

Multi Objective Hyperparameter Optimization in Online Environments

Maria Joana Clímaco Baceira

Dissertation

Master in Data Analysis

Supervised by:

PhD Bruno Miguel Delindro Veloso

Co-supervised by:

PhD João Manuel Portela da Gama

2026

Abstract

Hyperparameter Optimization (HPO) has become a fundamental component of modern Machine Learning (ML), as the selection of hyperparameters directly influences predictive performance, model generalization, and computational efficiency. HPO has reduced the need for manual hyperparameter tuning by automating the search for high-quality model configurations, enabling the effective deployment of increasingly complex Machine Learning (ML) models across diverse application domains.

However, modern ML applications in IoT and streaming environments face a critical challenge: balancing high predictive accuracy with stringent operational constraints such as energy consumption and computational latency.

Traditional HPO methods often fail in these scenarios as they typically focus on single-objective, offline processing and lack the mechanisms to adapt to non-stationary data distributions, known as concept drift. This thesis addresses these limitations by proposing a real-time Multi-Objective Hyperparameter Optimization (MOHPO) framework.

The proposed architecture leverages a evolutionary strategy to efficiently converge toward an optimal Pareto set of configurations, allowing decision-makers to navigate trade-offs between predictive quality and resource efficiency. A core contribution of this work is the integration of drift-aware mechanisms and the adoption of the prequential evaluation protocol with fading factors, ensuring that the optimization process remains resilient and computationally lightweight for production-grade environments.

The statistical evaluation of the results demonstrate that the proposed multi-objective framework achieves competitive predictive quality while significantly optimizing resource consumption, which is of critical importance in the modern ML applications in IoT and streaming environments.

Keywords: multi-objective optimization, hyperparameter optimization, data streams

Resumo

A Otimização de Hiperparâmetros (HPO) tornou-se uma componente fundamental na Aprendizagem Máquina (Machine Learning; ML) dos dias de hoje, dado que o conjunto de hiperparâmetros de um modelo ML influencia diretamente o desempenho preditivo, a capacidade de generalização e a eficiência computacional. A HPO automatizou a procura de configurações de hiperparâmetros de alta qualidade, reduzindo a necessidade pelo ajuste manual destes, e com isto, permitiu a implementação de modelos ML cada vez mais complexos ao longo de diversas áreas.

No entanto, as aplicações mais recentes de algoritmos ML enfrentam um desafio crítico: equilibrar uma elevada capacidade de predição com restrições operacionais rigorososas, tais como o consumo de energia ou a latência computacional.

Os métodos tradicionais de HPO normalmente falham em ter em conta essas restrições, uma vez que tendem a focar-se em problemas de objetivo único e em processamento offline, não dispondo de mecanismos adequados para se adaptarem a distribuições de dados não estacionárias, isto é, *concept drift*. Tendo isto em conta, esta tese aborda estas limitações através da proposta de um framework de Otimização Multiobjetivo de Hiperparâmetros (MOHPO).

A arquitetura proposta recorre a uma abordagem evolutiva para convergir de forma eficiente para um conjunto ótimo de configurações de modelos de ML, permitindo aos utilizadores encontrar um compromisso entre a qualidade de predição pretendida com as restrições operacionais existentes, bem como à adoção do protocolo de avaliação prequential, garantindo que o processo de otimização se mantém resiliente às alterações na distribuição dos dados e, simultaneamente, computacionalmente leve, de modo a ser adequado a ambientes de produção.

A análise estatística dos resultados demonstrou que a arquitetura multi-objetivo proposta atinge uma qualidade preditiva competitiva ao mesmo tempo que otimiza significativamente o consumo de recursos, o que é de importância crítica nas aplicações modernas de ML em IoT e ambientes de streaming.

Palavras-chave: otimização multi-objetivo, otimização de hiperparâmetros, data streams

Contents

Abstract iii
Resumo v

1 Introduction **1**
1.1 Motivation 1
1.2 Problem Definition and Objectives 2
1.3 Dissertation Structure 4

2 Theoretical Foundations **5**
2.1 Hyperparameter Optimization 5
2.1.1 Multi-Objective Hyperparameter Optimization 6
2.2 Machine Learning in Dynamic Environments 7
2.2.1 Concept Drift 7
2.2.2 Nelder-Mead Algorithms 8
2.3 Real-time Hyperparameter Optimization 8

3 Literature Review **11**
3.1 Real-time Hyperparameter Optimization 11
3.2 Multi-objective Optimization 13
3.3 Discussion 16

4 Methodology **17**
4.1 Framework Architecture and Implementation 17
4.2 Experimental Protocol: Prequential Evaluation 18
4.3 Multi-Metric Performance Indicators 19
4.4 Proposed Scenarios and Baselines 19
4.5 Datasets 20
4.5.1 Dataset 1: Agrawal (Synthetic) 20
4.5.2 Dataset 2: Credit Card 20
4.5.3 Dataset 3: Higgs 21
4.5.4 Dataset 4: Hyperplane (Synthetic) 21
4.5.5 Dataset 5: Random Radial Basis Function (Synthetic) 21
4.5.6 Dataset 6: SEA (Synthetic) 22
4.5.7 Dataset 7: Shuttle 22

5 Results **23**
5.1 Results Analysis 23
5.2 Statistical Results Analysis 24

5.2.1 Accuracy Metric 24
5.2.2 RAM-hours Metric 25
5.3 Discussion 26

6 Conclusion **29**

Bibliography **31**

Glossary

AI — Artificial Intelligence
BO — Bayesian Optimization
BOHB — Bayesian Optimization Hyperband
EA — Evolutionary Algorithms
HPO — Hyperparameter Optimization
IoT - Internet of Things
ML — Machine Learning
MO — Multi-Objective
MOBO — Multi-Objective Bayesian Optimization
MOHPO — Multi-Objective Hyperparameter Optimization
MoFiBay — Multi-objective Multi-Fidelity Bayesian Optimization and Hyperband
MOO — Multi-Objective Optimization
OHL — Online Hyperparameter Learning
PESMO — Multi-objective Bayesian Optimization via Predictive Entropy Search
SPT — Self Parameter Tuning

List of Figures

4.1 Proposed MOHPO framework architecture 18

5.1 Accuracy: Statistical Comparison 25
5.2 RAM-Hours: Critical Difference Diagram 26

List of Tables

5.1 Performance Comparison: Accuracy 23
5.2 Performance Comparison: RAM-Hours 24
5.3 Descriptive statistics across conditions. 25
5.4 Friedman test mean ranks (MR) across conditions. 26

Chapter 1

Introduction

This chapter presents the context, the challenges and the objectives of this dissertation. It begins by explaining the concepts related to Hyperparameter Optimization, including the limitations of modern Machine Learning applications. Finally, this chapter specifies the objectives of this study and concludes with an overview of the dissertation's structure.

1.1 Motivation

Hyperparameters (HP) constitute a large part of the configurations of a Machine Learning (ML) algorithm, influencing and controlling the complexity, behavior, efficiency, and even the overall performance of the algorithm. Improving the performance of an algorithm by fine-tuning these hyperparameters still relies primarily on the expertise and time-consuming efforts of data scientists, usually requiring multiple rounds of experimentation and adjustment (Bischl et al. (2023)). Progressive automation of machine learning (AutoML) has simplified this burden with automation, and within this context, Hyperparameter Optimization HPO stands out as one of the most important and resource-intensive stages of a ML workflow, creating the opportunity for the proposal of many HPO methods.

However, HPO is commonly referred to as a black-box optimization problem, since the underlying relationship between the hyperparameters and the performance cannot be expressed analytically and can only be observed through the evaluation of the outputs. Given the high dimensionality and complexity of the search space, delegating the optimization process to specialized algorithms is both efficient and necessary to ensure reproducibility. Despite this, sophisticated HPO methods remain underutilized in practice, mostly due to factors such as the limited understanding of these methods, skepticism about their benefits compared to simpler approaches, lack of guidance on method selection and configuration, and difficulties in defining appropriate search spaces (Bischl et al. (2023)).

Moreover, nowadays, machine learning applications are no longer optimized solely for predictive accuracy. Although predictive accuracy is still decisive, practical deployment introduces additional requirements, such as robustness, reliability, interpretability, and efficiency, and in resource-constrained environments, factors such as power consumption, memory usage, and inference latency can be as critical. These considerations naturally lead to trade-offs, making hyperparameter optimization a multi-objective problem rather than a single-objective one (Karl et al. (2023)).

Multi-objective optimization (MOO) offers a principled framework to address these chal-

lenges by explicitly modeling multiple, potentially conflicting objectives. Instead of producing a single optimal solution, MOO algorithms aim to approximate the set of Pareto-optimal solutions, each representing a different trade-off among objectives. This allows domain experts and decision makers to analyze these trade-offs post hoc and select the most suitable configuration for a given application, without the need to define preferences in advance.

Framing hyperparameter optimization as a multi-objective problem therefore provides greater flexibility and aligns more closely with real-world requirements. By identifying a diverse set of high-quality trade-off configurations, Multi-Objective Hyperparameter Optimization (MOHPO) supports informed decision-making and facilitates the deployment of machine learning models that balance performance with practical constraints.

