Department of Computing

Individual Research Report

Model-Based Uncertainty Quantification for Quality-Diversity Optimisation

---

10/05/2023 - 11/09/2023

Author:

Louis BERTHIER

Email : louis.berthier22@imperial.ac.uk

CID : 02285087

Supervisors:

Dr. Antoine CULLY

Email : a.cully@imperial.ac.uk

PhD candidate Manon FLAGEAT

Email : manon.flageat18@imperial.ac.uk

PhD candidate Bryan LIM

Email : bryan.lim16@imperial.ac.uk

Second Marker:

Dr. Mark VAN DER WILK

Email : m.vdwilk@imperial.ac.uk

Submitted in partial fulfilment of the requirements for the MSc degree in Computing (Artificial Intelligence and Machine Learning) of Imperial College London

Abstract

QD algorithms are optimization algorithms that return a collection of solutions that are both optimal and varied, unlike conventional optimization algorithms which return a single and unique solution. This set of solutions enables the agent to adjust its behaviour in an environment to meet one or more objectives under different constraints, in the most optimal way possible. Like most algorithms, these face the problem of uncertain and noisy environments, which disrupt the evaluation of solutions and their reliability. One proposal for limiting the impact of this uncertainty is to adapt sampling approaches to avoid lucky solutions, enhance their reliability, and bring them closer to the expected ground truth solutions. Unfortunately, these algorithms are not very data-efficient by design and operation, and adding sampling on top of them may increase performance in uncertain environments, but it considerably reduces data efficiency. This research project focuses on the use of model-based methods coupled with sampling approaches within QD algorithms and uncertainty quantification to cope with uncertain environments. The aim is, on the one hand, to enable the neural network to propose robust and reliable solutions despite noise, and on the other hand, to improve the data efficiency of QD algorithms by limiting solution evaluations across the environment. To this end, eight algorithms are proposed in this research project. These contributions propose different combinations of three mechanisms, namely: (1) a repertoire reset, (2) the model design and (3) a sampling method. The performances of these algorithms are evaluated using a redundant robotic arm task, under deterministic and uncertain conditions, although the main environment of this project is the non-deterministic one. These performances are compared with two baselines, MAP-Elites and MAP-Elites-Sampling. The best-performing contribution beats the baseline established by MAP-Elites but performs slightly worse than the second baseline defined by MAP-Elites-Sampling. However, the behaviour of the model and the results associated with this best contribution are promising and demonstrate the value of Model-Based Quality-Diversity algorithms with uncertainty quantification, especially in uncertain environments.

Acknowledgments

Foremost, I would like to extend my heartfelt gratitude to **Dr. Antoine Cully** for giving me the opportunity to join his laboratory and for extending the scope of this project following my ISO.
I would also like to express my appreciation to him, **Manon Flageat** and **Bryan Lim** for listening to me, guiding me throughout this individual research project and for their exceptional mentorship. Their astute support and insightful advice enabled me to overcome the many challenges I faced.

My gratitude extends to **the members of the Adaptive and Intelligent Robotic Lab**, including the master and PhD students for their support and presence, during our meetings and engaging discussions.

I would also like to thank **Dr. Mark van der Wilk** for his co-supervision and the attention he gave me to ensure the supervision, my well-being and the progress of this project.

Last but certainly not least, I would like to thank **my family and friends** who have supported me tirelessly throughout this project and enabled me to give my best.

Table of Contents

List of Figures

List of Equations

List of Tables

Chapter A

Introduction

This Individual Research Project represents a continuation of the Independent Study Option conducted from January to May [1]. Thus, this master's thesis focuses on enhancing QD algorithms, particularly in uncertain environments, by combining such algorithms with both model-based and sampling approaches to assess and use the uncertainty associated with the predicted solutions.

A.1 Motivation and Contribution

Optimization is the process of **finding the most optimal solution to a given environment**, task and its associated objectives and constraints. From an industrial standpoint, this generally involves maximizing yield and minimizing costs, although other criteria such as quality, safety, efficiency, and numerous other considerations can also be addressed. Therefore, optimization is a key element in the development of new products and the proposition of novel technologies and methods.

Diverging from classical algorithms like reinforcement learning (RL), quality-diversity (QD) algorithms are optimization algorithms inspired by evolutionary algorithms (EAs). They return **a collection of solutions that must be both diverse**, offering several approaches, **and high-quality**, to effectively solve the problem at hand [2, 3]. Unfortunately, some of the environments in which QD algorithms are applied are **uncertain**. Indeed, this uncertainty is **a ubiquitous feature of the real world**, manifesting as measurement noise, incomplete information, external interference and other relevant aspects contingent upon the environment under consideration [4, 5]. **Such uncertainty impacts algorithms**, particularly in the evaluation of their solutions. Consequently, QD algorithms are not immune to these undesirable effects and tend to be **inefficient in uncertain environments** [6].

This is where model-based (MB) approaches come in. **They approximate the dynamics of the environment** through model learning. Such approaches aim to guide the agent according to the information it receives as input and the plans predicted by the trained model. To tackle uncertain environments, **the goal is to combine QD algorithms with these MB approaches and quantify the uncertainty of the obtained solutions**. This quantification serves as guidance for the algorithm and the agent during the simulation of the task at hand, **avoiding unrepresentative solutions** and enabling the model to **better capture the relationship between inputs and outputs** [7, 8].

In chapter B, we recall the challenges of uncertainty in non-deterministic environments and review the principle of QD algorithms. This chapter also presents MB approaches and various considerations for using QD algorithms in an uncertain environment. Then, chapter C introduces the work proposed in this research project, notably through a combination of 3 mechanisms that form 8 new methods for dealing with a non-deterministic environment. Additional considerations are also introduced to ensure the proper functioning and assessment of these algorithms, via new hyperparameters and metrics respectively. Chapter D presents the redundant robotic arm task used to compare algorithms.
Chapter E presents the results via an ablation study, starting with an in-depth analysis of the most-performing contribution. Finally, chapter F summarizes and discusses the proposed algorithms, before exploring new perspectives to improve the approaches implemented through this research project.

A.2 Social Stakes and Ethical Considerations

Concerning the ethical considerations surrounding this project, it is important to emphasize that it is devoid of any involvement with humans or animals. This research project aims to improve QD algorithms, particularly for uncertain environments. However, it should be noted that there is still a significant amount of work needed before such contributions can be effectively applied in real-world environments. Additionally, this project does not use any external data sources and does not possess any military or hazardous social or environmental objectives. Instead, it focuses on the development of a robotic application within simulated and non-real environments such as a simulated robotic arm. Although robotics can be used for non-ethical purposes, this is not the case here, thanks to the environments and tasks considered. Such working environments are accessible through the QDax research framework. QDax is the result of a collaboration between InstaDeep and the Adaptive & Intelligent Robotics Lab (AIRL) of Imperial College London. This framework serves research purposes and is open-source, allowing users to freely access and utilize it. On the whole, this work is more theoretical. The aim is to improve the resilience and performance of algorithms in the presence of noise. The final decision on how to apply the contributions introduced through this project rests with the man himself. However, neither AIRL nor I have any intention of using the contributions proposed through this project for purposes that might (1) endanger people or the environment, or (2) negatively impact any other ethical considerations. As a result, the ethical integrity of the project is respected and it is aligned with responsible research practices.
A checklist verifying potential ethical issues surrounding the project is available in appendix A.

Chapter B

Literature Review - ISO Continuation

As emphasized in Chapter A, this Individual Research Project is a continuation of my previous work conducted during my Independent Study Option. Therefore, this chapter draws on and contains information and references from the previous work. Although the content has been modified and paraphrased to avoid self-plagiarism, the idea remains the same, resulting in an overlap in terms of the background information between the two projects. Consequently, this section serves primarily as a summary of the relevant literature from the ISO with additional information [1].

B.1 Uncertainty as a Key Challenge for Algorithms

QD optimization algorithms are used in a wide range of fields, including (1) **video games** for generating environments or agents [9, 10, 11], (2) **robotics** for controlling simulated or real agents [12, 13, 14], (3) **industrial design**, particularly in the context of fluid mechanics [15, 16, 17], (4) **human-robot communication** in the same environment [18, 19], and many **other and various domains** [20]. These algorithms are designed for use in both simulated and real environments.

Unfortunately, such algorithms face a significant challenge: **uncertainty**. Indeed, although real environments are certainly more affected by this problem, simulated environments are not exempt. Uncertainty can arise through several factors, such as (1) **noise in the measurement** of data or the evaluation of solutions, (2) **absence or lack of information**, (3) **parasitic external interference** as well as other factors more specific to the environment, as Fu et al. and Matott et al. pointed out [4, 5]. All these sources of uncertainty prevent the efficient use of conventional algorithms. Particularly for classical QD algorithms, this uncertainty **makes the evaluation of solutions inefficient and unreliable**, as well as **prevents proper evaluation of algorithms** and methods through conventional metrics [6, 21, 22, 23].

As reminded by Chua et al., this uncertainty can be divided into two categories: **aleatoric uncertainty** and **epistemic uncertainty**. The former corresponds to **data uncertainty** and arises **from the inherent randomness of the environment** within the data and observations. On the other hand, epistemic uncertainty stems **from a lack of data and information** about the environment, and it is also known as **model uncertainty**, as reminded by Chua et al. [24].
Abdar et al. conducted a review of uncertainty quantification (UQ) using deep neural networks (DNNs). This review covers various aspects, including a comparison of uncertainty quantification methods and their respective advantages and disadvantages [25]. They outlined that **epistemic uncertainty can be reduced** through appropriate model selection and the availability of extensive and representative data that adequately reflect the environment. Regarding aleatoric uncertainty, Abdar et al. underlined that it is **assumed to be irreducible** since it is inherent in the observed data by definition [25]. Nonetheless, techniques have been developed to address this issue, such as the approach proposed by Sambyal et al. in the medical field, which leverages **data augmentation methods** [26].

The graph shown in figure 1 is inspired by figure 3 in the article by Tuna et al., which looks at attacks on DNNs through the maximization of epistemic uncertainty [27]. It provides a visual summary of the differences between these two types of uncertainty, as well as the associated levels depending on the data available.

