation of data are provided in C.

The datasets are randomly split into training (50%), validation (25%) and test (25%) datasets. A dataset paired with a random seed is referred to as an *instance*, and is denoted p. Also, an approach paired with a model is called an *approach-model*, and is denoted s.

6.5.2 Classification experiments

The first experiment evaluates the classification accuracy on test datasets using a K-nearest neighbors model. Training datasets contain points consisting of available neighbors, and validation datasets are used to adjust the parameters of the approach-models. The budget of evaluations to train an approach-model $s \in \{\mathsf{Sub}\text{-KNN}, \mathsf{Hybrid}\text{-KNN}, \mathsf{Meta}\text{-KNN}\}$ on a given instance p is $100n_{p,s}$, where $n_{p,s}$ is the number of parameters of s on p.

The datasets considered are the HPD-**X-VS-MLP** and HPD-**X-S-MLP**, *i.e.*, the very small and small ones with the MLP architecture. The experiments are performed using 5 labels. The images $f(x) \in [0, 100]$ in the datasets are transformed into labels, by partitioning uniformly the interval $[0, 100]$ into five subintervals of length 20 and assigning labels. For example, an image $f(x) \in [0, 20[$ is replaced by the label 0, and an image $f(y) \in [80, 100]$ by

the label 4. The MLP architecture is used for classification since it produces images that are more uniformly distributed over $[0, 100]$ than the CNN architecture. Table 6.3 presents the accuracies obtained on test datasets, and these accuracies are computed as an average over 20 random seeds.

Table 6.3 Average accuracies on test datasets with 20 random seeds.

(a) Very small size.

Variant	**Sub**	**Hybrid**	**Meta**
#1	0.57	0.67	**0.68**
#2	0.60	**0.67**	0.62
#3	0.52	**0.56**	0.51
#4	0.54	**0.62**	0.57
#5	**0.58**	0.57	**0.58**

(b) Small size.

Variant	**Sub**	**Hybrid**	**Meta**
#1	0.67	0.72	**0.73**
#2	0.67	**0.71**	0.70
#3	0.59	**0.62**	0.60
#4	0.57	**0.64**	0.61
#5	0.53	**0.55**	0.54

While Meta achieved the highest accuracy on variant #1, Hybrid performed best overall. Sub obtained the worst performance across the experiments. The standard deviations obtained are consistently similar across all variants, sizes and approaches, ranging from 0.05 to 0.1.

6.5.3 Data profiles on regression problems

The main computational experiments are presented in this section. Root Mean Squared Errors (RMSE) on validation and test datasets, respectively referred to RMSE validation and RMSE test, are computed and viewed as functions.

Before computing the RSME test, parameters are adjusted with respect to a corresponding RMSE validation. The optimization is expressed as a least-squares problem focused on minimizing the RMSE validation. The decision variables are the parameters of an approach-model $s \in \{$Sub-IDW, Hybrid-IDW, Meta-IDW$\}$. Each approach-model has weight parameters for the hyperparameters. For Sub and Hybrid, some hyperparameters are repeated across subproblems, leading to distinct weight parameters per subproblem. Meta requires parameters for the excluded-included distances between hyperparameters that can take the special value `EXC`, and a parameter for the categorical distance of the optimizer (in the situations where the categorical distance is not fixed).

The derivatives of the objective functions are unknown. The optimization of the RMSE validation is done with the open-source blackbox optimization software NOMAD [30] that is based on the Mesh Adaptive Direct Search algorithm [21]. RMSE tests are used to assess the

generalization ability of the approach-models, and they are not subject to any optimization.

Each dataset is instantiated by 5 seeds, for a total of 200 instances contained in the set $\mathcal{P}$. An approach-model s is said to have τ-solved an instance $p \in \mathcal{P}$ when its RMSE is within a relative error of τ with respect to the best RMSE obtained by any s on this instance p. At evaluation $k \geq 0$, the convergence test of an approach-model s on a instance p is expressed as

$$\frac{\text{RMSE}_{p,s}(k) - \text{RMSE}_p}{\text{RMSE}_p} \leq \tau, \tag{6.7}$$

where $\text{RMSE}_{p,s}(k) > 0$ is the RMSE of s on instance p at an evaluation $k \geq 0$, $\text{RMSE}_p > 0$ is the best RMSE obtained on that instance p and $\tau \in [0, 1]$ is a given tolerance. Data profiles [179] represent the proportion of τ-solved instances by the approach-models:

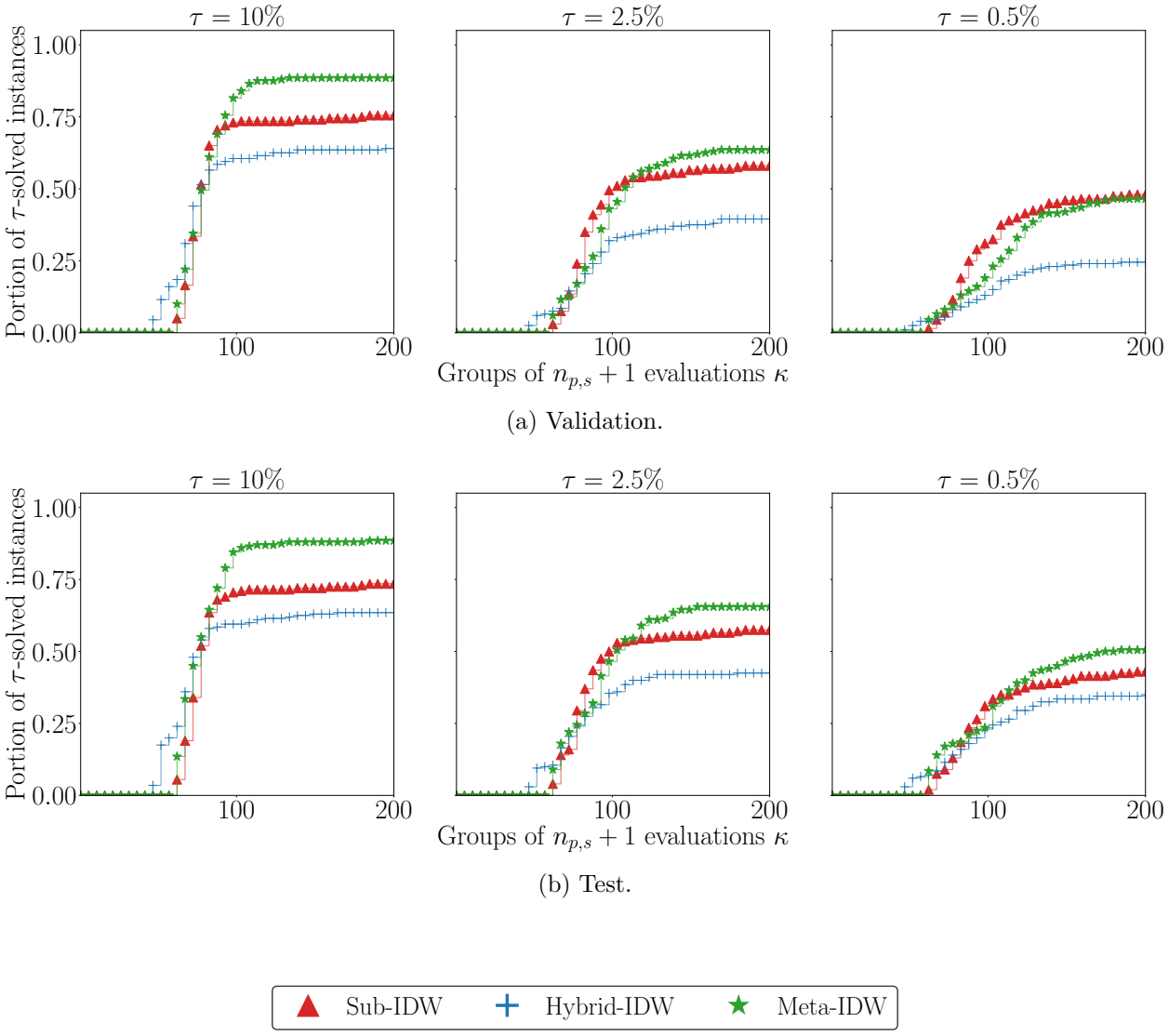
$$\text{data}_s(\kappa) \coloneqq \frac{1}{|\mathcal{P}|} \left| \left\{ p \in \mathcal{P} \ : \ \frac{k_{p,s}}{n_{p,s}+1} \leq \kappa \right\} \right| \in [0, 1] \tag{6.8}$$

where $k_{p,s} \geq 0$ is the number of evaluations required for an approach-model s to τ-solved an instance $p \in \mathcal{P}$ and $n_{p,s}$ is the number of parameters of a solver s on the instance p. The horizontal axis κ of a data profile represents groups of $(n_{p,s}+1)$ evaluations, which depends on the instance p and the approach-model s, since the number of parameters $n_{p,s}$ (decision variables) depends on both of these.

In the experiments, the budget of evaluations for an approach-model s on instance p is $200n_{p,s}$ evaluations. Each optimization starts with Latin hypercube sampling (LHS) of 33% of the total budget of evaluations. See Appendix A.3. in [26] for more details.

Data profiles are presented in Figure 6.10 for both the RMSE validation (objective function) and the RMSE test. The optimization problem is only performed on the RMSE validation. Depending on the variant and size, solving an instance required between a few minutes and 75 minutes on an 11th generation Intel i7-11800H (2.30 GHz) CPU. The maximum runtime of approximately 75 minutes was observed for all three approaches for instances with variant #5 and the size set to large. RMSE tests are evaluated each time an improvement on the RMSE validation is found, in order to build the data profiles for the RMSE tests.

The data profiles for the RMSE validation and test clearly indicates that Meta outperforms the two other approaches. For example, Figure 6.10b with $\tau = 0.5\%$ shows that Meta-KNN τ-solves approximately 30% of the instances with $100(n_{p,s}+1)$ evaluations, and after $200(n_{p,s}+1)$ evaluations it τ-solves approximately 50% of the instances, whereas Hybrid only τ-solves 30%. The Hybrid provides the worst performance on all profiles. All data profiles exhibit a plateau during initial iterations, due to the LHS initialization.



(a) Validation.

(b) Test.

Figure 6.10 Data profiles of the solvers on the regression problems with IDW models.

6.5.4 Data aggregation with selected parameters experiments

In this section, computational experiments are performed to study the aggregation of data. The RMSE test is plotted against the number of points in a (partial) training dataset. IDW models are constructed with a partial training dataset, which is increased iteratively with an additional random data point drawn from the training dataset without replacement. The partial training dataset begins with a random data point from each subproblem, and points are added until it becomes the training dataset. Training datasets represent interpolation points for IDW models. In early iterations, the numbers of points are limited, particularly

when distributed across different subproblems.

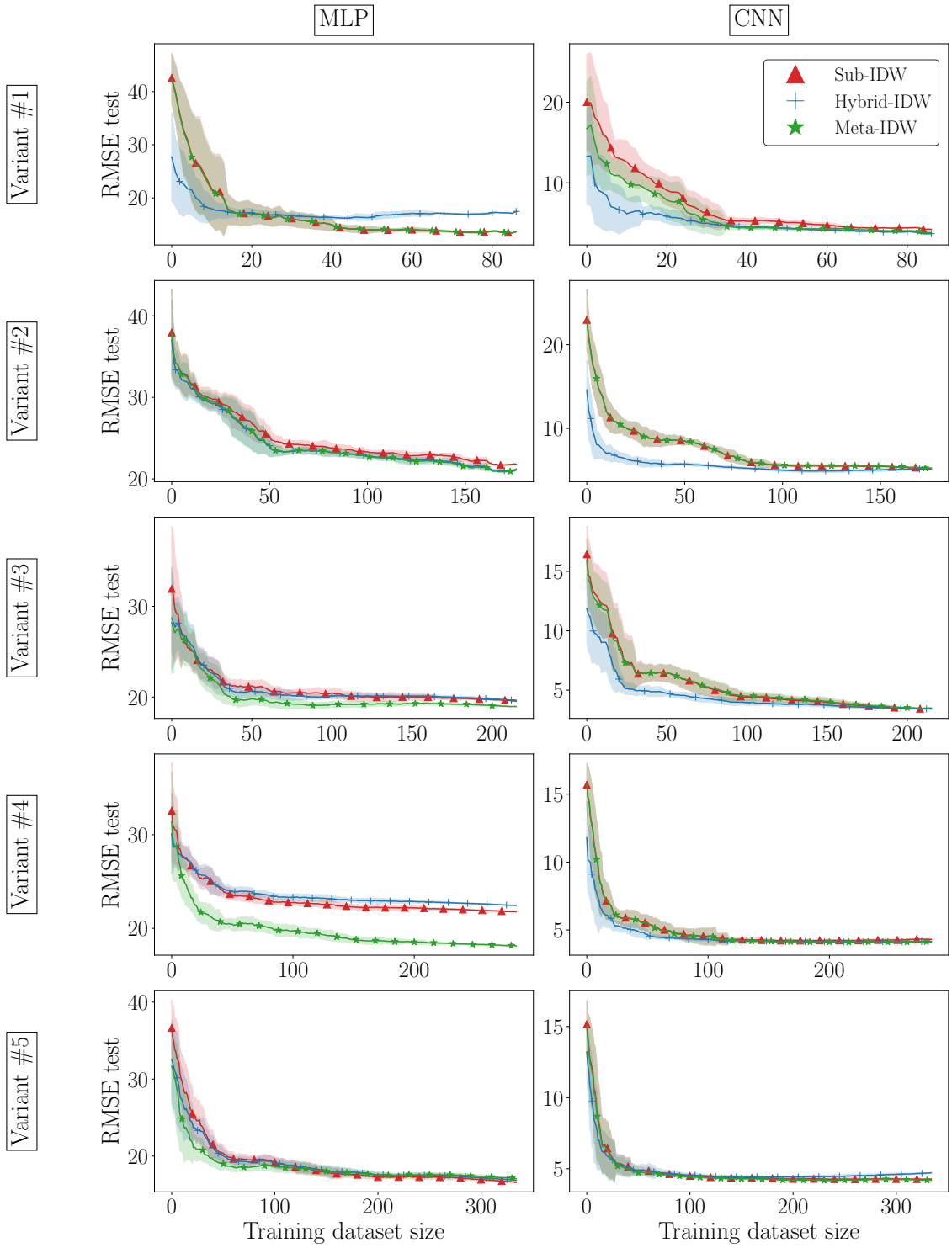


Figure 6.11 RMSE tests with iteratively increasing training data points and selected parameters. Each subfigure is composed of 20 runs on different random seeds.

The datasets are the HPD-**X**-M-**A**, *i.e.*, all variants and both architecture are considered, while size is fixed at medium. The adjusted parameters obtained using the first random seed from Section 6.5.3 are selected and fixed for all iterations.

In Figures 6.11, solid lines represent means, and shaded areas represent standard deviations. Each point in the graphs is averaged with 20 random seeds. For the MLP architecture, Meta performs best, whereas Sub and Hybrid have similar performances. For the CNN architecture, Hybrid performs best at early iterations. Towards the final iterations, all three approaches have similar performances, except for variant #5, where Meta has a slightly best RMSE test.

6.6 Discussion

The present work focused on modeling mixed-variable and hierarchical domains in which two points do not share the same variables. The research gap addressed is the lack of a distance function that is interpretable and is computed in constant-time. To formalize the distance, a modeling framework is proposed, generalizing the following state-of-the-art frameworks: mixed-variable domains with (strictly) meta variables [25]; tree-structured spaces [45]; hierarchical spaces [135]; variable-size design space [38, 196]; and conditional search space (or neural architecture search) [143].

The approach employed in this work can be summarized in Figure 6.12, reprising Figure 6.2b from the introduction.

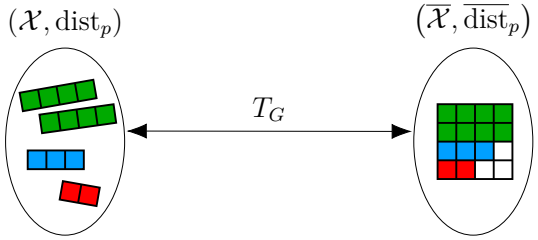


Figure 6.12 A synthesis of the work.

A complicated heterogeneous dataset in a hierarchical domain $\mathcal{X}$ is mapped in the extended domain $\overline{\mathcal{X}}$, where the meta distance $\overline{\mathrm{dist}}_p$ is used to compare any pair of points of the dataset. A mapping T_G based on a graph structure G establishes a one-to-one correspondence between the original domain $\mathcal{X}$ and the extended domain $\overline{\mathcal{X}}$. With this mapping, a distance dist_p is induced on the domain $\mathcal{X}$.

Theoretically, the meta distance should provide better generaliziability than simpler approaches partitioning domains, as it it allows to use more data simultaneously. The compu-

tational experiments seems to confirm this. Our approach using the meta distance performed globally better than an approach using subproblems and another one based on the variable-size framework [196].

This work is an important stepping stone towards efficient machine learning and optimization involving mixed-variable and hierarchical domains. However, much more work remains. Currently, more complicated hierarchical interrelationships between variables, such as incompatibilities between values of different variables, are being developed in a subsequent work. Moreover, further computational experiments on diverse problems, tasks and models are required. For instance, Gaussian processes are commonly used for such mixed-variable and/or hierarchical domains [196, 224, 226], and these models will be studied in future work, with a focus on Bayesian optimization.

Data availability statement

Scripts and data are publicly available at https://github.com/bbopt/graph_distance.

Declaration of interest statement.

On behalf of all authors, the corresponding author states that there is no conflict of interest.

Funding.

This research is funded by a Natural Sciences and Engineering Research Council of Canada (NSERC) PhD Excellence Scholarship (PGS D), a Fonds de Recherche du Québec (FRQNT) PhD Excellence Scholarship and an Institut de l'Énergie Trottier (IET) PhD Excellence Scholarship, as well as by the NSERC discovery grants RGPIN-2020-04448 (Audet), RGPIN-2024-05093 (Diouane) and RGPIN-2018-05286 (Le Digabel). The work of Saves is part of the activities of ONERA - ISAE - ENAC joint research group. His research, presented in this paper, has been performed in the framework of the COLOSSUS project (Collaborative System of Systems Exploration of Aviation Products, Services and Business Models) and has received funding from the European Union Horizon Europe program under grant agreement n° 101097120 and in the MIMICO research project funded by the Agence Nationale de la Recherche (ANR) n^o ANR-24-CE23-0380.

Acknowledgments

We express our gratitude to Amaury Diopus'kin for the PyTorch implementation, produced during its internship in summer 2022, that was used for data generation.

A Variants of the HPD.

Table 6.4 details the five variants and the parameters (decision variables) for the Sub and Meta approaches. The variants are of increasing difficulty, notably in terms of number of subproblems and variables present. Variant #1 fixes the optimizer, hence the number of layers is meta instead of meta-decreed. There are 3 subproblems, each assigned to a fixed number of layers. Variant #2 adds the hyperparameters $\alpha_{1:3}$. Variant #3 frees the optimizer $o \in \{\texttt{ASGD}, \texttt{ADAM}\}$, and the hyperparameters $\beta_{1:3}$ are introduced via $o = \texttt{ADAM}$. There are 4 subproblems, each assigned to a pair (o, l). Variant #4 adds a subproblem by allowing $l = 3$ when $o = \texttt{ADAM}$. The number of layers l becomes a meta-decreed variable, since $L_{\texttt{ASGD}} \neq L_{\texttt{ADAM}}$. Variant #5 adds the dropout ρ.

For Sub, the variants are treated by their disconnected graphs separately. In contrast, Meta, considers each variant has a single problem, represented by a rectangular border. For Hybrid, only one layer of hierarchy is considered. Hence, in variant 1 and 2, all subproblems are considered since there is one layer of hierarchy. For the other variants, Hybrid has two subproblems, one per optimizer, since the layer of hierarchy tackle is the one related to the number of layers.

B Dataset sizes of the instances.

Table 6.5 details the dataset sizes of all variants for Meta and Sub. For variants #1 and #2, Hybrid aggregates all data as Meta. For the other variants, Hybrid aggregates over the hidden layers, but separates the data for each optimizer in Table 6.5a.

In Table 6.5b, the values in size-related columns are the sums of the values in the rows of the corresponding columns in Table 6.5a, *i.e.*, Meta aggregates the data of the subproblems of Sub.

C Data generation.

The generation of a data couple $(x, f(x))$ is done by the following three steps: 1) a deep model, either MLP or CNN, is constructed with respect to its given hyperparameters $x \in \mathcal{X}$ with PyTorch; 2) the deep model is trained and validated with 25% on either the Fashion-MNIST training dataset or CIFAR10 training dataset; 3) the performance score $f(x) \in [0, 100]$, that represents the percentage of well-classified images, is computed on the deep model with another 25% of either the corresponding Fashion-MNIST or CIFAR10 test datasets. In order to reduce data generation time, only 25% of the Fashion-MNIST and CIFAR10

Table 6.4 Variants of HPD with $u_{1:2} = (u_1, u_2)$, $u_{1:3} = (u_1, u_2, u_3)$, $\alpha_{1:3} = (\alpha_1, \alpha_2, \alpha_3)$ and $\beta_{1:3} = (\beta_1, \beta_2, \beta_3)$.

(a) Parameters for the Sub approach.

Var.	o	l	Variables	# of variables
#1	ASGD	1	$r\ u_1$	2
		2	$r\ u_{1:2}$	3
		3	$r\ u_{1:3}$	4
#2	ASGD	1	$r\ u_1\ \ \alpha_{1:3}$	5
		2	$r\ u_{1:2}\ \alpha_{1:3}$	6
		3	$r\ u_{1:3}\ \alpha_{1:3}$	7
#3	ASGD	1	$r\ u_1\ \ \alpha_{1:3}$	5
		2	$r\ u_{1:2}\ \alpha_{1:3}$	6
	ADAM	1	$r\ u_1\ \ \beta_{1:3}$	5
		2	$r\ u_{1:2}\ \beta_{1:3}$	6
#4	ASGD	1	$r\ u_1\ \ \alpha_{1:3}$	5
		2	$r\ u_{1:2}\ \alpha_{1:3}$	6
	ADAM	1	$r\ u_1\ \ \beta_{1:3}$	5
		2	$r\ u_{1:2}\ \beta_{1:3}$	6
		3	$r\ u_{1:3}\ \beta_{1:3}$	7
#5	ASGD	1	$r\ u_1\ \ \alpha_{1:3}\ \rho$	6
		2	$r\ u_{1:2}\ \alpha_{1:3}\ \rho$	7
	ADAM	1	$r\ u_1\ \ \beta_{1:3}\ \rho$	6
		2	$r\ u_{1:2}\ \beta_{1:3}\ \rho$	7
		3	$r\ u_{1:3}\ \beta_{1:3}\ \rho$	8

(b) Parameters for the Meta approach.

Var.	Variables	# of variables	# of θ_i^r
#1	$l, r, u_{1:3}$	5	2
#2	$l, r, u_{1:3}, \alpha_{1:3}$	8	2
#3	$o, l, r, u_{1:2}, \alpha_{1:3}, \beta_{1:3}$	11	7
#4	$o, l, r, u_{1:3}, \alpha_{1:3}, \beta_{1:3}$	12	8
#5	$o, l, r, u_{1:3}, \alpha_{1:3}, \beta_{1:3}, \rho$	13	8

datasets are used, and the number of epochs and batch size are respectively set to 25 and 128. Fashion-MNIST and CIFAR10 are strictly used for generating data couples $(x, f(x))$ that form datasets. Each dataset is heterogeneous and it is split into a training, validation and test datasets.

Table 6.5 Dataset sizes of the instances, characterized by a variant-size, with sizes amongst Very Small (VS), Small (S), Medium (M) and Large (L).

(a) Dataset sizes for Sub approach.

Variant	o	l	VS	S	M	L
#1	ASGD	1	20	30	40	50
		2	30	45	60	75
		3	40	60	80	100
#2	ASGD	1	50	75	100	125
		2	60	90	120	150
		3	70	105	140	175
#3	ASGD	1	50	75	100	125
		2	60	90	120	150
	ADAM	1	50	75	100	125
		2	60	90	120	150
#4	ASGD	1	50	75	100	125
		2	60	90	120	150
	ADAM	1	50	75	100	125
		2	60	90	120	150
		3	70	105	140	175
#5	ASGD	1	60	90	120	150
		2	70	105	140	175
	ADAM	1	60	90	120	150
		2	70	105	140	175
		3	80	120	160	200

(b) Dataset sizes for Meta approach.

Variant	VS	S	M	L
#1	90	135	180	225
#2	180	270	360	450
#3	220	330	440	550
#4	290	435	580	725
#5	340	510	680	850

CHAPTER 7 ARTICLE 4: MODELING HIERARCHICAL SPACES: A REVIEW AND UNIFIED FRAMEWORK FOR SURROGATE-BASED ARCHITECTURE DESIGN

The view from Hell is blue sky
So ominously blue
Daydream until all the blue is gone

— Queens of the Stone Age, *Keep Your Eyes Peeled*

Authors. Paul Saves, Edward Hallé-Hannan, Jasper Bussemaker, Youssef Diouane and Nathalie Bartoli

Submitted on August 4, 2025; accepted on January 3, 2026 in *Structural and Multidisciplinary Optimization* (see doi:10.1007/s00158-026-04249-2)

Abstract. Simulation-based problems involving mixed-variable inputs frequently feature domains that are hierarchical, conditional, heterogeneous, or tree-structured. These characteristics pose challenges for data representation, modeling, and optimization. This paper reviews extensive literature on these structured input spaces and proposes a unified framework that generalizes existing approaches. In this framework, input variables may be continuous, integer, or categorical. A variable is described as meta if its value governs the presence of other decreed variables, enabling the modeling of conditional and hierarchical structures. We further introduce the concept of partially decreed variables, whose activation depends on contextual conditions. To capture these inter-variable hierarchical relationships, we introduce design space graphs, combining principles from feature modeling and graph theory. This allows the definition of general hierarchical domains suitable for describing complex system architectures. Our framework defines hierarchical distances and kernels to enable surrogate modeling and optimization on hierarchical domains. We demonstrate its effectiveness on complex system design problems, including a neural network and a green-aircraft case study. Our methods are available in the open-source Surrogate Modeling Toolbox (SMT 2.0).