The significance of this study stems from the rapid development of modern technological domains, including the growing prominence of Artificial Intelligence (AI), the automation of business processes, and the continuously increasing number of electronic devices that generate data at high speed across diverse application areas, such as automation, manufacturing, healthcare, and surveillance. Therefore, in light of the recent advances in MOHPO, the development of an algorithm that can optimize hyperparameters in a multi-objective manner, while being sufficiently flexible and scalable to be deployed both locally and online, represents a valuable contribution to ongoing research.

1.2 Problem Definition and Objectives

Modern machine learning success heavily relies on the appropriate configuration of hyperparameters. However, the current landscape of Hyperparameter Optimization (HPO) primarily focuses on single-objective functions, such as maximizing accuracy. In real-world scenarios, this is often insufficient, as models must simultaneously satisfy competing constraints such as computational efficiency, latency, and fairness. The core problem lies in the lack of integrated frameworks that can efficiently navigate these trade-offs without requiring exhaustive manual intervention. Moreover, conventional HPO approaches typically rely on offline processing, lacking the capacity for real-time, online hyperparameter fine-tuning.

This gap stems from the inherent design of traditional solvers, which fail to account for non-dominated solutions, that is, trade-offs where no single metric can be improved without degrading another. In the ever growing era of the Internet of Things (IoT), where millions of devices generate high-frequency data streams across domains like automation and healthcare (Veloso, Gama, Malheiro, and João (2021)), the prioritization of accuracy must give way to a multi-dimensional evaluation, where predictive performance is optimized alongside competing constraints, such as energy efficiency, computational latency, and memory footprint, while accounting for concept drift.

The real-world deployment of machine learning in these environments introduces complex trade-offs. Practitioners must reconcile the need for high predictive power with the challenges of concept drift, where the underlying data distribution changes over time (Gama, Žliobaitė, Bifet, Pechenizkiy, and Bouchachia (2014)), and stringent operational constraints, such as energy consumption, memory footprint, and latency. Current HPO tools lack the necessary integration to optimize these disparate and often conflicting metrics simultaneously.

To address such gap, this dissertation proposes and implmements a **Multi-objective Hyperparameter Optimization (MOHPO)** algorithm. By combining a Multi-objective Opti-

mization (MOO) framework with HPO techniques, this proposal aims to optimize simultaneously disparate objectives rather than collapsing them into a single scalar value.

To achieve this, the initial step will be to design and implement a MOHPO algorithm by selecting suitable MOO algorithms and integrating them into a hyperparameter search and optimization framework. The goal is for the algorithm to execute in parallel and converge towards an optimal Pareto set by leveraging a parallelized execution structure. Furthermore, the algorithm is intended to be scalable and adaptable, enabling execution both locally and in an online environment.

By automating this selection process, which factors for one of the most time-consuming tasks of a ML pipeline, the proposed MOHPO algorithm aims to alleviate the burden on data scientists, moving beyond the time-consuming trial-and-error rounds toward a truly automated and sustainable AutoML paradigm.

To achieve this, this dissertation proposes a **Multi-objective Hyperparameter Optimization (MOHPO)** framework specifically designed to address these multifaceted requirements. The core contribution of this work is the development of an algorithm capable of:

1. **Simultaneous Multi-metric Optimization:** Navigating the trade-offs between predictive quality, with specific mechanisms to handle concept drift, and resource efficiency (e.g., energy consumption and computational cost).

2. **Scalability and Flexibility:** Leveraging a parallelized execution structure that ensures the framework is robust enough for production-grade environments, which goal is to leverage a parallelized execution structure to converge efficiently towards an optimal Pareto set.

3. **Online Deployment:** Providing an adaptable architecture that can be seamlessly deployed in online environments, allowing for real-time model adjustment in dynamic data stream scenarios. The algorithm must adapt to streaming data or evolving search spaces.

The effectiveness of this approach will be assessed by applying the developed MOHPO framework to various ML algorithms and benchmark datasets. The evaluation will focus on the algorithm's ability to improve model performance across the entire Pareto front, demonstrating that the trade-offs achieved are superior or more diverse than those produced by standard single-objective or weighted-sum methods.

In order to assess whether this objective is achieved, the following research questions are formulated:

1. **Research Question 1:** How can Multi-Objective Hyperparameter Optimization be efficiently performed in real time for data stream learning?

2. **Research Question 2:** To what extent does the proposed MOHPO framework improve the trade-offs between predictive performance and resource efficiency compared with conventional HPO approaches?

1.3 Dissertation Structure

This dissertation is organized into six chapters, each chapter addressing a specific component of the research process and development of this dissertation.

Chapter 2 explores the theoretical foundations relevant to the study, as well as the importance of Hyperparameter Optimization, developing the concept of Multi-objective Hyperparameter Optimization problem. Chapter 3 introduces a systematic literature review, synthesizing the existing methodologies, frameworks and software on MOHPO and provides a discussion regarding important points related to this study.

Chapter 4 outlines the methodology used during the development of this dissertation, while Chapter 5 presents and discusses the results obtained, including evaluation components and comparison between methods.

Chapter 6 concludes the dissertation, presenting the key contributions, the limitations and the future research paths of this work.

Chapter 2

Theoretical Foundations

This chapter reviews the theoretical foundations essential to this dissertation, with the objective of highlighting the significance of HPO and the rise of multi-objective Hyperparameter Optimization (MOHPO) within today's fast-paced, dynamic technological landscape.

By exploring how multi-objective hyperparameter optimization paradigms can be incorporated into machine learning algorithms, it presents the theoretical foundations for building a real-time MOHPO framework that addresses the multiple needs of modern ML algorithms, with a special emphasis on operation in dynamic environments.

The first part of this chapter reviews the main concepts closely related to Hyperparameter Optimization, while the second part reviews concepts related to machine learning in dynamic environments, while intersecting the core ideas essential to this dissertation's study. The third and final part of this chapter introduces the theoretical foundations on real-time Hyperparameter Optimization.

2.1 Hyperparameter Optimization

As the core theme of this dissertation relies on Hyperparameter Optimization (HPO), this section introduces the theoretical foundations of closely related to HPO in the context of this work. The majority of ML algorithms have a defined set of hyperparameters, which must be carefully selected, since this configuration impacts the performance of the algorithm.

Bischl et al. (2023) reviews the foundations of HPO, and introduces HPO as a black-box problem aimed at minimizing an estimated generalization error. The core theoretical framework defines HPO as the process of searching for the optimal configuration of hyperparameters λ^* within a defined space Λ. The mathematical equation can be described as follows:

$$\lambda^* = \arg\min_{\lambda \in \Lambda} c(\lambda) \tag{2.1}$$

where $c : \Lambda \to \mathbb{R}$ is the objective function representing the expected loss on unseen data.

This is a distinct process from the "inner" learning of model parameters, such as weights, due to the fact that the objective function $c(\lambda)$ is unknown, non-differentiable, and of expensive evaluation, rendering therefore HPO as a black-box optimization problem.

2.1.1 Multi-Objective Hyperparameter Optimization

Presented as an evolutionary step to HPO, Multi-objective Hyperparameter Optimization (MOHPO) encompasses the need for trade-offs between two or more objectives, for example, between predictive accuracy and energy consumption.

In such case, MOO is applied in order to search for the Pareto-front, that is, the set of configurations which are optimal trade-offs between the objectives, as defined by Feurer and Hutter (2019), at which no other configuration performs better for at least one of and at least as well for all the other objectives, and finally, a configuration is chosen, from the Pareto front.

Ultimately, MOHPO goal is to simultaneously minimize (or maximize) multiple, often conflicting, objective functions, in order to identify the optimal collection of points at which no single objective can be improved without deteriorating another. MOHPO is particularly relevant for resource-limited IoT environments.

Pareto Dominance and Pareto Front

The set of all non-dominated solutions is known as the Pareto Set, and its projection in the objective space is the Pareto Front, representing the optimal trade-offs available.

As introduced just previously, the Pareto set corresponds to a set $\mathcal{X}^*$ of optimal points, that is, all the possible solutions that are not dominated by any other solution in the search space, representing the set of optimal solutions considering multiple conflicting objectives (Hernandez-Lobato, Hernandez-Lobato, Shah, and Adams (2016), Coello Coello (2001)). A solution x belongs to the Pareto set if there is no other solution x' in the search space that is better than x in all objectives and strictly better in at least one objective.

Meanwhile, the Pareto Front represents the set of objective vectors, that is, the solutions in the Pareto set defined by a particular constraint. The Pareto front can be defined as the trade-offs between different objectives, where improving one objective comes at the expense of worsening another. (Coello Coello (2001), Teich (2001))

Hypervolume

The hypervolume indicator provides a comprehensive measure of approximation based quality by quantifying the objective-space region dominated by a set of solutions relative to a reference point. Its strict monotonicity with respect to Pareto dominance gives it particularly strong theoretical foundations, while its ability to simultaneously account for convergence and diversity makes it a widely adopted performance indicator in multi-objective optimization (Guerreiro, Fonseca, and Paquete (2021)).

However, its computational complexity, especially when dealing with higher-dimensional objective spaces, and its dependence on the reference point constitute important practical limitations.

To evaluate and compare MOHPO algorithms, the Hypervolume (HV) metric is employed. It measures the size of the portion of the objective space that is dominated by a solution set A relative to a reference point $r \in \mathbb{R}^m$:

$$HV(A, r) = \Lambda \left(\bigcup_{a \in A} [a_1, r_1] \times \cdots \times [a_m, r_m] \right) \tag{2.2}$$

where Λ denotes the Lebesgue measure. A higher HV indicates a superior set of solutions that are both close to the true Pareto front and well-distributed across the objective space.