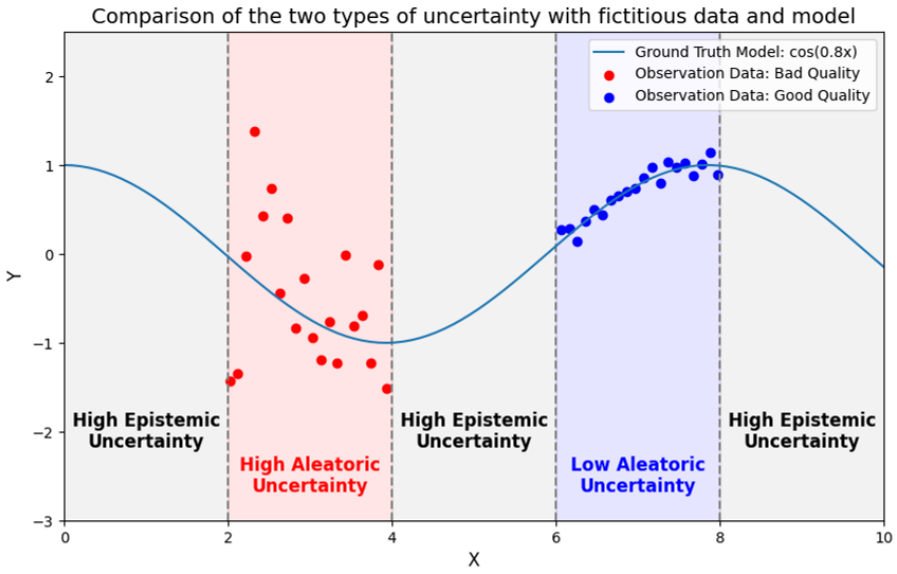


Figure 1: Graph showing uncertainties and the level associated with each, using fictitious data and a fictitious model. The shaded areas [0;2], [4;6], and [8;10] in the environment have high epistemic uncertainty due to the lack of observation data for model training. Red zone [2;4] has observation data, but they deviate from the ground truth model, assuming noise or disturbances during measurement, resulting in high aleatoric uncertainty. On the other hand, blue zone [6;8] has observation data that closely match the reality and dynamics of the environment, resulting in low aleatoric uncertainty.

Thus, uncertainty manifests itself in two categories as described by equation B.1, and at several levels depending on multiple factors such as the environment, the observed data or the selected model.

$$\text{Total Uncertainty} = \text{Epistemic Uncertainty} + \text{Aleatoric Uncertainty} \tag{B.1}$$

Unfortunately, conventional QD algorithms like MAP-Elites were not designed for uncertain but for deterministic environments in the first place, where a single solution evaluation yielded exactly the ground truth solution [28]. As emphasized by Flageat et al., MAP-Elites is affected and **sensitive to noise by its elitism**, which favours solutions with the highest fitness [6]. Consequently, **a single evaluation** of solutions proves ineffective in uncertain environments and **may be unrepresentative** of actual performance due to noise.

Therefore, researchers have been interested in developing methods to overcome this problem of uncertainty. These include the following methods:

1. **Corrected metrics**, which are adapted to evaluate the actual performance of algorithms by re-evaluating the solutions encountered [6, 21, 22, 23].
2. **Sampling methods**, which improve the evaluation of a solution by taking into account past information via old evaluations, or additional information via re-evaluations [21].
3. **MB methods**, which quantify uncertainty to guide the algorithm and the agent [7, 8].

These methods will be discussed in sections B.3.1, B.3.2 and B.4.1 respectively.

B.2 Deterministic Quality-Diversity Algorithms

QD algorithms are **based on the theory of evolution**, with the idea of a population evolving through the selection of individuals, their descendants and the survival of the best-performing candidates over time. Therefore, such algorithms are closely related to EAs, except that they consider additional criteria in addition to the fitness of solutions, notably the behavioural descriptor (BD) [2, 3].
The advantages of these algorithms over RL algorithms are that (1) they provide **a collection of the most optimal solutions**, unlike RL algorithms, which only suggest the single best-performing solution, and (2) QD algorithms can be applied to **a wider range of environments**, notably by considering non-Markov Decision Processes (MDPs) frameworks.
Finally, these QD algorithms are also efficient because they can **address optimization problems with one or several objectives**, depending on the task under consideration, the constraints imposed and the expected behaviour.

B.2.1 Algorithmic Foundations: Principles and Operations

As emphasized by Cully et al., the **objective function**, $f(\theta)$, of QD algorithms is constructed from two elements, namely (1) f_θ **the fitness**, and (2) b_θ **the BD** of the identified solution. The variable θ represents the **genotypes**, or parameters, in the search space and depends on the task to be performed and the environment. Several arrangements are proposed for these parameters to test several solutions and identify the most promising ones [2].
As reminded by Chatzilygeroudis et al., the solutions are represented in the **feature space**, which corresponds to the projection of the scores associated with each of the solutions identified and retained. The dimensions of this space depend intrinsically on the dimensions of the BD. In general, solutions are represented through a 2D space where the BD corresponds to the position of one part of the agent. Higher-dimensional spaces and BDs are possible depending on the task under consideration, but visually it becomes impossible to interpret for a space of dimension greater than 3. Finally, this feature space also contains the performance attached to each solution through its fitness, which is expressed through a colour bar established by the set of fitnesses stored in this space [3].
Finally, unlike the BD, fitness is represented by a single value. The latter assesses **the effectiveness of the corresponding performance of the solution**, whereas the BD provides insight into **how this solution accomplishes its performance** [2, 3].

Cully et al., and Chatzilygeroudis et al. highlighted **multiple criteria for modifying QD algorithms and evaluating their performance**, specifying the benefits of each method [2, 3]:

1. **Container: It stores the different solutions** according to the BD and the associated score. It also associates the solution with its genotype, to identify the combination that produced the result and provide it to the user. There are two types of containers: **the Grid** and **the Archive**, which are respectively discretized and non-discretized containers. Unlike the Grid, where the structure of the space is pre-defined by the user with a uniform division of each dimension of the space, the structure of the Archive is derived directly from a distance threshold between the BDs of the solutions generated and those stored.
   Other strategies have been adopted to improve solution management and algorithm performance, notably using **the Centroidal Voronoi Tessellation** construction highlighted by Vassiliades et al. This variation consists of dividing the Grid into regions whose shape and area vary according to the dispersion of solutions in the feature space [29].

2. **Selection Operator: It chooses a group of individuals to serve as parents** for a subsequent mutation. Depending on the preferences of the user and what he wants to promote within the algorithm, several operators are available: (1) **random selection**, which selects random genotypes potentially not stored in the container, (2) **uniform random selection**, which chooses individuals from the container with a probability inverse to the number of stored solutions, (3) **tournament selection**, which compares candidates two by two and only the one with the highest fitness is retained as a parent, (4) **roulette wheel selection**, where individuals have a greater chance of being selected in proportion to their fitness, (5) **rank selection** where selection probability is also determined based on the fitness, and a higher fitness corresponds to a higher probability of an individual being chosen [30].

3. **Alteration Operator: It modifies the parents selected with the previous operator to generate offspring**, i.e. new solutions. Here are some of the most commonly used operators to generate new genotypes: (1) **mutation operator** which introduces noise into the genotype of each parent, (2) **crossover operator** which combines the information contained in the genotype of two parents, (3) **variation operator** which is a combination of the two previous operators.

4. **Comparison Score: It measures the performance of new solutions** produced by the previous operator and **compares them with those already stored in the container**. Again, several scores are applicable depending on what the user is trying to promote to solve the task, notably: (1) **the fitness score** intrinsically linked to the task, which evaluates the performance of a solution concerning the objectives and constraints, (2) **the novelty score**, which assesses the difference of a solution to the others in the container by utilizing the average distance among the k-nearest solutions as presented in equation B.2, (3) **the curiosity score**, which evaluates the discovery of new solutions by introducing a positive reward if a solution is capable of generating offspring retained in the container and a negative reward otherwise.

$$\text{Novelty}_k(i) = \frac{1}{k}\sum_{n=1}^{k} d(i, s_n) \tag{B.2}$$

k : Number of nearest neighbours considered

$d(i, s_n)$: Distance between the new individual i and the n-th closest solution s_n in the container.

5. **Metric:** It does not measure the performance of the solutions, but that of the algorithm, to compare the different methods proposed by the researchers. It is calculated from the container and its chosen solutions, although **it is not involved in the algorithm itself**. The following is a non-exhaustive list of metrics that can be considered to assess algorithms: (1) **diversity** and **coverage**, which look respectively at the number and percentage of cells occupied by an individual in the container, and measure the difference and diversity of the solutions between containers as described in equations B.3 and B.4, (2) **performance**, through the mean, median or maximum fitness of the solutions in the container, to evaluate their efficiency about the task as denoted by the equation B.5, (3) **convergence speed**, which evaluates the number of environment steps needed to obtain a collection of solutions with a certain diversity and performance, (4) **QD score** is calculated as the sum of the fitness values of all the individuals in the container, considering both diversity and performance as shown in equation B.6.

$$\text{Diversity}(C) = N_C \tag{B.3}$$

$$\text{Coverage}(C) = 100 \times \frac{N_C}{T_C} \tag{B.4}$$

$$\text{Performance}(C) = \underset{1 \leq n \leq N_C}{\text{Ope}} \ (f_n) \tag{B.5}$$

$$\text{QD Score}(C) = \sum_{n=1}^{N_C} f_n \tag{B.6}$$

N_C : Number of cells occupied by an individual in container C
T_C : Total number of cells in container C
f_n : Fitness of the n-th solution occupying a cell in container C
Ope : An operation such as max, median or mean

Although QD algorithms may change, they share the same idea of evolving a population, as introduced in section B.2 and represented in the algorithm 1.

Algorithm 1 General Quality-Diversity Algorithm

```
Initialize randomly a container C
for Generation = 1 until Max Generation G_max do
    Select a population P (parents) from the container C using a selection operator
    Generate new individuals I (offspring) from the population P using an alteration operator
    for each individual i in I do
        Evaluate the scores S(i), for instance fitness f(i) and behavioural descriptor BD(i) of individual i
        if BD(i) is new then
            Add individual i to the container C
        else
            if f(i) > f(old) then
                Add individual i to the container C
            else
                Keep the old solution already stored in the container C and forget new individual i
return C
```

In conclusion, **QD algorithms can be customized based on several criteria**, and these choices significantly **influence their efficiency**. They can be tailored to follow different philosophies, such as prioritizing exploration for the discovery of new solutions and diversity or favouring exploitation and quality to identify the most promising solutions. In general, QD algorithms seek a compromise between the two to return multiple and different solutions, all of which are highly efficient.

This research project focuses on improving QD algorithms for uncertain environments. For the remainder of this work, **MAP-Elites** (ME) algorithm is considered, presented in section B.2.2 and modified to propose new approaches.

B.2.2 A Classical Algorithm: MAP-Elites

MAP-Elites, proposed by Mouret et al., is referred to as an **"illumination algorithm" returning "diverse and high-performing" solutions** and is based on the NSLC algorithm introduced by Lehman et al. [31]. It is elitist and particularly interested in obtaining solutions with the highest fitness for each BD. Unlike NSLC, which has two archives and favours the diversity of solutions, MAP-Elites seeks to fill its grid with solutions with the highest fitness [28].
It is depicted in figure 2 and inspired by figure 1 proposed by Berthier et al. [1].