Keywords. Numerical modeling, feature model, hierarchical domain, meta variables, mixed variables, design space graph, Gaussian process

Nomenclature

Abbreviation	Meaning
ADSG	Architecture Design Space Graph
BO	Bayesian Optimization
cat	Categorical variable
cont	Continuous variable
CR	Continuous Relaxation
DAG	Directed Acyclic Graph
dec	Decreed variable
DoE	Design of Experiments
DOT	DAG of tomorrow
DRAGON	Distributed fans Research Aircraft with electric Generators by ONERA
DSG	Design Space Graph
EI	Expected Improvement
EXC	Excluded
EMI	Earth Mover Intersection
FAST-OAD	Future Aircraft Sizing Tool with Overall Aircraft Design
FM	Feature Model
GED	Graph Edit Distance
GD	Gower Distance
HPO	Hyper-Parameters Optimization
GP	Gaussian Process
HH	Homoscedastic Hypersphere
int	Integer variable
KPLS	Kriging with Partial Least Squares
LCA	Lowest Common Ancestor
LHS	Latin Hypercube Sampling
met	Meta variable
MAP	Maximum *A Posteriori*
MLP	Multi-Layer Perceptron
MMD	Maximum Mean Discrepancy
NSGA2	Non-dominated Sorting Genetic Algorithm II
ord	Ordinal variable
PLS	Partial Least Squares
qtn	Quantitative variable
SBArchOpt	Surrogate-based Architecture Optimizer
SEGOMOE	Super Efficient Global Optimization with Mixture of Experts
SMT	Surrogate Modeling Toolbox
SPD	Symmetric Positive Definite
SVM	Singular Vector Machine
UCB	Upper Confidence Bound
WB2S	Watson and Barnes 2^{nd}-criterion Scaled

7.1 Introduction

Optimizing the architecture of complex systems, such as those related to aerospace design, requires the exploration of vast and highly intricate design spaces to identify optimal solutions [61,72]. These design processes often depend on *expensive-to-evaluate* computer simulations, which are used to explore innovative concepts in early-stage studies and analyses [172]. Given input values, these computer models output key performance metrics, such as the aerodynamic efficiency of an aircraft. Many of such models could be opaque, expensive-to-evaluate, or derivative-free and therefore are treated as *blackbox* functions [26]. The design and optimization of such complex systems is further complicated by the large number of variables influencing their overall performance and feasibility. As system complexity grows, traditional methods for exploring design spaces become less practical due to the high computational cost and inherent complexity of the data structures involved [172, 236]. To address the inefficiencies of directly evaluating such simulations, the optimization process increasingly leverages *surrogate models*, which provide inexpensive approaches to assess the expensive objective or constraint functions. These surrogate models enable the efficient exploration of complex and high-dimensional design spaces, making them indispensable for the optimization of complex systems [172].

One of the critical challenges in this domain is the structured *heterogeneity* of the design spaces, characterized by diverse input data types and complex interdependencies. Real-world optimization problems may involve a combination of continuous, integer, and categorical variables. Moreover, some variables, referred to as *decreed*, may be conditionally included or excluded depending on the state of other *meta* variables, thereby introducing hierarchical structure into the design space [125]. Consequently, such mixed-variable domains pose significant challenges that traditional optimization and modeling techniques often struggle to capture in a scalable and efficient manner, especially when the design space is structured and the relationships between variables are non-trivial [46, 131]. Although the literature offers several approaches for tackling these challenges, ranging from Bayesian optimization and surrogate modeling to feature-based methods, no single solution has been consistently outperforming [248].

As a result, the field lacks a comprehensive framework that can unify these varied techniques and address the full spectrum of mixed-variable and hierarchical optimization problems encountered in complex system design. This paper also addresses a critical gap between the SMT library [52] and the Design Space Graph (DSG) software (adsg-core) [60, 62], which is essential for fully integrating hierarchical surrogate modeling into the existing ecosystem. Our goal is to define distances and models for heterogeneous data points in mixed-variable

domains, where two points do not necessarily share the same variables or bounds [125]. Such a domain is equipped with a single graph whose purpose is to model the complex hierarchical dependencies. Although closely related, the proposed framework does not compare graphs, such as in graph matching. The framework focuses on the variables that can be present across the different data configurations. However, note that graph distances are covered in the literature review, since some notions of these are integrated in the framework, and they share similar purposes. In consequence, our framework provides a consistent mathematical foundation for representing and comparing hierarchical, conditional, and heterogeneous variable domains, enabling their direct use within surrogate models such as Gaussian processes, inverse distance weighting, or k-nearest neighbors. It can also structure the input space of original simulations, serving as a formal interface between architecture definition and model evaluation.

This work is motivated by various real-world applications where meta variables play a pivotal role. For example, thermal insulation design involves meta variables that dictate the number of heat intercepts, each introducing new design variables [3, 155]. Similarly, in aerospace design, meta variables influence critical decisions such as the number of engine shafts or the inclusion of a fan [40, 63, 66]. We position this work as a unified representation and modeling framework for mixed-variable, variable-size design spaces, primarily intended to support surrogate-based optimization of expensive, derivative-free simulators. While model-agnostic, the framework is not intended to replace methods that operate on completed graph objects (for example, graph-matching or specialized graph kernels) when structural isomorphism is the central modeling requirement; in such cases, complementary graph-centric techniques are preferable [69].

The general context of this research lies in system architecture, which defines how systems meet stakeholder requirements by specifying components, functions, and their interrelationships [88]. Designing system architectures involves complex decision-making, requiring cross-disciplinary collaboration and effective management of requirements and functions [72, 215]. Our approach is validated through a case study on green aircraft design optimization using Bayesian Optimization (BO) techniques, demonstrating the framework's ability to manage hierarchical dependencies and complex variable structures. This paper builds upon prior work connecting hierarchical variable frameworks with surrogate-based modeling and optimization [66, 125, 223], aiming to integrate these frameworks within more general system architecture environments [61, 62, 73]. Importantly, the proposed distance of [125] is not merely a technical device but a practical enabler for surrogate modeling and optimization in mixed-variable, hierarchical domains. It provides a principled way to compare two heterogeneous design points that may not share the same active variables. Because of this,

whole datasets can be used without partitioning by active-variable pattern. The distance also permits construction of valid correlation kernels for Gaussian-process surrogates over variable-size domains. This enables information transfer across subspaces, improving sample efficiency and making surrogate-based optimization (e.g., Bayesian optimization) more reliable. The distance was developed and validated in our prior work [125], where additional experiments and implementation details are provided.

This paper provides a comprehensive review of multiple research domains, identifying critical intersections and synergies in the existing literature. Then, this paper proposes a unified framework for surrogate modeling in architecture optimization, building on these findings. Leveraging *graph theory* concepts, its goal is to extend existing concepts through a generalized approach capable of handling mixed-variable and hierarchical blackbox functions and simulations. To finish, this paper delivers the first open-source implementation of the developed framework, enabling the resolution of diverse heterogeneous modeling problems in practical applications. Our main contributions are:

1. Conduct an extensive review of graph kernels and hierarchical variable models to contextualize their applications and to showcase their applicability in managing hierarchical dependencies and complex variable structures, particularly for defining distances or correlation kernels.
2. Propose a unified mathematical framework enabling more expressive and flexible modeling of dependencies to integrate them within surrogate modeling workflows. This generalization provides a robust foundation for addressing hierarchical and conditional variable relationships. A distance is proposed to compare two hierarchical input points and build a correlation kernel for surrogate modeling.
3. Implement this framework in the open-source SMT toolbox [226], extending its capabilities to support hierarchical surrogate models. This implementation is documented in a dedicated tutorial notebook[1], easing its practical application.
4. Showing the application of our method on two different test cases: (i) a neural network hyperparameter modeling problem to illustrate the design space modeling capabilities and (ii) a green aircraft multidisciplinary problem to illustrate the surrogate model capabilities for optimization purposes.

The remainder of this document is structured as follows. Section 7.2 reviews related work, focusing on surrogate modeling in variable-size spaces, representations of structured data,

[1] https://github.com/SMTorg/smt-design-space-ext/blob/main/tutorial/SMT_DesignSpace_example.ipynb

and general distance measures over such structures. Section 7.3 introduces our extended framework, detailing the hierarchical distance and kernel, the automation of the underlying graph construction, and demonstrating its use through a case study on multi-layer perceptron architectures. Section 7.4 applies the framework to a neural network model and extends it to surrogate emulation for the Bayesian optimization of a green aircraft. Finally, Section 7.5 summarizes our contributions and outlines directions for future research.

7.2 Literature review

This section provides a literature review of related works and previous studies. Specifically, it reviews key approaches to handling variable-size spaces, data representations, and distance measures over complex structures.

7.2.1 Background and previous works

This section describes meta and meta-decreed variables, which model hierarchical dependencies in domains of interest, as introduced in previous works [25, 124] and [125], respectively. This framework has been formally analyzed through a so-called design space graph introduced for an application to hybrid-electric distributed propulsion aircraft [225] and compared with the classical Feature Model (FM) [43].

Types and roles of variables

For a given problem featuring n variables, a point $\mathbf{X}$ can be decomposed as $\mathbf{X} = (X_1, \ldots, X_n)$, where X_i is the value taken by the point $\mathbf{X}$ in its corresponding i^{th} variable. To incorporate structure into the problem, it is necessary to establish a terminology that describes the types and roles of variables. Each variable is assigned a *type* that reflects its quantitative or qualitative nature [66, 224], four types that can be assigned to a variable:

- continuous, if the variable can take any real value between lower and upper bounds;
- integer, if the variable can take any integer value between lower and upper bounds;
- ordinal, if the variable can take any value in an ordered finite set of either strings or values;
- categorical, if the variable can take any value in an unordered finite set of either strings, modalities, or values.

Concerning variables that take values in a finite set—namely *integer*, *ordinal*, and *categorical* variables—the key distinction is the presence (or absence) of an intrinsic ordering among their levels. Categorical (nominal) variables have *no* natural order: for example, a variable *shape* with levels {square, circle, star} is unordered. By contrast, integer and ordinal variables are ordered. An integer variable carries a known, equally spaced order (*e.g.* {1, 2, 3}), whereas an ordinal variable admits an order but the spacing between successive levels need not be equal or even known (for instance {1, 2, 4} has non-uniform spacing, while {small, medium, large} is ordered but the distances between levels are qualitative). When the metric structure of an ordinal scale is unknown, one may learn a monotone embedding or distance function from data; for example, the authors of [216] propose methods to estimate a strictly monotone mapping for ordinal levels.

In addition to their variable type, each variable is assigned a *role* that reflects its hierarchy with other variables in points of the hierarchical domain [25, 125]. An underlying graph structure models the hierarchy between variables through parent-child relations. In this graph, a variable x_i is a parent of another variable x_j (a child), if x_i controls the inclusion or bounds of x_j, with $i, j \in \{1, 2, \ldots, n\}$. In such cases, x_j is said to have a *decree dependency* with x_i, and an arc from x_i to x_j is present in the graph. This arc represents the decree dependency. More details on the graph structure are provided in the next section. The roles of the variables, *variable roles* for short, are determined based on whether they have parents and/or children in the graph. Each variable is assigned one of the following four roles:

- (strictly) meta, if it has no parent and at least one child;
- meta-decreed, if it has at least one parent and at least one child;
- (strictly) decreed, if it has at least one parent and no children;
- neutral, it has neither parents nor children.

This taxonomy with four roles (meta, meta-decreed, decreed, neutral) captures the structural position of each input within the role graph and has direct modeling consequences. Intuitively, a *meta* variable is a top-level controller (no parent, at least one child) whose values determine which downstream variables are admissible; a *decreed* variable is a child-only input (at least one parent, no children) whose admissible set depends entirely on its parents; a *meta-decreed* variable plays both roles (it is controlled by parents and in turn controls children); and a *neutral* variable is independent of the hierarchical structure (no parents, no children). Concretely, these roles determine whether a variable can be included or excluded. Role assignment also guides dimensionality reduction and problem decomposition (for example, excluded decreed variables are omitted from surrogate evaluations and acquisition

computations), and it supports interpretable modeling choices that respect the semantics provided by domain experts [25, 125]. Notwithstanding, this work is limited as it does not consider partially decreed variables, that is, the variables that are included but can only take a subset of their admissible values. Also, this previous work cannot handle incompatibilities properly, and the next sections will show how we can extend the previous work to generalize it.

Modeling Design Spaces featuring Graph Structures

This section presents references addressing the general class of problems that involve hierarchical spaces in the context of data-driven regression and optimization. In this study, a hierarchical space, originally introduced in [135], is defined as a space with at least one meta variable, following the formalization in [125]. A hierarchical space may also be referred to as a variable-size space [196], as a conditional search space problem [38], as a heterogeneous dataset [208], or as a hierarchical domain [125]. More details about other data structure representations are given in Appendix A.

Decreed and meta-decreed variables may have multiple parents, and both types can influence numerous other variables. Building on these observations, we introduce the role graph that is a structure that captures all information about variable roles and decree dependencies. Formally, the role graph $G = (N, A)$ is a couple, where N is the set of all the variables represented as nodes of the problems and A is the set of decree dependencies represented as arcs.

In [125], a general class of mixed-variable hierarchical structures within domains was introduced. This work rigorously defines a hierarchical domain, denoted as $\mathcal{X}_G$, that represents a domain $\mathcal{X}$ equipped with a graph structure G. In this domain, each point $\mathbf{X}$ is properly defined within $\mathcal{X}$, and the variables are hierarchically ordered in G, where a variable is a child of another if it is decreed by it. Equivalently, the domain is *hierarchical* if its corresponding role graph contains at least one decreed variable (*i.e.*, a variable with the decree property). Additionally, the hierarchical domain framework introduced in [125] models mixed-variable domains with meta variables, implying that points do not necessarily share the variables or bounds. This hierarchical domain $\mathcal{X}_G$ can be extended to contain the variables that can be considered excluded, *i.e.*, the variables decreed excluded due to the values of the corresponding meta variables at a given point.

This extension yields a space, called the *extended domain*, containing all the possible variables that are present in at least one point. The role graph contains the decree dependencies and allows for conditioning universal sets into restricted sets. It is assumed to be directed and

acyclic to avoid pathological situations where meta-decreed variables influence each other. Nevertheless, the Design Space Graph (DSG) addresses this issue by parsing the design variables through a recursive process, stepping through the included (but not-yet-parsed) choices. Typically, parsers process design variables in a sequential, left-to-right manner. However, due to the complex interdependencies among design choices in systems engineering like Architecture Design Space Graph (ADSG), a simple sequential parsing approach may not suffice. To address this, the DSG employs a recursive parsing strategy. This method allows the parser to navigate through the network of design variables, processing each based on its dependencies rather than its position in a sequence. This recursive approach ensures that all design variables, including those not initially parsed due to their position or dependencies, are eventually analyzed, leading to a more comprehensive understanding of the system's design structure [60].

The `DRAGON` aircraft concept [229] relies on a distributed electric propulsion aircraft, improving fuel efficiency by optimizing propulsive performance. `DRAGON` introduces multiple compact electric fans on the wing pressure side, increasing the bypass ratio as an alternative to large turbofans. This design overcomes challenges of under-wing turbofans while enabling transonic flight. In such aircraft design examples, a meta-decreed variable could be the number of motors that is used for a given wing, where each motor has a set of decreed variables. Figure 7.1 illustrates the roles of variables for a toy example of a wing aircraft design with meta-decreed variables. The number of motors per wing m is a meta variable, since it controls how many variables, peculiar to each motor, are in the problem ($m = 0$ means single-engined aircraft, as the engine is not supported by the wing). The number of propellers p_i assigned to the i-th motor is a meta-decreed variable since 1) its inclusion depends on the number of motors m, *i.e.*, it is decreed, and 2) it controls the number of variables regarding the radii of every included propeller. Finally, the radius r_{ij} of the j-th propeller in the i-th motor is a decreed variable, since its inclusion depends on the number of propellers p_i for the motor i. In Figure 7.1, to compute the meta distance, we consider the extended points that contain all included and excluded variables. The variables that are excluded for a given point are assigned the special value `EXC`. In other words, some variables may be excluded for a point $\mathbf{X} \in \mathcal{X}$ but present in another point $\mathbf{X}'$. For example, in Figure 7.1, the variable representing the number of propellers of the second motor is set to $m_2 = \text{EXC}$ when the number m of motors is one. Considering the excluded variables provides the necessary information for computing distances between points that do not share the same variables. This trick is always possible by adding bounds for every variable: for example, if we only have data for an aircraft with 2 motors and 2 propellers, we can always restrict the extended design space without loss of generality. Consequently, the extended domain, in which extended points reside, is

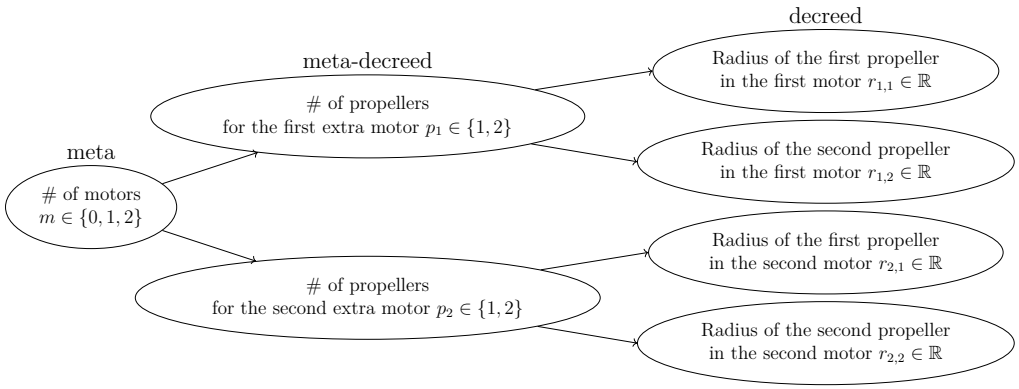


Figure 7.1 Variables roles for aircraft design problem.

constructed directly with Cartesian products on the bounded restricted sets of each variable. The hierarchical distance is defined on the extended domain, but fortunately, a bijection based on the role graph G enables the mapping of points to extended points by adding `EXC` to the excluded variables, and conversely, maps extended points back to original points by removing variables assigned the value `EXC`. Thus, a distance function that considers all included and excluded variables is also induced on the original domain within the restricted set. For example, one such distance combines angular and length information, and is well-defined for heterogeneous data comparison [125]. As such, it has the property of leading to unit induced distances when comparing an included variable's value with a non-included variable whose value is `EXC`. This unit distance is effective for multiplication-based comparisons, as it will be non-influential if multiplying the remaining distances by 1.

Hierarchical spaces with feature models

To describe complex configuration spaces—where selecting a high-level option determines the availability of related sub-options—we rely on *feature models*, a formalism widely used in software engineering [56]. In this context, a *feature* represents a configurable aspect of a system, such as a module, parameter, or capability that can be enabled or disabled. This directly corresponds to the notion of variables in hierarchical modeling, where the activation of one variable governs the inclusion or structure of others [43]. For example, in aircraft design, the feature "Energy source" might offer alternatives such as "Electricity" or "Fuel", each unlocking its own specific set of design parameters. In this sense, features may correspond to variable levels, variables themselves, or even the domains (supports) of those variables. While feature models often define these roles loosely, we adopt a more structured interpretation in this work by treating features as the *nodes* of a DSG, a formalism we build upon in subsequent

sections [17, 62]. In a feature model, features are organized via a tree structure, where nodes represent features and edges define dependencies. A child feature is only accessible if its parent is selected. Some are *mandatory*—always included when the parent is included—while others are *optional*. Groups of child features can also form exclusive (XOR) or inclusive (OR) sets, expressing constraints like "select exactly one" or "select at least one." In addition to the tree, *cross-tree constraints* allow logical rules between features in different branches. These include relations such as "A requires B" or "A excludes B", as well as more complex logical relationships, as in "A and B implies not C" [41]. Feature models provide a compact way to describe rich, structured design spaces without enumerating every valid configuration. In our setting, they naturally map to hierarchical variable spaces, where high-level meta-choices conditionally activate relevant subsets of parameters.

7.2.2 Hierarchical surrogate modeling

This section first reviews surrogate modeling strategies for representing architectural decision variables. It then tackles the core challenge of defining valid distance metrics in these complex, non-standard design spaces.

The motivation for developing a hierarchical surrogate arises from the need to accurately model and optimize complex engineering architectures such as hybrid-electric propulsion systems, where multiple physical domains (mechanical, electrical, and thermal) interact. Traditional flat representations fail to capture these nested dependencies and conditional relationships between subsystems. As illustrated in Fig. 7.2, the authors applied a function-based architecture modeling approach to the design of a hybrid-electric propulsion system for the Beechcraft King Air C90GT aircraft. Their framework systematically generated and evaluated thousands of propulsion architectures under mission constraints (*e.g.*, climb, cruise, descent) and subsystem-level design variables (*e.g.*, motor rating, battery mass, gearbox ratio). Such systems naturally combine discrete architectural choices like power-split topology and component inclusion with continuous design variables, leading to heterogeneous and nested design spaces that challenge traditional surrogate modeling. The present work therefore aims to construct surrogate models and distance metrics that preserve this hierarchical structure, enabling efficient learning, comparison, and optimization across large and multi-level configuration spaces [65].

Hierarchical design spaces for surrogate modeling

Interest in modeling "variable-size" spaces emerged from various disciplines around 2010. For example, the work by [134] introduced Latin Hypercube Sampling (LHS) and Gaussian

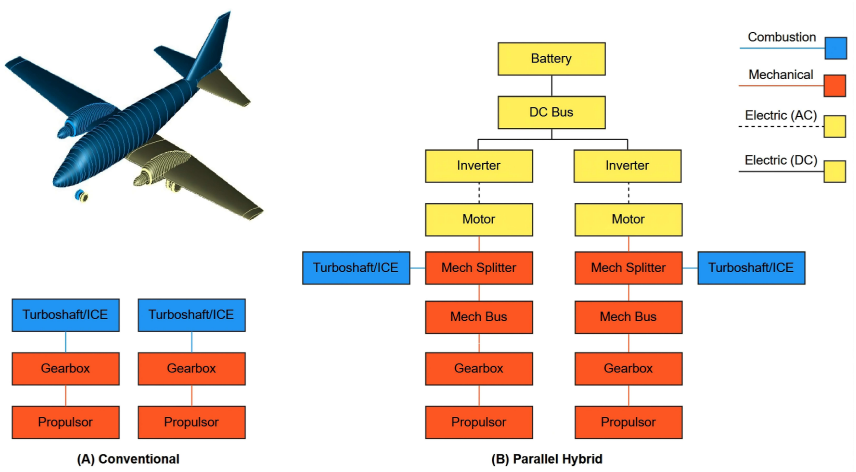


Figure 7.2 An example of hierarchical surrogate in the context of aircraft design. Adapted from [65, Figure 4].

process (or Kriging) for *nested* variables, where variables depend hierarchically on other variables. Although the mechanisms differ, these nested variables can be considered akin to decreed variables in this context. Similarly, the technical report [135] from 2013 proposes a mixed-variable kernel function for varying-size spaces, referred to as hierarchical. The latter report has been a key reference for the previous work [125]; thus, it is also important for this current work. In this report [135], the inclusion and exclusion of a variable is controlled by variable-wise Kronecker delta functions. This variable-wise Kronecker delta denotes a per-variable indicator of mutual inclusion: only variables active in both sample points contribute to their kernel term. This idea resonates with the kernel designs in variable-size Bayesian optimization [195], which allow covariance across samples with partially distinct dimensional supports [196]. More precisely, the domain is equipped with a Directed Acyclic Graph (DAG) structure that contains hierarchical dependencies between the variables and where the variables are viewed as nodes. The inclusion of a (child) variable is determined by a delta function that takes the values of its ancestors, that is, variables of higher hierarchy that share a path in the DAG. The root and intermediate nodes are restricted to categorical variables. The kernel is constructed variable-wise with multiplication and addition of one-dimensional kernel. Each one-dimensional kernel is computed with an assigned isometry, which is a distance-preserving function that maps a variable into a Euclidean space [101]. The computation of a one-dimensional kernel is divided into three cases: the variable is included for both points or the variable is excluded for both points, or the variable is included for one and only one point.

In a Bayesian optimization context, the paper [196] proposes kernel functions that allow

the construction of a GP on domains with dimensional variables [166, 167], which are a special case of strictly discrete meta variables that influence the dimension, and propose two kernels. First, the subproblem-wise kernel computes similarity measures between dimensional variables, as well as between the other included variables, whose inclusion can be determined by the dimensional variables. Second, the dimensional variable-wise kernel decomposes the computation dimensional variable per dimensional variable, which allows for regrouping the specific decreed variables that are included by their corresponding dimensional variables [134]. For SVM regression, the work [111] proposes the Earth Mover's Intersection (EMI or discrete Wasserstein) that computes similarity measures between sets of different sizes, similarly to the Jaccard index [250]. A kernel function is induced from the EMI, and SVM regression experiments are realized.

Going back to GP regression for hierarchical design spaces, defining valid and efficient covariance kernels over unordered sets of vectors presents unique challenges. Traditional kernels, like the squared exponential kernel, often fall short when applied to sets with no inherent ordering and varying cardinalities. Methods such as the Maximum Mean Discrepancy (MMD) and the Wasserstein distance offer promising solutions by treating these sets as samples from distributions, enabling the comparison of sets with different sizes and structures [100]. For instance, the MMD-based kernel has shown strong performance in adapting to the geometric properties of underlying functions, making it a robust choice for surrogate modeling in complex engineering problems [240]. However, such methods are not structure-specific and may struggle with high-dimensional data or scenarios with significant missing information, underscoring the need for more specialized kernels that can effectively capture the underlying structure and relationships within the data.

In the complex systems architecting context, the works [60, 62, 64] detail the Architecture Design Space Graph (ADSG) and its application for modeling complex architectural design spaces with a semantic approach. In this approach, three types of architectural decisions are available: function-component mapping, component characterization, and component connection. The graph model is constructed from a design space definition, and discrete architectural decisions are automatically inserted according to specified rules. The DSG abstracts away from the system architecture context, and uses a directed graph to model general-purpose hierarchical selection choices (*i.e.*, selections between mutually-exclusive options) and connection choices (*i.e.*, connection patterns between sets of source and target nodes). The current work integrates the architectural decisions of the DSG, notably for the automatic construction of the domain in the software part of this work. However, the DSG supports cyclic graphs and therefore includes, but is not limited to, DAGs. A notable tool in this domain is `ConfigSpace` [165], an open-source Python library that facilitates the definition

and management of hierarchical design spaces. `ConfigSpace` supports mixed-discrete variables, conditional activation of design variables, and value constraints. It allows for querying and validating design vectors, correcting and imputing design vectors, and generating valid design vectors. It is widely used by surrogate modeling and optimization frameworks such as SMAC3 [165], BOAH [164], OpenBox [139], and SMT [226]. Even if explicit hierarchical models are not available, problems often contain some implicit hierarchical structure [67]. Various design space modeling techniques, including the Architecture Decision Graph [236], the Adaptive Reconfigurable Matrix of Alternatives [173], or the aforementioned DSG, to cite a few, provide useful frameworks for understanding and optimizing complex systems. More relevant background on data representations within structured spaces is given in Appedix A.