2.2 Machine Learning in Dynamic Environments

In traditional Machine Learning, models are typically trained in a batch setting, assuming that data is independent and identically distributed (i.i.d.). However, in the context of data streams, data arrives continuously and potentially at high velocity, requiring models to learn incrementally.

2.2.1 Concept Drift

A critical challenge in stream learning is concept drift, defined as a change in the joint distribution of the data over time. Formally, given two time points t and $t + 1$, concept drift occurs if:

$$P_t(X, y) \neq P_{t+1}(X, y) \tag{2.3}$$

where X is the vector of input features and y is the target variable. This phenomenon necessitates continuous model adaptation and is categorized into four main types:

- **Sudden:** The distribution changes abruptly (e.g., sensor failure).
- **Gradual:** A new distribution replaces the old one over a period.
- **Incremental:** The shift occurs in small, steady steps.
- **Recurring:** Previously observed patterns reappear periodically.

Concept drift is an inherent characteristic of non-stationary dynamic data streams and cannot be prevented, since it originates from changes in the underlying environment, such as evolving user behavior, market conditions, sensor degradation, or seasonal effects, among other possibilities.

Instead of trying to prevent concept drift, learning systems must be designed to cope with these changes, by continuously monitoring the data stream, determining what changed, and adapting accordingly the predictive model, whenever a significant distribution shift is detected (Lu et al. (2019)).

Additionally, Lu et al. (2019) identifies three complementary components for learning under concept drift:

- **Drift detection:** Identify when a statistically significant change has occurred in the data distribution or model performance.
- **Drift understanding:** Characterize the detected drift by estimating its location, severity, type, and affected features, supporting the development of better-informed adaptation strategies.
- **Drift adaptation:** Update the learning model to reflect the new concept. Common approaches include incremental model updates, sliding windows, which prioritize recent observations, ensemble methods, that replace outdated classifiers, and selective retraining using newly acquired data.

Handling concept drift is essential to avoid a common assumption where machine learning models generally assume that training and future data are under the same distribution.

When this assumption no longer holds, previously learned decision boundaries become obsolete, leading to a progressive degradation in predictive accuracy, increased classification errors, and unreliable decision-making.

Effective drift detection and adaptation enable models to remain accurate over long-term deployment while avoiding unnecessary retraining when the underlying concept remains stable.

2.2.2 Nelder-Mead Algorithms

The Nelder-Mead algorithm is a derivative-free optimization method for the unconstrained minimization of nonlinear functions. It evaluates the objective function at a set of points and uses these function values to determine the direction of the search. This makes the method particularly suitable for optimization problems in which derivative information is unavailable or difficult to obtain (Singer and Nelder (2009)).

The method was originally proposed by Nelder and Mead in 1965 Nelder and Mead (1965), and is a simplex-based direct-search method, where a simplex is used to explore the search space and progressively move towards a region with lower objective-function values. It should be distinguished from the simplex method used in linear programming, as the two methods address different classes of optimization problems (Singer and Nelder (2009)).

Consider the unconstrained optimization problem of finding the value of $\mathbf{x}$ that minimizes an objective function $f(\mathbf{x})$. The algorithm starts by creating a simplex containing $n+1$ points for an n-dimensional problem. The points are then ordered according to their objective-function values, from the best point, which has the lowest value, to the worst point, which has the highest value.

At each iteration, the algorithm tries to improve the worst point of the simplex. First, it calculates the centroid of all points except the worst one, then the worst point is reflected through this centroid, producing a new candidate point, and if this new point improves the objective function, the worst point is replaced.

If the reflected point is particularly good, the algorithm attempts to move further in the same direction through an *expansion*. However, if the reflection does not provide sufficient improvement, the algorithm instead performs a *contraction*, moving the simplex towards its center. Finally, if neither reflection nor contraction produces an improvement, the entire simplex is *shrunk* towards its best point.

These four operations, reflection, expansion, contraction, and shrinkage, allow the simplex to move through the search space and adapt its size and shape according to the objective function, and this process is repeated until a chosen stopping criterion is reached (Singer and Nelder (2009)).

Overall, the Nelder-Mead algorithm provides a relatively simple and computationally efficient approach to derivative-free optimization. Its ability to operate using only objective-function evaluations makes it particularly useful for problems where derivative information is unavailable or impractical to calculate.

2.3 Real-time Hyperparameter Optimization

Traditional HPO treats hyperparameter selection as an offline process, in which a set of candidate configurations is evaluated before the final model is trained. However, this process

becomes less suitable when the learning environment is dynamic and hyperparameters need to adapt as new data and model states become available.

Im, Savin, and Cho (2021) motivated an online formulation of HPO, in which hyperparameters are updated concurrently with model parameters, allowing the optimization process to adapt continuously to the evolving learning dynamics.

In their work, Im et al. (2021) establishes a theoretical connection between HPO and online parameter learning in recurrent neural networks, allowing hyperparameters to be optimized through real-time recurrent learning, without repeatedly restarting the underlying optimization procedure. This establishes a foundation for real-time HPO as a computationally tractable and asymptotically exact approach to dynamically adapting hyperparameters.

Therefore, HPO in data-stream environments needs to consider the temporal and dynamic nature of the data. Instead of selecting a single fixed configuration, the optimization process can, and should, adapt the hyperparameters as new data become available, allowing machine learning models to maintain their predictive performance when changes occur in the data stream.

The theoretical foundation of the framework proposed in this dissertation is based on the idea that model configuration should not necessarily remain fixed in a changing environment, and instead, hyperparameter optimization can be treated as a continuous and adaptive process. This perspective is particularly relevant for applications in which data are generated continuously and their characteristics may change over time.

This observation motivates the extension of the SSPT framework towards a multi-objective optimization setting. Rather than searching for a single hyperparameter configuration that optimizes one performance criterion, a multi-objective formulation can consider several objectives simultaneously and identify configurations representing different trade-offs between them. This approach allows for a more flexible framework for hyperparameter optimization. Therefore, the theoretical foundation concepts explained in this chapter provide the basis for the approach developed in this dissertation: extending the adaptive and single-pass principles of SSPT to a multi-objective optimization framework. The following chapters build upon these foundations to define, implement, and evaluate this proposed extension.

Chapter 3

Literature Review

This chapter provides an overview of MOO frameworks and HPO algorithms, with a particular emphasis on recently developed MOHPO approaches. The literature review aims to identify dominant trends, strengths, and limitations of various optimization algorithms, as well as their applications within both MOO and HPO frameworks.

The chapter is organized into three main sections. The first sectionfocuses on real-time HPO, tracing the evolution from gradient-based efficiency and population-based adaptation to drift-aware self-tuning mechanisms in data streams. The second section examines MOHPO methodologies, defining the criteria for Pareto efficiency and the trade-offs inherent in multi-faceted systems. Finally, the chapter provides a synthesis of the reviewed literature, identifying the research gaps in current MOHPO methodologies and outlining how this research addresses these challenges to enhance model resilience in resource-constrained IoT devices.

3.1 Real-time Hyperparameter Optimization

The shift from static to dynamic environments introduces a need for a transition in how hyperparameters are optimized. In a comprehensive review of the field, Lin et al. (2019) distinguish between batch and online (as real-time) HPO, emphasizing that the latter is essential for scenarios where computational budgets are limited or where the learning process is non-stationary. Unlike traditional "train-then-evaluate" cycles, online methods aim to adapt configurations during the learning process itself, often in a single-pass setting, overcoming the need for re-training the model in a batch setting.

Moreover, Bischl et al. (2023) categorize these methodologies into three primary families, each offering distinct mechanisms for dynamic adaptation. Gradient-based methods treat hyperparameters as differentiable variables, allowing the use of hyper-gradients to update them alongside the model's weights during training. By computing the gradient of a validation loss with respect to the hyperparameters, the system can perform a "seamless" optimization in a single pass. However, as noted by the authors, these methods are often restricted to continuous hyperparameters and can be computationally intensive due to the need for second-order derivative approximations.

Bischl et al. (2023) suggest that in Reinforcement Learning (RL) based approaches, the HPO process is modeled as a Markov Decision Process (MDP). An RL agent observes the state of the model (e.g., current accuracy, training loss trajectory) and takes actions by adjusting

hyperparameter values to maximize a long-term reward. While highly flexible and capable of handling complex search spaces, these methods suffer from a "cold-start" problem and require a significant warm-up period before the agent becomes effective.

Finally, Bischl et al. (2023) refers that Heuristic-based schedules rely on pre-defined or adaptive rules to adjust parameters, such as learning rate annealing or population-based perturbations. Although heuristic-based methodologies are computationally efficient and easy to implement, they lack the theoretical guarantees of convergence found in gradient-based methods and are often less responsive to unexpected concept drifts since they follow a more rigid adaptation logic.

The main argument of this work are based on the observation that, although these methods significantly reduce the overhead associated with global search strategies, avoiding multiple full training runs on offline settings, they introduce a new layer of complexity, namely the sensitivity to "hyper-hyperparameters" and the risk of convergence instability in non-stationary environments.

A significant advancement in operational efficiency for real-time systems is presented by Zhan, Gomes, Li, Madduri, and Wu (2018), who propose the Online Hyperparameter Learning (OHL) algorithm. Alternatively to traditional protocols, the OHL framework shifts this paradigm towards a continuous, incremental hyperparameter search and update mechanism.