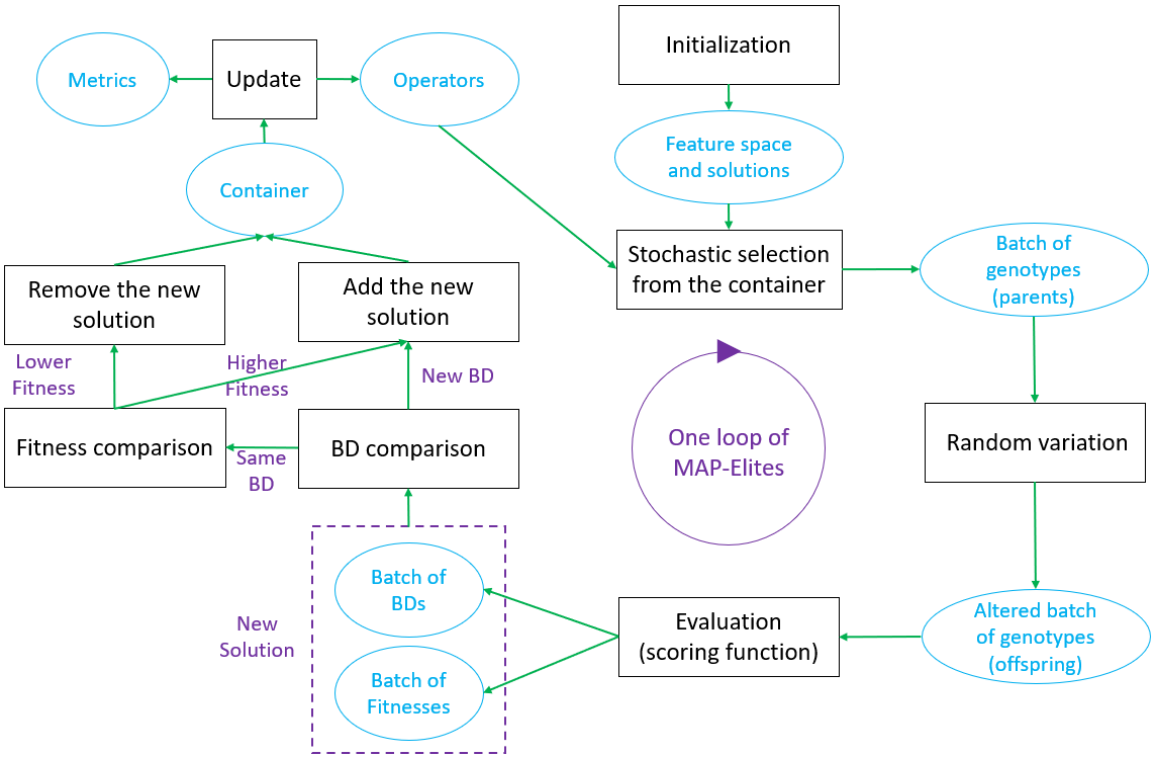


Figure 2: Diagram illustrating **the conventional loop of MAP-Elites**. This algorithm involves the following key steps: (1) initialize the feature space filled with a few solutions, (2) randomly select a batch of genotypes from the container, (3) apply random variations to the previously selected batch, (4) evaluate the scores of each individual in the altered batch, (5) compare BD and fitness, (6) delete or add the new solution to the container according to the comparisons, (7) update the operators for the algorithm and the metrics for the user. The next step is **to repeat the loop with steps (2) to (7)** for a certain number of generations.

Although MAP-Elites is prone to certain problems, notably due to its **elitism** and **low data efficiency**, it remains one of the most promising algorithms, with a mechanism that is simple and accessible to understand. As a result, many researchers have turned their attention to this algorithm, proposing new versions through the addition of new methods [29, 32, 33, 34, 35, 36, 37].

One of the most interesting versions for this research project is an approach where MAP-Elites includes **solution sampling** in its loop. This new algorithm, proposed by Justesen et al. involves re-evaluating each solution many times to get an idea of its true performance [21]. It is important to note that this variation only makes sense in the context of uncertain environments, as demonstrated by the experiments carried out by Berthier et al. [1].
Indeed, re-evaluating a solution in a deterministic environment makes no sense, since by definition of the environment, the solution obtained will always be the same. On the contrary, the re-evaluation of a solution, potentially lucky because of the noise in an uncertain environment, leads to a new one.

B.3 Uncertain Quality-Diversity Algorithms

In section B.1, it was noted that QD algorithms were originally not designed to handle uncertain environments and are not effective in such conditions. As time went by, QD algorithms were democratized and adapted to meet new requirements, particularly in real environments, where uncertainty is prevalent. Flageat et al. conducted a thorough examination of the application of QD algorithms in non-deterministic environments and proposed a comprehensive framework for addressing the problem, including a fair comparison of the different algorithms proposed. At the same time, they reminded important concepts for dealing with noise and correctly evaluating both solutions and algorithms, notably through **sampling methods**, **corrected metrics** and **loss metrics** [23].

Flageat et al. posed the problem as follows: since noise prevents a reliable and correct evaluation of solutions, it is no longer possible to consider fixed values for both fitness and BD. Instead, they have reframed the problem in terms of solutions whose fitness and BD follow individual **distribution**, with each distribution having its mean and variance. In the deterministic case, the variance is zero [23].
Thus, as outlined by Flageat et al., two key challenges need to be addressed: (1) **the performance** of each solution by evaluating the expected fitness and BD, and (2) **reproducibility** by quantifying the variance of the fitness and each dimension in the BD. To obtain these measures, each solution must be sampled several times. A desirable outcome is a solution with **high expected performance** and **low variance**, indicating both high performance and reproducibility [23].
Finally, it's worth mentioning that **reproducibility cannot always be enhanced** as it is contingent upon the nature of uncertainty. In instances where noise arises from an external source and uniformly affects all observed data, it becomes impossible to enhance reproducibility. However, if the uncertainty is specific to particular components of the system, reproducibility can be improved by opting for more robust and stable elements [23].

B.3.1 Emerging Insights for Uncertain Environments

The main way of dealing with uncertainty is through **solution sampling**, which attempts to evaluate and approximate solutions as reliably as possible, **taking into account additional information**. These sampling methods are discussed in section B.3.2. Unfortunately, not all algorithms use sampling approaches and not necessarily the same ones. Consequently, algorithms are not necessarily evaluated on the same basis, and this is one of the problems pointed out by Flageat et al. when comparing different methods. To take re-evaluations into account and **allocate the same resources to each algorithm**, they suggested the use of **sampling size**, the concept of which is presented in equation B.7, instead of batch size [23].

$$\text{Sampling Size } (S) = \text{Batch Size } (B) \times \text{Number of Reevaluations } (N) \tag{B.7}$$

Although sampling is a method for limiting the effect of uncertainty, and sampling size provides a basis for comparison between algorithms, it is not enough to have an idea of the actual behaviour and efficiency of an algorithm applied in an uncertain environment. **Corrected metrics** demonstrated their effectiveness in this context. Their calculation methods are identical to those of the metrics presented in section B.2.1, except that instead of being calculated from a container, they are calculated from a **corrected container**. The latter is simply a copy of the original container at the end of a generation, also known as an **uncorrected** or **illusory container**, where each retained solution is re-evaluated several times to approach its true fitness and BD value. It's the same principle as sampling approaches, except that **this sampling is "external"** to the algorithm and is not used by it. Like metrics, corrected metrics are only there to inform the user, without intervening in the algorithm. This principle of corrected metrics was put forward by Justesen et al. and taken up by other researchers [6, 22, 23].

Thus, the idea is to take the mean obtained for each solution after a certain number of re-evaluations, but other operations are possible, such as the median, recommended by Flageat et al. [23]. Depending on the algorithms, the sampling process changes, as in Deep-Grid Map-Elites (DGME), and in this specific case, the idea is not to take the average obtained after re-evaluations, but the average for a certain number of solutions belonging to the same cell [6, 23]. Finally, new metrics, and indeed corrected metrics, have been introduced, depending on the problem and performance sought, through new implementations [6, 21, 22, 23].

Eventually, the last point outlined through the formalism proposed by Flageat et al. [23] are **the loss metrics**, introduced by Flageat et al. [38]. These additional metrics **emphasize the effectiveness of algorithms** in addressing the two presented challenges, **performance and reproducibility**. A general example of a loss metric is given in equation B.8, and the aim is **to reduce this loss** as much as possible [23].

$$\text{Loss}_{\text{Metric}}(C) = \frac{\text{Metric}(C) - \text{Metric}(\overline{C})}{\text{Metric}(C)} \tag{B.8}$$

C : Container

$\overline{C}$: Corrected Container

As with corrected metrics, metrics can be adapted to transform them into loss metrics [22, 23, 38].

B.3.2 Sampling Approaches as a Solution to Uncertainty

As mentioned in section B.3 and reiterated by Flageat et al., QD algorithms and their solutions **must consider not static values but distributions**. Uncertainty makes it impossible to evaluate solutions reliably and accurately through a single evaluation. Therefore, one way to address this is **to combine these algorithms with sampling approaches**, the aim of which is **to avoid random solutions by considering additional information** to approximate each solution as closely as possible [23].
The nature of this information also determines the type of sampling approaches to be used, i.e. (1) **explicit sampling** or (2) **implicit sampling**. The first approach consists of **creating additional information by re-evaluating a solution** several times to determine the mean and variance of its distribution. In contrast, the second approach doesn't seek to create information, but **to take advantage of information already encountered**, storing it as the algorithm and agent evolve.

Some explicit sampling methods are considered to be **sampling-based approaches**, which by definition consider a small portion of the individuals and not the entire population to infer certain information. One solution introduced by Cantu-Paz in the context of genetic algorithms is **naive sampling**. This method, as its name suggests, is simple to understand and implement, but its inherent naivety can quickly make it **unsuitable** in terms of available resources for problems where the environments have complex dynamics and where evaluation is too costly [39]. This method is **static by definition**, since the number of re-evaluations remains constant and identical for each solution, assuming a uniform distribution of re-evaluations, which is not automatically valid in real and complex environments.
Moreover, as reminded by Cantu-Paz, uncertainty is lower as the standard deviation decreases, and this decrease is **proportional to the square root of the number of re-evaluations** considered [39].
QD algorithms already **perform poorly from a data efficiency aspect**, and combining them with naive sampling would only worsen this data inefficiency. Rakshit et al. compared different approaches to improve or at least **limit the lack of data efficiency** of sampling methods. They highlighted the benefits of **adaptive sampling**. Unlike naive sampling, adaptive sampling is **dynamic** and does not assume a uniform distribution of solutions. It aims to modify the choice and reassessment of solutions while taking into account the information obtained [40].

Thus, the naive approach is simpler to implement and more effective in an environment where solutions are assumed to be uniformly distributed, whereas the adaptive approach is more difficult to implement but is more effective in terms of re-evaluation in environments where the population is not uniformly distributed [39, 40]. Although these two approaches are too demanding in terms of computing resources and inefficient in terms of data efficiency, researchers have proposed algorithms whose sampling idea is based on them to cope with uncertainty [21, 23].