General distances over structures

One of the objectives of this paper is to propose a distance to compare two hierarchical input points and to build a correlation kernel on top of such a distance. The goal of this section is to give more insights into these particular domains.

To begin with, recall that a function d is called a distance if and only if d respects the four criteria of positivity (1), identity (2), symmetry (3), and the triangular inequality (4).

In particular, when a function only respects the points 1, 3, and 4, it is called a pseudo-distance, and when it only respects the points 1, 2, and 4, it is called a quasi-distance [96].

Another closely related concept is the similarity s [116] defined as a function that respects the boundness (instead of positivity), identity, symmetry, and respects the triangular inequality. However, several authors do not include the symmetry in the definition of the similarity measure [253], while some authors add the positivity but remove the triangular inequality [188]. Furthermore, some authors introduce dissimilarity measures that respect only positivity and reflexivity (instead of identity) and divergence measures that respect both positivity and identity [91]. Therefore, distances must be preferred over similarity distances. Most definitions of similarity measures include correlation, and, for example, Pearson's R correlation [54] is bounded, symmetric, and respects the identity, but not the positivity and the triangular inequality [75]. This is the case with most correlation similarities (*e.g.*, distance correlation, Rho-vectorial coefficient [211], Brownian covariance [247], polychloric correlations [83]).

Both distances and similarity measures can be distinguished according to their deterministic or stochastic nature. For example, distances between vectors include Minkowski and Mahalanobis distances, generalizing Euclidean distances [57] in, respectively, a deterministic and a stochastic setup. General objects could also be compared with respect to the precise features

they have. For example, the cosine similarity (Otsuka–Ochiai coefficient) [243] is a measure between vectors with respect to the angle that they form. Distances tend to form families of distance and to rely on parameters and thus are connected to the fields of hyperparameter optimization, feature selection, and metric learning. More details about common distances are given hereinafter.

In the previous section, we mentioned the EMI [111] measure that extends the Jaccard similarity index, and the Tanimoto measure [108,113] to compare generic variable-size vectors or deterministic sets [254]. Similar literature worth noting includes several basic distances over non-structured set. Among others, we can cite the Hamming distance [127], Morisita's overlap index [68], Szymkiewicz–Simpson coefficient [218], Renkonen similarity index [209], Dice-Sørensen index [239], Lee-Mannheim distance [184] or Tversky index [253].

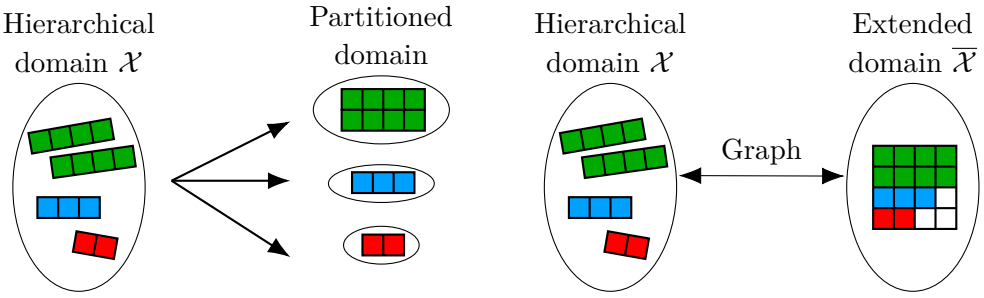
Distances between strings or sequences can be measured using various probabilistic or deterministic methods. For time-series, deterministic approaches include resampling followed by Minkowski or cosine distances [152], frequency-based features like Fourier transforms [6], and elastic measures such as time-warped edit distance [170]. Stochastic approaches often rely on auto-regressive models [231], such as using Kullback-Leibler divergence between Markov chains as a Bayesian similarity measure [205], or the Wasserstein-Fourier distance for stationary series [70]. For ordered data, rank-based metrics like Kendall's Tau and Spearman's Rho [169] compare element ranks instead of values. More generally, information-based measures like Kolmogorov complexity can be approximated by the normalized compression distance [81]. Simple distances can also be extended to hierarchical data, from sets [36] to more complex structures discussed in the next section.

Graph-based distances to compare structured data

Graph representation is amongst the most popular hierarchical database representations, even if not the only solution to model such hierarchies. However, it is the most appropriate to our context of hierarchical variables for architectural problems (see Appendix A for more background information). In fact, rather than comparing individual decision variables, these methods treat each architecture as a complete graph and quantify similarity through operations such as edge insertion, deletion, and permutation-based alignment of shortest-path or adjacency matrices. By focusing on holistic graph topology, this alternative approach captures structural differences and commonalities in hierarchical designs, complementing variable-level metrics with a well-established graph similarity perspective.

A central difficulty when working with hierarchical, variable-size domains is that two design points need not share the same active variables or the same dimensional support. A com-

mon workaround is to partition the hierarchical domain into homogeneous subspaces and treat each subspace separately; however, this sacrifices statistical efficiency (data is split) and complicates downstream modeling and optimization (Fig. 7.3a). Our objective is instead to place heterogeneous points into a single, *extended* domain and to define a principled distance on that domain so all data can be used jointly (Fig. 7.3b). Concretely, we construct an extended representation that embeds each hierarchical point into a common extended hierarchical space while retaining the notion of a variable's admissible *support* (active, partially active, or excluded). The distance is then computed as a structured aggregation of per-variable contributions that account for the graph-structured space. The latter construction may yield a correlation kernel tailored surrogate models that admit practical scalability strategies when exhaustive pairwise computations become expensive.



(a) **Motivation:** avoid partitioning a hierarchical domain and splitting heterogeneous data.

(b) **Objectives:** define a distance on the extended domain, in which data is aggregated.

Figure 7.3 Motivation for developing a unified distance. Adapted from [125, Figure 2].

Various methods measure similarity or distance between elements in hierarchies and taxonomies, categorized by feature-based [189], morphism-based [257], or structural properties, including deterministic and stochastic distances.

Deterministic distances, like Rada's distance [203], compute the distance in a hierarchy by counting edges up to the Lowest Common Ancestor (LCA). The LCA distance [44] measures the path length between two nodes starting from their common ancestor, with the idea that nodes with a more specific common ancestor are more similar. Path distance also fits into this category, as it measures the length of the path between two nodes, often considering their depth or level in the hierarchy [262]. Rada's distance is defined as:

$$d_{\text{Rada}}(\mathbf{X}, \mathbf{X}') = \text{path}(\mathbf{X}, \text{LCA}(\mathbf{X}, \mathbf{X}')) + \text{path}(\mathbf{X}', \text{LCA}(\mathbf{X}, \mathbf{X}')), \tag{7.1}$$

where $\text{LCA}(\mathbf{X}, \mathbf{Y})$ represents the lowest common ancestor of $\mathbf{x}$ and $\mathbf{y}$ in the hierarchy.

Stochastic distances, such as Resnik's distance [210], introduce the use of information content (IC) to measure similarity, based on the probability of encountering concepts and their hierarchical significance. Resnik's distance is defined as:

$$d_{\text{Resnik}}(\mathbf{X}, \mathbf{X}') = -\log\left(p(\text{LCA}(\mathbf{X}, \mathbf{X}'))\right). \tag{7.2}$$

Hybrid distances, *e.g.*, Jiang-Conrath distance [140], combine edge counting and IC for a more comprehensive measure. Additionally, in the realm of stochastic approaches, Wasserstein-based distances, like the Sliced Wasserstein Weisfeiler-Lehman (SW-WL) kernel [69], provide a powerful framework for measuring structural similarity between meshes. These distances are commonly used in transport theory to compare distributions over graph features. These approaches are also useful when working with non-attributed graphs and enable the efficient computation of graph distances using multi-level refinements of graph structure [183].

Graphs are powerful tools for representing hierarchical structures, especially for heterogeneous data. Graph Edit Distance (GED) [109] is widely used to quantify the similarity between two graphs by calculating the minimal cost to transform one graph into another through edit operations like node and edge insertions, deletions, or substitutions. GED generalizes several similarity measures, such as Hamming and Jaro-Winkler distances [261]. It satisfies properties like non-negativity, identity, symmetry, and the triangle inequality [232]. The computation of GED depends on the type of graph considered. For attributed graphs, similarity is based on node and edge attributes [50], with approaches including self-organizing maps [181], and graph kernels [182]. For non-attributed graphs like directed acyclic graphs or trees, GED can be computed using string-based methods, such as tree edit distance, which takes advantage of polynomial-time tree isomorphism problems [103]. Methods like Dijkstra's algorithm [213] are often more efficient for these cases. These methods convert graphs into strings and compute the shortest edit path using algorithms such as Dijkstra's or maximum *a posteriori* (MAP) [212], which avoid matrix normalization to reduce complexity.

In addition to graph edit distance, one can measure similarity by aligning global matrix representations of the graphs. Let $A_{\mathbf{X}}$ and $A_{\mathbf{X}'}$ be the adjacency matrices of graphs $G_{\mathbf{X}}$ and $G_{\mathbf{X}'}$, respectively, and let $D_{\mathbf{X}}$ and $D_{\mathbf{X}'}$ denote their corresponding shortest-path distance matrices. A family of permutation-based distances seeks a permutation matrix P that best aligns these representations under the Frobenius norm. For instance, the *chemical distance* [157] is

$$d_{\text{Chem}}^{P_n}(G_{\mathbf{X}}, G_{\mathbf{X}'}) \;=\; \min_{P \in P_n} \| A_{\mathbf{X}}\, P - P\, A_{\mathbf{X}'} \|_F, \tag{7.3}$$

where P_n is the set of all $n \times n$ permutation matrices over the n vertices of the considered

graphs and $\|\cdot\|_F$ denotes the Frobenius norm, selecting the node alignment that minimizes adjacency discrepancies. Likewise, the Chartrand–Kubiki–Shultz (CKS) distance [74] applies the same principle to shortest-path information,

$$d_{\mathrm{CKS}}^{P_n}(G_{\mathbf{X}}, G_{\mathbf{X}'}) \;=\; \min_{P \in P_n} \| D_{\mathbf{X}}\, P - P\, D_{\mathbf{X}'} \|_F, \tag{7.4}$$

emphasizing global connectivity differences. Other related measures include Poole's most interesting common generalization distance [198], which abstracts both graphs to their least general common ancestor before measuring edit cost, and the Champin–Solnon mapping cost [71], which allows flexible (non-bijective) vertex correspondences via a cost matrix. Wang and Ishii's formulation [259] unifies both adjacency- and path-based distances under the same permutation framework, highlighting the versatility of Frobenius norm alignment for capturing holistic graph similarity. Still, these distances are related to the graph isomorphism problem [120], making their computation challenging.

This review introduced key concepts and relevant works, identifying a critical gap: the lack of a unified framework or software for handling mixed-variable and hierarchical challenges in complex system design, such as architecture design. The following section shows how to extend and generalize further the frameworks introduced in this section with application to surrogate modeling.

7.3 A unified framework for hierarchical design spaces

Building on the literature reviewed in Section 7.2, our framework unifies feature models, mixed-variable kernels, and graph-based similarity. It uses a design space graph representation that encodes conditional inclusion and partially-decreed relationships. The representation supports continuous, integer, and categorical data and yields distances and kernels tailored for surrogate modeling. In that sense, the present work generalizes prior mixed-variable and hierarchical kernels, *e.g.*, imputation and role-agnostic strategies, while remaining complementary to full graph-matching and graph-kernel methods when explicit topology matching is required, *e.g.*, Gaussian process over meshes [69].

This section presents a unified graph-based framework to model structured design spaces with mixed variables, hierarchical dependencies, and custom distances, enabling the handling of structured data representations in architectural modeling and optimization [47, 62]. Specifically, excluded variables offer valuable information for computing similarity measures between points that do not share the same variables. This approach enables hierarchical and mixed discrete variables to be effectively incorporated into a correlation kernel or distance

metric, facilitating comparisons between architectural configurations. Such correlation kernels belong to the family of graph-induced kernels [204], which are frequently employed in Bayesian Optimization (BO) [163, 234] for heterogeneous datasets, making them a powerful tool for managing complex, structured design spaces [149].

7.3.1 Adding relationships in-between variables

In this section, we introduce the new relationships between the variables and their levels in the graph-structured design space, that is a hierarchical domain equipped with a graph structure modeling the hierarchical dependencies between the variables. In particular, we introduce these three new relationships.

Decree dependencies This is the most general meta to decreed relationship between any two variables, capturing multi-level and nested inclusion or exclusion dependencies as detailed in Section 7.2.1. To be more precise, we introduce *partially-decreed* variables for when a parent does not fully enable or disable its child, but rather restricts the child's admissible values without entirely excluding it. For the toy example in Figure 7.4, the number of motors per wing m is a meta variable parent of the number of propellers p_1 and p_2, since it controls their inclusion or exclusion. The variable m is also a meta variable parent of the length of wings l since it determines its admissible values. The variable l is partially-decreed, since it is always included, but its admissible values are controlled by m.

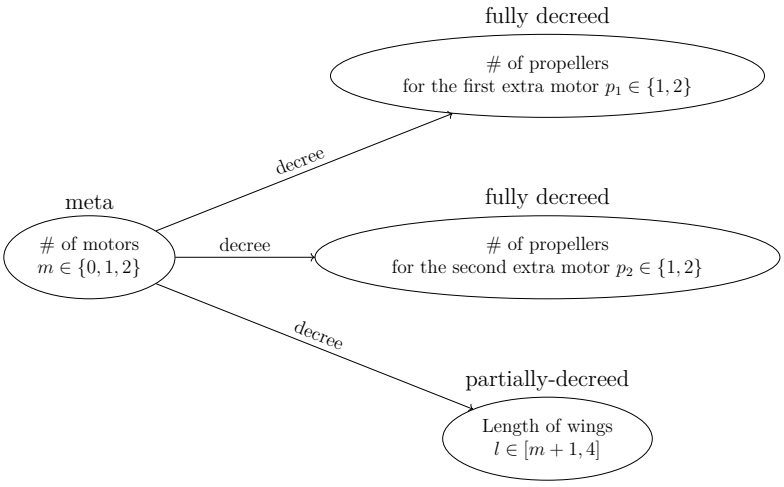


Figure 7.4 Variables roles for an aircraft design problem with a varying number of motors.

Incompatibility An incompatibility relationship, similar to the "Excludes" constraint in feature models (Section 7.2.1), prevents two variables from taking certain combinations of values. For instance, a specific battery type may be incompatible with high fuel levels for hybrid energy due to safety concerns, as in Figure 7.5. *Mandatory* constraints can also be captured indirectly: if selecting a value excludes all but one option of another variable, the remaining option becomes implicitly required, effectively modeling a "Requires" relationship.

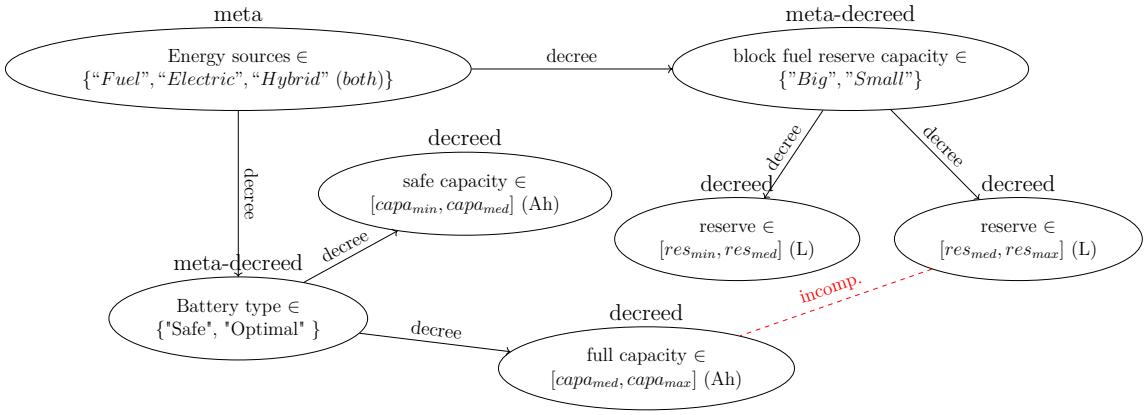


Figure 7.5 Variables roles for an aircraft design problem with a varying energy source.

Order relationship For continuous variables, whose supports cannot be enumerated, incompatibilities are expressed through order relations (*e.g.*, inequalities). Equality constraints can, similarly to before, be constructed from the conjunction of two opposite inequalities, enabling *Mandatory* relationships in a continuous setting. For instance, in designing a hydrogen tank, if there are three pressure variables such that $P_{min} < P_{input} < P_{max}$ [192], this relationship holds true with the structure of Figure 7.6 structure.

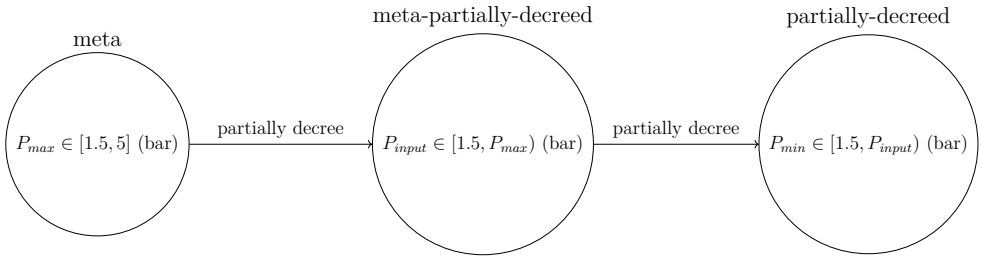


Figure 7.6 Roles of variables for a pressure order example.

We can go even further by mixing categorical variables and partially-decreed continuous bounds. A dummy example inspired by [106] is as follows. The energy used for propulsion

could be either "Fuel", "Electric" or "Hybrid" (both fuel and electricity). If the energy is either electric or hybrid, a back-up battery needs to be sized, but if the energy is hybrid and the fuel reserve is sufficient, no back-up battery is needed. Therefore, the back-up battery capacity is a decreed variable that is included if the energy is electric or if the energy is hybrid but the reserve is too small. To model the "and" interaction, an "intermediate node" as shown in Figure 7.7, needs to be added to the model. The intermediate variable is decreed by the energy variable and is included only if the source is a hybrid electric, and its value is 1 if the fuel reserve is insufficient and 0 otherwise. The backup battery is included if the intermediate value is included and is of value 1 or if the energy source is electric.

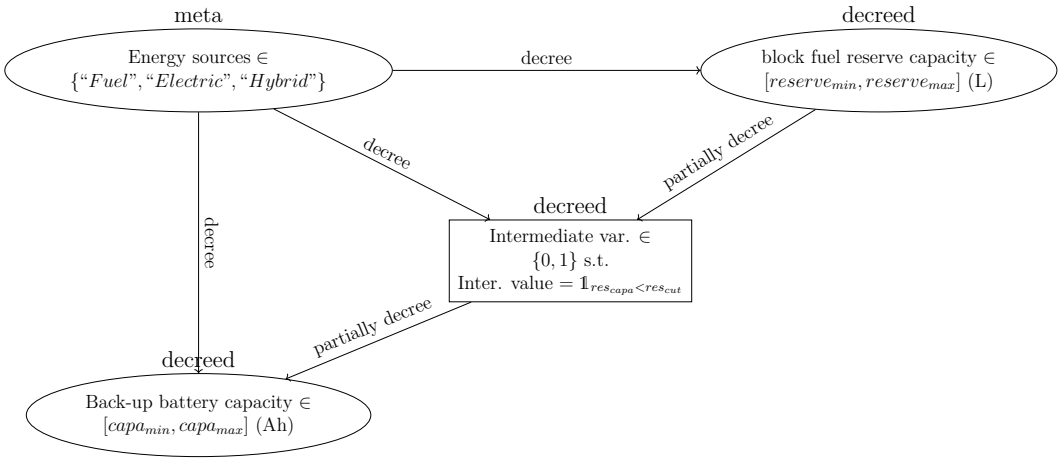


Figure 7.7 Roles of variables for a hybrid-electric technology choice example.

This work extends the notion of variable roles to accommodate a wider range of applications, including aerospace system design [63, 64, 236] and software architecture [10]. Unlike traditional feature models, our representation is a DAG rather than a tree, allowing features to have multiple parents. The framework is more formally described in the following section.

7.3.2 Generalizing the hierarchical definition

Compared to the previous work [125], we extend the graph to include more concepts and have a structure that is more complex but more flexible and more general. This new structure adds the novel concepts developed in the previous section and is partly inspired by feature model and graph theory concepts. Consequently, it is not a plain directed graph (N, A) but a so-called mixed graph (N, A, E), that mixes arcs A and undirected edges E. Notwithstanding, we limit our work to graphs that are directed and acyclic if we do not consider the undirected edges.

Definition 44 (Generalized role graph)**.** *The generalized role graph* $\mathcal{G} = (N, A, E)$ *is a graph structure, where*

- N *is the set of nodes that contains 1) all the included and excluded variables, together with their levels if discrete or bounded supports if continuous, and 2) all intermediate variables,*
- A *is the set of decree dependencies that contains references for all inclusion-exclusion or admissible values dependencies between two nodes, represented as arcs,*
- E *is the set of incompatibilities that contains references for all incompatibilities between two nodes, represented as edges.*

A node $n \in N$*, which refers to any variable or support, can be of three possible kind:* $N = N_{var} \cup N_{levels} \cup N_{bnd} \cup N_{mid}$ *where* N_{var} *is the set of possible variables,* N_{levels} *is the set of the levels of any discrete variable,* N_{bnd} *is the set of continuous bounds for any continuous variable and* N_{mid} *is the set of intermediate nodes modeling composite (e.g. "and/or") activation conditions. Likewise, features in features model in Section 7.2.1, a node is more general and not limited to a variable.*

An arc $a \in A$*, which refers to any dependency, connects a parent (variable) to a child (variable), whose inclusion or admissible values are influenced by the parent or to a child (level/bounds) that is part of the complete support of the variable. We then have* $A = A_{dec\text{-}met} \cup A_{bnd\text{-}lim}$ *where* $A_{dec\text{-}met}$ *is the set of meta/decreed relationships,* $A_{bnd\text{-}lim}$ *is the set of partially meta/decreed relationships.*

An edge $e \in E$ *which refers to any incompatibity, can be of two kinds* $E = E_{opposed} \cup E_{order}$ *where* $E_{opposed}$ *is the set of completely incompatible two nodes and* E_{order} *is the set of order relationships between two variables.*

As in Section 7.2.1, a variable is assigned one type and one role. Their types are continuous (cont), ordinal (ord), integer (int), or categorical (cat) as explained in Section 7.2.1. In particular, ordinal variables are intermediate variables that can be either integer, such as {2,4,8} or categorical, such as {small, medium, high} but with an order relation (high>medium>small) and are therefore quantitative (qnt) like integer or continuous variables. Variables also come with bounds or support/levels definitions. The role is amongst meta (m), meta-decreed (md), decreed (dec) or neutral (neu).

In this framework, each node can have parents that are the other nodes in N that directly influence its activation or admissible values. A node's parents are simply those connected to

it by a directed arc in the graph. These node parents must themselves be included (*i.e.*, not excluded from the design). To handle the heterogeneous design space, we extend each point to include all possible dimensions, enabling direct comparisons between them—a strategy formally validated in [125], which relies on the notion of support as a variable admissible values depend on its parent values. The support, knowing the role graph $\mathcal{G}$, is defined as follows:

Definition 45 (Support)**.** *Let x_i be a variable with full value space $\mathcal{X}_i$ and let $P_i \subset N$ be the set containing its parents in the graph $\mathcal{G}$. The support of x_i, that depends on the values taken by its parents, is written as $\mathcal{S}_\mathcal{G}(x_i \mid P_i) \subseteq \mathcal{X}_i$ or $\mathcal{S}_i \subseteq \mathcal{X}_i$ for short. This support is the set of admissible values that x_i can take under the design space graph $\mathcal{G}$ and the values taken by its parents. A variable is considered* excluded *from the design space when $\mathcal{S}_\mathcal{G}(x_i \mid P_i) = \emptyset$.*

Here, the *full value space* of a variable is the set of all possible values that this variable could take in the absence of any constraints from parent variables. The support $\mathcal{S}_\mathcal{G}(x_i \mid P_i)$ captures the subset of these values that are admissible given the values of its parents in the design space graph $\mathcal{G}$. This distinction allows explicit modeling of excluded or partially-decreed variables. In fact, a variable could be excluded, and in that case its support is the empty set, but it also could be partially included, and in such case its support is a subset of the full support. This leads to the following definition of a partially-decreed variable, a variable that is always present, but whose admissible values are constrained by the values of its parent.

Definition 46 (Partially-decreed variable)**.** *A variable x_i is said to be* partially-decreed *by its parents $x_j \in P_i$, if*

$$\mathcal{S}_\mathcal{G}(x_i \mid x_j) \subset \mathcal{X}_i \quad \textit{and} \quad \mathcal{S}_\mathcal{G}(x_i \mid x_j) \neq \emptyset.$$

This generalizes the decree logic [25]: if $\mathcal{S}_\mathcal{G}(x_i \mid x_j) = \mathcal{X}_i$, then x_i is fully included (conditionally included); if $\mathcal{S}_\mathcal{G}(x_i \mid x_j) = \emptyset$, it is excluded (conditionally excluded).

The source-to-target assignment problem features such variables [66]. In that problem, either 1 or 2 consumers are assigned to either 1 or 2 energy sources. Two hierarchical interactions arise: if there is only one source, then both consumers have to be assigned to that source, and if there is only one consumer, the source choice for the second consumer is not included. Figure 7.8 illustrates the roles of variables for this source-to-target example. This problem includes 4 variables: the number of energy sources between 1 or 2, the number of consumers between 1 or 2, the source of energy used by the first consumer, and the source of energy used by the second consumer. There are 4 meta possibilities: if 1 source and 2 consumers, the assignment variables are partially-decreed because only the possibility "source number 1"

can be chosen. If 2 sources and 2 consumers, the assignment variables are fully decreed, both "source number 1" and "source number 2" are available for both consumers. If 2 sources and 1 consumer, the variable "source of energy for the first consumer" is fully decreed but the variable "source of energy for the second consumer" is completely excluded because there is no second consumer, in that case, the only possibility for that variable is "no possibilities" or empty set. If 1 source and 1 consumer, the variable "source of energy for the first consumer" is partially-decreed, but the variable "source of energy for the second consumer" is completely excluded. As shown by this example, the same variable can be either partially or completely decreed depending on the problem and meta variable. Therefore, contrarily to other roles, partially-decree and decree property are not mutually exclusive roles and as such, we may also consider meta-partially-decreed variables, that is variables that are both meta and partially-decreed. In particular, if a meta variable prohibits a value of another meta-partially-decreed variable, and if this value was decreeing another variable, then, the first meta variable is in fact excluding the latter variable: nested variable can create complex graph relations.