This algorithm is particularly suited for high-frequency data streams, such as traffic sensor measurements data, which was the focus of Zhan et al. (2018), due to its "one-pass" nature. The core methodology of OHL involves hyper-gradient computation, where the algorithm computes the partial derivatives of the loss function with respect to the hyperparameters, in real time as new data comes in. Moreover, to mitigate aggressive updates, OHL performs a time-based gradient descent step, based on the historical hyper-gradients. The experimental results have demonstrated that OHL succeeds in achieving similar or superior prediction accuracy while requiring only a fraction of the computation time, when compared to state-of-the-art tuning methods.

As emphasized by Zhan et al. (2018), the relationship between hyperparameters and model performance is inherently non-convex, characterized by a complex landscape with multiple local optima, and the assumption of convexity, which underpins many traditional optimization algorithms, is frequently violated in the context of HPO, and the case is no different in the OHL paradigm.

In the context of this study, OHL presents two main limitations. The first limitation relies on the differentiability of the objective function, making it unsuitable for discrete hyperparameter spaces, and the second limitation relies on the fact that the focus remains strictly on a single predictive objective.

While gradient-based methods offer high efficiency, their reliance on differentiability can be a limitation in complex learning scenarios. Zhou, Liu, and Li (2019) address this by proposing an accelerated version of Population-based Training (PBT), framed for Deep Reinforcement Learning in their study.

Although their application domain differs, the underlying challenge is identical to that of non-stationary data streams: the optimal hyperparameter configuration shifts dynamically as the model evolves or the environment changes. In DRL, this instability arises from the agent's shifting policy, whereas in IoT streams, it is driven by concept drift.

By demonstrating that population-based search can be optimized for promptness, Zhou et al. (2019) provide a robust alternative for scenarios where the objective function is too

complex or "noisy" for direct gradient computation, a common trait in real-world sensor data. Their work emphasizes that hyperparameter configurations must "self-adapt" promptly to the current learning state, as fixed sets are insufficient for the high-dynamic and non-stationary nature of modern neural networks.

In their work, Zhou et al. (2019) introduce a recombination operation inspired by Genetic Algorithms, which significantly increases the speed of convergence compared to standard evolutionary strategies. For the real-time paradigm, this contribution was crucial, as it demonstrates that even complex, population-based search strategies can be optimized for promptness. However, maintaining multiple parallel agents is a challenge on resource-limited devices.

In the context of data streams, particularly those generated by IoT devices, the static optimization of hyperparameters is often insufficient due to the evolving nature of the data. As noted by Veloso et al. (2021), the online processing of these streams requires machine learning models that can adapt their configurations in real-time. This necessity arises primarily from the occurrence of concept drift, where shifts in the underlying data distribution render previously optimal hyperparameter settings obsolete, leading to significant performance degradation.

To address this, Veloso et al. (2021) proposed the SSPT algorithm, an extension of the Self Parameter Tuning (SPT) optimization algorithm for data streams, which integrates the Nelder-Mead optimization method with dynamic sampling. The core principle of this approach is "self-tuning", where the system continuously monitors performance and adjusts hyperparameters on-the-fly without requiring human intervention or a complete retraining of the model from scratch. By employing a sliding window mechanism, the algorithm can react to drifts by re-evaluating the hyperparameter space whenever a change in the stream's statistics is detected.

However, while SSPT and similar real-time HPO frameworks provide robust mechanisms for maintaining accuracy in dynamic environments, they typically focus on a single performance metric. In real-world deployment scenarios, especially in resource-constrained IoT devices, optimizing solely for accuracy is often inadequate. There is a clear research gap in extending these real-time, drift-aware methodologies to a multi-objective perspective, where competing constraints such as energy consumption, computational latency, and model complexity must be balanced simultaneously.

It is important to note that these online methodologies lay the groundwork for the subsequent discussion on multi-objective optimization. By shifting the focus from expensive, static global searches to efficient, continuous updates, they provide the necessary operational framework for MOHPO. However, as most online methods currently prioritize a single accuracy-based metric, there is a fundamental need to integrate the multi-objective techniques discussed in Section 3.2 to address the multi-faceted constraints of real-time ML systems.

3.2 Multi-objective Optimization

The theoretical foundation of this dissertation is grounded in the comprehensive study by Karl et al. (2023), which provides an extensive analysis of the state-of-the-art in MOHPO. The authors argue for the necessity of transcending single-objective optimization, contending that predictive performance must be balanced against operational constraints, such as energy efficiency, and ethical considerations, like fairness. According to this survey, MOHPO method-

ologies can be categorized into three primary categories, each presenting distinct trade-offs between sample efficiency and architectural flexibility.

The first one, Bayesian Optimization (BO), is highlighted for its high sample efficiency, making it the preferred choice when model evaluations are computationally expensive. It works by making use of surrogate models, by approximating the relationship between hyperparameter configurations and performance, to estimate uncertainty and guide the search. However, the authors identify scalability limitations as the number of observations or objectives increases.

Meanwhile, Evolutionary Algorithms (EA), which includes methods such as NSGA-II and NSGA-III, are recognized for their robustness in navigating complex search spaces, including discrete, categorical, or hierarchical hyperparameters. Although they require a higher number of evaluations compared to Bayesian approaches, and therefore proving to be computationally expensive when dealing with large search spaces, their inherent parallelism and population-based nature allow for a diverse approximation of the Pareto front.

The final category refers to Multi-fidelity Approaches. This kind of optimization approach focuses on resource efficiency by employing "low-fidelity" approximations, such as reduced training epochs or data subsets, to quickly discard unpromising configurations. However, Karl et al. (2023) emphasizes that the efficacy of these techniques depends on the correlation between low and high-fidelity performance.

This taxonomy clarifies that while these algorithms are mature for static scenarios, their integration into online environments subject to concept drift remains a significant gap in the current literature.

Within the domain of Multi-Objective Bayesian Optimization (MO-BO), Hernandez-Lobato et al. (2016) introduced the Predictive Entropy Search for Multi-objective Optimization (PESMO), an information-theoretic acquisition function designed for functions which evaluation is considered expensive. Unlike traditional methods that focus on improving hypervolume, PESMO aims to identify evaluation points that maximize the expected reduction in entropy (uncertainty) regarding the entire Pareto set.

The primary advantage of this approach lies in its computational efficiency, since the algorithm's cost scales linearly with the number of objectives, offering a significant improvement over the exponential complexity typical of hypervolume-based methods. Furthermore, PESMO supports decoupled evaluations, a feature that allows each objective to be evaluated independently and potentially at different costs. This capability is particularly beneficial in scenarios where certain performance metrics, such as computational latency, are significantly cheaper to measure than others, such as model accuracy on a large dataset.

Despite its efficiency, the method relies on complex approximations to estimate the entropy of the posterior distribution over the Pareto set, which can be computationally demanding in the acquisition phase. Nevertheless, PESMO remains a state-of-the-art benchmark for sample-efficient multi-objective optimization, providing a robust framework for balancing competing machine learning objectives under a limited evaluation budget.

Within the domain of Evolutionary Algorithms (EA), the family of Non-dominated Sorting Genetic Algorithms (NSGA) has emerged as a fundamental milestone in evolutionary MOO. NSGA-II, introduced by Deb, Pratap, Agarwal, and Meyarivan (2002), revolutionized the field by addressing the high computational complexity and lack of elitism in its predecessor. Its architecture is built upon three main pillars: a fast non-dominated sorting procedure, an elitist strategy that preserves high-quality solutions across generations, and a crowding

distance operator. This operator serves as a diversity-preserving mechanism, favoring solutions located in less dense regions of the Pareto front to ensure a uniform spread. While highly effective for bi-objective and tri-objective problems, a significant limitation of NSGA-II is its performance degradation in "many-objective" scenarios. In these high-dimensional spaces, the crowding distance loses its ability to effectively discriminate between solutions, often leading to a loss of selection pressure and diversity.

To overcome these challenges, Deb and Jain (2014) and Jain and Deb (2014) proposed NSGA-III, an evolution of NSGA-II, in a comprehensive two-part study. While Part I establishes the theoretical shift toward a reference-point-based decomposition to handle many-objective spaces, Part II provides the formal validation of the algorithm. The core innovation of NSGA-III is the replacement of the crowding distance with a set of predefined reference points that guide the selection process, and this mechanism ensures that the population remains well-distributed across the entire Pareto front, even when dealing with four or more conflicting objectives.

For MOHPO tasks that require balancing multiple metrics—such as accuracy, energy consumption, and latency—NSGA-III offers a superior framework for discovering diverse trade-offs, although it shares the inherent computational intensity of most population-based evolutionary strategies.

Despite the robustness of evolutionary strategies, their high computational demand often becomes prohibitive when evaluating each hyperparameter configuration requires full model training. To mitigate this, a third family of methods focuses on resource efficiency by leveraging low-fidelity approximations. These multi-fidelity approaches aim to accelerate the optimization process by quickly pruning sub-optimal candidates based on cheaper, albeit noisier, performance estimates.

To address the data inefficiency inherent in traditional multi-objective solvers within this context,Chen, Zhou, Huang, and Xia (2022) proposed MoFiBay. This algorithm serves as an advanced evolution of the BOHB framework, specifically tailored for multi-objective landscapes. The core innovation of MoFiBay lies in its treatment of multi-fidelity data, as while conventional methods often rely solely on high-fidelity measurements to build surrogate models, MoFiBay implements a multi-fidelity ensemble model. This approach integrates information from both computationally cheap low-fidelity evaluations and expensive high-fidelity runs to refine the search process.