In contrast to explicit sampling, some implicit sampling methods are entitled **population-based approaches**. They consider the entire population, or at least a very large part of it, to obtain more reliable estimates of the value of solutions despite the noise. These methods are intended to be faster, as **they do not involve re-evaluation**, as demonstrated by the DGME algorithm proposed by Flageat et al. [6]. This new version of MAP-Elites introduces a depth to each cell to store solutions. During the evaluation, DGME will consider a local area of cells and associated solutions to approximate the score of new solutions as reliably as possible. The fact that DGME uses the entire locally defined population makes it a population-based method. This new mechanism with depth implies considering (1) **a new selection method** for solutions and (2) **a new addition method** for solutions [6].
The performance challenge is addressed by the fact that DGME considers past information to reliably estimate the scores of new solutions, as well as by the new selection mechanism, and the reproducibility challenge is addressed through the two new mechanisms. Similar to MAP-Elites, the process of creating a new batch of parents involves randomly selecting a cell followed by a selection that is proportional to the fitness of the solutions contained in this randomly chosen cell [28]. Finally, if a cell is not fully occupied, new solutions can be added without any restrictions. However, once the maximum capacity of a cell is reached, the fitness of solutions is no longer compared, deviating from the approach used in MAP-Elites and shown in figure 2. In contrast, a random selection is made within the selected cell to determine which old solution will be substituted for the new one. These two mechanisms **favour the solutions with the highest fitness** and **improve their stability by limiting the impact of non-representative solutions**. In conclusion, this algorithm is designed to be **"fast and stable"** as outlined by Flageat et al., particularly in uncertain environments [6].

To summarize, population-based methods are ideal for tasks that necessitate a set of solutions exhibiting great average performance, especially when evaluating individual solutions incurs significant computational costs. Conversely, sampling-based methods demonstrate superior performance in tasks where the evaluation of a single solution is expeditious and inexpensive, and the objective is to discover the most optimal solution [6]. Such results underline the significance of selecting the appropriate approach, considering the environment, task requirements and computational resources.
Unfortunately, despite the effectiveness of combining QD algorithms with such approaches in the context of uncertain environments, their applicability in real and complex environments is limited. Such algorithms necessitate a significant amount of time and do not exhibit optimal data efficiency. Consequently, it is crucial to explore new methods to address these challenges in non-deterministic and complex environments. This is where **MB approaches** become valuable, as **they aim to enhance both the identification of distributions** and their parameters through input-output relationships, as well as **data efficiency**.

B.4 Model-Based Strategies

Developing an agent and its interactions with a specific environment is both a **time-consuming** and **tedious** process. In addition, this approach severely limits the ability of the robot to interact with other environments, since it will only be effective for the environment on which the specifications depend. As a result, this method is **unviable** for requirements where the aim is to enable the robot **to adapt multiple behaviours** depending on its state and the environment. There is a real growing need for robots capable of autonomous exploration and task-solving. One solution is the utilization of **MB approaches**, which involve training the agent to capture the underlying dynamics of an environment.

These MB strategies allow the agent **to gain knowledge from limited experience** to train the model and be as close as possible to the dynamics of the environment. Once properly trained, it can be used to **plan and simulate scenarios** through sequential agent actions. To do this, the model **estimates transitions and expected rewards by approximating the dynamics of the environment**.
These strategies aim **to capture the relationship between input and output** to generalize and extend estimates from limited experience, resulting in a **data-efficient method**. Furthermore, acquiring a model enables robots to make decisions that span extended planning horizons, resulting in enhanced decision-making, particularly in challenging environments as emphasized by Swazinna et al. [41]. Nevertheless, the implementation of MB methods can be challenging and **might not be appropriate for environments with highly intricate dynamics**.
Another benefit of MB learning is its potential for **transferability**. In the presence of knowledge about a comparable environment, **a trained model can be repurposed and further trained to tackle new and different tasks**. This knowledge transfer improves data efficiency and minimizes the training time required to adapt a robot to a different environment, as outlined by Yao et al. [42].

In section B.4.1, the benefits of such strategies will be studied in the context of QD algorithms, before turning to the use of such approaches for UQ in section B.4.2.

B.4.1 Leveraging Model-Based Approaches in Quality-Diversity Algorithms

QD algorithms aim to return a collection of solutions, both "diverse and high-performing" [2, 3]. This collection translates into a discretized container. The interest of MBQD approaches is **to generalize this discrete space to a continuous space**, accurately approximating the dynamics of the environment.
Several methods have been explored to generalize this discrete space: (1) the training of **surrogate models** to simplify the expected model in complex environments and alleviate the computational burden and time constraints [43], (2) the use of a **Generative Adversarial Policy Network** which confronts a generator with a discriminator to produce policies from the continuous latent space once the generator has been correctly trained [44], (3) an upgraded version of MAP-Elites involving Bayesian Optimization and **Gaussian Processes** (GPs) as models, which are trained based on the last container produced to select the most interesting solutions in the classical MAP-Elites loop [12, 45], and (4) **locally approximate with a linear function the objective function** associated with the task to be solved for each solution identified, enabling **the application of numerous gradient descent steps** and leading to more promising solutions before adding them to the container [46].

The generalization offered by these MBQD approaches should make it possible, firstly, **to improve and reliably ensure the predictions made by the model** for both explored and unexplored regions, and secondly, **to improve the data efficiency of the algorithms**. In recent times, there has been an increasing emphasis on utilizing **surrogate models**, as suggested by the work of Tabatabaei et al. and Kent et al., to effectively represent complex environments with greater simplicity [43, 45].

Keller et al. have proposed **M-QD**, which consists of using a surrogate model after training and updating it [47]. This model **estimates the score of each solution** and calculates the distance associated with the nearest neighbours to filter new solutions. According to the conditions and thresholds for BD and quality respectively, solutions are dropped or added to the container, depending on whether they contribute to the development of diversity or quality [28]. This algorithm is intended to be **more data efficient** thanks to the use of this model and filtering steps, which prevent unnecessary solutions evaluation. One issue is that this model is trained with the genotype of the solutions as input, which **prevents it from being transferable** to another task [47].
To make the model transferable to other tasks, Lim et al. proposed **DA-QD**, which uses a model whose **inputs correspond to variables belonging to the state-action space and not to the genotype space**. More precisely, this algorithm alternates between (1) learning a model through a group of non-deterministic NNs and (2) operating QD to generate a collection of solutions. First, the model is trained based on transitions contained in a replay buffer. Then, this model is used to estimate solutions and fill a container belonging to the "imagination", thus conducting QD in imagination. Finally, the agent acts in the environment to fill the original container, not the imaginary one, with the most promising solutions from the imaginary container. These solutions are chosen **based on the uncertainty** associated with them and quantified by the set of NNs [24]. In the end, the simulation of QD in imagination **improves data efficiency** since evaluation by the model in imagination is less costly than evaluation where the agent acts in the environment [48].
Finally, one of the last approaches is **GDA-QD**, proposed by Lim et al. [49]. This algorithm is an improved version of DA-QD and **introduces a new NN, a critic**, in the non-imaginary part of DA-QD. DA-QD provided good exploration and was more data-efficient than M-QD, but its exploitation was not optimal [48]. The role of this critic is **to improve exploitation** by perturbing a certain proportion of the solutions chosen to act in the real world with policy-gradient (PG) updates. Unlike random and undirected perturbations (GA) in imagination, which promote exploration, PG updates are objective-centred and **improve exploitation** by converging on the most encouraging solutions. The balance between the two categories of updates makes it possible **to get the best of both worlds**, with good exploration and good exploitation at the same time, resulting in an even **better algorithm from a data efficiency point of view** [49].

As a final point, MB strategies provide significant benefits by utilizing models that continuously learn from encountered data. As opposed to sampling approaches, MB strategies retain observed data to enhance the model and its performance during exploration and exploitation. This not only **improves generalization** but also **strengthens the data efficiency** of algorithms.
Consequently, an interesting idea on which this research project is based is to use MB strategies to quantify the uncertainty of the solutions generated and to merge such approaches with QD algorithms. The aim is to improve the capture of distributions and the relationship between inputs and outputs despite the noise, to properly guide the agent and the algorithm, and to improve data efficiency, particularly in uncertain environments.

B.4.2 Enhancing Robustness via Uncertainty Quantification

As a reminder of the section B.1, two types of uncertainty need to be considered: **the epistemic uncertainty** resulting from the model and a limited understanding of the environment, and **the aleatoric uncertainty** emerging from observation data and therefore inherent to the environment [24].
MB approaches can **quantify both uncertainties**. From a general perspective, aleatoric uncertainty is identified through the entropy of probabilistic NNs or the incorporation of a non-deterministic element such as Gaussian noise, while epistemic uncertainty is characterized by the standard deviation associated with model predictions [24, 25, 50, 51].
Thus, an overall method for improving QD algorithms via UQ using MB approaches can be outlined as follows: (1) **specify the environment**, the task and its constraints, as well as the comparison met-

rics, (2) **train a model** sufficiently to approximate the dynamics of the environment, (3) **quantify the uncertainty** of the model predictions, and (4) **guide QD algorithm** for both exploration and exploitation via the uncertainty associated with the solutions.
Regarding the second point, **model selection is variable**. Numerous researchers have looked into this question and proposed various models, including:

1. Probabilistic NNs

    - **Bayesian Neural Networks (BNNs):** They merge Bayesian inference and NNs. They quantify epistemic uncertainty via network parameters, and they calculate aleatoric uncertainty from the distribution of the predictions based on the posterior distribution over the parameters. Then uncertainty is used to sort and select the best solutions [52, 53].
    - **Gaussian Processes (GPs):** These are non-parametric models that adapt their number of parameters according to the data available and become costly as the size of the training data expands. They quantify aleatoric uncertainty via the variability intrinsic to the input data, and epistemic uncertainty through the uncertainty present in the intrinsic function that produces the observation data [12, 24, 45].
    - **Generative Adversarial Policy Network (GAPN):** It has two parts as introduced in section B.4.1, namely a generator and a discriminator. Epistemic uncertainty is derived from the parameters of the generator and the aleatoric uncertainty arises from the inherent randomness of the environment, reflected in the outputs of the generator [44].
    - **Variational Auto-Encoders (VAEs):** Like GAPN, they are generative models with two elements, an encoder that reduces input data into a lower-dimensional latent space, while the decoder reproduces the original inputs from this latent space. The encoder enables the estimation of epistemic uncertainty through parameters of the learned posterior distribution. Conversely, the decoder network allows for the quantification of aleatoric uncertainty by examining the distribution of reconstructed samples it generates [54, 55, 56].

2. Set of models

    - **Implicit:** It involves training only a unique model with a constant number of shared parameters. The main idea is to apply the model several times to a prediction to obtain a set of values. From this set, it is possible to deduce the distributions followed by each solution and more precisely, the standard deviation, to rank them according to the degree of epistemic uncertainty and select the most reliable ones [51].
    - **Explicit:** It requires training multiple models where the number of parameters is linearly proportional to the number of models. The idea is the same as that of implicit ensembles, except that distributions are built on the basis of the predictions of each model in this explicit ensemble, instead of applying a model several times and comparing its own predictions [24, 57].