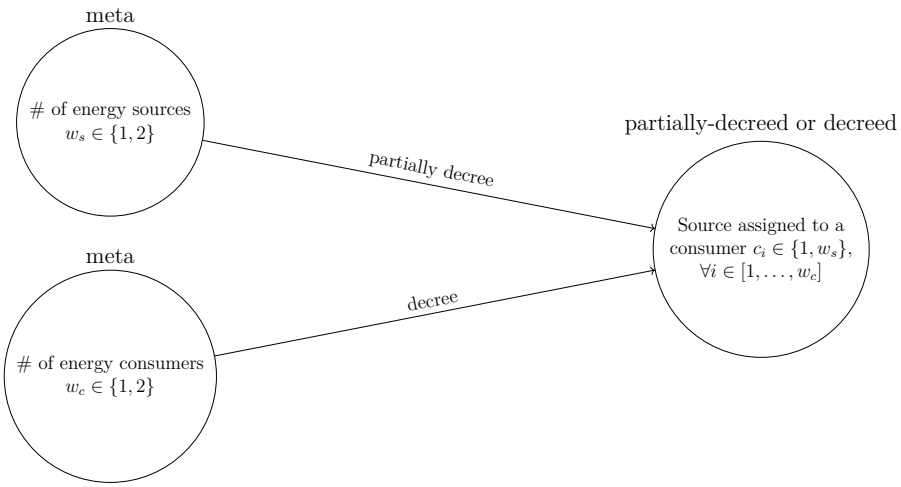


Figure 7.8 Roles of variables for a source-to-consumer example.

However, in our novel extended framework, it is possible to have completely incompatible variables and therefore we need to be able to compare two points without any choice-dependence. The latter remark excludes some classical approaches to compare heterogeneous data such as the imputation method [66, 241]. Consequently, in the next section, we will show how to construct a well-behaved distance and kernel to deal with mixed hierarchical points.

7.3.3 Mixed hierarchical kernels for hierarchical domains

SMT 2.0 introduced a hierarchical correlation kernel that accounts for both meta and decreed variables, and was further extended in [221] to handle distances between decreed continuous variables. In this work, we generalize the SMT framework to support fully hierarchical domains. This extension enriches the `DesignSpace` definition with support for meta-decreed, partially-decreed, and order-based relationships, as described in this section[2]. With these variable types and structural relations, we can now define a generalized design space suitable for hierarchical surrogate modeling.

Definition 47 (Graph-structured design space)**.** *The design space $\mathcal{X}$ represents the set of all possible and admissible configurations of variables based on the graph $\mathcal{G}$. Each point $\mathbf{X} \in \mathcal{X}$ is a configuration defined by a tuple of variable values $(X_1, X_2, \ldots, X_n)$, with X_i is the value of $\mathbf{X}$ for the i-th variable x_i, and it may include continuous, discrete, or categorical variables.*

The span of the design space is the Cartesian product of the supports of the variables possibly included within it, where we define a distance function

$$d_i : \mathcal{S}_i \times \mathcal{S}_i \longrightarrow \mathbb{R}_+, \tag{7.5}$$

The function $d_i(X_i, X_i')$ is defined as:

$$d_i(X_i, X_i') = \begin{cases} d(X_i, X_i'), & \text{if } X_i, X_i' \in \mathcal{S}_i, \\ \delta_i, & \text{if exactly one of } X_i, X_i' = \emptyset, \\ 0, & \text{if } X_i = X_i' = \emptyset, \end{cases} \tag{7.6}$$

where d is any 1-dimensional distance function (see Section 7.2.2) and where $\delta_i = \max\{d(X_i, X_i') : X_i, X_i' \in \mathcal{S}_i\}/2 \geq 0$ is constant ensuring the triangular inequality is respected between included and excluded variables, as proven in [125]. Note that a particular case of the proof for δ_i is given in Appendix C. Finally, for any $p \geq 1$, the hierarchical function $\text{dist}_p : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$ defined by

$$\text{dist}_p(\mathbf{X}, \mathbf{X}') := \left(\sum_{i=1}^{n} d_i \left(X_i, X_i'\right)^p \right)^{1/p}. \tag{7.7}$$

In [110], the authors found good results with kernel approaches because "Unlike the traditional edit distance, kernel functions make good use of statistical learning theory in the inner

[2] https://smt.readthedocs.io/en/latest/_src_docs/applications/Mixed_Hier_usage.html

product rather than the graph space directly". Therefore, this work was motivated by the need to extend graph distances to handle mixed hierarchical variables. We generalize the SMT 2.0 hierarchical kernel as the product of three Symmetric Positive Definite (SPD) sub-kernels: a neutral kernel for variables without dependencies, a meta kernel for high-level configuration choices, and a meta–decreed kernel that ties each choice to its activated children via our graph-distance. Because each factor is SPD, their product is SPD as well, and the resulting kernel naturally handles mixed variable types, conditional activations, and hierarchical structure. More details and the SPD proof are given in Appendix C. Also, in this work, we will use the so-called algebraic distance introduce in [221], namely

$$d(\mathbf{X}, \mathbf{X}') = \begin{cases} 1, & \text{if } \mathbf{X}^\top \mathbf{X}' = 0 \\ \frac{||\mathbf{X}-\mathbf{X}'||}{\sqrt{||\mathbf{X}||^2+1}\sqrt{||\mathbf{X}'||^2+1}}, & \text{otherwise.} \end{cases} \tag{7.8}$$

This distance is well-defined as proven in [125] and the kernel that results from such distance, known as the Alg-Kernel is SPD as proven in Appendix C. This distance is a particular case of the more general hierarchical distance introduced in [125].

7.4 Modeling and optimizing with graph-structure inputs

In this section, we will connect the implementations of the mixed hierarchical surrogate models with the graph-structured design space. In particular, we will describe the method implemented in SMT to build a hierarchical GP and show an illustration on two test cases. First, the hyperparameters modeling of a neural network and, second, an application to a green aircraft optimization through Bayesian optimization. Section 7.4.1 and Section 7.4.2 respectively introduce the two problems, that are MLP modeling and aircraft design modeling. Then, Section 7.4.3 presents the graph-structured design space introduced in this work put into practice, and Section 7.4.4 showcases its application to the MLP model. To finish with, Section 7.4.5 presents the practical details of the graph-structured Bayesian optimization, and Section 7.4.6 showcases its application to the aircraft design model.

7.4.1 Modeling a complex MLP system architecture

To illustrate our approach, we introduce a complex system design problem with different variable types and dependencies. For that, we leverage the Multi-Layer Perceptron (MLP) example that has been introduced in [25, 125, 235], where an MLP is tuned for a given supervised learning task by optimizing its hyperparameters. Identifying optimal hyperparameters is a problem called HyperParameter Optimization (HPO) that is known to be

challenging [115]. In fact, the training, validation, and performance testing are typically conducted with a predetermined set of hyperparameters [159]. These hyperparameters vary widely in type, with some even acting as meta variables, such as the number of hidden layers which is a good illustration for a particular framework. Gaussian Processes (GPs) offer valuable assistance in this optimization process. GPs can efficiently evaluate multiple sets of hyperparameters at a significantly reduced computational cost. Furthermore, GPs can be integrated into Bayesian optimization frameworks to automate the search for optimal hyperparameters [154]. The deep model's performance relative to its hyperparameters can be conceptualized as a blackbox function with mixed hierarchical variable inputs and therefore heterogeneous data.

Consider a blackbox function $f : \mathcal{X} \to \mathbb{R}$, which outputs a performance score $f(\mathbf{X}) \in \mathbb{R}$ for a given vector of hyperparameters $\mathbf{X} \in \mathcal{X}$. For the MLP problem, the function f is evaluated by training the deep model knowing the specific set of hyperparameters $\mathbf{X}$ and then test it to obtain a resulting performance score $f(\mathbf{X})$. Therefore, the latter represents the model's accuracy on a previously unseen data set [25]. While the internal mechanics of f are understood, the function is treated as an expensive-to-evaluate blackbox problem due to the complexity involved in adjusting billions of model parameters and of fine-tuning them through backpropagation during training [25, 158]. To facilitate the comprehension of our modeling work, Feature Model (FM) has been used to build the tree presented in Figure 7.9 that shows how modeling can give insights for a system such as the MLP. First, the selection of the optimizer determines which hyperparameters are included in the optimization domain. For instance, the decay parameter λ is only considered if the optimizer $o \in \{\text{Adam}, \text{ASGD}\}$ is ASGD. Second, the optimizer influences the model's architecture. Specifically, it affects the set of possible values for the number of hidden layers and number of units in each included layer that is represented by a "requires" relationship, as it changes the possible bounds. Choices are represented by "alternative" features, as the optimizer that is either ASGD or Adam but not both, whereas the included or not layers are represented by OR features. For the sake of illustration, we have no incompatibilities and therefore, no "excludes" relationship on this problem.

The feature model of Figure 7.9 presents several limitations. First, continuous variables have non-discrete support that can not be addressed by feature models. However, this remark can partially be addressed by a feature model extension called attribute feature models [1, 245].

Attribute feature models address a key limitation by explicitly encoding dependencies that are otherwise implicit. For example, they capture that including a hidden layer requires specifying how many units it may contain and they also clarify how the choice of optimizer

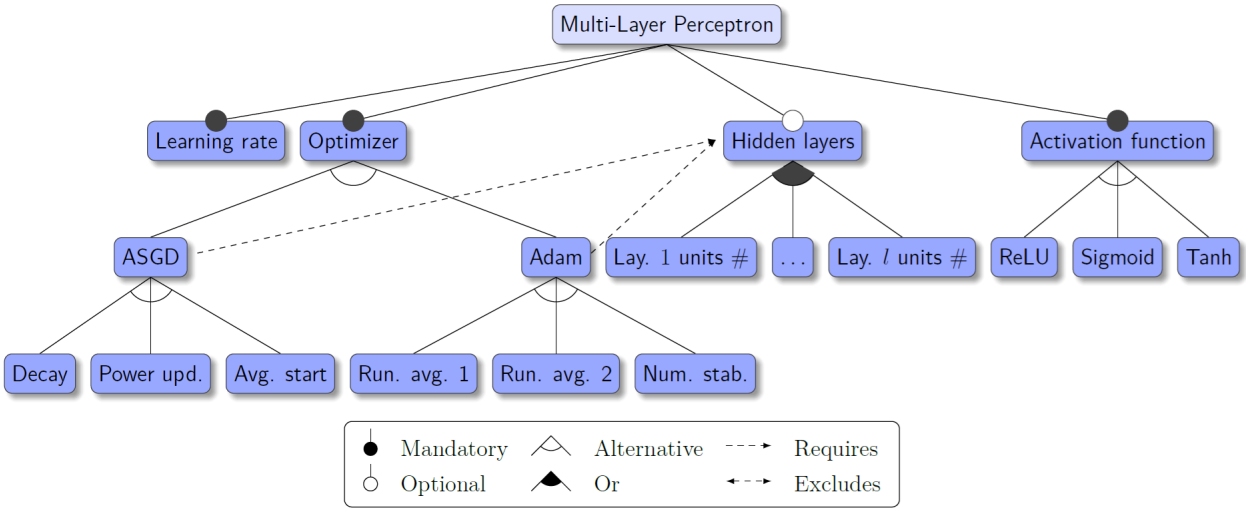


Figure 7.9 Feature model for the MLP working example.

constraints which architectures are valid, and enable more flexible, partially specified variable definitions by explicitly defining allowable value ranges (supports). Attributes do not necessarily serve as variable supports, and features do not always correspond to optimization or modeling variables. For example, in the Feature Model (FM), "Adam" is considered a feature, whereas in the design space definition, it represents a level of the variable "Optimization". Similarly, "hidden layers" may be a feature in the FM but not a design variable in the design space. In the design space, the number of hidden layers is treated as a variable, while in the FM, this choice is governed by an "OR" relationship, meaning that determining the number of hidden layers requires counting how many have been selected. These distinctions highlight the need for the hierarchical model introduced in this paper. Unlike traditional feature models and design space graphs, this model is more directly aligned with design variables, making it a complementary approach that bridges the gap between existing representations.

With the design space graph approach, the first modeling step is to build the role graph $G = (N, A)$, which is presented schematically in Figure 7.10. Note that the index i in u_i indicates the i-th hidden layer. For example, if $l = 3$, there are three hidden layers, each with its own hyperparameter for the number of units, *i.e.*, u_1, u_2, u_3. Furthermore, the number of hidden layers $l \in L_o$ (influenced by the chosen optimizer) also determines the number of variables associated to the units u_i in the hidden layers, creating a 2-level hierarchy.

The role graph G in Figure 7.10 models a great deal of information, including 1) the decree relations, represented by arcs, between the variables, 2) the role of each variable via the position of its corresponding node in the DAG, 3) the depth of each variable, and 4) the

dependencies of the belonging sets of meta-decreed and decreed variables with their lower-depth variables. As such, it is more explicit than the feature model. The FM has 4 first-level features, whereas our hierarchical model has 3 roots because the number of hidden layers that we allow depends from the choice of the optimizer. Therefore, to solve these incompatibilities, the graph model proposes to choose first the optimizer and then the number of hidden layers.

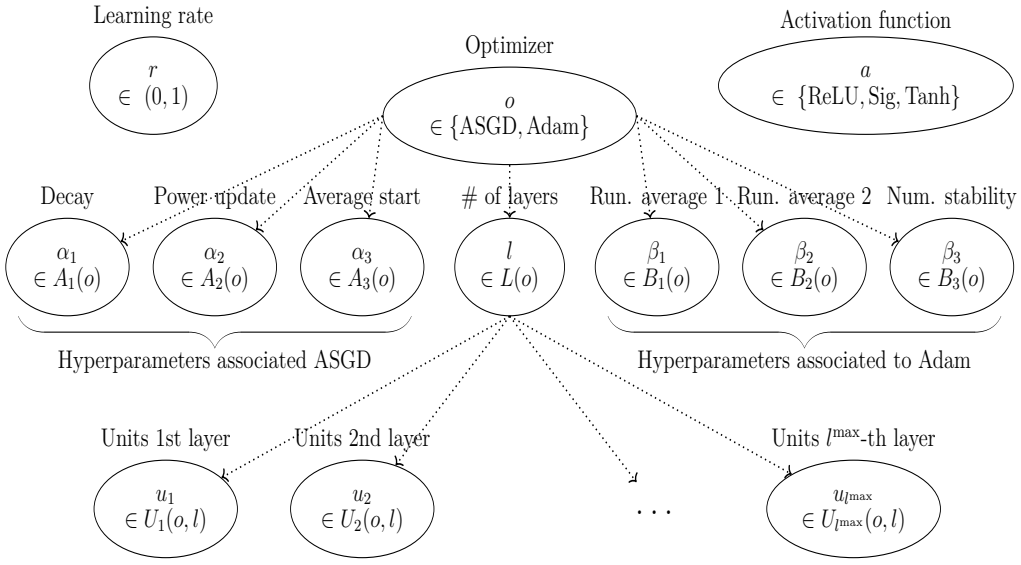


Figure 7.10 Role graph G for the MLP working example.

Then, within the design space graph approach, the next modeling step consists in formulating the problem—such as an optimization or analysis task—based on specific variables whose admissible bounds are explicitly defined for exploration. This step resulted in the construction of Table 7.1, which summarizes the problem description and all its variable roles, types, and bounds.

7.4.2 Modeling a multidisciplinary aircraft system architecture

Designing complex engineering systems, such as aircraft, requires coordinating multiple interacting disciplines, including aerodynamics, structures, propulsion, and controls, where decisions in one area produce feedback that propagates through others and creates tightly coupled, often nonlinear behavior. To manage this complexity, eXtended Design Structure Matrix (XDSM) models offer a concise yet expressive way to represent modules, data flows, solver loops, and iteration order, making it easy to see which subsystems must be solved iteratively and which can be treated more loosely [160]. These abstract system models serve as blueprints that can be translated into executable Multidisciplinary Design Analysis (MDA) and optimization simulations in frameworks such as OpenMDAO, where each

Table 7.1 Definition of the "MLP" optimization problem.

	Function/variable	Type	Role	Quantity	Range
Minimize	Loss function	cont		1	
	Total objectives			**1**	
with respect to	Learning rate	cont	neutral	1	$(0, 1)$
	Decay	cont	decreed	1	$(0, 1)$
	Power update	cont	decreed	1	$(0, 1)$
	Average start	cont	decreed	1	$(10^3, 10^8)$
	Run. average 1	cont	decreed	1	$(0, 1)$
	Run. average 2	cont	decreed	1	$(0, 1)$
	Num. stability	cont	decreed	1	$(0, 1)$
	Total continuous variables			7	
	Activ. funct.	cat	neutral	3 levels	{ReLU,Sigmoid,Tanh}
	Optimizer	cat	meta	2 levels	{ASGD,Adam}
	Number of layers	int	meta-decreed	1	{1,2,3}
	Neur. Units layer 1	int	decreed	1	$\{25, 30, \ldots, 45\}$
	Neur. Units layer 2	int	decreed	1	$\{25, 30, \ldots, 45\}$
	Neur. Units layer 3	int	decreed	1	$\{25, 30, \ldots, 45\}$

discipline is implemented as a component, drivers coordinate solvers and optimizers, and system-level derivatives or surrogates can be propagated for efficient gradient-based design exploration [118]. Together, XDSM and an MDAO improve traceability and reproducibility, guide solver and surrogate choices, and accelerate disciplined trade studies before committing to costly high-fidelity analyses.

The DRAGON baseline configuration is an aircraft featuring two turboshafts, four generators, four propulsion buses with cross-feed and forty fans. This configuration was selected for the initial study as it satisfies the safety criterion. However it was not designed to optimize aircraft weight. The turboelectric propulsive chain being an important weight penalty, it is of particular interest to optimize the chain and particularly the number and type of each component, characterized by some mixed hierarchical values. In our workflow, the high-level XDSM representations are translated into executable MDA simulations, forming the computational core of the DRAGON co-design model. These simulations integrate aerodynamic, structural, propulsion, and electrical models within mission-analysis and sizing loops, ensuring that all estimated quantities reflect a physically consistent equilibrium achieved through multidisciplinary coupling and convergence towards consensus across disciplines. The simulations rely on the open-source Future Aircraft Sizing Tool with Overall Aircraft Design (FAST-OAD) [93], enhanced with expert knowledge and recent extensions for hybrid-electric and distributed propulsion concepts. Built upon the OpenMDAO framework [118], FAST-OAD provides a modular structure where each discipline acts as a component, and where the data exchanges and solver couplings are explicitly defined. This setup enables flexible

management of multidisciplinary interactions and clear control of convergence behavior.

The consideration of the variables related to architecture was revised to take full advantage of mixed variables optimization. Three configurations with different numbers of electric components were considered, each with its own sizing rules. The default configuration is preserved for the analysis, and two new configurations, one with low distribution and the other with high distribution, were created and analyzed to establish the sizing rules. The number of motors was to remain an optimization variable as it is an important driver of the propulsive and aerodynamic efficiency of the aircraft, but the type of architecture constrains it. In this analysis, the number of motors could only be a multiple of 4, 8, or 12, depending on the selected architecture. The solution consisted of expanding the levels of the categorical variable representing the architecture and assigning a specific and valid number of motors to each level.

In a previous work, we optimized the Distributed fans Research Aircraft with electric Generators by ONERA (DRAGON), but without considering any hierarchy [227], and an additional goal of this article is to show how our framework can be applied in practice. The definition of the turboshaft layout is given in Table 7.2 and the definition of the architecture variable is given in Table 7.3, where, for the sake of simplicity, we fixed the number of generators to be equal to the number of cores.

Table 7.2 Definition of the turboshaft layout variable and its 2 associated levels.

Layout number	position	y ratio	tail	VT aspect ratio	VT taper ratio
1	under wing	0.25	without T-tail	1.8	0.3
2	behind	0.34	with T-tail	1.2	0.85

Table 7.3 Definition of the architecture variable and its 17 associated levels.

Architecture number	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**11**	**12**	**13**	**14**	**15**	**16**	**17**
Number of cores	2	2	2	2	2	2	2	2	2	4	4	4	4	4	4	4	4
Number of generator	2	2	2	2	2	2	2	2	2	4	4	4	4	4	4	4	4
Number of motors	8	12	16	20	24	28	32	36	40	8	16	24	32	40	12	24	36

At the time, the propulsive architecture had to be encoded as a single categorical variable with 17 possible levels (Table 7.3). Notwithstanding, for modeling electric architectures, it is more efficient to represent the architectural choices using two integer variables instead of one categorical variable. However, taking this approach expands the range of potential architectures beyond the initial 17 configurations. Yet, there are two important constraints to consider for these possible setups. The first constraint relates to the electrical connections between components: ensuring a certified electric architecture is crucial, and figuring out how

to connect, for example, 8 motors to 6 generators is not straightforward. The second constraint is connected to the distributed propulsion system, especially the numerous propellers. Managing this system involves addressing a substantial number of potential failures in the electro-mechanical architecture, as for both stability and redundancy, not all electric connections are allowed. Using the hierarchical framework introduced in this paper, this single categorical choice can be decomposed into the two hierarchical integer variables that encode the propulsive chain. Not only do we have a better representation of the architecture system, but, as a side effect, in the continuous relaxation setup, this reduces the number of design variables associated with the propulsive architecture from 17 to 3, thereby simplifying and accelerating continuous relaxation based optimization for the overall problem. The problem defined in Table 7.4 targets an expensive-to-evaluate, high-fidelity blackbox simulation. The objective is to minimize the aircraft fuel mass over the mission—a key driver of both environmental impact and operating cost—subject to five expensive-to-evaluate system-level constraints. These constraints are outputs of the simulator (*i.e.*, they result from the simulator's internal computations and couplings given the input variables) and therefore differ from algebraic or structural constraints imposed directly on the inputs.

Table 7.4 Definition of the `DRAGON` optimization problem.

	Function/variable	Type	Role	Quantity	Range
Min.	Fuel mass	cont		1	
	Total objectives			**1**	
w.r.t	Fan operating pressure ratio	cont	neutral	1	$[1.05, 1.3]$
	Wing aspect ratio	cont	neutral	1	$[8, 12]$
	Angle for swept wing	cont	neutral	1	$[15, 40]$ $(°)$
	Wing taper ratio	cont	neutral	1	$[0.2, 0.5]$
	HT aspect ratio	cont	neutral	1	$[3, 6]$
	Angle for swept HT	cont	neutral	1	$[20, 40]$ $(°)$
	HT taper ratio	cont	neutral	1	$[0.3, 0.5]$
	TOFL for sizing	cont	neutral	1	$[1800., 2500.]$ (m)
	Top of climb vertical speed for sizing	cont	neutral	1	$[300., 800.]$ (ft/min)
	Start of climb slope angle	cont	neutral	1	$[0.075., 0.15.]$ (rad)
	Total continuous variables			10	
	Turboshaft layout	cat	neutral	2 levels	{1,2}
	Number of cores	int	meta	1	{2,4,6}
	Number of motors	int	decreed	1	{8,12,16,20,...,40}
s.t.	Wing span < 36 (m)	cont		1	
	TOFL < 2200 (m)	cont		1	
	Wing trailing edge occupied by fans < 14.4 (m)	cont		1	
	Climb duration < 1740 (s)	cont		1	
	Top of climb slope > 0.0108 (rad)	cont		1	
	Total constraints			**5**	

7.4.3 The graph-structured design space: implementation and usage

Here we extend the framework from [226] by incorporating the key features reviewed in Section 7.2. Our implementation builds on SMT 2.0's design-space tools [52, 226], the ConfigSpace library [165], and the DSG available in the software `adsg-core` [61, 62], making it practical for real-world use[3]. Table 7.5 outlines the main features of the Python modeling software that can handle hierarchical and mixed variables within SMT, chosen within the open-source software literature based on its mixed variable handling capacities as detailed in [226, Table 1] where several are compared for such modeling task. The default option in SMT is a basic DSG that is limited to one level of hierarchy and is not able to handle complex relationships such as incompatibility constraints even if hierarchical. The ConfigSpace design space may be slow and is not based on the graph structure. Overall, the best option for architecture modeling is the DSG of the `adsg-core` toolbox that is fast and gives explicit visualizations of the DSG in DOT language, draw.io or GML [98] and our main software contribution has been combining DSG to work within SMT to define hierarchical surrogate models.

Table 7.5 Comparison of software packages for hierarchical and mixed design-space definition.

Package	**License**	`SMT` interface name	Mixed hier. var.	Nested hier.	Incomp. (excl.)	Graph struct.	Explicit & visu.
SMT [226]	BSD	`DesignSpace`	✓	–	–	✓	–
ConfigSpace [165]	BSD	`ConfigSpaceDesignSpaceImpl`	✓	✓	✓	–	–
adsg-core [62]	MIT	`AdsgDesignSpaceImpl`	✓	✓	✓	✓	✓

We can automatically define a DSG from the variables specifications[4].

The DSG models hierarchical choices using selection and connection choice nodes, that select between mutually exclusive option nodes and connect sets of source nodes to target nodes, respectively [62]. Nodes are subject to selection and only obtain meaning in the context of an optimization problem. Its original context was function-based modeling of system architecture design spaces, called the Architecture Design Space Graph (ADSG), where various types of nodes, such as functions, components, and ports, were defined [64].

7.4.4 Application of the graph-structured design space to the MLP problem

For illustrating the method, let `design_space` be the SMT 2.0 `AdsgDesignSpaceImpl` defined in Appendix B (Figure 7.14) for the MLP problem of Section 7.4.1. In a DSG, both neutral

[3] https://adsg-core.readthedocs.io/
[4] https://github.com/SMTorg/smt-design-space-ext/

and purely meta variables serve as graph roots, also referred to as "permanent nodes" or "start nodes", while decreed and meta-decreed variables function as "conditional nodes". A choice that depends on a meta or meta-decreed variable is represented by a choice node (in blue) and is connected to the variable name and its possible choices through derivation edges. Future work may include automated formal verification of the graph structure. However, some verification steps are already handled within the DSG. For instance, the DSG is considered infeasible if an incompatibility edge is defined between two permanent nodes, ensuring that conflicting configurations are detected early.