The primary advantage of MoFiBay is its ability to maximize the utility of every evaluation, significantly improving the surrogate model's accuracy in predicting the Pareto front without increasing the total computational budget. By leveraging the Hyperband mechanism for aggressive resource allocation alongside a multi-objective BO sampling strategy, the framework effectively navigates complex trade-offs in real-time or live environments. However, the complexity of maintaining an ensemble of surrogates for multiple conflicting objectives introduces additional computational overhead during the acquisition phase. Furthermore, its performance remains sensitive to the correlation between different fidelity levels, a challenge that is particularly relevant in dynamic deployment scenarios where data distributions may shift.

3.3 Discussion

The comprehensive literature review of the MOHPO strategies state-of-the-art reveals a significant improvement opportunity in the development of optimization methodologies. While real-time HPO has evolved to handle the velocity and non-stationarity of data streams, and MOO has matured in its ability to navigate complex trade-offs, their intersection remains largely unexplored. This section synthesizes the identified limitations and establishes the gaps that this dissertation aims to address.

A critical finding from the analyzed literature is that the relationship between hyperparameters and model performance is inherently non-convex (Zhan et al. (2018)). This complexity justifies the employment of sophisticated meta-heuristics over simple gradient-based updates, especially in environments where the search space is irregular. However, a fundamental limitation of current MOO frameworks, such as NSGA-III (Deb and Jain (2014)) and MoFiBay (Chen et al. (2022)), is their static stance design.

In online scenarios, the Pareto front is not a fixed target, but rather, it is, most likely, non-stationary. As data distributions shift due to concept drift, a configuration that was previously optimal may become obsolete at the very next moment in execution. Most of the current MOHPO tools lack native mechanisms to detect these shifts and trigger re-optimization. This dissertation addresses this gap by proposing a framework that integrates drift-awareness into the multi-objective loop, fulfilling the need for resilient Online Deployment.

The second major gap lies in the tension between algorithmic complexity and operational promptness. The transition from single-pass gradient methods (Zhan et al. (2018)) to population-based adaptation (Zhou et al. (2019)) demonstrates that diversity is essential for robustness. However, population-based EA often incur a computational expense that is prohibitive for real-time IoT applications and devices, which resources are limited.

In contrast, lightweight self-tuning methods like SSPT (Veloso et al. (2021)) achieve the desired "promptness", tailored for resource-constrained IoT devices, but remain strictly single-objective in the HPO logic. This research aims to fill this void by leveraging a parallelized execution structure, allowing the framework to maintain a diverse approximation of the Pareto set without sacrificing the convergence speed necessary for real-time response.

Finally, most online methodologies currently prioritize a single predictive metric. As emphasized by Karl et al. (2023), practical deployment in resource constrained environments requires balancing predictive quality against resource efficiency. Therefore, there is a clear lack of integrated frameworks that treat energy consumption and computational latency as primary objectives alongside accuracy, creating space for Multi-objective Hyperparameter Optimization(MOHPO) to thrive.

By treating these operational constraints as non-dominated objectives rather than secondary filters, the MOHPO framework proposed on this dissertation enables a simultaneous Multi-objective Hyperparameter Optimization. This approach ensures that the resulting models are not only accurate but also sustainable and viable for deployment in the resource-limited devices that characterize the modern AI landscape.

In summary, the gaps identified during this research, namely the lack of drift awareness in MOO, the excessive computational cost of population-based methods and the dominance of single-objective HPO methodologies constitute the core motivation for this study.

The following chapters detail the development of the framework designed to unify these disparate domains into a scalable and adaptable solution.

Chapter 4

Methodology

This chapter introduces the planned research design, the proposed framework architecture, and the evaluation protocol in order to evaluate the Multi-Objective Hyperparameter Optimization (MOHPO) approach being developed under this dissertation.

By defining the specific performance metrics and outlining the experimental scenarios, this chapter delivers the necessary foundation to validate the proposed framework's ability to balance predictive quality with operational efficiency in real-time environments.

4.1 Framework Architecture and Implementation

The proposed framework is designed to overcome the gap between multi-objective optimization and real-time hyperparameter optimization. The development is carried out in Python, hand-in-hand with RiverML's specialized toolkits, which provide the needed structures for hyperparameter optimization in a online-manner, without the need for full retraining.

The planned architecture for the proposed MOHPO framework consist of three functional layers, and is depicted on figure 4.1. The three functional layers were established according to the objectives presented in the Problem Definition and Objectives section of this dissertation.

1. **Model Layer:** The entry point of the system where streaming data is processed. The model performs incremental learning. It is initialized with a certain hyperparameter (HP) configuration ($HP_1, \ldots, HP_n$), however this configuration can be optimized. The optimized hyperparameter configuration is provided by the optimization layer at the end of the loop, and updates its internal structure for every new instance without full retraining.

2. **Monitoring and Trigger Layer:** This layer acts as the interface between the model's output and the optimizer. It computes multiple objectives, typically Accuracy, and an additional operational or ethical metric such as Fairness[1]. Crucially, this layer also acts as a trigger, once a performance drop or concept drift is detected, it dispatches a signal to initiate a new MOHPO cycle.

[1]Fairness is used as an exemplary metric; the framework is designed to support various operational constraints like latency or energy consumption.

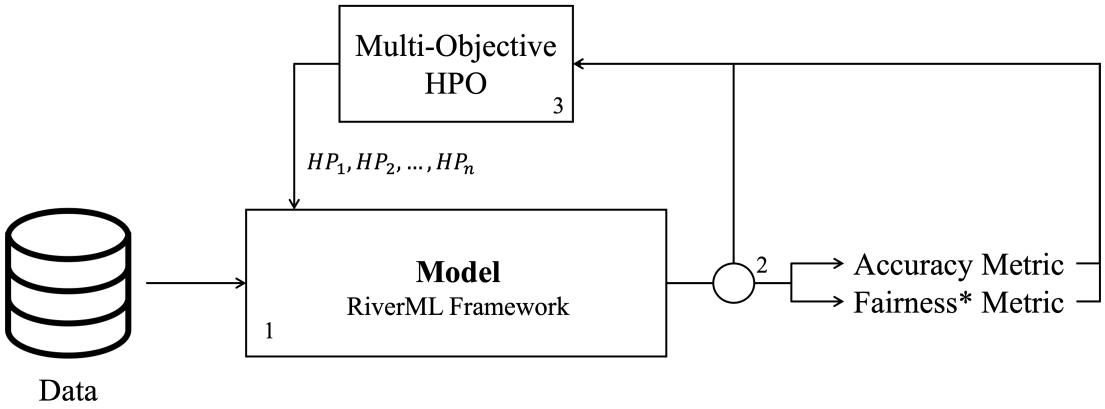


Figure 4.1: Proposed MOHPO framework architecture

3. **Multi-Objective HPO Framework:** This is the core contribution of this dissertation. Unlike traditional HPO, this module treats the current hyperparameter configuration as a multi-objective problem. The system is then triggered to optimize a population of hyperparameter configurations toward a Pareto-optimal set. It then updates the model with these superior non-dominated solutions ($HP_1, \ldots, HP_n$), ensuring the system remains synchronized with the current data stream.

Weighted Aggregation Strategy: Combining two metrics

The proposed framework, described above, adopts a weighted aggregation strategy to incorporate the importance of the metric, that is, a weight, into the optimization algorithm. Given two optimization objectives, an weight parameter $\alpha \in [0, 1]$ specifies the relative importance assigned to the first objective, while the second objective is weighted by $1 - \alpha$, ensuring that the weights remain normalized throughout the MOHPO process.

This formulation is based in the classical weighted-sum scalarization approach, widely used in multi-objective literature (Marler and Arora (2004)), including ML and HPO problems (Shahhosseini, Hu, and Pham (2022)).

4.2 Experimental Protocol: Prequential Evaluation

Evaluating algorithms in non-stationary environments dictates a departure from traditional offline evaluation metrics. As argued by Gama, Sebastiao, and Rodrigues (2013), static holdout sets are insufficient for data streams as they fail to capture the temporal evolution of the model. This dissertation adopts the sequential (Prequential) framework, based on Gama et al. (2013) framework, as the primary benchmark to evaluate the overall performance of the proposed MOHPO framework.

In this context, the Sequential protocol is used to monitor the performance of the framework, comprising the MOHPO optimizer, by processing the metrics generated throughout the stream. The execution flow for each observation d_i strictly follows these steps:

The execution flow for each observation d_i strictly follows these steps:

1. **Predict (Test):** The model, configured with a candidate set from the MOHPO optimizer, performs a prediction for instance d_i.

2. **Evaluate:** The prediction is compared with the ground truth, and performance metrics are updated incrementally.

3. **Compare:** At the end of the experimental simulations and respective evaluation, the results according to this protocol are compared and reported.

To ensure that the evaluation remains sensitive to concept drift, forgetting mechanisms as proposed by Gama et al. (2013) are implemented, as well as fading factors (α), which provide a memory-less and computationally efficient estimation suitable for resource-constrained IoT devices.

These factors allow the sequential benchmark to prioritize recent metrics, ensuring that the final assessment of the MOHPO framework reflects its ability to maintain or improve model performance in response to dynamic shifts in the data distribution.