In conclusion, each model has its own set of advantages and disadvantages, varying in terms of ease of training, implementation simplicity, and suitability for modelling the dynamics of the environment. One of the most common approaches to estimating epistemic uncertainty via model parameters is the Monte Carlo dropout. Finally, users have multiple options in the model selection and construction process, which are influenced by their preferences, the overall problem specification and available computational resources.
As a new approach, it could be interesting to combine MBUQ with sampling methods whose conditions would be based on uncertainty levels. The idea would be, depending on the uncertainty thresholds, to force the model to re-evaluate solutions and ensure certain reliability of estimates.

Chapter C

Design and Contribution

As presented in section B.4.1, the main idea behind an MB version of QD algorithms is to enable the model **to capture the relationship between the inputs and the outputs despite noise**. The second objective, through the use of a trained model, is **to improve the data efficiency** of MBQD algorithms. This research project proposes **8 new algorithms**. As a reminder, the following algorithms are inspired by and use the QDax library, which is available and open-source [58].

The new contributions build on the work of Kent et al., Keller et al. and Lim et al. with initial MBQD approaches [45, 47, 48]. Thus, the model-based approaches proposed through this project seek **to identify and approximate the link between a genotype and the scores** of a solution, so the fitness and BD, coming closer to M-QD than DA-QD [47, 48]. Firstly, the idea is to compare performance by combining these MBQD algorithms with two sampling approaches: (1) **an implicit approach** that consists of accumulating a huge number of solutions to train the model, inspired by the way DGME works [6], and (2) **an explicit approach** which considers a smaller set of solutions where each of them has been re-scored several times and is, therefore, more reliable, as in ME-Sampling [21, 39].
Then, for each of these two algorithms, an additional version is proposed to compare (1) **a deterministic model** which predicts the value of the scores, and therefore of the fitness and BD, and (2) **a probabilistic model** which predicts the parameters of a Gaussian distribution linked to each score, as proposed by Chua et al., thus quantifying the uncertainty of the solutions [24].
Finally, for each of these four new implementations, the possibility of resetting the repertoire is studied, leading to (1) a classic approach with **no repertoire reset**, and (2) an approach where **the repertoire is emptied after each model training** and solutions are re-evaluated by the model. This second version is inspired by the work of Lim et al. with DA-QD and the management of an imaginary archive and another real archive, although the objective is different [48].
To sum up, here are **the 8 new contributions** proposed and studied in this project [47, 48]:

1. **Model-Based MAP-Elites Explicit** (MBME Explicit) [21, 39].
2. **Model-Based MAP-Elites Wipe Explicit** (MBMEW Explicit) [21, 39, 48].
3. **Model-Based MAP-Elites Implicit** (MBME Implicit) [6].
4. **Model-Based MAP-Elites Wipe Implicit** (MBMEW Implicit) [6, 48].
5. **Model-Based MAP-Elites Uncertainty Quantification Explicit** (MBMEUQ Explicit) [21, 23, 24, 39].
6. **Model-Based MAP-Elites Uncertainty Quantification Wipe Explicit** (MBMEUQW Explicit) [21, 23, 24, 39, 48].
7. **Model-Based MAP-Elites Uncertainty Quantification Implicit** (MBMEUQ Implicit) [6, 23, 24].
8. **Model-Based MAP-Elites Uncertainty Quantification Wipe Implicit** (MBMEUQW Implicit) [6, 23, 24, 48].

Wipe (W) indicates that the algorithm uses the repertoire reset and Uncertainty Quantification (UQ) indicates that the algorithm uses the probabilistic model for its predictions.

Although there are variations in these algorithms, they are all based on the same principle of **training a model during simulation, and using it instead of the intrinsic scoring function of the environment to evaluate solutions** and potentially add them to the repertoire.
This MBME process is illustrated in figure 3 and described in algorithm 2.

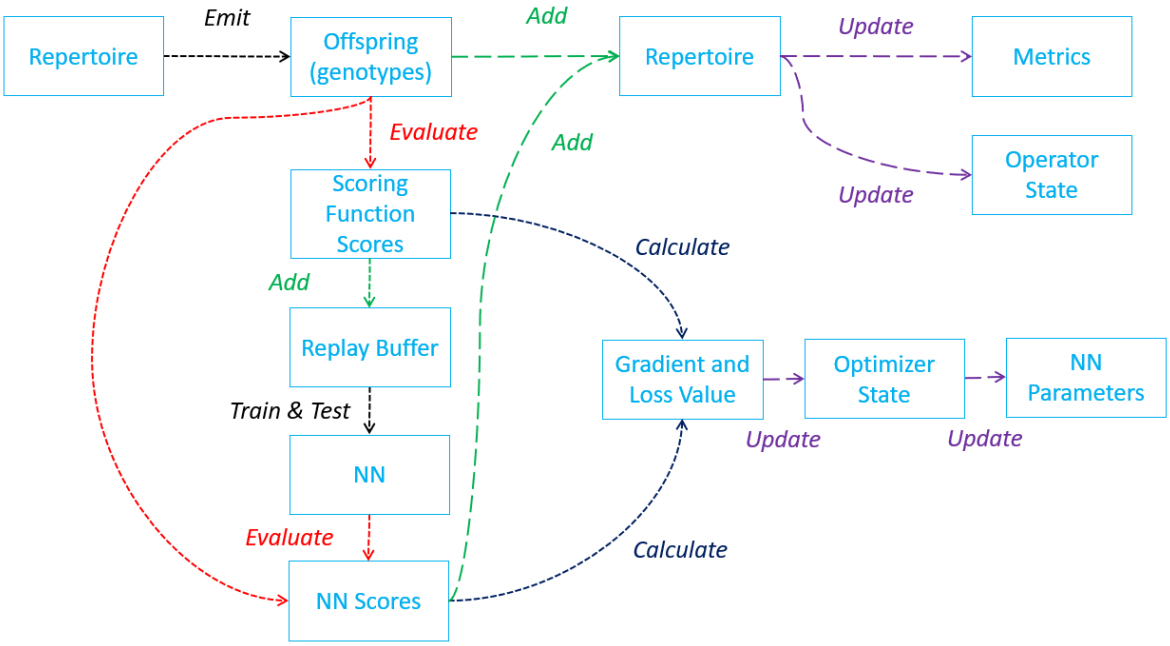


Figure 3: Diagram depicting the main loop of MBME algorithm after initialization. The idea is to alternate between (1) training a model from a batch of genotypes and their associated scores stored in a buffer $\mathscr{D}$, and (2) performing MAP-Elites from the solutions predicted by the model and filling the container accordingly. The solutions stored in the buffer $\mathscr{D}$ come from the evaluation of the genotypes with the scoring function of the environment.

The addition mechanism in the container for MBME remains the same as for ME described in algorithm 1 and figure 2 [28]. There is **a first condition on the occupation of the BD** and **a second one on the fitness value** of the new solution compared to the old one already stored in the container.
Moreover, **a first-in-first-out** (FIFO) **buffer** $\mathscr{D}$ fed by scoring function evaluations constitutes **a training and test dataset**, $\mathscr{D}_{train}$ and $\mathscr{D}_{test}$, for the model. At the beginning, the model is not trained and the buffer has not accumulated any data, so the container is filled with scores obtained via the scoring function and not the model. Following the first training of the model, **the scoring function is used only to feed the buffer, while the model is used to fill the container**.
Finally, the model is trained with a certain update frequency after the first training generation to give the buffer $\mathscr{D}$ time **to accumulate new data**.

The use or non-use of the repertoire reset has no effect on either model predictions or training data. On the other hand, **the sampling approach and model type do have an impact on model training**, as shown in table C.0.1.
Indeed, since all algorithms are compared with the same sampling size, defined by equation B.7 and shown in table D.0.1, reevaluations in the case of the explicit algorithms have two consequences: (1) **the genotypes batch is 8 times smaller** than in the implicit case, and (2) the scoring function gives **access to uncertainty** through the standard deviations of fitness and BD [23].
In addition, the type of model used modifies the predictions associated with the evaluation of a solution, enabling uncertainty to be quantified or not.

Design \ MBME Algorithms	Explicit	UQ Explicit	Implicit	UQ Implicit
Scoring Function Version	sampling		classical	
Number of Evaluations	8		1	
Scoring Function Predictions	$\overline{fitness}$, σ_{fit} $\overline{BD_x}$, σ_{BD_x} $\overline{BD_y}$, σ_{BD_y}		fitness BD_x BD_y	
Model Predictions	fitness BD_x BD_y	$\overline{fit}$, $\log(\sigma_{fit})$ $\overline{BD_x}$, $\log(\sigma_{BD_x})$ $\overline{BD_y}$, $\log(\sigma_{BD_y})$	fitness BD_x BD_y	$\overline{fit}$, $\log(\sigma_{fit})$ $\overline{BD_x}$, $\log(\sigma_{BD_x})$ $\overline{BD_y}$, $\log(\sigma_{BD_y})$

Table C.0.1: Design of the various MBME algorithms, where $\overline{k}$ represents the mean of element k and σ_k the standard deviation of the same element. The version of the **algorithms with container reset** retains exactly **the same characteristics as without reset**.

Now that the algorithms and the ideas behind their design have been presented, it's important to look at the uncertainty quantification introduced in section C.1, the hyperparameters used and presented in section C.2, and the performance assessment developed in section C.3.

C.1 Design of the Neural Network and Quantification of the Uncertainty

MBME and MBMEUQ algorithms **rely on the use of a model** to approximate the dynamics of the environment and evaluate solutions for potential addition to the container. Here, the model is represented by a NN and, more precisely, by an MLP. The task of the NN is straightforward since it involves **a linear regression** with a batch of genotypes as input, to predict the score of solutions. According to table C.0.1, the models offer different outputs depending on whether they quantify uncertainty or not. These two types of models, **deterministic or probabilistic**, are displayed in figure 4.

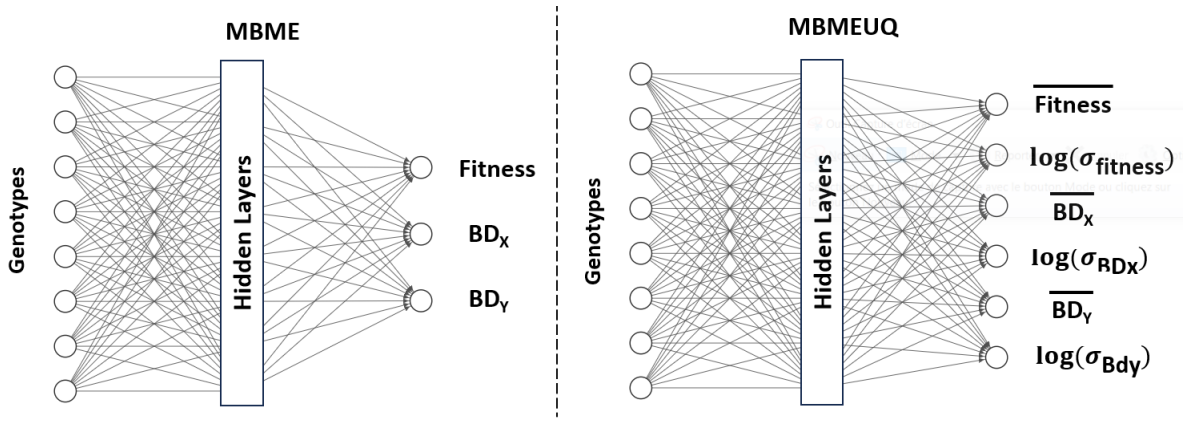


Figure 4: Representations of the NNs used for the **MBME** (left) and **MBMEUQ** (right) algorithms, respectively **deterministic** and **probabilistic**. Models have (1) **an input size of 8** corresponding to the DoFs, (2) **fully connected hidden layers of various sizes**, and (3) an output of **size 3 or 6 respectively for MBME and MBMEUQ algorithms**.