To do this, the *encoder* maps architectural choices, such as selections and connections in the graph, into a numerical design vector $\mathbf{x}$ that optimizers can handle. Therefore, selection and connection choice encoders enable a full enumeration of valid design vectors and ensure any source-to-target connection problem can be encoded such that optimization algorithms can effectively search the design space. The encoder plays a crucial role in mapping selection choices from the hierarchical space to discrete variables in the input vectors, ensuring correct associations with input points for tasks such as sampling. Currently, two encoding methods are implemented. The *complete encoder* performs an exhaustive exploration of the design space, identifying all valid configurations and providing accurate imputation ratios. In contrast, the *fast encoder* offers a more efficient but less comprehensive approach by directly mapping selection choices to design variables. While the fast encoder significantly reduces computation time, it may overlook some valid configurations, making it more suitable for large design spaces with computational constraints. The complete encoder implements a constrained enumeration: it derives local connection choice formulations from the DSG, builds an influence matrix to prune infeasible partial assignments, and enumerates the set of valid connection matrices to produce a full mapping that permits exact decoding or nearest-valid lookup. By contrast, the fast encoder avoids global enumeration and decodes greedily, trading completeness for much lower memory and runtime requirements and serving as the practical fallback when exhaustive enumeration is infeasible. More details are given in [60, Appendix B].

To construct the DSG shown in Figure 7.11, we only need to define the variables, levels and relationships. Then, the DSG is processed to have a visual graph with the nodes in N_{var} in blue if categorical and white otherwise, the nodes in N_{levels} in white and those in N_{bnd} in straw color. The edges in N_{opposed} are colored in red. The DSG also automatically processes the graph to generate outputs, as presented in Table 7.6.

This analysis provides useful insights, including the imputation ratio (`imp_ratio`), which measures the ratio between the total number of declared categorical and integer possibilities

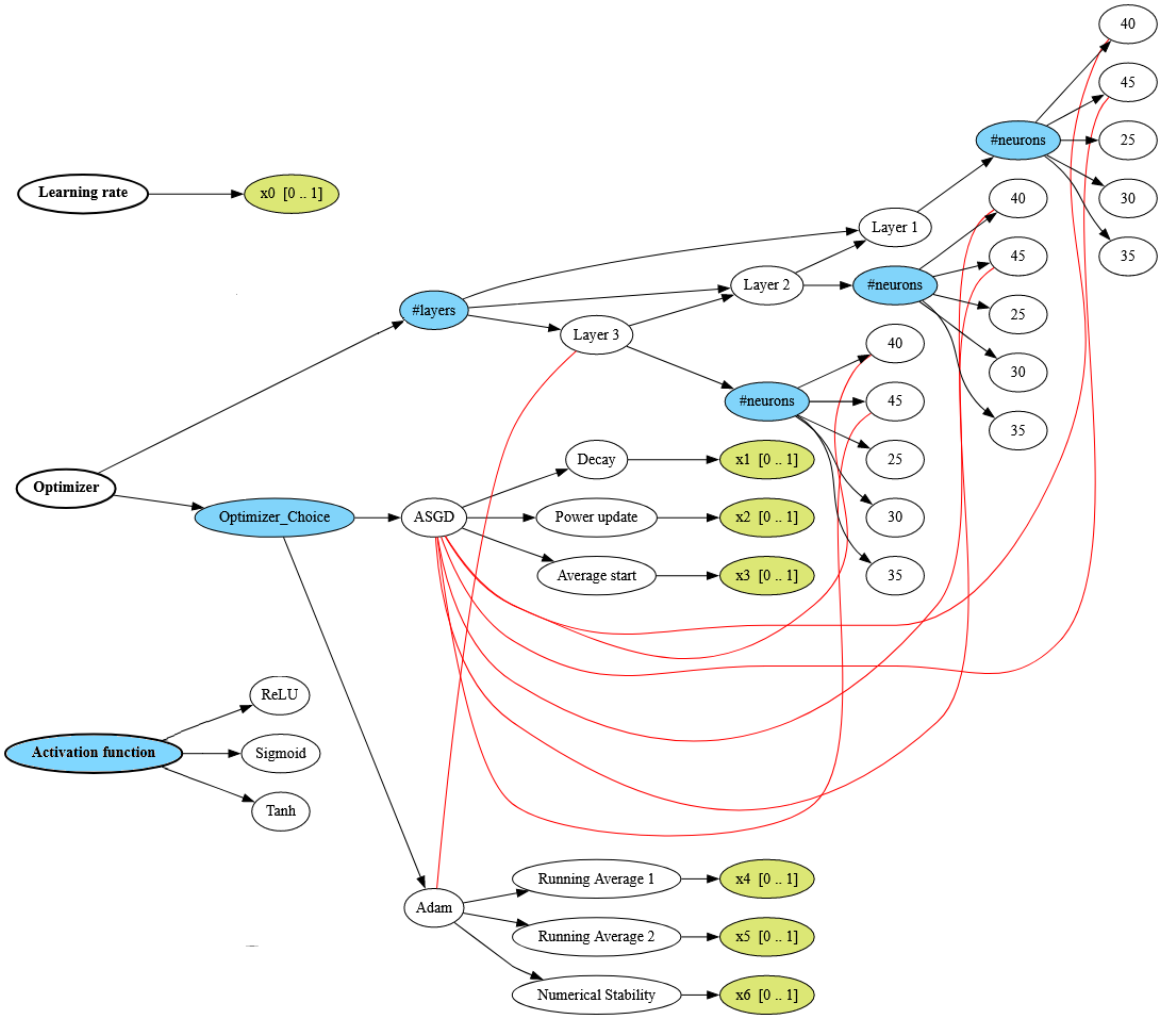


Figure 7.11 DSG definition for the neural network problem.

Table 7.6 Processed DSG on the neural network problem.

Type	n_valid	n_declared	n_discrete	n_dim_cont	n_dim_cont_mean	imp_ratio	encoder
option-decisions	207	1350	6	0	0.0	6.521739	complete
additional-dvs	207	0	0	7	4.0	1.750000	
total-design-space	207	2250	6	7	4.0	10.869565	complete

(`n_declared`) and the number of feasible instances (`n_valid`). In this case, 2250 possibilities are declared, but only 207 are valid, meaning that approximately 1 in 10.9 instances is feasible. The table also displays the total number of variables, which consists of 13 variables: 6 discrete (`n_discrete`) and 7 continuous (`n_dim_cont`). The continuous imputation ratio is 1.75, indicating that only 4 out of the 7 continuous variables are included (`n_dim_cont_mean`). Specifically, the included variables correspond to either the combination of "decay", "Power update", and "Average start" or the combination of "Running average 1", "Running average

2", and "Numerical Stability". The `encoder` column in Table 7.6 indicates use of the complete encoder; exhaustive enumeration therefore ran efficiently and found all valid design vectors among the 2250 possible combinations.

Thanks to the DSG, we can make explicit the definition of the hierarchical model by giving the model directly and generating a valid configuration as in Figure 7.12. This new class upgrades and improves on the prior `ConfigSpaceDesignSpaceImpl` class that did not take into consideration hierarchical spaces. The two can be compared on processing tasks as described in Table 7.7. The most important task consists in correcting a non-valid input point and returning both a corrected input and its corresponding included variable within the extended space (called active). This task of projecting an input from the extended domain to the hierarchical space is really important since the hierarchical distance is computed on the extended domain and then projected as this projection is an isomorphism [125, Theorem 2]. On this task, we observe, in Table 7.7 an improvement by 76 % with the `AdsgDesignSpaceImpl` that relies on adsg-core implementation compared to our legacy `ConfigSpaceDesignSpaceImpl` method relying on ConfigSpace. Another processing task is enumeration of all discrete possibilities, and in Table 7.7, we obtained a 36% speed up with the `AdsgDesignSpaceImpl` class for such a task. To finish with, one could want to generate a design of experiments that respects the graph-structure domain. On this task, we obtain a 15% improvement with our class, but to achieve this performance, we first generated one random point for each of the 207 possible discrete configurations and then subsampled 100 points from that set: an approach also adopted by SBArchOpt [59]. However, one might alternatively choose to sequentially sample additional points. While it is straightforward to parallelize the sampling of 100 points, doing so sequentially is approximately 15 times slower than the subsampling approach.

Table 7.7 Comparison of processing duration for several tasks with `ConfigSpaceDesignSpaceImpl` and `AdsgDesignSpaceImpl`.

Class name	**Correct and compute activeness of 1000 invalid points**	**Generate 1 point by discrete possibility (207 points)**	**Generate 100 valid points**
`ConfigSpaceDesignSpaceImpl`	2.2300s	0.0081s	0.0054s
`AdsgDesignSpaceImpl`	0.5356s (-76%)	0.0053s (-36%)	0.0046s (-15%)

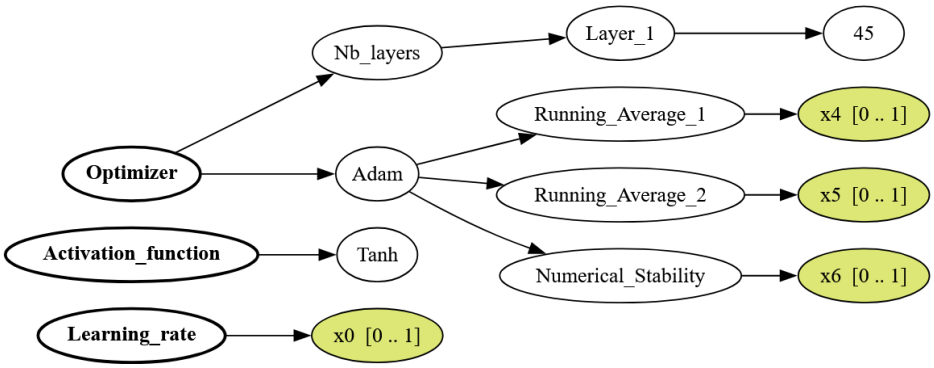


Figure 7.12 `AdsgDesignSpaceImpl` sampling.

In the next section, we will use the graph-structured DSG of SMT, which is based on the underlying DSG of ADSG, to create surrogate Gaussian process models. These surrogate models, therefore, support the modeling of hierarchical problems while preserving their structure. They are then applied to surrogate-based Bayesian optimization.

7.4.5 Bayesian optimization: implementation and usage

Bayesian optimization has emerged as a highly effective technique for optimizing blackbox functions, particularly when these functions are expensive to evaluate, non-convex, or lack gradient information [107]. Its growing popularity in various fields, from machine learning hyperparameter tuning (HPO) to engineering design, is largely due to its ability to achieve near-optimal solutions with a minimal number of evaluations [78]. At its core, BO builds a probabilistic surrogate model, typically a GP, to approximate the objective function that is being optimized. The surrogate model provides not only predictions of the function's value at any given point in the search space but also an estimate of the uncertainty associated with these predictions. This uncertainty quantification method is the foundation of BO ability to balance exploration (sampling areas of the search space with high uncertainty) and exploitation (focusing on areas predicted to yield high objective values). The BO process typically starts with a Design of Experiments (DoE) to obtain initial evaluations of the objective function. These evaluations are used to fit the GP model, which is then iteratively updated as new function evaluations are conducted. At each iteration, the next point to evaluate is chosen by maximizing an acquisition function to guide the search towards the optimum as fast as possible.

GP are well-suited for Bayesian optimization because they can flexibly represent unknown objective functions via a mean function and correlation kernel. The mean gives a prior es-

timate while the kernel encodes correlations across the search space, allowing uncertainty quantification and efficient predictions. In this work, we adopt the kernel described in Section 7.3.3, which balances smoothness assumptions with adaptability to mixed continuous, categorical, and hierarchical inputs. Once the surrogate is set up using SMT's `DesignSpace` class, defining a design of experiments and instantiating corresponding GP models—complete with graph-structured kernels—becomes a single, streamlined step. The acquisition function is a crucial component of BO, responsible for determining the next point to evaluate. Common acquisition functions include Expected Improvement (EI), Probability of Improvement (PI), Upper Confidence Bound (UCB) or Watson and Barnes 2$^{\text{nd}}$ criterion scaled (WB2s) [39, 144, 233]. In this work, we will use WB2s that is more smoothed and less multimodal than the other classical criteria, that is tractable and easy to optimize with a local optimizer such as COBYLA [199] and the optimizer SEGOMOE [39].

In many real-world optimization tasks, the design variables span continuous, integer, and categorical types, so standard continuous Bayesian optimization has been extended for them. Categorical inputs use specialized kernels, and, for comparison with our method, we use, from the simplest to the most complex method: Gower Distance (GD), Continuous Relaxation (CR), and Homoscedastic Hypersphere (HH) to capture categorical correlations, while integer variables employ ordinal-aware, distance-based kernels. More details on these methods are presented in [224] as well as their unification in both modeling and software terms. High dimensionality compounds the challenge, since modeling accuracy typically requires exponentially more samples. To mitigate this "curse of dimensionality", dimensionality reduction methods such as Partial Least Squares (PLS) are integrated into the Gaussian-process surrogate. In particular, Kriging with PLS (KPLS) has recently been generalized to mixed-variable and even hierarchical spaces, enabling efficient BO over complex, heterogeneous domains [225]. However, the baseline surrogate model presented in this work, without any dimension reduction is limited to around one hundred variables, which may be sufficient in most applications, while the Architecture design space graph has been applied for up to 144 million configurations.

7.4.6 Application of the Bayesian optimization approach to the aircraft design problem

Once we have defined the surrogate model based on the `DesignSpace` class in SMT, we can directly define both DoE and surrogate models such as GP models while accounting for the hierarchical model. In [226], a GP model has been built for the hierarchical neural network problem and promising results have been obtained, and in [221], the GP has been extended

to handle decreed continuous variables and meta-decreed variables.

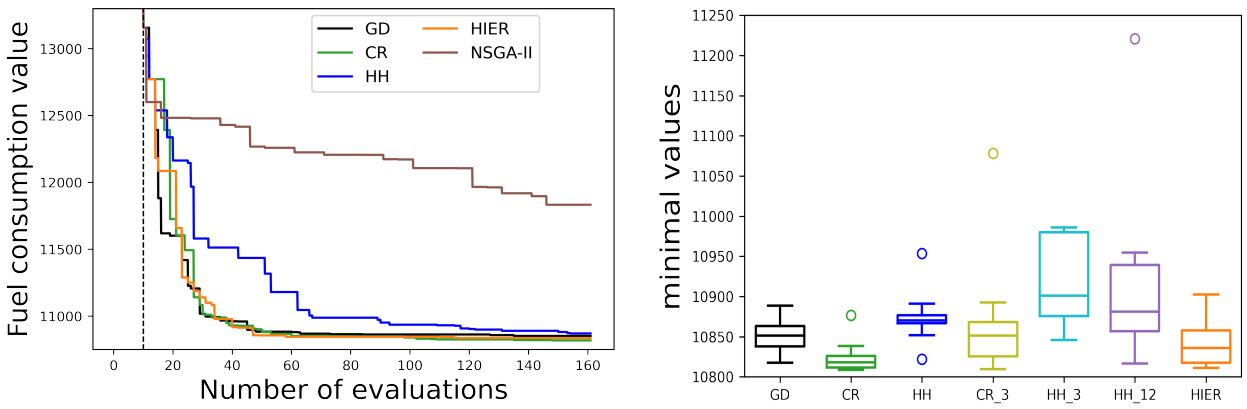
To validate our method, we compare the 8 methods described in Table 7.8 on the optimization of the DRAGON aircraft concept, with 10 points for the initial DoE. The first six are categorical kernels with different structures introduced and detailed in [224]. The method HIER is the one based on the graph-structure design space and the last NSGA-II is an evolutionary algorithm posed as a baseline [95]. We also tested dimension reduction, as such, several methods employ the PLS (Partial Least Squares) dimension reduction techniques [225]. These reduced-order models are given a target dimension that is the reduced size of their corresponding embeddings; for instance, HH with PLS 3D (or HH_3 for short) is the HH method reduced to 3 dimensions. Even with these dimension reduction techniques, the matrix based methods are too costly for such problems: HH takes 320h, HH with PLS 3D takes 102h and HH with PLS 12D takes 258h.

Table 7.8 The various kernels compared for optimizing DRAGON.

Name	# of cat. params	# of cont. params	Tot. # of params	Comp. time
GD	2	10	12	36h
CR	19	10	29	62h
CR with PLS 3D	Not applicable	Not applicable	3	14h
HH	137	10	147	320h
HH with PLS 3D	2	1	3	102h
HH with PLS 12D	2	10	12	258h
NSGA-II	Not applicable	Not applicable	Not applicable	16h
HIER	1	12	13	40h

Figure 7.13 shows promising results with the hierarchical domain. Namely, it is a bit more costly than GD for a better result and in terms of convergence behaviour GD, CR and HIER are highly similar. Note that CR with PLS 3D is faster than HIER for little degraded performance and that CR is the best in terms of convergence but is slightly more costly. In the end, HIER is a really good trade-off for BO.

The DRAGON case study illustrates the benefit of using hierarchical surrogate modeling within a Bayesian optimization framework for complex aircraft design problems. The hierarchical kernel (HIER) achieved a strong balance between accuracy and computational efficiency: while slightly more expensive than GD, it consistently provided better optima and convergence stability across all runs for only one parameter more (see Fig. 7.13). In practical terms, the optimization led to a 439 kg reduction in fuel consumption for a single mission compared to the reference configuration, reaching 10,809 kg against 11,248 kg in the baseline configuration of the first studies [230]. The optimal configuration (option 10) features eight engines, four



(a) Convergence curves.

(b) Boxplots.

Figure 7.13 DRAGON optimization results using a DoE of 10 points over 10 runs. The boxplots are generated, after 150 iterations, using the 10 best points.

of which are hybrid turbo-generators placed at the rear, leveraging a favorable moment arm between the wing and horizontal tail. This design improves overall energy efficiency while maintaining feasible aerodynamic and mission performance. Although structural effects of high sweep and aspect ratio were simplified, the resulting architecture remains close to the reference optimum (10,816 kg) obtained in previous optimization work [226], confirming that the hierarchical surrogate and kernel-based exploration effectively capture trade-offs between structural layout, powertrain topology, and mission-level performance.

7.5 Conclusion and Perspectives

This work introduced a novel operations research framework designed to define graph-structured design spaces, with application to surrogate modeling and optimization. We built upon previous works, extending their definitions to encompass any DAG, and we enhanced the flexibility and scalability of the framework. Specifically, this work extended the concept of meta and decreed variables presented in earlier research [25, 221], bridging the gap with design space graphs. Afterwards, we demonstrate the improved framework capabilities and we proposed an industrial application for sizing and optimizing complex system architectures [64, 66]. Our graph-structured framework provides a flexible representation for mixed-variable and variable-size input feature spaces. It integrates naturally with surrogate-based optimization and is primarily intended for pointwise comparison rather than full graph-isomorphism tasks. However, it can become computationally burdensome for combinatorially large configuration spaces or when similarity depends on explicit topology matching. In such cases,

we recommend combining the framework with scalable approximations or using specialized graph-kernel methods as appropriate [69].

A key contribution of this research is the implementation of the extended framework within the open-source Surrogate Modeling Toolbox software [226]. This open-source toolbox is collaborative and is an easy to use and well documented platform for users and developers. Notably, SMT remains the only open-source toolbox capable of constructing both hierarchical and mixed surrogate models, and it has demonstrated its value in numerous practical applications, such as the optimization of a green aircraft design. In short, this paper lays the groundwork for further developments involving hierarchical models or data. By improving both the theoretical foundation and practical implementation of this framework, the potential applications of this work are vast, ranging from industrial design optimization to more complex systems requiring advanced modeling and optimization techniques.

Several areas of future research and development have been identified. For instance, the new framework still lacks automated formal verification for graph structures. Similar to approaches used in feature models, future work will focus on integrating anomaly detection algorithms to ensure the reliability of the design space [43]. Future research could focus on refining graph structure models and addressing limitations in current methodologies to improve their application in various domains or adding user interface to define the graph with visual tools. For instance, the role graph in this work is designed as a DAG to prevent cyclical dependencies, ensuring that meta-decreed variables do not influence one another. While DAGs provide clear hierarchical structures and efficient traversal [44], they limit the representation of complex interdependencies. The DSG addresses this by recursively parsing design variables in a flexible, non-linear sequence [62]. Future research will focus on expanding the DSG to incorporate more complex structures, including cyclic dependencies, leveraging advancements like hypergraphs [15] and cyclic graph models [37] to enhance the framework's applicability to sophisticated optimization and surrogate modeling challenges. However, enumerating all valid instances within such complex structures quickly becomes computationally intractable. To overcome this, advanced operations research techniques will be required to efficiently estimate the number of possibilities and explore these graphs effectively [191]. Another promising direction for future development is the generalization of the framework to enable more automatic learning. For instance, relationships between variables could be learned directly from data [53]. Moreover, adopting more general hierarchical representations could help tackle problems involving partial and incomplete data, particularly when numerous variables exist but are not fully observed [190]. Also, we can extend relationship definitions to include any function of the form $R(\phi(x_1), \psi(x_2))$.

A further area of interest lies in reducing graph complexity to improve scalability. Future work will explore techniques such as kernelization of the vertex cover problem [58] and propositionalization methods, which convert graph structures into flat propositions, making it easier to apply distance metrics over propositional data [150]. These strategies will enhance the framework's scalability when dealing with large, complex design spaces. Finally, the framework will be adapted to tackle highly dimensional problems with architectural constraints. This will involve using advanced techniques such as KPLS combined with hierarchical kernels [51, 225]. The goal is to enable the framework to handle more complex optimization tasks by integrating dimension reduction techniques with structural flexibility.

As future work, we plan to integrate the Design Space Graph within coupled system XDSM, thereby extending graph-structured design space for the treatment of coupling variables by explicitly tracking their relationships across surrogate-based co-simulation chains [97]. Still, our work is general and can be leveraged for many applications beyond coupled system. For instance, the distance introduced in this work is model-agnostic and applicable to a wide range of surrogate models beyond Gaussian processes [125]. Specifically, it can be used as a dissimilarity measure for instance-based learners, to construct alternative correlation kernels, or to generate finite-dimensional embeddings that produce feature vectors suitable for any parametric regressor. Future work will extend this framework to higher-dimensional problems and explore its integration with k-nearest neighbors, neural networks, and inverse-distance weighting surrogate models. Moreover, our methodology aligns with the principles of systems engineering by providing a general structured modeling procedure that spans from architecture definition to surrogate construction [72].

This framework formalizes the translation of component- and function-level dependencies into graph-structured design spaces, thereby reducing ambiguity and enabling analytical optimization. Consequently, future research will investigate the integration of this approach within system architecture design and optimization reasoning environments [61] and its application to co-design and socio-technical simulation problems [228]. Beyond formal representation, the proposed design space graphs and hierarchical distances directly support surrogate modeling and optimization by providing consistent variable embeddings and correlation structures. The framework can also interface with original simulations whenever hierarchical dependencies must be explicitly managed in Design of Experiments or system analysis workflows. This work is general and can be applied to any kind of complex system that involves a structured design space or heterogeneous datasets.

Replication of results

This framework is implemented in the open-source SMT toolbox [226], extending its capabilities to support hierarchical surrogate models. This implementation is documented in a dedicated tutorial notebook https://github.com/SMTorg/smt-design-space-ext/blob/main/tutorial/SMT_DesignSpace_example.ipynb, easing its practical application. Also one can automatically define a DSG from the variables specifications https://github.com/SMTorg/smt-design-space-ext/.

Fundings

This research is funded by a Natural Sciences and Engineering Research Council of Canada (NSERC) PhD Excellence Scholarship (PGS D), a Fonds de Recherche du Québec (FRQNT) PhD Excellence Scholarship and an Institut de l'Énergie Trottier (IET) PhD Excellence Scholarship, as well as by the NSERC discovery grant RGPIN-2024-05093. This work is part of the activities of ONERA - ISAE - ENAC joint research group. The research presented in this paper has been performed in the framework of the COLOSSUS project (Collaborative System of Systems Exploration of Aviation Products, Services and Business Models) and has received funding from the European Union Horizon Europe program under grant agreement n° 101097120 and in the MIMICO research project funded by the Agence Nationale de la Recherche (ANR) n^o ANR-24-CE23-0380.

Conflict of interest statement

The author declares no conflict of interest.

Ethics approval and Consent to participate

Not applicable.

Data Availability

All source code and demonstration data are openly available at https://github.com/SMTorg/smt-design-space-ext/. Additional datasets are available on request.

Author Contributions

P. Saves: conceptualization, software, writing, formal modeling; E. Hallé-Hannan: experimental analysis, writing, outline; J. Bussemaker: formal modeling, Contributions drafting; Y. Diouane: supervision, review; N. Bartoli: supervision, review. All authors approved the final manuscript.

A Data representation of structured spaces

In general, structured representations can be categorized into three major types: graph-based, logic-based, and frame-based representations [188], although some models fall outside this classification (*e.g.*, general matrices [121]).

Graph-based representations are particularly powerful in fields such as structural pattern recognition and network analysis. In these models, data is represented as graphs, where nodes correspond to entities and edges to the relationships between them. This approach is highly valued for its invariance to transformations, making it robust in various applications where the structure of the data is as important as the content itself [110]. Recent advances in causal representation learning extend these models by approximating latent causal structures from high-dimensional data [53]. This data-driven approach moves beyond correlations and supports out-of-distribution generalization and planning [193, 256].

Logic-based representations are another form of structured representation, widely used in domains such as inductive logic programming and semantic web technologies. These models use logical statements to represent data, enabling complex inferences and generalizations [177]. The key advantage of logic-based representations lies in their ability to encode generalization and inference naturally, allowing for the incorporation of background knowledge through rules. This makes them particularly effective in domains requiring complex reasoning and knowledge representation. For instance, within a data landscape, logic-based representations can show semantic relationships and logical dependencies across different systems or domains, enhancing the ability to spot overarching patterns in knowledge [114]. Horn clauses, description logics, and order-sorted feature structures are key logic-based formalisms. Horn clauses enable software verification and automated theorem proving [14]. Description logics balance expressive power and computational efficiency, sitting between propositional and first-order logic [35]. Order-sorted feature structures, a subset of first-order logic, are used in case-based reasoning and natural language processing [148].