4.3 Multi-Metric Performance Indicators

To fulfill the **Simultaneous Multi-metric Optimization (Goal 1)**, we evaluate the Pareto-optimal configurations using a multi-dimensional set of metrics:

- **Predictive Quality:** Measured via a baseline accuracy metric, such as *RMSE* or Accuracy, allowing for an accuracy assessment.

- **Operational Efficiency:** *Training Latency* (time per example) and *RAM-Hours* (memory footprint over time) are tracked to ensure the framework's suitability for real-time deployment.

4.4 Proposed Scenarios and Baselines

The MOHPO framework will be benchmarked against two distinct strategies to validate its contributions:

- **Random Search:** A stochastic baseline to justify the necessity of evolutionary metaheuristics.

- **Single-Objective Online HPO:** Utilizing the *SSPT* algorithm to compare the benefits of explicit multi-objective trade-offs against accuracy-only tuning.

To establish a standard baseline for evaluation the performance of the proposed framework and compare its results with the other selected HPO methodologies, the Hoeffding Tree Classifier, available in the River ML API was selected and used as the reference stream classification model.

To ensure fair and reproducible performance comparisons across streaming iterations, accuracy as a metric for predictive quality and RAM-hours for resource consumption were measured accross all test instances and all datasets used.

Other experimental execution metrics besides accuracy for predictive quality and RAM-hours for resource consumption were monitored during the executions, such as latency per instance and peak memory consumption

4.5 Datasets

To evaluate the performance of the proposed framework, both synthetic and real-life datasets available in the River ML interface were selected.

The synthetic datasets were designed to simulate data stream scenarios, and do not represent real-world observations. Although, they do not represent real-world observations, their controlled generation mechanisms allow specific characteristics of streaming environments to be reproduced, providing a suitable setting for evaluating the performance of the proposed framework, under controlled experimental conditions.

On the other hand, the real-life datasets were designed to simulate data stream scenarios, and represent real-world observation. The data contained in these datasets were retrieved from multiple sources and has been through proper data treatment and data anonymization procedures before being available for use in the River ML API.

The datasets are described further in the following sub-sections, and all of the datasets are available in the River ML API.

4.5.1 Dataset 1: Agrawal (Synthetic)

The Agrawal dataset is a synthetic data stream generator, introduced by Agrawal, Imielinski, and Swami (1993) in 1993, and was a common source of data for early work on scaling up decision tree learners.

Each observation containing nine features representing demographic and financial attributes, six numeric and three categorical. There are 10 functions defined for generating binary class labels from the features, and, presumably, these determine the target variable, that is, whether the loan should be approved or not.

An important characteristic of this dataset for the present study is its support for concept drift, making it particularly suitable for evaluating online learning algorithms in evolving data stream environments.

In this study, classification function 0 was selected, with a random seed of 42. A total of 50 000 observations were generated and used in the experiments.

4.5.2 Dataset 2: Credit Card

The Credit Card dataset is a real-world benchmark dataset collected from European cardholders, designed for binary classification and streaming fraud detection tasks.

Each observation represents a credit card transaction consisting of 30 continuous numerical features. To preserve user privacy, 28 of these features are the result of a PCA transformation, while the remaining two features represent the transaction amount and the elapsed time between the given transaction and the first record in the stream. The target variable is binary, indicating whether a transaction is legitimate or fraudulent.

An important characteristic of this dataset for the present study is its extreme class imbalance, making it particularly suitable for evaluating online learning algorithms under severely skewed class distributions.

In this study, the dataset was processed in its standard streaming configuration, and a total of 50 000 observations were generated and used in the experiments.

4.5.3 Dataset 3: Higgs

The Higgs dataset is a large-scale real-world benchmark dataset produced via Monte Carlo simulations, designed for binary classification tasks in high-energy physics and streaming machine learning.

Each observation consists of 28 continuous kinematic features measured by particle detectors. The first 21 features correspond to kinematic properties of physical particles, while the remaining 7 high-level features are functions derived by physicists to enhance discrimination. The target variable is binary, representing whether a physical process produces a Higgs boson signal or a background noise event.

An important characteristic of this dataset for the present study is its massive scale combined with subtle feature interactions, making it particularly suitable for evaluating the scalability and convergence speed of online learning algorithms under big data streaming conditions.

In this study, the dataset was processed in its standard streaming configuration. A total of 50 000 observations were generated and used in the experiments.

4.5.4 Dataset 4: Hyperplane (Synthetic)

The Hyperplane dataset is a synthetic data stream generator designed for binary classification tasks in streaming environments, where it a problem of prediction class of a rotation hyperplane.

Each observation consists of a set of continuous features, where the target variable is determined by whether a weighted linear combination of these features exceeds a specified threshold. Points that fall on one side of the hyperplane boundary are labeled positive, while those on the other side are labeled negative.

An important characteristic of this dataset for the present study is its ability to simulate continuous concept drift, making it particularly suitable for evaluating online learning approaches in evolving environments. Concept drift is introduced by incrementally altering the feature weights over time, which causes the orientation and position of the hyperplane decision boundary to rotate smoothly throughout the stream.

In this study, the dataset was configured with 2 features,and a random seed of 42 were used. A total of 50 000 observations were generated and used in the experiments.

4.5.5 Dataset 5: Random Radial Basis Function (Synthetic)

The Random RBF (Radial Basis Function) dataset is a synthetic data stream generator designed for classification tasks in streaming environments.

Each observation consists of a set of continuous features generated around a set of random centroids in the feature space, and each centroid is assigned a random central position, a standard deviation, a weight, and a class label. Instances are synthesized by selecting a centroid at random, but taking into account their weights, and offsetting its attributes in a random direction using a Gaussian distribution.

This process will create a normally distributed hypersphere of samples on the surrounds of each centroid.

In this study, the dataset was configured with 4 features, 4 target classes, 50 centroids, and random seed of 42. A total of 50 000 observations were generated and used in the experiments.

4.5.6 Dataset 6: SEA (Synthetic)

The SEA dataset is a synthetic data stream generator, and is designed for binary classification tasks in streaming environments.

Each observation consists of three features, of which only the first two contribute to the classification decision, while the third feature is irrelevant to the target. The target variable is binary and is determined according to whether the sum of the first two features exceeds a predefined threshold.

This generator provides four variants, corresponding to different classification thresholds. For instance, in the selected variant, variant 0, an observation is assigned to the positive class when $x_1 + x_2 > 8$. Therefore, the different variants define different decision boundaries while maintaining the same general structure of the classification problem.

An important characteristic of this dataset for the present study is its ability to simulate abrupt concept drift, particularly suitable for evaluating approaches designed to operate in evolving data stream environments. This can be achieved by changing the classification threshold during the stream, modifying the relationship between the input features and the target variable.

In this study, variant 0 was selected, with a noise level of 10% and a random seed of 42. A total of 50 000 observations were generated and used in the experiments.

4.5.7 Dataset 7: Shuttle

The Shuttle dataset is a real-world dataset originating from the UCI Statlog benchmark, cast as a binary anomaly detection task following the ODDS (Outlier Detection Data Sets) standard.

Each observation consists of nine numerical features representing sensor readings recorded by a NASA space shuttle. In its standard online streaming adaptation, the original multiclass setup is converted into a binary problem by treating the dominant majority class ("Rad Flow") as the normal inlier class, while grouping the rare classes to form the anomaly target.

An important characteristic of this dataset for the present study is its highly imbalanced class distribution, making it particularly suitable for evaluating online anomaly detection algorithms in streaming environments. Because the majority class accounts for approximately 80% of the original data, models must effectively identify rare events without being overwhelmed by normal operating conditions.

In this study, the dataset was processed sequentially in its standard streaming configuration. A total of 49 097 observations were used in the experiments, the totality of the dataset.

Chapter 5

Results

This chapter evaluates the predictive performance, adaptability, and resource efficiency of the proposed real-time MOHPO framework within streaming and IoT environments.

To assess these characteristics, the evaluation focuses on predictive accuracy as the primary measure of predictive quality and RAM-hours as the metric for computational resource consumption. Meanwhile, the performance of the proposed framework is benchmarked across multiple operational configurations and directly compared against well-established single-objective HPO.

Throughout this chapter, a thorough statistical analysis of the results is presented, detailing the trade-offs between predictive quality and resource efficiency under real-time streaming conditions. These key findings are therefore the foundation for the following discussion, where we review the key findings, connecting these with real-life applications and academic research and discussing their broader impacts and limitations. At the end of this chapter, future works are also presented to leverage further research in the MOHPO field.

5.1 Results Analysis

This section analyzes the results across the performance indicators described in section 4.3. Table 5.1 and table 5.2 present, respectively, the metric values for Accuracy and RAM-hours, that is, a predictive quality metric and an operational efficiency measure. Additionally, further statistical analysis was also performed on these results, in order to analyze their statistical relevance.

Regarding accuracy values (Table 5.1), it's possible to see that the results were consistent across all methods, despite the different values for each dataset. This is due to the different characteristics of each dataset, for which another ML algorithm might work better than the one used in the scope of this dissertation (Hoeffding Tree Classifier).