These representations come from the tool implemented by LeNail to visualize NNs in several formats [59]. The size of the input element corresponds to the DoFs indicated in table D.0.1, and the output element corresponds to the solution's scores. These scores are of dimension 3 if the model is **deterministic and only interested in solution values** (MBME), or of size 6 if the model is **probabilistic, quantifies uncertainty and seeks to identify the parameters of each Gaussian distribution** associated with a solution (MBMEUQ) as described in table C.0.1.

Since the type of prediction varies according to the choice of algorithm, the model does not use the same loss for training. For MBME algorithms, the model is trained using **Mean Squared Error** (MSE), while MBMEUQ algorithms use **Negative Log-Likelihood** (NLL) for training. These losses are detailed in equations C.1 to C.4, and the objective for the model is **to minimize them**.
Also, to limit overfitting and improve model training, an **L2 regularization term** is added to the previous losses to form the **Training Loss** shown in equation C.5.

$$\mathscr{L}_{\text{MSE}}^{\text{Score}}(x,y) = \frac{1}{|\text{B}|}\sum_{i=1}^{|\text{B}|}\left(\text{x}_i^{Score} - \text{y}_i^{Score}\right)^2 \tag{C.1}$$

$$\mathscr{L}_{\text{MSE}}^{\text{Total}} = \mathscr{L}_{\text{MSE}}^{\text{Fitness}} + \mathscr{L}_{\text{MSE}}^{\text{BD}_x} + \mathscr{L}_{\text{MSE}}^{\text{BD}_y} \tag{C.2}$$

$$\mathscr{L}_{\text{NLL}}^{\text{Score}}(x,y) = \frac{1}{2}\sum_{i=1}^{|\text{B}|}\left[\log(2\pi[e^{\log\sigma(x_i^{Score})}]^2) + \left(\frac{y_i^{Score} - \mu(x_i^{Score})}{e^{\log\sigma(x_i^{Score})}}\right)^2\right] \tag{C.3}$$

$$\mathscr{L}_{\text{NLL}}^{\text{Total}} = \mathscr{L}_{\text{NLL}}^{\text{Fitness}} + \mathscr{L}_{\text{NLL}}^{\text{BD}_x} + \mathscr{L}_{\text{NLL}}^{\text{BD}_y} \tag{C.4}$$

$$\mathscr{L}_{\text{Training}} = \begin{cases} \mathscr{L}_{\text{MSE}}^{\text{Total}} + \alpha\sum_{n=1}^{K}\beta_n^2 & \text{if MBME Algorithm} \\ \mathscr{L}_{\text{NLL}}^{\text{Total}} + \alpha\sum_{n=1}^{K}\beta_n^2 & \text{if MBMEUQ Algorithm} \end{cases} \tag{C.5}$$

B : Batch of individuals considered for the training
α : Regularization weight, indicated as L2 Rate
β_n : Value of the n-th model parameter
K : Number of model parameters
x_i^{Score} : Model prediction for the i-th individual and corresponding score
y_i^{Score} : Environment evaluation for the i-th individual and corresponding score
$\log\sigma(x_i^{Score})$: Model log std prediction for the i-th individual and corresponding score
$\mu(x_i^{Score})$: Model mean prediction for the i-th individual and corresponding score

Adding the exponential so that the model predicts the logarithm of the uncertainty instead of the uncertainty limits numerical instabilities. As a result, **the model trains better**.

The different algorithms allow a comparison of approaches that do or do not capture the uncertainty associated with the solutions, notably through the use of NLL to generate the standard deviation of the distribution of each solution. This quantified uncertainty does not present a clear separation between the aleatoric and epistemic aspects. **This uncertainty is a mix of these two types of uncertainty**, the proportions of which we don't explicitly know.
It's also important to note that **the aleatoric uncertainty is not the same between explicit sampling and implicit sampling** since the latter considers the direct distribution of individuals since there are

no re-evaluations, whereas the former proposes a new distribution obtained through re-evaluations and the average of several individuals.

In conclusion, **the algorithms do not explicitly take advantage of this uncertainty**. It is quantified, gives the user an idea of the model's performance, and **the model implicitly takes advantage of it** when training and adjusting its parameters. However, apart from this implicit aspect and the metrics, uncertainty is not explicitly re-used in the algorithm to find new or more promising solutions.

C.2 Hyperparameters Selection

The training of a model and therefore the identification and selection of its hyperparameters are key elements in (1) **proposing appropriate training** and (2) **ensuring correct and reliable predictions**, whose error in comparison with the environment and the scoring function is low.
However, there are many hyperparameters and **adjusting each of them is no easy task**, especially when the algorithms propose different approaches, whether via training data or the training itself, notably with the loss used as described in section C.1. Moreover, the algorithms have to contend with two environments: the deterministic arm and the uncertain arm. When searching for hyperparameters, **a trade-off was sought to maximize results in both environments**. On the other hand, if no trade-off was possible, **the uncertain environment was favoured** since this project is mainly concerned with the use of MBQD approaches for non-deterministic environments.

The proposed contributions are designed to use the scoring function to feed both the buffer and the container for a certain number of generations, before using only the model to fill the container after an initial training. Such a design requires the consideration of new hyperparameters used and represented in equations C.6 and C.7.

$$\text{Model Training Frequency: } \text{MTF} = \frac{\text{FTG}}{2} \tag{C.6}$$

$$\text{Buffer Size: } \mathscr{D}_{\text{Size}} = \frac{1}{1-\text{PODR}} \times \text{MTF} \times \text{BS} \tag{C.7}$$

$$\begin{aligned} \text{FTG} &: \text{First Training Generation} \\ \text{PODR} &: \text{Percentage of Old Data Retained} \\ \text{BS} &: \text{Batch Size} \end{aligned}$$

FTG represents **the generation at which the model trains for the first time** and the scoring function ceases to be used to feed the container. MTF indicates **the frequency, in terms of generations, at which the model trains** throughout the entire simulation. Finally, PODR allows **a percentage of the old data stored in the buffer to be retained for the next time the model is trained**.

When designing an algorithm, the question arises as to how many times solutions should be evaluated by the environment, **to ensure a fair comparison between algorithms**. Because of reevaluations, this is where sampling size comes into play [23]. However, these new MBQD algorithms no longer use the environment to fill the container. The advantage in addition to the robustness of predictions against noise is to improve data efficiency by proposing solutions predicted by the model and not by the environment scoring function. This difference may not be obvious for simulated environments such as the robotic arm, but it becomes relevant when **the environment is real and the agent has to interact physically** and move around in the environment [1, 60].

Therefore, when evaluations are done by the model and not by the environment, it was decided **to apply a batch size comparison** rather than a sampling size comparison. This approach is justified by the fact that it seems **fairer**, especially when comparing implicit and explicit approaches to sampling. The model predicts several solutions corresponding to the initial design of the algorithm, through the presence or absence of sampling for solutions.
Establishing a sampling size comparison for the model would force explicit algorithms to predict 8 times as many solutions as they currently do, to match the number predicted by implicit algorithms. This would **increase the cost in time and computational resources**, and above all, it would **break down the idea of equitable comparison**. Without this fair comparison, explicit algorithms would offer as many solutions as implicit ones while benefiting from training with theoretically more reliable solutions, particularly in uncertain cases thanks to reevaluations.

Although the model uses inputs proportional to batch size and not sampling size, to ensure a fair comparison between the explicit and implicit approaches, it is **important to consider the comparison between using the model and using the scoring function**. Indeed, the model is by definition more data-efficient than the scoring function of the environment. Thus, an evaluation using the model **can be considered "free" and is not equivalent to an evaluation by the scoring function**. This notion of equivalence and fairness between the model and the scoring function requires the consideration of a new hyperparameter, namely **the number of subdivisions**.
In this project, several arbitrarily chosen values were tested to compare the performance of the different algorithms. A more appropriate approach would be to measure how much time and computational resources the scoring function requires to evaluate a batch of solutions, measure these same quantities for the model and deduce proportionally the maximum number of batches a model can use to equal the resources consumed by the scoring function. Finally, this hyperparameter represents **the maximum number of solution batches that the model can evaluate, to approximate the resources used by the scoring function to evaluate one batch of solutions**.

In addition to the hyperparameters specific to these contributions, other more general hyperparameters have also been studied to improve training. All the hyperparameters and the values used by each of the 8 algorithms are presented in table C.2.1.

MBME Algorithms / Hyperparameters	Explicit	Implicit	UQ Explicit	UQ Implicit
Loss	MSE		NLL	
Batch Size (BS)	64	512	64	512
Inputs Dimensions	(BS,8)			
Outputs Dimensions	(BS,3)		(BS,6)	
Fully Connected Hidden Layers Size	(64,32,16)	(32,16,8)	(64,32,16)	(16,16,16)
Optimizer	Adam			
Learning Rate	$7e-3$	$1e-4$	$7e-3$	$5e-3$
L2 Rate	$1e-6$			
Number of Epochs	125		175	225
First Training Generation (FTG)	1000	500	1000	500
Percentage of Old Data Retained (PODR)	20%	80%	20%	40%
Number of subdivisions	30	5	30	5
	30	5	100	100

Table C.2.1: Set of hyperparameters considered for the model of each MBME algorithm. Algorithms with repertoire reset have the same hyperparameter values as their version without repertoire reset, except for the number of subdivisions. Shaded cells correspond to hyperparameters of algorithms with repertoire reset.

The first 4 hyperparameters defined in table C.2.1 are imposed either by (1) **the hyperparameters of the QD algorithms** described in table D.0.1 and derived from the environment and configuration presented in chapter D or by (2) **the design of the model quantifying or not the uncertainty** described in table C.0.1 and section C.1. The rest of the hyperparameters have been determined through the state-of-the-art and several comparisons of simulations in an uncertain environment.
Right from the start, **the Adam optimizer** is used. Even if other optimizers exist, it is one of those that works best throughout the literature, as well as several personal projects. On top of that, it offers an adaptive individual learning rate for each model parameter and optimizes each of them [61].