Frame-based representation is a classical approach to knowledge modeling, commonly used in domains like case-based reasoning and Artificial intelligence [176]. Knowledge is organized

using attribute-value pairs within a hierarchical structure, where each frame represents a concept or entity. This structure allows properties to be inherited from parent frames, facilitating efficient representation of complex knowledge [237]. Frames also support default values and data types, enhancing their utility in knowledge-based systems [237]. Frames enable reasoning and inference, making them valuable for tasks like automated theorem proving and decision-making [176]. Though similar to other methods like knowledge graphs [151], semantic networks [202], and concept maps [186], frame-based systems remain integral to structured knowledge representation, particularly in case-based reasoning and statistical relational learning [114].

Structured data representations play a crucial role across various domains of machine learning and artificial intelligence, each tailored to the specific demands of different tasks and applications. Recent advances in supervised learning of representations have shown great promise in automatically extracting task-specific features from structured spaces, especially in categorical and graph-based data. These methods not only enhance flexibility but also significantly improve the predictive performance of models [187, 263]. Moreover, supervised learning approaches have been particularly effective in identifying meaningful, task-driven features from complex data structures, offering robust solutions in fields like graph-based modeling and few-shot learning [77, 99]. As research progresses, selecting the appropriate data representation remains a critical factor, as it directly influences the effectiveness of learning algorithms and their ability to generalize across various applications. For example, deep representation learning continues to expand in its ability to handle structured data, which poses unique challenges due to its high dimensionality and inherent complexity [193].

A critical data-issue when working with structured spaces is the presence of *missing values* and the mechanism behind them. Following Rubin's classical taxonomy [217], missingness is characterised as Missing Completely At Random (MCAR) when the fact of missing does not depend on any observed or unobserved data, Missing At Random (MAR) when the missingness depends only on observed values, and Missing Not At Random (MNAR) when missingness depends on the unobserved (missing) values themselves which is exactly the case in our study [66]. In many high-dimensional or structured-data settings (*e.g.*, graphs, feature matrices, heterogeneous data types) these distinctions remain critical for modeling, imputation and inference, because the mechanism determines whether standard approaches (*e.g.*, deletion, mean-imputation, matrix-completion) are valid or biased. In particular, in our case, under MNAR one must explicitly model the missingness mechanism or risk biased learning or poor generalisation [147]. Recognising and properly handling missingness is thus important in representation learning from structured spaces, for example, when elements of a graph (nodes, edges) or frame attributes are missing non-randomly, this can distort learned

embeddings or downstream predictions [146]. A major interest of such statistical tools in our context is that they can be used if the graph structure is both implicit and unknown to uncover it without any knowledge nor specifications.

B SMT Design Space Graph implementation usage

In this section, we present, in Figure 7.14, the code that generates the hierarchical design space graph of the Multi-Layer Perceptron hierarchical design space. This design space is adapted for visualization, distance computation, and surrogate modeling. Recall that the best option for architecture modeling is the DSG of the adsg-core toolbox that is fast and gives explicit visualizations of the DSG in DOT language, draw.io or GML [98], and our main software contribution is to allow for combining DSG with SMT to define hierarchical surrogate models. For architecture modeling, we recommend the DSG implementation provided by the `adsg-core` library: it extends the DSG concept with model-based systems engineering primitives. Also, this implementation offers a well-documented library API for programmatic graph construction and can export graphs into DOT language, draw.io, and GML formats [98]. Moreover, ADORE is a user-friendly GUI for building, visualizing and wiring ADSG models to evaluation and optimization backends; however, exhaustive enumeration of very large configuration families remains an implementation and performance challenge governed by the ADSG APIs and the chosen encoder implementations. Still, in practice, the ADORE/ADSG/DSG toolchain automates generation and sampling of parameterised families, while the encoder choice (complete vs. fast/lazy) controls whether combinations are enumerated exhaustively or handled through more scalable decoding and correction strategies [60, 62].

```
# Define the mixed hierarchical design space
design_space = DesignSpaceGraph([
    FloatVariable (0, 1), #Learning rate
    CategoricalVariable (["ReLU","Sigmoid","Tanh"]), #3 possible choices for the activation
        function
    CategoricalVariable (["ASGD","Adam"]), #2 possible choices for the optimizer
    FloatVariable (0, 1), #ASGD Decay
    FloatVariable (0, 1), #ASGD Power update
    FloatVariable (0, 1), #ASGD Average start
    FloatVariable (0, 1), #Adam Running Average 1
    FloatVariable (0, 1), #Adam Running Average 2
    FloatVariable (0, 1), #Adam Numerical Stability
    IntegerVariable (1, 3), #for the number of hidden layers  (l=x9)
    OrdinalVariable (['25','30','35','40','45']), #number of hidden neurons layer 1
    OrdinalVariable (['25','30','35','40','45']), #number of hidden neurons layer 2
    OrdinalVariable (['25','30','35','40','45']), #number of hidden neurons layer 3
])

#ASGD vs Adam optimizer options activated or deactivated
design_space.declare_decreed_var(decreed_var =3, meta_var =2, meta_value =["ASGD"])
design_space.declare_decreed_var(decreed_var =4, meta_var =2, meta_value =["ASGD"])
design_space.declare_decreed_var(decreed_var =5, meta_var =2, meta_value =["ASGD"])
design_space.declare_decreed_var(decreed_var =6, meta_var =2, meta_value =["Adam"])
design_space.declare_decreed_var(decreed_var =7, meta_var =2, meta_value =["Adam"])
design_space.declare_decreed_var(decreed_var =8, meta_var =2, meta_value =["Adam"])

# Number of hidden layers: Activate x11 when x9 in [2, 3] and x12 when x9 == 3
design_space.add_value_constraint(var1=9, value1=3, var2=2, value2=["Adam"])  # Forbid 3
    hidden layers with Adam
design_space.declare_decreed_var(decreed_var =10, meta_var =9, meta_value =[1,2,3])
design_space.declare_decreed_var(decreed_var =11, meta_var =9, meta_value =[2,3])
design_space.declare_decreed_var(decreed_var =12, meta_var =9, meta_value =3)
design_space.add_value_constraint(var1=10, value1=['40','45'], var2=2, value2=["ASGD"])  #
    Forbid more than 35 neurons with ASGD
design_space.add_value_constraint(var1=11, value1=['40','45'], var2=2, value2=["ASGD"])  #
    Forbid more than 35 neurons with ASGD
design_space.add_value_constraint(var1=12, value1=['40','45'], var2=2, value2=["ASGD"])  #
    Forbid more than 35 neurons with ASGD
```

Figure 7.14 `AdsgDesignSpaceImpl` definition for the neural network problem.

C Symmetric Positive Definiteness of hierarchical kernels

In this section, we show that our proposed hierarchical kernel—built from neutral, meta-level, and decreed-variable components—yields a symmetric positive-definite covariance function suitable for Gaussian-process models on arbitrarily nested, mixed-type variable spaces. The new kernel k that we propose for hierarchical variables extends a previous work [226] and is given by

$$\begin{aligned} k(\mathbf{X}, \mathbf{X}') = \quad & k_{\text{neu}}(\mathbf{X}_{\text{neu}}, \mathbf{X}'_{\text{neu}}) \times k_{\text{m}}(\mathbf{X}_{\text{m}}, \mathbf{X}'_{\text{m}}) \times \\ & k_{\text{m,dec}}([\mathbf{X}_{\text{m}}, \mathbf{X}_{\text{inc}}(\mathbf{X}_{\text{m}})], [\mathbf{X}'_{\text{m}}, \mathbf{X}'_{\text{inc}}(\mathbf{X}'_{\text{m}})]), \end{aligned} \tag{G.9}$$

where k_{neu}, k_{m} and $k_{\text{m,dec}}$ are as follows:

- k_{neu} represents the neutral kernel that encompasses both categorical and quantitative neutral variables, *i.e.*, k_{neu} can be decomposed into two parts $k_{\text{neu}}(\mathbf{X}_{\text{neu}}, \mathbf{X'}_{\text{neu}}) = k^{\text{cat}}(\mathbf{X}^{\text{cat}}_{\text{neu}}, \mathbf{X'}^{\text{cat}}_{\text{neu}}) \times k^{\text{qnt}}(\mathbf{X}^{\text{qnt}}_{\text{neu}}, \mathbf{X'}^{\text{qnt}}_{\text{neu}})$. The categorical kernel, denoted k^{cat}, could be any Symmetric Positive Definite (SPD) [224] mixed kernel. For the quantitative (integer or continuous) variables, a distance-based kernel is used. The chosen quantitative kernel (*e.g.*, exponential or Matérn) always depends on a given distance d. For example, the n-dimensional exponential kernel gives

$$k^{\text{qnt}}(\mathbf{X}^{\text{qnt}}, \mathbf{X'}^{\text{qnt}}) = \prod_{i=1}^{n} \exp(-d(\mathbf{X}^{\text{qnt}}_i, \mathbf{X'}^{\text{qnt}}_i)). \tag{G.10}$$

- k_{m} is the meta variables related kernel. It is also separated into two parts: $k_{\text{m}}(\mathbf{X}_{\text{m}}, \mathbf{X'}_{\text{m}}) = k^{\text{cat}}(\mathbf{X}^{\text{cat}}_{\text{m}}, \mathbf{X'}^{\text{cat}}_{\text{m}}) k^{\text{qnt}}(\mathbf{X}^{\text{qnt}}_{\text{m}}, \mathbf{X'}^{\text{qnt}}_{\text{m}})$ where the quantitative kernel is ordered and not continuous because meta variables take value in a finite set.
- $k_{\text{m,dec}}$ is an SPD kernel that models the correlations between the meta levels (all the possible subspaces) and the decreed variables. It is a quantitative kernel depending on the aforementioned graph-structure distance $\overline{\text{dist}}$. In particular, we have developed several continuous relaxations to make any distance, such as the new graph-structure one, to induce a categorical kernel. This allows us to generalize here the previous kernel [221] to also handle decreed categorical variables as described in [66].

Theorem 12. *The kernel K^{naive} defined as follows is not a SPD kernel.*

$$\begin{cases} K^{naive}(\mathbf{X}, \mathbf{X'}) = k_{\text{dec}}(\mathbf{X}_{\text{inc}}(\mathbf{X'}_{\text{m}}), \mathbf{X'}_{\text{inc}}(\mathbf{X'}_{\text{m}})) k_{\text{neu}}(\mathbf{X}_{\text{neu}}, \mathbf{X'}_{\text{neu}}), & \text{if } \mathbf{X}_{\text{m}} = \mathbf{X'}_{\text{m}} \\ K^{naive}(\mathbf{X}, \mathbf{X'}) = k_{\text{m}}(\mathbf{X}_{\text{m}}, \mathbf{X'}_{\text{m}}) k_{\text{neu}}(\mathbf{X}_{\text{neu}}, \mathbf{X'}_{\text{neu}}), & \text{if } \mathbf{X}_{\text{m}} \neq \mathbf{X'}_{\text{m}} \end{cases} \tag{G.11}$$

Proof. K^{naive} is a kernel if and only if K^{naive} can be written as $K^{naive}(\mathbf{X}, \mathbf{X'}) = \kappa(d(\mathbf{X}, \mathbf{X'}))$ with κ being an SPD function and d being a distance in a given Hilbert space. If the two continuous decreed inputs $\mathbf{X}_{dec}$ and $\mathbf{X'}_{dec}$ are in the same subspace ($\mathbf{X}_{\text{m}} = \mathbf{X'}_{\text{m}}$), then $k_{\text{m}}(\mathbf{X}_{\text{m}}, \mathbf{X}_{\text{m}}) = 1$ and $K^{naive}(\mathbf{X}, \mathbf{X'}) = k_{\text{dec}}(\mathbf{X}_{\text{inc}}(\mathbf{X}_{\text{m}}), \mathbf{X'}_{\text{inc}}(\mathbf{X'}_{\text{m}}))$. This works without any problem, a categorical variable being fully correlated with itself.

On the contrary, if the two continuous inputs $\mathbf{X}_{\text{dec}}$ and $\mathbf{X'}_{\text{dec}}$ lie in two different subspaces ($\mathbf{X}_{\text{m}} \neq \mathbf{X'}_{\text{m}}$), then, we have $k_{dec}(\mathbf{X}_{\text{dec}}, \mathbf{X'}_{\text{dec}}) = 1 \implies d(\mathbf{X}_{\text{inc}}(\mathbf{X}_{\text{m}}), \mathbf{X'}_{\text{inc}}(\mathbf{X'}_{\text{m}})) = 0$.

We know that a distance is defined such that $d(\mathbf{X}, \mathbf{X'}) = 0 \Leftrightarrow \mathbf{X} = \mathbf{X'}$ (Identity of indiscernibles). Yet, we have $d(\mathbf{X}_{\text{inc}}(\mathbf{X}_{\text{m}}), \mathbf{X'}_{\text{inc}}(\mathbf{X'}_{\text{m}})) = 0$ and $\mathbf{X}_{\text{inc}}(\mathbf{X}_{\text{m}}) \neq \mathbf{X'}_{\text{inc}}(\mathbf{X'}_{\text{m}})$. d is not a distance because it is always equal to 0 for points in distinct decreed spaces. Therefore, the kernel vanishes to 1 and the correlation matrix is degenerated and can not be SPD.

Based on this proof, the idea of the arc-kernel is to have a constant residual distance between distinct subspaces. In other words, there is a non-zero distance between $\mathcal{X}(\mathbf{X}_\mathrm{m})$ and $\mathcal{X}(\mathbf{X'}_\mathrm{m})$ if $\mathbf{X}_\mathrm{m} \neq \mathbf{X'}_\mathrm{m}$.

□

Theorem 13. *Our* `Alg-Kernel` *kernel is SPD and so is* k *defined in* (G.9).

Proof. Let $I_\mathbf{X}$ be the subset of indices $i \in I_\mathrm{dec}$ that are decreed-included by $\mathbf{X}_\mathrm{m}$ such that $I^{inter}_{\mathbf{X},\mathbf{X'}} = I_\mathbf{X} \bigcap I_{\mathbf{X'}}$. Let $d_E([x_\mathbf{X}, x_{\mathbf{X'}}], [y_\mathbf{X}, y_{\mathbf{X'}}]) = \theta_i \sqrt{(x_\mathbf{X} - x_{\mathbf{X'}})^2 + (y_\mathbf{X} - y_{\mathbf{X'}})^2}$ be an Euclidean distance in $\mathbb{R}^2 \times \mathbb{R}^2$. Due to [135, Proposition 2], we only need to show that, for any two inputs $\mathbf{X}, \mathbf{X'} \in \mathcal{X}$, the isometry condition $d_E(f_i^{alg}(\mathbf{X}), f_i^{alg}(\mathbf{X'}))$ holds for a given function f_i^{alg}, that is equivalent to having a Hilbertian metric. In other words, d_E is isomorphic to an L^2 norm [123]. Such kernels are well-known and referred as "substitution kernels with Euclidean distance" [240].

$\forall i \in I_\mathrm{dec}, f_i^{alg}(\mathbf{X}_\mathrm{dec})$ is defined by

$$\begin{cases} f_i^{alg}(\mathbf{X}_\mathrm{inc}) = [\frac{1-((\mathbf{X}_{dec})_i)^2}{1+((\mathbf{X}_{dec})_i)^2}, \frac{2(\mathbf{X}_{dec})_i}{1+((\mathbf{X}_{dec})_i)^2}] \text{ if } i \in I_u \\ f_i^{alg}(\mathbf{X}_\mathrm{dec}) = [0,0] \text{ otherwise} \end{cases} \tag{G.12}$$

Case 1: $i \in I_\mathrm{dec}, i \notin I_\mathbf{X}, i \notin I_{\mathbf{X'}}$.

$$d_E(f_i^{alg}(\overline{\mathbf{X}}_{dec}), f_i^{alg}(\overline{\mathbf{X'}}_{dec})) = d_E([0,0],[0,0]) = 0.$$

Therefore, the complementary space is not relevant, as for the arc-kernel.

Case 2: $i \in I_\mathrm{dec}, i \in I_\mathbf{X}, i \notin I_{\mathbf{X'}}$.

$$\begin{aligned} d_E(f_i^{alg}(\mathbf{X}_{dec}), f_i^{alg}(\overline{\mathbf{X'}}_{dec})) &= d_E([\frac{1-((\mathbf{X}_{dec})_i)^2}{1+((\mathbf{X}_{dec})_i)^2}, \frac{2(\mathbf{X}_{dec})_i}{1+((\mathbf{X}_{dec})_i)^2}], [0,0]) \\ &= \theta_i. \end{aligned}$$

This case corresponds to the kernel $K_\mathrm{m}^{alg}(\mathbf{X}_\mathrm{m}, \mathbf{X'}_\mathrm{m})$ when $\mathbf{X}_\mathrm{m} \neq \mathbf{X'}_\mathrm{m}$.

Case 3: $i \in I_\mathrm{dec}, i \in I^{inter}_{\mathbf{X},\mathbf{X'}}$.

$$
\begin{aligned}
&d_E(f_i^{alg}(\mathbf{X}_{dec}), f_i^{alg}(\mathbf{X'}_{dec})) \\
&= d_E\left(\left[\frac{1-((\mathbf{X}_{dec})_i)^2}{1+((\mathbf{X}_{dec})_i)^2}, \frac{2(\mathbf{X}_{dec})_i}{1+((\mathbf{X}_{dec})_i)^2}\right], \left[\frac{1-((\mathbf{X'}_{dec})_i)^2}{1+((\mathbf{X'}_{dec})_i)^2}, \frac{2(\mathbf{X'}_{dec})_i}{1+((\mathbf{X'}_{dec})_i)^2}\right]\right) \\
&= 2\theta_i \frac{|(\mathbf{X}_{dec})_i - (\mathbf{X'}_{dec})_i|}{\sqrt{((\mathbf{X}_{dec})_i)^2+1}\sqrt{((\mathbf{X'}_{dec})_i)^2+1}}
\end{aligned}
$$

This case corresponds to the kernel $K_{dec}^{alg}(\mathbf{X}_{dec}(\mathbf{X}_m), \mathbf{X'}_{dec}(\mathbf{X'}_m))$ when $\mathbf{X}_m = \mathbf{X'}_m$.

Therefore, there exists an isometry between our algebraic distance and the Euclidean distance. The distance being well-defined for any given kernel, the matrix obtained with our model is SPD.

Although, in [135], they also demonstrate the metric property of their distance formally. The non-negativity and symmetry of d^{alg} are trivially proven knowing that the hyperparameters θ are strictly positive. To prove the triangle inequality, consider $(u, v, w) \in \mathcal{X}^3$.

Case 1: $i \in I_{\text{dec}}, i \notin I_u, i \notin I_v$.

$$d^{\text{alg}}(u_i, v_i) = 0 \leq d^{\text{alg}}(u_i, w_i) + d^{\text{alg}}(w_i, v_i) \text{ by non negativity.}$$

Case 2: $i \in I_{\text{dec}}, i \in I_u, i \notin I_v$.

- $i \in I_w,\ d^{\text{alg}}(u_i, v_i) = \theta_i \leq d^{\text{alg}}(u_i, w_i) + \theta_i$ by non negativity.
- $i \notin I_w,\ d^{\text{alg}}(u_i, v_i) = \theta_i \leq \theta_i + d^{\text{alg}}(w_i, v_i)$ by non negativity.

Case 3: $i \in I_{\text{dec}}, i \in I_{u,v}^{inter}$.

- $i \notin I_w,\ d^{\text{alg}}(u_i, v_i) = 2\theta_i \frac{|u_i - v_i|}{\sqrt{(u_i)^2+1}\sqrt{(v_i)^2+1}} \leq \theta_i + \theta_i$ for $(u_i, v_i) \in [0,1]^2$.
- $i \in I_w$, Knowing that $\frac{|a-c|}{\sqrt{a^2+1}\sqrt{c^2+1}} + \frac{|c-b|}{\sqrt{c^2+1}\sqrt{b^2+1}} \geq \frac{|a-b|}{\sqrt{a^2+1}\sqrt{b^2+1}}$, we have $d^{\text{alg}}(u_i, v_i) = 2\theta_i \frac{|u_i - v_i|}{\sqrt{(u_i)^2+1}\sqrt{(v_i)^2+1}} \leq d^{\text{alg}}(u_i, w_i) + d^{\text{alg}}(w_i, v_i)$.

Using [125, Theorem 3], any graph-structure distance is well-defined if and only if $\sup\{d(\mu, \nu) : \mu, \nu \in \mathcal{X}^2\}/2 \leq \theta_i$. This can make the proof easier and faster as follows. First, we can compute the maximum of the algebraic distances which gives $\max_{(x,y)\in\mathbb{R}^2} d^{\text{alg}}(x, y) = \max_{(x,y)\in\mathbb{R}^2} 2\theta_i|x-y|\sqrt{\frac{1}{x^2+1}}\sqrt{\frac{1}{y^2+1}} = d^{\text{alg}}(x, -\frac{1}{x}) = 2\theta_i$. Then, we just have to check that $2\theta_i/2 \leq \theta_i$ which is the case. This theorem brings two interesting insights on our distance,

first, it is a limit case as the distance between two non-interacting subspaces is properly maximized, and second, the algebraic circle-based distance is well-constructed as the maximal distance corresponds to the maximal arc of 1 radian, making it a generalization of the Arc-distance [267].

□

CHAPTER 8 ADDITIONAL CONTRIBUTIONS

Now is that a real poncho or is that a Sears poncho?
Don't you know?
You could make more money as a butcher

— Frank Zappa, *Cosmik Debris*

This chapter presents secondary contributions that are not submitted or published in peer-reviewed outlets, but are still worth mentioning. Section 8.1 presents the technical report containing the created problems for benchmarking CatMADS and CatMADS$_{\text{GP}}$ in projects 1 and 2, respectively presented in Chapters 4 and 5.

8.1 Cat-Suite: A collection of optimization problems with categorical and quantitative variables for benchmarking

Authors. Edward Hallé-Hannan, Charles Audet, Youssef Diouane, Sébastien Le Digabel and Christophe Tribes

Abstract. Benchmarking new optimization methods on test problems is essential for assessing their performance and tuning their parameters. Yet, few problems are available in the literature for mixed-variable optimization. This manuscript introduces Cat-Suite, a collection of optimization problems involving categorical, integer, and continuous variables. Currently, Cat-Suite includes 60 analytical problems, half of which are constrained and half unconstrained. Each of these problems is either a modification of an existing one from the literature or derived from classical continuous test functions, such as Rosenbrock. Cat-Suite is an ongoing project, and additional problems will be added in the future. The goal is to facilitate benchmarking and testing of novel mixed-variable optimization methods with a diverse problem collection. The problems are implemented in Python and available at https://github.com/bbopt/Cat-Suite.

Keywords. Optimization, benchmarking, mixed-variable, categorical variables, constraints.

The corresponding technical report is available at

https://www.gerad.ca/en/papers/G-2025-39.

8.2 Contributions to the software NOMAD

All the methodological advancements in the articles 1 (CatMADS) and 2 (Surrogate-based neighborhoods) have been fully integrated in the NOMAD software package [31]. These include:

1. categorical neighborhoods with IDW and GPs;
2. alternating categorical and quantitative polls;
3. generalized extended poll with the progressive barrier;
4. ordering poll candidate points with mixed-variable surrogate models;
5. adapting many quantitative search methods into mixed-variable methods, such as the speculative and quadratic search.

A beta release in Python will be available with the next NOMAD 4.6 release, expected in June 2026.

CHAPTER 9 CONCLUSION AND GENERAL DISCUSSIONS

To think I did all that
And may I say, not in a shy way
Oh, no, oh, no, not me
I did it my way

— Frank Sinatra, *My Way*

9.1 Summary of the Thesis

The first research project introduces CatMADS, a generalization of MADS for constrained blackbox optimization problems involving categorical, integer, and continuous variables. The method structures categorical variables through distance-based neighborhoods. Polling mechanisms are developed from these neighborhoods to handle categorical variables within a direct-search framework. Integer and continuous variables are treated using generalized MADS mechanisms. The progressive barrier and the extended poll are also extended so that they work together in the mixed-variable setting. Different definitions of mixed-variable minima are introduced to establish convergence results with varying trade-offs between evaluation cost and the strength of the targeted minimum.

The second research project focuses on improving how categorical variables are handled in mixed-variable optimization. The work proposes the automatic construction of categorical neighborhoods using distances induced by surrogate models. These neighborhoods are problem-specific structures that capture similarities between categorical variables with respect to both the objective and the constraints. They are particularly well suited for constrained problems. The neighborhoods are incorporated into CatMADS as a proof of concept. Gaussian processes are used as surrogate models to construct the neighborhoods, leading to a variant of CatMADS called $\text{CatMADS}_{\text{GP}}$. This variant outperforms several popular state-of-the-art solvers. This suggests that surrogate-based neighborhoods provide a principled approach for improving the optimization of categorical variables.

The last two projects address problems that are more generally hierarchical. The third project focuses on the fundamental issue of comparing points in hierarchical domains that do not necessarily share the same variables or are subject to the same bounds. Standard distances assume shared variables and bounds. The proposed meta distance maps hierarchical domains to an extended domain containing all possible variables. This distance can be used

in various machine learning tasks and optimization methods for hierarchical mixed-variable domains. The final project is a direct continuation of the hierarchical framework developed for the meta distance. The work introduces a framework integrating several key concepts of the literature of hierarchical domains and generalizing the framework of the meta distance. This framework was implemented within SMT and used to develop a mixed-variable and hierarchical optimization method. The method is successfully applied to a real-life aircraft design engineering problem.

9.2 Limitations and future work

In the first and second projects, the neighborhoods only allow changes to the categorical variables. The extended poll is used to calibrate the quantitative variables when the decrease in the objective function falls within a given parameter. This parameter may be highly problem dependent and difficult to tune properly. Mechanisms that do not rely on such parameters and that optimize categorical and quantitative variables simultaneously may therefore be desirable. For instance, a trust-region step based on a quadratic model of the quantitative variables could be launched from categorical neighbors [86,87]. This could make it possible to identify categorical variables that are discarded by the parameter of the extended poll or that require more exploration than a quantitative poll allows. After spending additional low-cost effort calibrating the quantitative variables using a model, these categorical variables could become quite promising. Moreover, CatMADS and CatMADS$_{\mathsf{GP}}$ have no complexity analysis, and there is currently no strong justification for using MADS as the underlying method for optimizing quantitative variables. The categorical neighborhood mechanisms could also be integrated with other derivative-free approaches. Exploring alternatives to MADS, such as sufficient decrease methods, could enable broader convergence analysis.