Table 5.1: Performance Comparison: Accuracy

Dataset	RandomSearch	SSPT	MOSSPT_70	MOSSPT_90	MOSSPT_60	MOSSPT_30	MOSSPT_10	MOSSPT_40
Shuttle	0.995906	0.994867	0.995234	0.995329	0.995397	0.995421	0.995451	0.995481
RandomRBF	0.656680	0.642560	0.648900	0.651027	0.649600	0.648800	0.648123	0.648849
Agrawal	0.997360	0.992740	0.993700	0.993073	0.992755	0.992584	0.992470	0.992169
HIGGS	0.650080	0.637800	0.632330	0.633913	0.632150	0.631092	0.630367	0.629849
Hyperplane	0.955740	0.951820	0.952160	0.951967	0.951755	0.951728	0.951657	0.951606
Credit Card	0.997020	0.997040	0.997040	0.997040	0.997040	0.997040	0.997040	0.997040
SEA	0.880980	0.880260	0.879760	0.879107	0.879020	0.879064	0.879127	0.879134

However, regarding the RAM-hours results (Table 5.2), the findings suggest an improvement over traditional HPO methods like Random Search, and hint at a better performance on complex, large-scale datasets such as HIGGS compared with SSPT, a more lightweight HPO approach.

Table 5.2: Performance Comparison: RAM-Hours

Dataset	RandomSearch	SSPT	MOSSPT_70	MOSSPT_90	MOSSPT_60	MOSSPT_30	MOSSPT_10	MOSSPT_40
Hyperplane	0.000066	0.000002	0.000003	0.000003	0.000005	0.000004	0.000004	0.000004
Agrawal	0.000103	0.000005	0.000004	0.000041	0.000008	0.000011	0.000011	0.000042
HIGGS	0.003265	0.000195	0.000025	0.000045	0.000026	0.000026	0.000025	0.000025
Credit Card	0.000150	0.000006	0.000007	0.000010	0.000005	0.000006	0.000005	0.000007
SEA	0.000100	0.000005	0.000004	0.000005	0.000003	0.000015	0.000005	0.000008
Shuttle	0.000068	0.000003	0.000006	0.000006	0.000003	0.000005	0.000004	0.000005
RandomRBF	0.000136	0.000005	0.000007	0.000006	0.000007	0.000005	0.000012	0.000009

5.2 Statistical Results Analysis

To obtain the statistical results, the Python package *autorank* by Herbold (2020) was used, and the statistical analysis was conducted for 8 populations with 7 paired samples for both metrics. The family-wise significance level of the tests is $\alpha = 0.05$.

The package *autorank* (Herbold (2020)) automatically checks the data's distribution and variance in order to select the correct parametric or non-parametric statistical test for comparing algorithms. It then calculates post-hoc pairwise comparisons and formats the results into publication-ready reports.

5.2.1 Accuracy Metric

When checking the distribution of the data for the accuracy metric, we did not reject the null hypothesis that the population is normal for all populations, since the minimal observed $p - value = 0.014$. Therefore, we assume that all populations are normal.

Additionally, we applied Bartlett's test for homogeneity and failed to reject the null hypothesis ($p = 1.000$) that the data is homoscedastic. Thus, we assume that our data is homoscedastic.

Having that in mind, we use repeated measures ANOVA as omnibus test to determine if there are any significant differences between the mean values of the populations. If the results are significant, we use the post-hoc Tukey HSD test to infer which differences are significant.

We reject the null hypothesis ($p = 0.003$) of the repeated measures ANOVA that there is a difference between the mean values of the populations. Descriptive statistics are provided in Table 5.3.

Therefore, we assume that **there is a statistically significant difference** between the mean values of the populations.

Based on the post-hoc Tukey HSD test, we assume that there are no significant differences within the following groups: MOSSPT_10, MOSSPT_30, MOSSPT_40, MOSSPT_60, SSPT, MOSSPT_90, MOSSPT_70, and RandomSearch. All other differences are significant.

Table 5.3: Descriptive statistics across conditions.

	Mean ± CI	**SD**
MOSSPT_10	0.871 ± 0.137	0.163
MOSSPT_30	0.871 ± 0.137	0.163
MOSSPT_40	0.871 ± 0.137	0.163
MOSSPT_60	0.871 ± 0.137	0.163
MOSSPT_70	0.871 ± 0.137	0.163
SSPT	0.871 ± 0.137	0.163
MOSSPT_90	0.872 ± 0.137	0.162
RandomSearch	0.876 ± 0.137	0.158

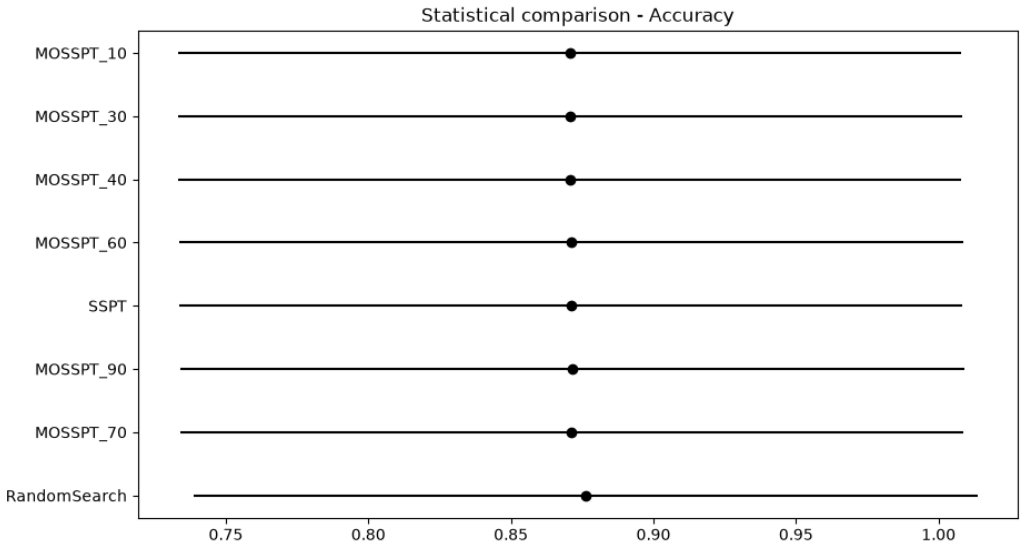


Figure 5.1: Accuracy: Statistical Comparison

5.2.2 RAM-hours Metric

When checking the data's distribution for the RAM-hours metric, we rejected the null hypothesis that the population is normal ($p < 0.05$) for the following populations:

- MOSSPT_30 ($p < 0.001$)
- MOSSPT_60 ($p < 0.001$)
- MOSSPT_70 ($p = 0.001$)
- MOSSPT_90 ($p = 0.002$)
- MOSSPT_40 ($p = 0.005$)

Consequently, non-normality was assumed across all populations.

Because we have more than two populations and some of them are not normal, we use the non-parametric Friedman test as an omnibus test to determine if there are any significant

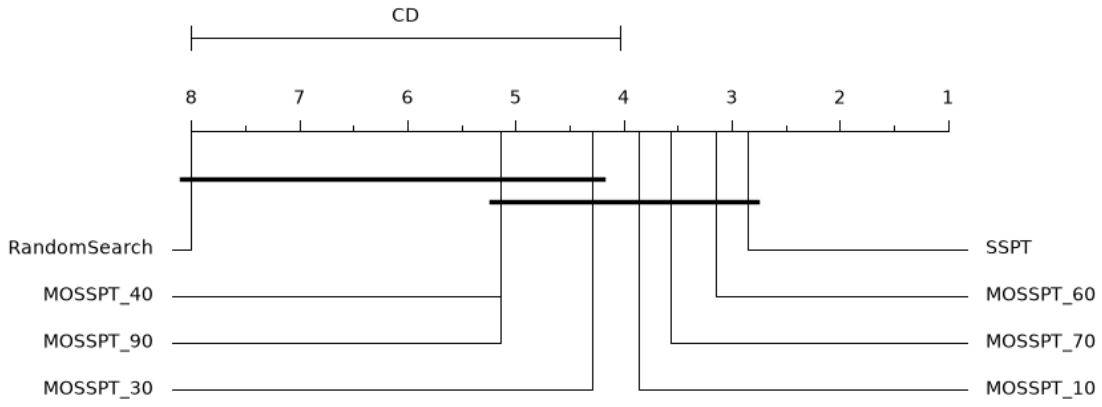


Figure 5.2: RAM-Hours: Critical Difference Diagram

differences between the median values of the populations, followed by the post-hoc Nemenyi test to infer which differences are significant. Differences between populations are significant if the difference of the mean rank is greater than the critical distance CD $= 3.968$ of the Nemenyi test.

The Friedman test revealed a statistically significant difference in central tendencies across the evaluated populations ($p = 0.002$). While median values were identical across groups (MD $= 0.000$, MAD $= 0.000$), mean ranks varied significantly, as shown in Table 5.4.

Table 5.4: Friedman test mean ranks (MR) across conditions.

Population	Mean Rank (MR)
SSPT	2.857
MOSSPT_60	3.143
MOSSPT_70	3.571
MOSSPT_10	3.857
MOSSPT_30	4.286
MOSSPT_90	5.143
MOSSPT_40	5.143
RandomSearch	8.000

Based on the post-hoc Nemenyi test, we assume that there are no significant differences within the following groups - all other differences are significant:

- SSPT, MOSSPT_60, MOSSPT_70, MOSSPT_10, MOSSPT_30, MOSSPT_90, and MOSSPT_40;
- MOSSPT_30, MOSSPT_90, MOSSPT_40, and RandomSearch.