It's not easy to combine all these hyperparameters and find the best possible combination, all the more so when we consider the search for 8 different algorithms and each hyperparameter has a fairly large set of values. Therefore, multiple orders of magnitude were **estimated from a few test simulations or from the literature to restrict these sets of values** and explore fewer combinations.
These initial estimates seemed interesting, except that in reality **the model was underfitting**, not allowing for sufficiently interesting learning. The main cause of this underfitting was **the complexity of the model** via the number of hidden layers and neurons associated with each of them. Consequently, an architecture search was carried out across a set of 15 possibilities in which the model took **2 to 3 hidden layers each consisting of 8 to 64 neurons**.
Once the model no longer appeared to be underfitting and the architecture was fixed, **two GridSearches** were performed to estimate the best possible combination of hyperparameters. First, a GridSearch to estimate **the number of epochs, FTG and PODR**, which also impact the other two hyperparameters presented in equations C.6 and C.7, **MTF and the buffer size** respectively. Once these hyperparameters had been established, a second GridSearch was performed to set **the learning rate and the L2 rate**. The values studied for each hyperparameter are described in table C.2.2.

Model Training Hyperparameters	Range values
Number of Epochs	[75, 125, 175, 225]
FTG	[200, 500, 800, 1000]
PODR	[20%, 40%, 60%, 80%]
Learning Rate	[2e-4, 7e-4, 2e-3, 7e-3, 2e-2]
L2 Rate	[1e-6, 5e-6, 1e-5, 5e-5, 1e-4, 5e-4, 1e-3]

Table C.2.2: Values of the multiple hyperparameters considered for the two GridSearches

There were 64 simulations per algorithm for the first GridSearch and 35 for the second. To compare performances, only **the 10 best simulations** were retained for each GridSearch. To do so, the last value of the corrected QD score presented in equation B.6 and from an uncertain environment, was used to evaluate the actual performance of the algorithms, as presented in section B.3.1 [6, 22, 23].
The objective is to **maximize such a metric**, thus retaining only **the 10 simulations with the highest corrected QD score**. Additional simulations have identified hyperparameter values not necessarily belonging to the combinations defined in table C.2.2, whose final values are specified in table C.2.1.

In parallel with this search for hyperparameters, a data pre-processing method was tested, namely **Z-score standardization**. This method is designed to be **more robust to potential outliers than min-max normalization**, and above all, it **enables different features to be brought together on the same scale**. Here, 3 features are considered for an MBME algorithm and 6 for an MBMEUQ algorithm, corresponding either to fitness and BD values or to the parameters of the Gaussian distribution associated with the latter.

The idea is **to normalize the data stored in the buffer**, i.e. the scores assigned to the solutions by the scoring function, and to retain the μ and σ parameters of each score to denormalize the model's predictions after being trained. Theoretically, this method aims **to facilitate and improve model learning** by positioning features on the same scale and with a similar distribution: $\mathcal{N}(0, 1)$.
This method is presented in equation C.8.

$$\text{Z-Score Standardization: } y^{\text{Score}}_{\text{Standardized, i}} = \frac{y^{\text{Score}}_{\text{i}} - \mu^{\text{Score}}_{\mathscr{D}}}{\sigma^{Score}_{\mathscr{D}}} \tag{C.8}$$

$y^{\text{Score}}_{\text{i}}$: Environment evaluation for the i-th individual and corresponding score
$\mu^{\text{Score}}_{\mathscr{D}}$: Mean of the score obtained via buffer $\mathscr{D}$ before splitting it for model training
$\sigma^{\text{Score}}_{\mathscr{D}}$: Std of the score obtained via buffer $\mathscr{D}$ before splitting it for model training

Unfortunately, this method **did not produce convincing results**. Unlike the other mechanisms, i.e. **reset, uncertainty quantification and sampling**, this data pre-processing is not a central element of the project in terms of results. As a consequence of these results and **to favour a more in-depth analysis of the 3 other mechanisms**, this standardization proposal has been **discarded**.
However, it's important to bear in mind that other normalization or pre-processing methods are available and may be worth exploring.

To conclude, it is possible to **extend the resource allocation formalism** introduced by Flageat et al. by distinguishing between the world of the environment and the world of the model when making predictions or evaluations [23]. On the one hand, the **global resources** allocated to the algorithms follow **a sampling size comparison** to make the comparison between algorithms entirely equitable. On the other hand, **when the model is used** in each algorithm to predict new solutions, it follows a **batch size comparison** to make comparisons between different sampling approaches fair.
Finally, it is important to establish the new hyperparameter corresponding to **the number of subdivisions**, to take maximum advantage of **the model's performance and low cost** at each prediction.

C.3 Evaluation of the Performances

Now that the context is set for the new algorithms as well as each model and its training, let's look at the measurement of the different performances.
On the one hand, **the performances of the algorithm** to compare different approaches, and on the other, **the performances of the model** to ensure that its predictions and training are appropriate to the uncertain environment and the expected solutions.

The metrics used are those introduced in section B.2.1 and used by Berthier et al.: **QD score, max fitness and coverage** [1]. Both **the uncorrected and corrected versions** of each of these metrics will be available to measure the true performance of the different algorithms [6, 22, 23].
Moreover, **two new metrics** are introduced for MBMEUQ algorithms via equations C.9 and C.10.
In the same way as the QD score, these new metrics take into account **performance** via the uncertainty value, while also taking into account **coverage** via the sum of cells occupied by a solution [2, 3].
However, unlike the QD score, the aim is **to minimize these new metrics**.

1. **Uncertainty Fitness Score** (UFS):

$$\mathrm{UFS}(C) = \sum_{n=1}^{N_C} \sigma_n^f \tag{C.9}$$

2. **Uncertainty BD Score** (UBS):

$$\mathrm{UBS}(C) = \sum_{n=1}^{N_C} \sigma_n^{BD} \text{ with } \sigma_n^{BD} = \frac{\sigma_n^{BD_x} + \sigma_n^{BD_y}}{2} \tag{C.10}$$

N_C : Number of cells occupied by an individual in container C
σ_n^f : Fitness uncertainty (std) of the n-th solution occupying a cell in container C
$\sigma_n^{BD_x}$: X position BD uncertainty (std) of the n-th solution occupying a cell in container C
$\sigma_n^{BD_y}$: Y position BD uncertainty (std) of the n-th solution occupying a cell in container C
σ_n^{BD} : Mean BD uncertainty (std) of the n-th solution occupying a cell in container C

In addition to the corrected metrics, **the QD score loss metric** introduced in section B.3.1 and equation B.8 is studied. Indeed, QD score is considered to be **the main metric** in this research project for defining whether one algorithm is better than another.

Regarding model evaluation, it is pertinent to look at two different cases to understand **the training quality of the model and its effectiveness**, namely (1) **the last testing** where the model is theoretically the most effective in its predictions, and (2) **all the generations** where the model is used without being re-trained.
Through this report, the analysis **focuses mainly on the second case with all generations** rather than the first one with the last model test at the last generation. Indeed, the set of generations where the model is used without being re-trained is **much more representative** of the model's behaviour during simulation.

In this way, multiple **ancillary studies** have been implemented **to confirm that the model is correctly trained and that its predictions are consistent** both (1) during generations dedicated to training the model and (2) between generations of training. The studies are the following:

1. **A loss study** during training and testing to ensure that the model is neither underfitting nor overfitting.
2. **A study of the error for fitness and BD values** between model predictions and scoring function evaluations associated with the environment.
3. **A statistical study** of fitness and BD prediction errors to estimate the number of outliers.
4. **Another study of error for uncertainty quantification for fitness and BD** between model predictions and scoring function evaluations. This study concerns only MBMEUQ Explicit and MBMEUQW Explicit algorithms accessing uncertainty through standard deviations both via the scoring function and the model.
5. **A study of the comparison of predictions** to check their distribution and better understand the errors identified previously.
6. **Another comparison study for the prediction of uncertainty** associated with fitness and BD. Like the previous comparison, it is used to understand the errors obtained previously. However, this comparison is only possible for the MBMEUQ Explicit and MBMEUQW Explicit algorithms.

The study of errors and **the comparison of uncertainties can not be carried out for algorithms using the implicit sampling approach**. Indeed, uncertainties are not provided by the scoring function when it evaluates a solution, as reminded by table C.0.1

For predictions outside the generations, **the average error is calculated** to have only one point per generation and make the graph more readable, while giving a general idea of the performance of the model during the simulation. The calculation of the error is described in equation C.11.

$$\text{Average Absolute Error: AAE}_{\text{Score}}(x,y) = \frac{1}{|B|}\sum_{i=1}^{|B|} |x_i^{Score} - y_i^{Score}| \tag{C.11}$$

B : Batch size used by the model to generate solutions at each generation

x_i^{Score} : Model prediction for the i-th individual and corresponding score

y_i^{Score} : Environment evaluation for the i-th individual and corresponding score

On the one hand, this makes it possible **to check whether the model is correct**, thanks to its training and error curves, and on the other hand it also makes it possible **to identify where the model is lacking**, both with regard to training periods and predictions, thanks to comparison figures.
To conclude, this is **a double check, confirming or questioning the performances obtained with each algorithm**. The algorithms are compared via the environment and task introduced in chapter D, and the results and their analysis are proposed in chapter E.

Algorithm 2 Model-Based MAP-Elites Algorithm

Initialize randomly a container C
Initialize parameters of the neural network NN
Initialize the state of the optimizer Opt
Initialize an empty buffer $\mathscr{D}$
for Generation $= 1$ until Max Generation G_{max} **do**
  Select a population P (parents) from the container C using a selection operator
  Generate new individuals I (offspring) from the population P using an alteration operator
  if Generation $\leq$ First Training Generation **then**
    for each individual i in I **do**
      Evaluate the scores $S_{\text{environment}}(i)$ of individual i via **the scoring function**
      Add individual i and $S_{\text{environment}}(i)$ to the container C
      if Size of $\mathscr{D}$ < Max Size $\mathscr{D}_{max}$ **then**
        Add individual i and $S_{\text{environment}}(i)$ at the end of the buffer $\mathscr{D}$
      else
        Remove first individual from the buffer $\mathscr{D}$
        Add individual i and $S_{\text{environment}}(i)$ at the end of the buffer $\mathscr{D}$
  else
    for each individual i in I **do**
      Predict the scores $S_{\text{model}}(i)$ of individual i via **the neural network** NN
      Add individual i and score $S_{\text{model}}(i)$ to the container C
      Evaluate the scores $S_{\text{environment}}(i)$ of individual i via **the scoring function**
      if Size of $\mathscr{D}$ < Max Size $\mathscr{D}_{max}$ **then**
        Add individual i and score $S_{\text{environment}}(i)$ at the end of the buffer $\mathscr{D}$
      else
        Remove first individual from the buffer $\mathscr{D}$
        Add individual i and $S_{\text{environment}}(i)$ at the end of the buffer $\mathscr{D}$
  if Generation == Training Generation **or** Generation == Last Generation **then**
    Split the buffer $\mathscr{D}$ into a training dataset $\mathscr{D}_{train}$ and a test dataset $\mathscr{D}_{test}$
    for Epoch $= 1$ until Max Epoch E_{max} **do**
      for Batch $= 1$ until Max Batch B_{max} from $\mathscr{D}_{train}$ **do**
        Calculate the value and gradient of the training loss $\mathscr{L}_{train}$, $\nabla\mathscr{L}_{train}$
        Update the state of the optimizer Opt based on $\nabla\mathscr{L}_{train}$
        Update parameters of the neural network NN based on the new state of the optimizer Opt
      Calculate the average training loss of all batches $\overline{\mathscr{L}_{train}}$
    for Batch $= 1$ until Max Batch B_{max} from $\mathscr{D}_{test}$ **do**
      Calculate the test loss $\mathscr{L}_{test}$
    Calculate the average test loss of all batches $\overline{\mathscr{L}_{test}}$
return C, NN, $\overline{\mathscr{L}_{train}}$, $\overline{\mathscr{L}_{test}}$,

Chapter D

Experimental Setup

In this research project, the environment studied corresponds to the simulation of a **robotic arm** made up of N joints, and the objective is to reach multiple positions on a 2D plane by controlling the angle of each joint. To consider the impact of noise on the algorithms and their performance, two tasks will be used to compare the different approaches proposed in this project and in the literature: (1) **a deterministic** robotic arm and (2) **an uncertain** version of the same arm.
However, it is important to remind that **the main task corresponds to the uncertain arm** as this project is mainly concerned with uncertainty quantification and non-deterministic environments. Hence the design of the algorithms and a more in-depth study of the results for the case of the uncertain environment as specified in chapters C and E respectively.