CatMADS, CatMADS$_{\mathsf{GP}}$, and the meta distance do not account for noisy functions, which are typically present in real-life blackbox applications [24, 76]. The initial results obtained are promising, but most experiments were done on analytical problems that do not necessarily reflect the noisy blackbox problems. Benchmarking and testing on true blackbox applications are necessary to confirm that the proposed methods and structures significantly improve performance. It is quite possible that extensions for handling stochasticity are needed to tackle such applications efficiently [76].

The hierarchical BO method from the fourth project does not address convergence guarantees. The categorical neighborhoods developed in the earlier projects cannot handle hierarchical domains. A major research direction would be to generalize these neighborhoods so that they can handle meta variables. Such neighborhoods could be defined between subspaces that do

not share the same variables or bounds, and could drive new optimization mechanisms for hierarchical domains. That said, hierarchical optimization is currently not well formalized. Even the standard notion of a local minimum has not yet been established. Further work on formalizing hierarchical optimization may therefore be required before developing hierarchical methods with convergence guarantees, that is beyond the current BO approaches.

9.3 Synthesis and concluding remarks

The four projects of the thesis largely focus on structuring and modeling complex mixed-variable and hierarchical domains. A common theme across these projects is the exploitation of available information to introduce structure into variables that are otherwise difficult to handle. In particular, the first and second projects address categorical variables by constructing problem-specific distances that capture similarities between categorical variables. These distances are then used to define neighborhoods guiding the optimization of categorical variables. The meta distance expands this idea by introducing parameters that allow the comparison of points whose variables may be included or excluded. Overall, the contributions of the thesis are primarily modeling ideas that improve optimization performance.

Globally, the thesis introduces several contributions advancing mixed-variable and hierarchical optimization. However, this class of problems remains relatively underdeveloped. Many challenges remain to be addressed, particularly in light of the limitations discussed in the previous section. Nevertheless, the ideas proposed in this thesis demonstrate strong practical potential. The computational experiments suggest that significant improvements in optimization performance can be obtained using relatively simple modeling ideas that focus on comparisons and similarities. These results represent meaningful steps toward better methods for mixed-variable and hierarchical optimization. It is hoped that this work can be further developed, as many important applications belong to these classes of problems, such as hyperparameter optimization and several engineering design problems.

REFERENCES

[1] A. Abbas and I. F. Siddiquiand S. U. Lee. Multi-objective optimization of feature model in software product line: Perspectives and challenges. *Indian Journal of Science and Technology*, 2016.

[2] M.A. Abramson. *Pattern Search Algorithms for Mixed Variable General Constrained Optimization Problems*. PhD thesis, Department of Computational and Applied Mathematics, Rice University, August 2002.

[3] M.A. Abramson. Mixed Variable Optimization of a Load-Bearing Thermal Insulation System Using a Filter Pattern Search Algorithm. *Optimization and Engineering*, 5(2):157–177, 2004.

[4] M.A. Abramson, C. Audet, J.W. Chrissis, and J.G. Walston. Mesh Adaptive Direct Search Algorithms for Mixed Variable Optimization. *Optimization Letters*, 3(1):35–47, 2009.

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

[6] R. Agrawal, C. Faloutsos, and A. Swami. Efficient similarity search in sequence databases. In *Foundations of Data Organization and Algorithms, FODO'93*, 1993.

[7] A. Ahmad and L. Dey. A k-mean clustering algorithm for mixed numeric and categorical data. *Data & Knowledge Engineering*, 63(2):503–527, 2007.

[8] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama. Optuna: A Next-generation Hyperparameter Optimization Framework. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Association for Computing Machinery, 2019.

[9] S. Alarie, C. Audet, A.E. Gheribi, M. Kokkolaras, and S. Le Digabel. Two decades of blackbox optimization applications. *EURO Journal on Computational Optimization*, 9:100011, 2021.

[10] A. Aleti, B. Buhnova, L. Grunske, A. Koziolek, and I. Meedeniya. Software Architecture Optimization Methods: A Systematic Literature Review. *IEEE Transactions on Software Engineering*, 39(5):658–683, 2013.

[11] N. Ali, D. Neagu, and P. Trundle. Classification of Heterogeneous Data Based on Data Type Impact on Similarity. In *Advances in Computational Intelligence Systems*. Springer International Publishing, 2019.

[12] N. Ali, D. Neagu, and P. Trundle. Evaluation of k-nearest neighbour classifier performance for heterogeneous data sets. *SN Applied Sciences*, 1:1–15, 2019.

[13] N. Andrés-Thió, C. Audet, M. Diago, A.E. Gheribi, S. Le Digabel, X. Lebeuf, M. Lemyre Garneau, and C. Tribes. Solar: a solar thermal power plant simulator for blackbox optimization benchmarking. *Optimization and Engineering*, 26(3):1815–1861, 2025.

[14] E. De Angelis, F. Fioravanti, J. Gallagher, M. Hermenegildo, A. Pettorossi, and M. Proietti. Analysis and transformation of constrained horn clauses for program verification. *Theory and Practice of Logic Programming*, 22:974–1042, 2022.

[15] R. Angles and C. Gutierrez. *Graph Data Management: Fundamental Issues and Recent Developments*. Springer Series in Data-Centric Systems and Applications, 2018.

[16] N. Aronszajn. Theory of Reproducing Kernels. *Transactions of the American mathematical society*, 68(3):337–404, 1950.

[17] M. Asadi, G. Gröner, B. Mohabbati, and D. Gašević. Goal-oriented modeling and verification of feature-oriented product lines. *Software & Systems Modeling*, 15:257–279, 2016.

[18] C. Audet, V. Béchard, and S. Le Digabel. Nonsmooth optimization through Mesh Adaptive Direct Search and Variable Neighborhood Search. *Journal of Global Optimization*, 41(2):299–318, 2008.

[19] C. Audet and J.E. Dennis, Jr. Pattern Search Algorithms for Mixed Variable Programming. *SIAM Journal on Optimization*, 11(3):573–594, 2001.

[20] C. Audet and J.E. Dennis, Jr. Analysis of Generalized Pattern Searches. *SIAM Journal on Optimization*, 13(3):889–903, 2003.

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

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

[23] C. Audet, Y. Diouane, E. Hallé-Hannan, S. Le Digabel, and C. Tribes. CatMADS: Mesh Adaptive Direct Search for constrained blackbox optimization with categorical variables. Technical Report G-2025-42, Les cahiers du GERAD, 2025.

[24] C. Audet, K.J. Dzahini, M. Kokkolaras, and S. Le Digabel. Stochastic mesh adaptive direct search for blackbox optimization using probabilistic estimates. *Computational Optimization and Applications*, 79(1):1–34, 2021.

[25] C. Audet, E. Hallé-Hannan, and S. Le Digabel. A General Mathematical Framework for Constrained Mixed-variable Blackbox Optimization Problems with Meta and Categorical Variables. *Operations Research Forum*, 4(12), 2023.

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

[27] C. Audet and W. Hare. *Derivative-Free and Blackbox Optimization*. Springer Series in Operations Research and Financial Engineering. Springer, Cham, Switzerland, 2nd edition, 2026. In preparation, 425 pages.

[28] C. Audet, W. Hare, and C. Tribes. A summary of benchmarking constrained, multi-objective and surrogate-assisted optimization methods. *To appear in Optimization Letters*, 2026.

[29] C. Audet, M. Kokkolaras, S. Le Digabel, and B. Talgorn. Order-based error for managing ensembles of surrogates in mesh adaptive direct search. *Journal of Global Optimization*, 70(3):645–675, 2018.

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

[31] C. Audet, S. Le Digabel, V. Rochon Montplaisir, and C. Tribes. The NOMAD project. Software available at https://www.gerad.ca/nomad, 2022.

[32] C. Audet, S. Le Digabel, and C. Tribes. Dynamic scaling in the mesh adaptive direct search algorithm for blackbox optimization. *Optimization and Engineering*, 17(2):333–358, 2016.

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

[34] C. Audet and C. Tribes. Mesh-based Nelder-Mead algorithm for inequality constrained optimization. *Computational Optimization and Applications*, 71(2):331–352, 2018.

[35] F. Baader, D. Calvanese, D. Mcguinness, D. Nardi, and P. Patel-Schneider. *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge University Press, 2007.

[36] P. Balança and E. Herbin. A set-indexed ornstein-uhlenbeck process. *Electron. Commun. Probab.*, 17:1–14, 2012.

[37] J. Bang-Jensen and M. Kriesell. Disjoint directed and undirected paths and cycles in digraphs. *Theoretical Computer Science*, 410:5138–5144, 2009.

[38] L. Baraton, A. Urbano, L. Brevault, and M. Balesdent. Comparative review of Multidisciplinary Design Analysis and Optimization architectures for the preliminary design of a liquid rocket engine. In *Aerospace Europe Conference 2023*. EUCASS, 2023.

[39] N. Bartoli, T. Lefebvre, S. Dubreuil, R. Olivanti, R. Priem, N. Bons, J.R.R.A Martin, and J. Morlier. Adaptive modeling strategy for constrained global optimization with application to aerodynamic wing design. *Aerospace Science and Technology*, 90:85–102, 2019.

[40] N. Bartoli, T. Lefebvre, R. Lafage, P. Saves, Y. Diouane, J. Morlier, J.H. Bussemaker, G. Donelli, J. M. Gomes de Mello, M. Mandorino, and P. Della Vecchia. Multi-objective Bayesian optimization with mixed-categorical design variables for expensive-to-evaluate aeronautical applications. In *AeroBest 2023*, 2023.

[41] D. Batory. Feature models, grammars, and propositional formulas. In *International Conference on Software Product Lines*. Springer, 2005.

[42] G. Beer. The Structure of Extended Real-valued Metric Spaces. *Set-Valued and Variational Analysis*, 21:591–602, 2013.

[43] D. Benavides, S. Segura, and A. Ruiz-Cortés. Automated analysis of feature models 20 years later: A literature review. *Information Systems*, 35(6):615–636, 2010.

[44] M. Bender, M. Farach-Colton, G. Pemmasani, S. Skiena, and P. Sumazin. Lowest common ancestors in trees and directed acyclic graphs. *Journal of Algorithms*, 57(2):75–94, 2005.

[45] J. Bergstra, R. Bardenet, Y. Bengio, and K. Balázs. Algorithms for Hyper-Parameter Optimization. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, pages 2546–2554, Red Hook, NY, U.S.A., 2011. Curran Associates Inc.

[46] A. Bhosekar and M. Ierapetritou. Advances in surrogate based modeling, feasibility analysis, and optimization: A review. *Computers & Chemical Engineering*, 108:250–267, 2018.

[47] D. Bihanic and T. Polacsek. Models for visualisation of complex information systems. In *16th International Conference on Information Visualisation*, 2012.

[48] M. Binois and N. Wycoff. A survey on high-dimensional Gaussian process modeling with application to Bayesian optimization. Technical Report 2111.05040, arXiv, 2021.

[49] J. Blank and K. Deb. Pymoo: Multi-Objective Optimization in Python. *IEEE Access*, 8:89497–89509, 2020.

[50] D. Blumenthal and J. Gamper. On the exact computation of the graph edit distance. *Pattern Recognition Letters*, 134:46–57, 2020.

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

[52] M. A. Bouhlel, J. T. Hwang, N. Bartoli, R. Lafage, J. Morlier, and J. R. R. A. Martins. A Python surrogate modeling framework with derivatives. *Advances in Engineering Software*, 135:102662, 2019.

[53] T. Bourdais, P. Batlle, X. Yang, R. Baptistao, N. Rouquette, and H. Owhadi. Codiscovering graphical structure and functional relationships within data: A gaussian process framework for connecting the dots. *Proceedings of the National Academy of Sciences*, 121(32):e2403449121, 2024.

[54] A. Bravais. Analyse mathématique sur les probabilités des erreurs de situation d'un point. *Mémoires de l'Académie royale des sciences de l'Institut imperial de France*, 1844.

[55] D. Brigo, F. Mercurio, and F. Rapisarda. Parameterizing correlations: a geometric interpretation. *IMA Journal of Management Mathematics*, 18(1):55–73, 2007.

[56] J-M. Bruel, B. Combemale, E. Guerra, J-M. Jézéquel, J. Kienzle, J. De Lara, G. Mussbacher, E. Syriani, and H. Vangheluwe. Comparing and classifying model transformation reuse approaches across metamodels. *Software and Systems Modeling*, 19:441–465, 2020.

[57] H. Busemann. The isoperimetric problem in the minkowski plane. *American Journal of Mathematics*, 69:863–871, 1947.

[58] J. Buss and J. Goldsmith. Nondeterminism within P^*. *SIAM Journal on Computing*, 22:560–572, 1993.

[59] J.H. Bussemaker. SBArchOpt: Surrogate-based architecture optimization. *Journal of Open Source Software*, 8:5564, 2023.

[60] J.H. Bussemaker. *System Architecture Optimization: Function-Based Modeling, Optimization Algorithms, and Multidisciplinary Evaluation*. PhD thesis, Delft University of Technology, 2025.

[61] J.H. Bussemaker, L. Boggero, and P. D. Ciampa. From system architecting to system design and optimization: A link between MBSE and MDAO. In *INCOSE International Symposium*, 2022.

[62] J.H. Bussemaker, L. Boggero, and B. Nagel. System architecture design space exploration: Integration with computational environments and efficient optimization. In *AIAA AVIATION 2024 FORUM*, 2024.

[63] J.H. Bussemaker, P.D. Ciampa, T. De Smedt, B. Nagel, and G. La Rocca. System Architecture Optimization: An Open Source Multidisciplinary Aircraft Jet Engine Architecting Problem. In *AIAA AVIATION 2021 Forum*. American Institute of Aeronautics and Astronautics, 2021.

[64] J.H. Bussemaker, P.D. Ciampa, and B. Nagel. System architecture design space exploration: An approach to modeling and optimization. In *AIAA AVIATION 2020 Forum*, AIAA AVIATION Forum. American Institute of Aeronautics and Astronautics, 2020.

[65] J.H. Bussemaker, R. Sánchez, M. Fouda, L. Boggero, and B. Nagel. Function-based architecture optimization: An application to hybrid-electric propulsion systems. In *INCOSE international symposium*, volume 33, pages 251–272, 2023.

[66] J.H. Bussemaker, P. Saves, N. Bartoli, T. Lefebvre, and R. Lafage. System architecture optimization strategies: Dealing with expensive hierarchical problems. *Journal of Global Optimization*, 90:1–45, 2024.

[67] J.H. Bussemaker, T. De Smedt, G. La Rocca, P. D. Ciampa, and B. Nagel. System architecture optimization: An open source multidisciplinary aircraft jet engine architecting problem. In *AIAA AVIATION 2021 Forum*, page 3078, 2021.

[68] D. Butturi-Gomes and M. Petrere. Edge influence and population aggregation: On point and interval statistical performances of morisita patchiness index estimators in different sampling schemes. *Ecological Indicators*, 108:105736, 2020.

[69] R. Carpintero Perez, S. Da Veiga, J. Garnier, and B. Staber. Gaussian process regression with sliced wasserstein weisfeiler-lehman graph kernels. In *International Conference on Artificial Intelligence and Statistics*, 2024.

[70] E. Cazelles, A. Robert, and F. Tobar. The wasserstein-fourier distance for stationary time series. *IEEE Transactions on Signal Processing*, 69:709–721, 2020.

[71] P-A. Champin and C. Solnon. Measuring the similarity of labeled graphs. In *Case-Based Reasoning Research and Development: 5th International Conference on Case-Based Reasoning*, 2003.

[72] A. Chan, A. Pires, T. Polacsek, S. Roussel, F. Bouissière, C. Cuiller, and P-E Dereux. Goal modelling: Design and manufacturing in aeronautics. In *International Conference on Research Challenges in Information Science*, 2023.

[73] A. Chan, A. Fernandes Pires, and T. Polacsek. Trying to elicit and assign goals to the right actors. In *Conceptual Modeling: 41st International Conference, ER 2022*, 2022.

[74] G. Chartrand, G. Kubicki, and M. Schultz. Graph similarity and distance in graphs. *Aequationes Mathematicae*, 55:129–145, 1998.

[75] J. Chen, Y. Ng, L. Lin, X. Zhang, and S. Li. On triangle inequalities of correlation-based distances for gene expression profiles. *BMC bioinformatics*, 24:40, 2023.

[76] R. Chen, M. Menickelly, and K. Scheinberg. Stochastic optimization using a trust-region method and random models. *Mathematical Programming*, 169(2):447–487, 2018.

[77] X. Chen, C. Ge, M. Wang, and J. Wang. Supervised contrastive few-shot learning for high-frequency time series. In *Proceedings of the 37th AAAI Conference on Artificial Intelligence (AAAI 2023)*, 2023.

[78] H. Cho, Y. Kim, E. Lee, D. Choi, Y. Lee, and W. Rhee. Basic enhancement strategies when using Bayesian optimization for hyperparameter tuning of deep neural networks. *IEEE access*, 8:52588–52608, 2020.

[79] T.D. Choi and C.T. Kelley. Superlinear convergence and implicit filtering. *SIAM Journal on Optimization*, 10(4):1149–1162, 2000.

[80] J. Choo, S. Bohn, G.C. Nakamura, A.M. White, and H. Park. Heterogeneous Data Fusion via Space Alignment Using Nonmetric Multidimensional Scaling. In *Proceedings of the 2012 SIAM International Conference on Data Mining*, pages 177–188. SIAM, 2012.

[81] R. Cilibrasi and P. Vitányi. Clustering by compression. *IEEE Transactions on Information theory*, 51:1523–1545, 2005.

[82] F.H. Clarke. *Optimization and Nonsmooth Analysis*. John Wiley and Sons, New York, 1983. Reissued in 1990 by SIAM Publications, Philadelphia, as Vol. 5 in the series Classics in Applied Mathematics.

[83] G. Coenders and W. Saris. Categorization and measurement quality. the choice between pearson and polychoric correlations. *The Multitrait-Multimethod approach to evaluate measurement instruments*, pages 125–144, 1995.

[84] F. Comellas and J. Ozón. Graph Coloring Algorithms for Assignment Problems in Radio Networks. In *Applications of Neural Networks to Telecommunications 2*, pages 49–56, Hillsdale, NJ, USA, 1995. Lawrence Erlbaum Associates.

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

[86] A.R Conn, K. Scheinberg, and L.N. Vicente. Global convergence of general derivative-free trust-region algorithms to first and second order critical points. *SIAM Journal on Optimization*, 20(1):387–415, 2009.

[87] A.R. Conn, K. Scheinberg, and L.N. Vicente. *Introduction to Derivative-Free Optimization*. MOS-SIAM Series on Optimization. SIAM, Philadelphia, 2009.

[88] E. Crawley, B. Cameron, and D. Selva. *System architecture: strategy and product development for complex systems*. Prentice Hall Press, 2015.

[89] A.-S. Crélot, C. Beauthier, D. Orban, C. Sainvitu, and A. Sartenaer. Combining Surrogate Strategies with MADS for Mixed-Variable Derivative-Free Optimization. Technical Report G-2017-70, Les cahiers du GERAD, 2017.

[90] J. Cuesta-Ramirez, R. Le Riche, O. Roustant, G. Perrin, C. Durantin, and A. Gliere. A comparison of mixed-variables Bayesian optimization approaches. *Advanced Modeling and Simulation in Engineering Sciences*, 9(1):6, 2022.

[91] S. Da Veiga. Global sensitivity analysis with dependence measures. *Journal of Statistical Computation and Simulation*, 85:1283–1305, 2015.

[92] G.B. Dantzig. *Linear Programming and Extensions*. Princeton Landmarks in Mathematics and Physics. Princeton University Press, 1963.

[93] C. David, S. Delbecq, S. Defoort, P. Schmollgruber, E. Benard, and V. Pommier-Budinger. From FAST to FAST-OAD: An open source framework for rapid overall aircraft design. *IOP Conference Series: Materials Science and Engineering*, 1024:012062, 2021.

[94] C. Davis. Theory of positive linear dependence. *American Journal of Mathematics*, 76:733–746, 1954.

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

[96] M. Deza and E. Deza. Other distances. *Encyclopedia of Distances*, 1:661–699, 2014.

[97] S. Dubreuil, N. Bartoli, G. Berthelin, O. Chandre Vila, C. Gogu, T. Lefebvre, J. Morlier, and M. Salaün. Development of MDO formulations based on disciplinary surrogate models by Gaussian processes. In *AeroBest*, 2021.

[98] J. Ellson, E. Gansner, L. Koutsofios, S. North, and G. Woodhull. Graphviz—open source graph drawing tools. In *Graph Drawing: 9th International Symposium, GD 2001 Vienna, Austria, September 23–26, 2001 Revised Papers 9*, 2002.

[99] P. Esser, M. Fleissner, and D. Ghoshdastidar. Non-parametric representation learning with kernels. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

[100] N. Fellmann, C. Blanchet-Scalliet, C. Helbert, A. Spagnol, and D. Sinoquet. Kernel-based sensitivity analysis for (excursion) sets. *Technometrics*, 67:1–13, 2024.

[101] A. Feragen, F. Lauze, and S. Hauberg. Geodesic exponential kernels: When curvature and linearity conflict. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3032–3042, 2015.

[102] E. Fermi and N. Metropolis. Numerical solution of a minimum problem. Los Alamos Unclassified Report LA–1492, Los Alamos National Laboratory, Los Alamos, USA, 1952.

[103] M. Fernández and G. Valiente. A graph distance metric combining maximum common subgraph and minimum common supergraph. *Pattern Recognition Letters*, 22:753–758, 2001.

[104] M. Feurer and F. Hutter. Hyperparameter optimization. *Automated Machine Learning: Methods, Systems, Challenges*, pages 3–33, 2019.

[105] R. Fletcher and S. Leyffer. Nonlinear programming without a penalty function. *Mathematical Programming*, Series A, 91:239–269, 2002.

[106] M. E. A. Fouda, E. J. Adler, J.H. Bussemaker, J. R. R. A. Martins, D. F. Kurtulus, L. Boggero, and B. Nagel. Automated hybrid propulsion model construction for conceptual aircraft design and optimization. In *33rd Congress of the International Council of the Aeronautical Sciences, ICAS 2022*, 2022.

[107] P. I. Frazier. A Tutorial on Bayesian Optimization. *ArXiv preprint*, 2018.

[108] L. Friedli, A. Gautier, A. Broccard, and D. Ginsbourger. Crps-based targeted sequential design with application in chemical space. *arXiv preprint arXiv:2503.11250*, 2025.

[109] A. Sanfeliu K. Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE transactions on systems, man, and cybernetics*, 3:353–362, 1983.

[110] X. Gao, B. Xiao, D. Tao, and X. Li. A survey of graph edit distance. *Pattern Analysis and applications*, 13:113–129, 2010.

[111] A. Gardner, C.A. Duncan, J. Kanno, and R.R. Selmic. On the Definiteness of Earth Mover's Distance and Its Relation to Set Intersection. *IEEE Transactions on Cybernetics*, 48(11):3184–3196, 2018.

[112] E.C. Garrido-Merchán and D. Hernández-Lobato. Dealing with categorical and integer-valued variables in Bayesian Optimization with Gaussian processes. *Neurocomputing*, 380:20–35, 2020.

[113] R. Gómez-Bombarelli, J. Wei, D. Duvenaud, J. Hernández-Lobato, B. Sánchez-Lengeling, D. Sheberla, J. Aguilera-Iparraguirre, T. Hirzel, R. Adams, and A. Aspuru-Guzik. Automatic chemical design using a data-driven continuous representation of molecules. *ACS central science*, 4:268–276, 2018.

[114] F. Gonzalez-Lopez, L. Pufahl, J. Munoz-Gama, V. Herskovic, and M. Sepúlveda. Case model landscapes: toward an improved representation of knowledge-intensive processes using the fcm-language. *Software and Systems Modeling*, 20:1353–1377, 2021.

[115] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. MIT Press, 2016.

[116] A. Goshtasby. Similarity and dissimilarity measures. *Image registration: Principles, tools and methods*, 1:7–66, 2012.

[117] N.I.M. Gould, D. Orban, and Ph.L. Toint. CUTEst: a Constrained and Unconstrained Testing Environment with safe threads for mathematical optimization. *Computational Optimization and Applications*, 60(3):545–557, 2015. Code available at https://ccpforge.cse.rl.ac.uk/gf/project/cutest/wiki.

[118] Justin Gray, John Hwang, Joaquim Martins, Kenneth Moore, and Bret Naylor. OpenMDAO: an open-source framework for multidisciplinary design, analysis, and optimization. *Structural and Multidisciplinary Optimization*, 59(4):1075–1104, 2019.

[119] A. Gretton, K.M. Borgwardt, M.J. Rasch, B. Schölkopf, and A. Smola. A Kernel Two-Sample Test. *The Journal of Machine Learning Research*, 13(25):723–773, 2012.

[120] M. Grohe and P. Schweitzer. The graph isomorphism problem. *Communications of the ACM*, 63:128–134, 2020.

[121] R. Grosse, R. Salakhutdinov, W. Freemab, and J. Tenenbaum. Exploiting compositionality to explore a large space of model structures. In *Uncertainty in AI (UAI)*, 2012.

[122] O. Günlük, J. Kalagnanam, M. Li, M. Menickelly, and K. Scheinberg. Optimal decision trees for categorical data via integer programming. *Journal of Global Optimization*, 81:233–260, 2021.

[123] B. Haasdonk and C. Bahlmann. Learning with distance substitution kernels. *Joint pattern recognition symposium*, 3175:220–227, 2004.

[124] E. Hallé-Hannan. Cadre mathématique pour l'optimisation de boîtes noires avec variables métas et catégorielles. Master's thesis, Polytechnique Montréal, 2022. Available at https://publications.polymtl.ca/10286/.

[125] E. Hallé-Hannan, C. Audet, Y. Diouane, S. Le Digabel, and P. Saves. A distance for mixed-variable and hierarchical domains with meta variables. *Neurocomputing*, 653:131208, 2025.

[126] E. Hallé-Hannan, C. Audet, Y. Diouane, S. Le Digabel, and C. Tribes. Cat-Suite: A collection of optimization problems with categorical and quantitative variables for benchmarking. Technical Report G-2025-39, Les cahiers du GERAD, 2025.

[127] M. Halstrup. *Black-Box Optimization of Mixed Discrete-Continuous Optimization Problems*. PhD thesis, Technical University of Denmark, 2016.

[128] R. Hamano, S. Saito, M. Nomura, K. Uchida, and S. Shirakawa. CatCMA: Stochastic Optimization for Mixed-Category Problems. Technical Report 2405.09962, Arxiv, 2024.