5.3 Discussion

Considering the results previously presented, it is possible to understand that while predictive accuracy is virtually identical across all configurations, and thus the predictive quality

provided by the HPO and MOHPO frameworks is solid, the lightweight single-objective and the multi-objective frameworks, that is, SSPT and MOSSPT variants, provide superior resource efficiency compared to RandomSearch, which consumes substantially more RAM-hours without offering meaningful accuracy gains.

Having in consideration the operation mode of these different HPO methods, this is an expected result. It is expected for these well-established methods to improve the predictive quality of a ML algorithm, however, some traditional methods, such as *Random Search*, are computationally expensive. Lightweight approaches such as *SSPT*, proposed by Veloso et al. (2021), do not require as much computational resources due to their efficient architecture, however still lack the MOO component that this dissertation aimed to propose, where the balance between two objectives can be found allowing for a flexible and lightweight ML algorithm. The proposed MOHPO framework, MOSSPT, behaved as expected when balancing predictive quality and computational resources as the two objectives to coordinate when optimizing the set of hyperparameters for the used ML algorithm, a classification tree available at the River API. That is, when balancing these two objectives, MOSSPT was able to search the hyperparameter search in a computationally cheap manner while still providing good results at different importance values for each objective.

Therefore, the implementation proposed in this dissertation has the potential to be applied to online data stream scenarios, since it has proved its efficient resource consumption while still delivering good predictive results, allowing finetuning on both objectives' importance if required, a useful feature for times and fields where resource allocation isn't as broad and decisions have to be made.

Chapter 6

Conclusion

This dissertation aims to address a critical gap at the intersection of data stream learning and multi-objective optimization, condensing the objectives into the two research questions established in the introductory chapter.

To date, the literature review has confirmed the need for transcending single-metric optimization approaches, demonstrating that in IoT and resource-constrained environments, predictive accuracy must be balanced against ethical and operational efficiency, such as fairness, latency and energy consumption. This theoretical foundation served as the basis for designing an architecture that is not only capable of real-time hyperparameter tuning but is also resilient to concept drift.

Moreover, regarding the first research question, the developments under this work have proved that combining weighted sums with lightweight hyperparameter search methodologies is an efficient manner for MOHPO in real-time data stream environment.

Regarding the second research question, the results showed that the proposed framework behaved as expected, improving the trade-offs between predictive performance and resource efficiency when compared with conventional, however refined, HPO approaches.

However, this implementation has multiple improvement opportunities regarding its current limitations, leveraging future works. First, it would be very interesting to understand how MOSSPT would behave in offline settings, studying not only online but also offline scenarios, with larger datasets, both in features and in records. This suggestion is made having in mind that the datasets used for the purpose of the development of this dissertation were small in comparison with real-life data stream scenarios.

Additionally, MOSSPT can be upgraded to take even more objectives for optimization, upgrading the MOO component, and even take (multiple) complex objective functions for MOHPO. This suggestion stems from the fact that in its proposed architecture, MOSSPT actually accepts if the user defines a function as the objective function to be optimized besides the predictive quality metric, and therefore it has the potential to accept different kinds of objective functions, and complex ones at that. This would provide an extremely flexible framework for MOHPO.

Concluding, this framework has space for improvements regarding its limitations, as exposed in the previous chapter, however, serves as a foundation for the implementation of MOHPO in online settings. In the future, MOHPO could be the norm for hyperparameter optimization.

Bibliography

Agrawal, R., Imielinski, T., & Swami, A. (1993). Database mining: A performance perspective. *IEEE transactions on knowledge and data engineering*, *5*(6), 914–925.

Bischl, B., Binder, M., Lang, M., Pielok, T., Richter, J., Coors, S., … Lindauer, M. (2023). Hyperparameter optimization: Foundations, algorithms, best practices, and open challenges [Journal Article]. *WIREs Data Mining and Knowledge Discovery*, *13*(2), e1484. Retrieved from https://wires.onlinelibrary.wiley.com/doi/abs/10.1002/widm.1484 doi: https://doi.org/10.1002/widm.1484

Chen, Z., Zhou, Y., Huang, Z., & Xia, X. (2022). Towards efficient multiobjective hyperparameter optimization: A multiobjective multi-fidelity bayesian optimization and hyperband algorithm [Conference Proceedings]. In (p. 160-174). Springer International Publishing.

Coello Coello, C. A. (2001). A short tutorial on evolutionary multiobjective optimization. In *International conference on evolutionary multi-criterion optimization* (pp. 21–40).

Deb, K., & Jain, H. (2014). An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, *18*(4), 577-601. doi: 10.1109/TEVC.2013.2281535

Deb, K., Pratap, A., Agarwal, S., & Meyarivan, T. (2002). A fast and elitist multiobjective genetic algorithm: Nsga-ii. *IEEE Transactions on Evolutionary Computation*, *6*(2), 182-197. doi: 10.1109/4235.996017

Feurer, M., & Hutter, F. (2019). Hyperparameter optimization. In *Automated machine learning: Methods, systems, challenges* (pp. 3–33). Springer.

Gama, J., Sebastiao, R., & Rodrigues, P. P. (2013). On evaluating stream learning algorithms. *Machine learning*, *90*(3), 317–346.

Gama, J., Žliobaitė, I., Bifet, A., Pechenizkiy, M., & Bouchachia, A. (2014, March). A survey on concept drift adaptation. *ACM Comput. Surv.*, *46*(4). Retrieved from https://doi.org/10.1145/2523813 doi: 10.1145/2523813

Guerreiro, A. P., Fonseca, C. M., & Paquete, L. (2021, July). The hypervolume indicator: Computational problems and algorithms. *ACM Comput. Surv.*, *54*(6). Retrieved from https://doi.org/10.1145/3453474 doi: 10.1145/3453474

Herbold, S. (2020). Autorank: A python package for automated ranking of classifiers. *Journal of Open Source Software*, *5*(48), 2173. Retrieved from https://doi.org/10.21105/joss.02173 doi: 10.21105/joss.02173

Hernandez-Lobato, D., Hernandez-Lobato, J., Shah, A., & Adams, R. (2016, 20–22 Jun). Predictive entropy search for multi-objective bayesian optimization. In M. F. Balcan & K. Q. Weinberger (Eds.), *Proceedings of the 33rd international conference on machine learning* (Vol. 48, pp. 1492–1501). New York, New York, USA: PMLR. Retrieved from https://

proceedings.mlr.press/v48/hernandez-lobatoa16.html

Im, D. J., Savin, C., & Cho, K. (2021). Online hyperparameter optimization by real-time recurrent learning. *arXiv preprint arXiv:2102.07813*.

Jain, H., & Deb, K. (2014). An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part ii: Handling constraints and extending to an adaptive approach. *IEEE Transactions on Evolutionary Computation*, *18*(4), 602-622. doi: 10.1109/TEVC.2013.2281534

Karl, F., Pielok, T., Moosbauer, J., Pfisterer, F., Coors, S., Binder, M., … Bischl, B. (2023). Multi-objective hyperparameter optimization in machine learning—an overview [Journal Article]. *ACM Trans. Evol. Learn. Optim.*, *3*(4), Article 16. Retrieved from https://doi.org/10.1145/3610536 doi: 10.1145/3610536

Lin, C., Guo, M., Li, C., Yuan, X., Wu, W., Yan, J., … Ouyang, W. (2019). Online hyperparameter learning for auto-augmentation strategy. In *2019 ieee/cvf international conference on computer vision (iccv)* (p. 6578-6587). doi: 10.1109/ICCV.2019.00668

Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., & Zhang, G. (2019). Learning under concept drift: A review. *IEEE Transactions on Knowledge and Data Engineering*, *31*(12), 2346-2363. doi: 10.1109/TKDE.2018.2876857

Marler, R. T., & Arora, J. S. (2004). Survey of multi-objective optimization methods for engineering. *Structural and multidisciplinary optimization*, *26*(6), 369–395.

Nelder, J. A., & Mead, R. (1965). A simplex method for function minimization. *The computer journal*, *7*(4), 308–313.

Shahhosseini, M., Hu, G., & Pham, H. (2022). Optimizing ensemble weights and hyperparameters of machine learning models for regression problems. *Machine Learning with Applications*, *7*, 100251.

Singer, S., & Nelder, J. (2009). Nelder-Mead algorithm. *Scholarpedia*, *4*(7), 2928. (revision #91557) doi: 10.4249/scholarpedia.2928

Teich, J. (2001). Pareto-front exploration with uncertain objectives. In *International conference on evolutionary multi-criterion optimization* (pp. 314–328).

Veloso, B., Gama, J., Malheiro, B., & João, V. (2021). Hyperparameter self-tuning for data streams [Journal Article]. *Information Fusion*, *76*, 75-86. Retrieved from https://www.sciencedirect.com/science/article/pii/S1566253521000841 doi: https://doi.org/10.1016/j.inffus.2021.04.011

Zhan, H., Gomes, G., Li, X. S., Madduri, K., & Wu, K. (2018). *Efficient online hyperparameter optimization for kernel ridge regression with applications to traffic time series prediction.* Retrieved from https://arxiv.org/abs/1811.00620

Zhou, Y., Liu, W., & Li, B. (2019). Efficient online hyperparameter adaptation for deep reinforcement learning. In *International conference on the applications of evolutionary computation (part of evostar)* (pp. 141–155).