[129] N. Hansen and A. Ostermeier. Completely Derandomized Self-Adaptation in Evolution Strategies. *Evolutionary Computation*, 9(2):159–195, 2001.

[130] T. Hastie, R. Tibshirani, and J. Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., New York, NY, USA, 2001.

[131] M. Herrera, A. Guglielmetti, M. Xiao, and R. Filomeno Coelho. Metamodel-assisted optimization based on multiple kernel regression for mixed variables. *Structural and multidisciplinary optimization*, 49:979–991, 2014.

[132] K. Horio, S. Ishikawa, and R. Kubota. Effective Hierarchical Optimization using Integration of Solution Spaces and its Application to multiple Vehicle Routing Problem. In *2015 International Symposium on Intelligent Signal Processing and Communication Systems*. IEEE, 2015.

[133] L.-Y. Hu, M.-W. Huang, S.-W. Ke, and C.-F. Tsai. The distance function effect on k-nearest neighbor classification for medical datasets. *SpringerPlus*, 5:1–9, 2016.

[134] Y. Hung, V.R. Joseph, and S.N. Melkote. Design and Analysis of Computer Experiments With Branching and Nested Factors. *Technometrics*, 51(4):354–365, 2009.

[135] F. Hutter and M.A. Osborne. A Kernel for Hierarchical Parameter Spaces. Technical Report 1310.5738, arXiv, 2013.

[136] P. Jäckel and R. Rebonato. The Most General Methodology to Create a Valid Correlation Matrix for Risk Management and Option Pricing Purposes. *Journal of Risk*, 2(2):17–27, 1999.

[137] J. Jahn. *Introduction to the theory of nonlinear optimization*. Springer, third edition, 2007.

[138] M. Jamil and X.-S. Yang. A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation*, 4(2):150–194, 2013.

[139] H. Jiang, Y. Shen, Y. Li, W. Zhang, C. Zhang, and B. Cui. Openbox: A Python toolkit for generalized black-box optimization. *ArXiv preprint*, 2023.

[140] J. Jiang and D. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. *ArXiv preprint*, 1997.

[141] R. Jin and H. Liu. A Novel Approach to Model Generation for Heterogeneous Data Classification. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*. Morgan Kaufmann Publishers Inc., 2005.

[142] Y. Jina and P.V. Kumar. Bayesian optimisation for efficient material discovery: a mini review. *Nanoscale*, 15(26):10975–10984, 2023.

[143] K. Jing, J. Xu, and Z. Zhang. A neural architecture generator for efficient search space. *Neurocomputing*, 486:189–199, 2022.

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

[145] D.R. Jones, M. Schonlau, and W.J. Welch. Efficient Global Optimization of Expensive Black Box Functions. *Journal of Global Optimization*, 13(4):455–492, 1998.

[146] J. Josse, J. M. Chen, N. Prost, G. Varoquaux, and E. Scornet. On the consistency of supervised learning with missing values. *Statistical Papers*, 65(9):5447–5479, 2024.

[147] J. Josse and F. Husson. Handling missing values in exploratory multivariate data analysis methods. *Journal de la société française de statistique*, 153(2):79–99, 2012.

[148] K. Kaneiwa. Order-sorted logic programming with predicate hierarchy. *Artificial Intelligence*, 158:155–188, 2004.

[149] R. Karlsson, L. Bliek, S. Verwer, and M. de Weerdt. Continuous surrogate-based optimization algorithms are well-suited for expensive discrete problems. In *Artificial Intelligence and Machine Learning*, 2021.

[150] T. Karunaratne and H. Boström. Graph propositionalization for random forests. In *2009 International Conference on Machine Learning and Applications*, 2009.

[151] M. Kejriwal. Knowledge graphs: A practical review of the research landscape. *Information*, 13:161– 178, 2022.

[152] E. Keogh and S. Kasetty. On the need for time series data mining benchmarks: a survey and empirical demonstration. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, 2002.

[153] K. Kim and J.-s. Hong. A hybrid decision tree algorithm for mixed numeric and categorical data in regression analysis. *Pattern Recognition Letters*, 98:39–45, 2017.

[154] A. Klein, S. Falkner, S. Bartels, P. Hennig, and F. Hutter. Fast Bayesian Optimization of Machine Learning Hyperparameters on Large Datasets. Technical Report 1605.07079, arXiv, 2017.

[155] M. Kokkolaras, C. Audet, and J.E. Dennis, Jr. Mixed variable optimization of the Number and composition of heat intercepts in a thermal insulation system. *Optimization and Engineering*, 2(1):5–29, 2001.

[156] A. Krizhevsky and G. Hinton. Learning multiple layers of features from tiny images. Technical report, Citeseer, 2009.

[157] V. Kvasnička, J. Pospíchal, and V. Baláž. Reaction and chemical distances and reaction graphs. *Theoretica chimica acta*, 79:65–79, 1991.

[158] D. Lakhmiri. HyperNOMAD. https://github.com/bbopt/HyperNOMAD, 2019.

[159] D. Lakhmiri, S. Le Digabel, and C. Tribes. HyperNOMAD: Hyperparameter Optimization of Deep Neural Networks Using Mesh Adaptive Direct Search. *ACM Transactions on Mathematical Software*, 47(3), 2021.

[160] A. Lambe and J. R. R. A. Martins. Extensions to the design structure matrix for the description of multidisciplinary design, analysis, and optimization processes. *Structural and Multidisciplinary Optimization*, 46:273–284, 2012.

[161] S. Le Digabel. Algorithm 909: NOMAD: Nonlinear Optimization with the MADS algorithm. *ACM Transactions on Mathematical Software*, 37(4):44:1–44:15, 2011.

[162] S. Le Digabel and S.M. Wild. A taxonomy of constraints in black-box simulation-based optimization. *Optimization and Engineering*, 25(2):1125–1143, 2024.

[163] R. Le Riche and V. Picheny. Revisiting Bayesian optimization in the light of the coco benchmark. *Structural and Multidisciplinary Optimization*, 64:3063–3087, 2021.

[164] M. Lindauer, K. Eggensperger, M. Feurer, A. Biedenkapp, D. Deng, C. Benjamins, T. Ruhkopf, R. Sass, and F. Hutter. BOAH: A tool suite for multi-fidelity Bayesian optimization & analysis of hyperparameters. *ArXiv preprint*, 2019.

[165] M. Lindauer, K. Eggensperger, M. Feurer, A. Biedenkapp, D. Deng, C. Benjamins, T. Ruhkopf, R. Sass, and F. Hutter. SMAC3: A versatile Bayesian optimization package for hyperparameter optimization. *Journal of Machine Learning Research*, 23:1–9, 2022.

[166] S. Lucidi and V. Piccialli. A Derivative-Based Algorithm for a Particular Class of Mixed Variable Optimization Problems. *Optimization Methods and Software*, 17(3–4):317–387, 2004.

[167] S. Lucidi, V. Piccialli, and M. Sciandrone. An Algorithm Model for Mixed Variable Programming. *SIAM Journal on Optimization*, 15(4):1057–1084, 2005.

[168] L. Lukšan and J. Vlček. Test Problems for Nonsmooth Unconstrained and Linearly Constrained Optimization. Technical Report V-798, ICS AS CR, 2000.

[169] J. Gibbon M. Kendall. *Rank correlation methods*. Oxford University Press, 1948.

[170] P-F. Marteau and S. Gibet. On recursive edit distance kernels with application to time series classification. *IEEE transactions on neural networks and learning systems*, 26:1121–1133, 2014.

[171] J.R.R.A Martins and A.B. Lambe. Multidisciplinary design optimization: a survey of architectures. *AIAA journal*, 51(9):2049–2075, 2013.

[172] J.R.R.A. Martins and A. Ning. *Engineering Design Optimization*. Cambridge University Press, 2022.

[173] D. Mavris, C. de Tenorio, and M. Armstrong. Methodology for aircraft system architecture definition. In *46th AIAA Aerospace Sciences Meeting and Exhibit*, 2008.

[174] M.D. McKay, R.J. Beckman, and W.J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. *Technometrics*, 21(2):239–245, 1979.

[175] J. Mercer. Functions of positive and negative type, and their connection the theory of integral equations. *Philosophical transactions of the royal society of London A*, 209(441-458):415–446, 1909.

[176] M. Minsky. A framework for representing knowledge. *Artificial Intelligence*, 306:1–81, 1974.

[177] T. Mitchell, R. Keller, and S.Kedar-Cabelli. Explanation-based generalization: A unifying view. *Machine learning*, 1:47–80, 1986.

[178] J.J. Moré, B.S. Garbow, and Kenneth E. Hillstrom. Testing unconstrained optimization software. *ACM Transactions on Mathematical Software*, 7(1):17–41, 1981.

[179] J.J. Moré and S.M. Wild. Benchmarking Derivative-Free Optimization Algorithms. *SIAM Journal on Optimization*, 20(1):172–191, 2009.

[180] M. Munoz Zuniga and D. Sinoquet. Global optimization for mixed categorical-continuous variables based on Gaussian process models with a randomized categorical space exploration step. *INFOR: Information Systems and Operational Research*, 58(2):310–341, 2020.

[181] M. Neuhaus and H. Bunke. Self-organizing maps for learning the edit costs in graph matching. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 35:503–514, 2005.

[182] M. Neuhaus and H. Bunke. A convolution edit kernel for error-tolerant graph matching. In *18th International Conference on Pattern Recognition (ICPR'06)*, volume 4, pages 220–223, 2006.

[183] G. Nikolentzos, G. Siglidis, and M. Vazirgiannis. Graph kernels: A survey. *Journal of Artificial Intelligence Research*, 72:943–1027, 2021.

[184] S. Nishimura and T. Hiramatsu. A generalization of the lee distance and error correcting codes. *Discrete applied mathematics*, 156:588–595, 2008.

[185] G. Noroozi. Data Heterogeneity and Its Implications for Fairness. Master's thesis, Western University, 2023. Available at https://ir.lib.uwo.ca/etd/9623/.

[186] J. Novak. Concept mapping: A useful tool for science education. *Journal of research in science teaching*, 27:937–949, 1990.

[187] J. Oh and K. Lee. On the effectiveness of supervision in asymmetric non-contrastive learning. In *Proceedings of Machine Learning Research*, 2024.

[188] S. Ontañón. An overview of distance and similarity functions for structured data. *Artificial Intelligence Review*, 53:5309–5351, 2020.

[189] S. Ontañón and E. Plaza. Similarity measures over refinement graphs. *Machine learning*, 87:57–92, 2012.

[190] H. Owhadi. Computational graph completion. *Research in the Mathematical Sciences*, 9(2):27, 2022.

[191] C.H. Papadimitrios and K. Steiglitz. *Combinatorial optimization: algorithms and complexity*. Courier Corporation, 1998.

[192] R. Parello, Y. Gourinat, E. Benard, and S. Defoort. Structural sizing of a hydrogen tank for a commercial aircraft. In *Journal of Physics: Conference Series*, volume 2716, page 012040. IOP Publishing, 2024.

[193] A. Payandeh, K. Baghaei, P. Fayyazsanavi, S. Ramezani, Z. Chen, and S. Rahimi. Deep representation learning: Fundamentals, technologies, applications, and open challenges. *IEEE Access*, 11:137621–137659, 2023.

[194] J. Pelamatti, L. Brevault, M. Balesdent, E.-G. Talbi, and Y. Guerin. Efficient global optimization of constrained mixed variable problems. *Journal of Global Optimization*, 73(3):583–613, 2019.

[195] J. Pelamatti, L. Brevault, M. Balesdent, E.-G Talbi, and Y. Guerin. *Overview and Comparison of Gaussian Process-Based Surrogate Models for Mixed Continuous and Discrete Variables: Application on Aerospace Design Problems*, volume 833, pages 189–224. Springer, 2020.

[196] J. Pelamatti, L. Brevault, M. Balesdent, E.-G. Talbi, and Y. Guerin. Bayesian optimization of variable-size design space problems. *Optimization and Engineering*, 22:387–447, 2021.

[197] C.L. Pereira, G.D.C. Cavalcanti, and T.I. Ren. A New Heterogeneous Dissimilarity Measure for Data Classification. In *22nd IEEE International Conference on Tools with Artificial Intelligence*, pages 373–374. IEEE, 2010.

[198] J. Poole and J. Campbell. A novel algorithm for matching conceptual and related graphs. In *Conceptual Structures: Applications, Implementation and Theory: Third International Conference on Conceptual Structures, ICCS'95 Santa Cruz, CA, USA, August 14–18, 1995 Proceedings 3*, 1995.

[199] M.J.D. Powell. *A direct search optimization method that models the objective and constraint functions by linear interpolation*, volume 275, pages 51–67. Springer, 1994.

[200] R.P. Prager and H. Trautmann. Exploratory Landscape Analysis for Mixed-Variable Problems. *IEEE Transactions on Evolutionary Computation*, 2024.

[201] P.Z.G. Qiand, H. Wu, and C.F.J. Wu. Gaussian process models for computer experiments with qualitative and quantitative factors. *Technometrics*, 50(3):383–396, 2008.

[202] M. Quillian. The teachable language comprehender: A simulation program and theory of language. *Communications of the ACM*, 12:459–476, 1969.

[203] R. Rada, H. Mili, E. Bicknell, and M. Blettner. Development and application of a metric on semantic nets. *IEEE transactions on systems, man, and cybernetics*, 19:17–30, 1989.

[204] D. Ramachandram, M. Lisicki, T. J. Shields, M. R. Amer, and G. W. Taylor. Bayesian optimization on graph-structured search spaces: Optimizing deep multimodal fusion architectures. *Neurocomputing*, 298:80–89, 2018.

[205] M. Ramoni, P. Sebastiani, and P. Cohen. Bayesian clustering by dynamics. *Machine learning*, 47:91–121, 2002.

[206] C.E. Rasmussen and C.K.I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2006.

[207] T. Ray and K.M. Liew. A swarm metaphor for multiobjective design optimization. *Engineering Optimization*, 34(2):141–153, 2002.

[208] A. Remadi, K. E. Hage, Y. Hobeika, and F. Bugiotti. To prompt or not to prompt: Navigating the use of large language models for integrating and modeling heterogeneous data. *Data & Knowledge Engineering*, 152:102313, 2024.

[209] O. Renkonen. Statisch-kologische untersuchungen ber die terrestrische kaferwelt der finnischen bruchmoore. *Annales Botanici Societatis Zoologic-Botanic Fennic Vanamo*, 6:1231, 1938.

[210] P. Resnik. Semantic similarity in a taxonomy: An information-based measure and its application to problems of ambiguity in natural language. *Journal of artificial intelligence research*, 11:95–130, 1999.

[211] P. Robert and Y. Escoufier. A unifying tool for linear multivariate statistical methods: the rv-coefficient. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 25(3):257–265, 1976.

[212] A. Robles-Kelly and E.Hancock. Graph edit distance from spectral seriation. *IEEE transactions on pattern analysis and machine intelligence*, 27:365–378, 2005.

[213] A. Robles-Kelly and E. Hancock. String edit distance, random walks and graph matching. *International Journal of Pattern Recognition and Artificial Intelligence*, 18:315–327, 2004.

[214] R.T. Rockafellar. *Convex Analysis*. Princeton University Press, Princeton, NJ, 1970.

[215] S. Roussel, T. Polacsek, and A. Chan. Assembly line preliminary design optimization for an aircraft. In *CP 2023 (The 29th International Conference on Principles and Practice of Constraint Programming)*, 2023.

[216] O. Roustant, E. Padonou, Y. Deville, A. Clément, G. Perrin, J. Giorla, and H. Wynn. Group kernels for Gaussian process metamodels with categorical inputs. *Uncertainty Quantification*, 8(2):775–806, 2020.

[217] Donald B Rubin. Inference and missing data. *Biometrika*, 63(3):581–592, 1976.

[218] W. Sager and P Lockeman. Classification of rauking algorithms. *International Forum on Information and Documentation*, 1976.

[219] A. Sanfeliu and K.-S. Fu. A distance measure between attributed relational graphs for pattern recognition. *IEEE Transactions on Systems, Man, and Cybernetics*, 13(3):353–362, 1983.

[220] L.A. Sarrazin-Mc Cann. Optimisation et ordonnancement en optimisation sans dérivées. Master's thesis, Polytechnique Montréal, 2018.

[221] P. Saves. *High-dimensional multidisciplinary design optimization for aircraft eco-design.* PhD thesis, ONERA and ISAE-SUPAERO, 2024. Available at https://theses.hal.science/ONERA-MIP/tel-04439128v1.

[222] P. Saves, N. Bartoli, Y. Diouane, T. Lefebvre, J. Morlier, C. David, E. Nguyen Van, and S. Defoort. Constrained Bayesian Optimization Over Mixed Categorical Variables, with Application to Aircraft Design. In *AeroBest*, 2021.

[223] P. Saves, J.H. Bussemaker, R. Lafage, Thierry T. Lefebvre, Nathalie Bartoli, Youssef Diouane, and Joseph Morlier. System-of-systems modeling and optimization: An integrated framework for intermodal mobility. In *ODAS 2024: 24th joint ONERA-DLR Aerospace Symposium*, 2024.

[224] P. Saves, Y. Diouane, N. Bartoli, T. Lefebvre, and J. Morlier. A mixed-categorical correlation kernel for Gaussian process. *Neurocomputing*, 550:126472, 2023.

[225] P. Saves, Y. Diouane, N. Bartoli, T. Lefebvre, and J. Morlier. High-dimensional mixed-categorical gaussian processes with application to multidisciplinary design optimization for a green aircraft. *Structural and Multidisciplinary Optimization*, 67:81, 2024.

[226] P. Saves, R. Lafage, N. Bartoli, Y. Diouane, J.H. Bussemaker, T. Lefebvre, J.T. Hwang, J. Morlier, and J.R.R.A Martins. SMT 2.0: A Surrogate Modeling Toolbox with a focus on Hierarchical and Mixed Variables Gaussian Processes. *Advances in Engineering Software*, 188:103571, 2024.

[227] P. Saves, E. Nguyen Van, N. Bartoli, Y. Diouane, T. Lefebvre, C. David, S. Defoort, and J. Morlier. Bayesian optimization for mixed variables using an adaptive dimension reduction process: applications to aircraft design. In *AIAA SciTech 2022 Forum*, 2022.

[228] Paul Saves, Pramudita Satria Palar, Muhammad Daffa Robani, Nicolas Verstaevel, Moncef Garouani, Julien Aligon, Benoit Gaudou, Koji Shimoyama, and Joseph Morlier. Surrogate modeling and explainable artificial intelligence for complex systems: A workflow for automated simulation exploration. *arXiv preprint arXiv:2510.16742*, 2025.

[229] P. Schmollgruber, C. Döll, J. Hermetz, R. Liaboeuf, M. Ridel, I. Cafarelli, O. Atinault, C. François, and B. Paluch. Multidisciplinary exploration of DRAGON: an ONERA hybrid electric distributed propulsion concept. In *AIAA SciTech 2019 Forum*, 2019.

[230] P. Schmollgruber, D. Donjat, M. Ridel, I. Cafarelli, O. Atinault, C. François, and B. Paluch. Multidisciplinary design and performance of the ONERA hybrid electric distributed propulsion concept (DRAGON). In *AIAA SciTech 2020 Forum*, 2020.

[231] J. Serra and J. Arcos. An empirical evaluation of similarity measures for time series classification. *Knowledge-Based Systems*, 67:305–314, 2014.

[232] F. Serratosa. Redefining the graph edit distance. *SN Computer Science*, 2:438, 2021.

[233] B. Shahriari, K. Swersky, Z. Wang, R. P. Adams, and N. de Freitas. Taking the Human Out of the Loop: A Review of Bayesian Optimization. *Proceedings of the IEEE*, 104:148–175, 2016.

[234] H. Sheikh and P. Marcus. Bayesian optimization for mixed-variable, multi-objective problems. *Structural and Multidisciplinary Optimization*, 65:331, 2022.

[235] James M Shihua, Paul Saves, Rhea P Liem, and Joseph Morlier. Bayesian optimization of a lightweight and accurate neural network for aerodynamic performance prediction. *arXiv preprint arXiv:2503.19479*, 2025.

[236] W.L. Simmons. *A Framework for Decision Support in Systems Architecting*. PhD thesis, Massachusetts Institute of Technology, 2008. Available at https://dspace.mit.edu/handle/1721.1/42912.

[237] G. Song, D. Fu, and D. Zhang. From knowledge graph development to serving industrial knowledge automation: a review. In *2022 Chinese Control Conference*, 2022.

[238] Y.-Y. Song and Y. Lu. Decision tree methods: applications for classification and prediction. *Shanghai Archives of Psychiatry*, 27(2):130–135, 2015.

[239] T. Sorensen. A method of establishing groups of equal amplitude in plant sociology based on similarity of species content and its application to analyses of the vegetation on danish commons. *Biologiske skrifter*, 5:1–34, 1948.

[240] B. Sow, R. Le Riche, J. Pelamatti, S. Zannane, and M. Keller. Learning functions defined over sets of vectors with kernel methods. In *5th International Conference on Uncertainty Quantification in Computational Science and Engineering (UNCECOMP)*. European Community on Computational Methods in Applied Sciences (ECCOMAS), 2023.

[241] A. Sportisse, C. Boyer, and J. Josse. Imputation and low-rank estimation with missing not at random data. *Statistics and Computing*, 30(6):1629–1643, 2020.

[242] P. Sridevi, Z. Arefin, and S.I. Ahamed. An integrated machine learning and hyperparameter optimization framework for noninvasive creatinine estimation using photoplethysmography signals. *Healthcare Analytics*, 7:100395, 2025.

[243] H. Steck, C. Ekanadham, and N. Kallus. Is cosine-similarity of embeddings really about similarity? In *Companion Proceedings of the ACM on Web Conference 2024*, 2024.

[244] M. Stelmack, N. Nakashima, and S. Batill. Genetic algorithms for mixed discrete/continuous optimization in multidisciplinary design. In *7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization*. Aerospace Research Central, 1998.

[245] D. Streitferdt, M. Riebisch, and K. Philippow. Details of formalized relations in feature models using OCL. In *10th IEEE International Conference and Workshop on the*

Engineering of Computer-Based Systems, 2003. Proceedings., pages 297–304. IEEE, 2003.

[246] S. Surjanovic and D. Bingham. Virtual Library of Simulation Experiments: Test Functions and Datasets. Technical report, Simon Fraser University, 2025.

[247] G. Székely and M. Rizzo. Brownian distance covariance. *The Annals of Applied Statistics*, 3(4):1236–1265, 2009.

[248] E.-G. Talbi. Metaheuristics for variable-size mixed optimization problems: A unified taxonomy and survey. *Swarm and Evolutionary Computation*, 89:101642, 2024.

[249] B. Talgorn, S. Le Digabel, and M. Kokkolaras. Statistical Surrogate Formulations for Simulation-Based Design Optimization. *Journal of Mechanical Design*, 137(2):021405–1–021405–18, 2015.

[250] T. Tanimoto. Elementary mathematical theory of classification and prediction. *International Business Machines Corp.*, 1958.

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

[252] J.J. Torres, G. Nannicini, E. Traversi, and R.W. Calvo. A trust-region framework for derivative-free mixed-integer optimization. *Mathematical Programming Computation*, 16:369–422, 2024.

[253] A. Tversky and I. Gati. Studies of similarity. *Cognition and categorization*, 1:79–98, 1979.

[254] J. Valls-Vargas, J. Zhu, and S. Ontanon. Error analysis in an automated narrative information extraction pipeline. *IEEE Transactions on Computational Intelligence and AI in Games*, 9:342–353, 2016.

[255] B. Van Dyke and T.J. Asaki. Using QR Decomposition to Obtain a New Instance of Mesh Adaptive Direct Search with Uniformly Distributed Polling Directions. *Journal of Optimization Theory and Applications*, 159(3):805–821, 2013.

[256] J. von Kügelgen, M. Besserve, L. Wendong, L. Gresele, A. Kekić, E. Bareinboim, D. Blei, and B. Schölkopf. Nonparametric identifiability of causal representations from unknown interventions. *Advances in Neural Information Processing Systems*, 37:1–36, 2024.

[257] W. Wallis, P. Shoubridge, M. Kraetz, and D. Ray. Graph distances using graph union. *Pattern Recognition Letters*, 22:701–704, 2001.

[258] F. Wang, H. Zhang, and A. Zhou. A particle swarm optimization algorithm for mixed-variable optimization problems. *Swarm and Evolutionary Computation*, 60:100808, 2021.

[259] Y. Wang and N. Ishii. A method of similarity metrics for structured representations. *Expert Systems with Applications*, 12:89–100, 1997.

[260] D.J. Wild. Mining large heterogeneous data sets in drug discovery. *Expert Opinion on Drug Discovery*, 4(10):995–1004, 2009.

[261] W. Winkler. Matching and record linkage. *Wiley interdisciplinary reviews: Computational statistics*, 6:313–325, 2014.

[262] P. Winter and S. MacGregor. Path-distance heuristics for the steiner problem in undirected networks. *Algorithmica*, 7:309–327, 1992.

[263] H. Wu, H. Yang, X. Li, and H. Ren. Semi-supervised autoencoder: A joint approach of representation and classification. In *Proceedings of the 2015 International Conference on Computational Intelligence and Communication Networks (CICN)*, 2016.

[264] J. Wu, X.-Y. Chen, H. Zhang, L.-D. Xiong, H.L., and S.-H. Deng. Hyperparameter Optimization for Machine Learning Models Based on Bayesian Optimization. *Journal of Electronic Science and Technology*, 17(1):26–40, 2019.

[265] H. Xiao, K. Rasul, and R. Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. Technical Report 1708.07747, arXiv, 2017.

[266] L. Yang and A. Shami. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415:295–316, 2020.

[267] M. Zaefferer and D. Horn. A first analysis of kernels for Kriging-based optimization in hierarchical search spaces. *ArXiv preprint*, 2018.

[268] J. Zhang. Graph Neural Distance Metric Learning with Graph-Bert. Technical Report 2002.03427, arXiv, 2020.

[269] L. Zhang, Y. Xie, L. Xidao, and X. Zhang. Multi-source heterogeneous data fusion. In *International Conference on Artificial Intelligence and Big Data (ICAIBD)*. IEEE, 2018.

[270] Y. Zhang, S. Tao, W. Chen, and D.W. Apley. A Latent Variable Approach to Gaussian Process Modeling with Qualitative and Quantitative Factors. *Technometrics*, 62(3):291–302, 2020.