For the rest of the study, the arm configuration consists of **8 joints**. This choice simplifies the complexity of the task, thereby avoiding prolonged time simulations and the necessity of significant computational resources. Additionally, this environment has been previously used in the literature, allowing for a more comprehensive comparison of the performances of multiple algorithms [1, 6, 23, 62].
It is also important to remind that this project relies primarily on the use of the library developed by Lim et al., namely **QDax** [58]. This library, written in JAX, considerably reduces simulation times for QD algorithms.

The graphical representation of the redundant arm task is shown in figure 5.

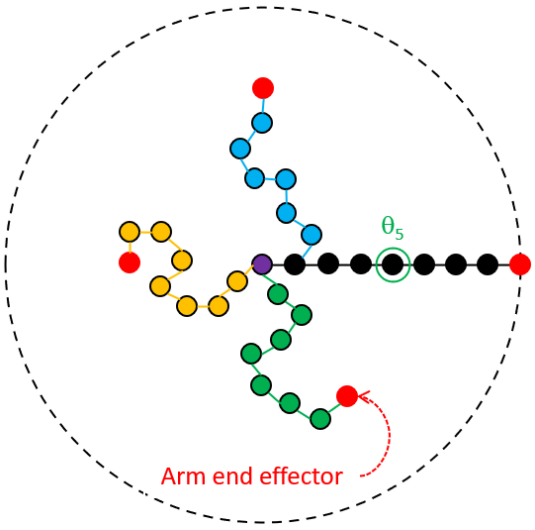


Figure 5: Diagram showing the robotic arm with its 8 joints in black and end effector in red.

The genotype used corresponds to **the angular position of each joint**, originally [0;2π] then normalized between [0;1]. Similarly, the BD, which represents **the final 2D position of the arm end effector** after the last joint, is normalized between [0;1]. Finally, fitness is expressed as **the negative variance of all 8 joints**, and the aim is to obtain zero fitness, not the highest when a solution is evaluated. For the uncertain version of this task, noise is artificially added to both the fitness and the BD once a solution has been evaluated.

The mathematical expressions of these different concepts are introduced through the D.1 to D.4 equations.

$$\text{Genotype: } \vec{\theta} = (\theta_n)_{1 \leq n \leq 8} \tag{D.1}$$

$$\text{Fitness: } f(\vec{\theta}) = -\mathbb{V}(\vec{\theta}) = -\frac{1}{8}\sum_{n=1}^{8}(\theta_n - \bar{\theta})^2 \tag{D.2}$$

$$\text{Noise: } \varepsilon_{\text{fitness}} \sim \mathcal{N}(0, 0.01) \text{ and } \varepsilon_{\text{BD}} \sim \mathcal{N}(0, 0.05) \tag{D.3}$$

$$\text{Uncertain fitness: } f_{\text{uncertain}}(\vec{\theta}) = f(\vec{\theta}) + \varepsilon_{\text{fitness}} \tag{D.4}$$

θ_n : Angle of the n-th joint
$\bar{\theta}$: Average of the 8 angles

To understand the usefulness and potential, and to compare the 8 contributions proposed through this project, it is also important to define **reference baselines**. To this end, **2 benchmark algorithms** are used [1]:

1. **MAP-Elites** (ME), which is based on the work of Mouret et al. [28].
2. **MAP-Elites-Sampling** (MES), which is based on the work of Justesen et al. and Cantu-Paz [21, 39].

To enable **a fair comparison** between the new algorithms and the results obtained by Berthier et al. via the benchmarks algorithms, **the same hyperparameters related to this environment and QD algorithms** are presented in table D.0.1 and used [1].

Tasks / QD Hyperparameters	Deterministic Arm	Uncertain Arm
Number of Generations	3000	
Type of the Grid	Cartesian	
Size of the Grid	100x100	
Number of joints (DoFs)	8	
Sampling size	512	
Frequency of Update (Corrected Metrics)	Every 10 Generations	
Number of reevaluations (Corrected Metrics)	256	
Fitness Noise	None	$\mathcal{N}(0, 0.01)$
BD Noise	None	$\mathcal{N}(0, 0.05)$

Table D.0.1: Set of hyperparameters considered for the QD algorithms respectively for the deterministic task and the uncertain task.

Now that the task has been presented, the proposed algorithms detailed and the various points of comparison explained, let's look at and analyze the results obtained in chapter E.

Chapter E

Experimental Results

On top of the QDax library, this project uses **Singularity** introduced by Kurtzer et al. [63]. Singularity makes it possible to create containers containing environments with packages, versions and all the other tools needed for the code to function properly, thus **making the project reproducible**.
The complete **source code** is available **here**.

The results are analyzed in two parts: (1) **a comparison of the best contribution with the two baselines** in section E.1, with an **in-depth analysis of the model** used and **the comparison with the ground truth results of the deterministic environment**, and (2) **an ablation study** in section E.2 to study the impact of the different mechanisms proposed across the contributions.
Berthier et al. have re-demonstrated the importance of corrected metrics, to evaluate the "true" behaviour of algorithms and their true performance, particularly for uncertain environments [1]. As this project is mainly concerned with the uncertain task of the robotic arm, **uncorrected metrics will not be studied**, although they are presented in appendix B. Nonetheless, uncorrected repertoires will be displayed to visualize the difference in the behaviour of MBQD approaches, without going into detail.

In addition, since the project is mainly concerned with the uncertain environment and the contributions have been designed for the latter, it is mainly **the associated metrics and repertoires that will be analyzed**. Unless otherwise stated in the analysis, the case of the deterministic task is not analyzed in detail, but rather **gives an idea of the behaviour of these MBQD approaches in a deterministic environment**. Indeed the proposed algorithms do not seek to address this issue, as it is **not the main objective of this research project**.

E.1 Best-Case Contribution vs. Benchmarks Algorithms

Among the 8 contributions, the **Model-Based MAP-Elites Uncertainty Quantification Wipe Explicit** (MBMEUQW Explicit) algorithm is considered **the best**. Indeed, this algorithm offers **the highest QD score for the uncertain environment**.

E.1.1 Overall Comparison

The results are displayed through the corrected metrics in figure 6 and the corrected repertoires in figure 7. Also, the two new metrics **UFS** and **UBS** introduced in section C.3 are presented in figure 8. However, **ME cannot be used as a baseline for these new metrics**, as the algorithm does not quantify uncertainty, unlike MES, which uses revaluations.
A summary of the different results is available in table E.1.1 and the calculation of the improvement between two results given a specific metric is described in equation E.1:

$$\text{Improvement: Imp} = \frac{\text{New Value} - \text{Reference Value}}{\text{Reference Value}} \tag{E.1}$$

Figures 6b and 7b show that the MBMEUQW Explicit algorithm is **better than the ME baseline but not as good as the MES baseline**. Table E.1.1 quantifies the extent to which the model stands out from the benchmark algorithms.

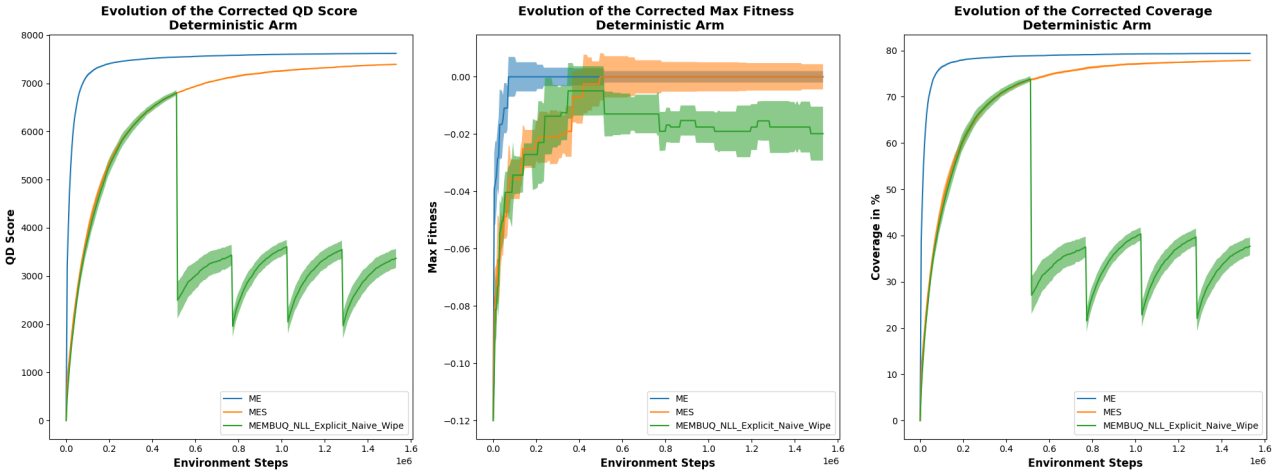
Metrics \ Algorithms	ME	MES	MBMEUQW Explicit
QD Score	4047.13	4798.36	4472.36
		+18.56%	+10.51%
	−15.66%		−6.79%
Max fitness	0.0017	0.0007	−0.0135
Coverage	42.15	50.54	48.94
		+19.91%	+16.11%
	−16.60%		−3.17%
UFS		50.35	48.79
UBS		251.90	243.95

Table E.1.1: Comparison of the different metrics between ME, MES and MEMBUQW Explicit with **the uncertain arm** task. The white cells correspond to **the final value** (first line), the slightly shaded cells correspond to **the improvement over ME** (second line) and the darker cells correspond to **the improvement over MES** (3rd line). Black cells correspond to **undefined results**.

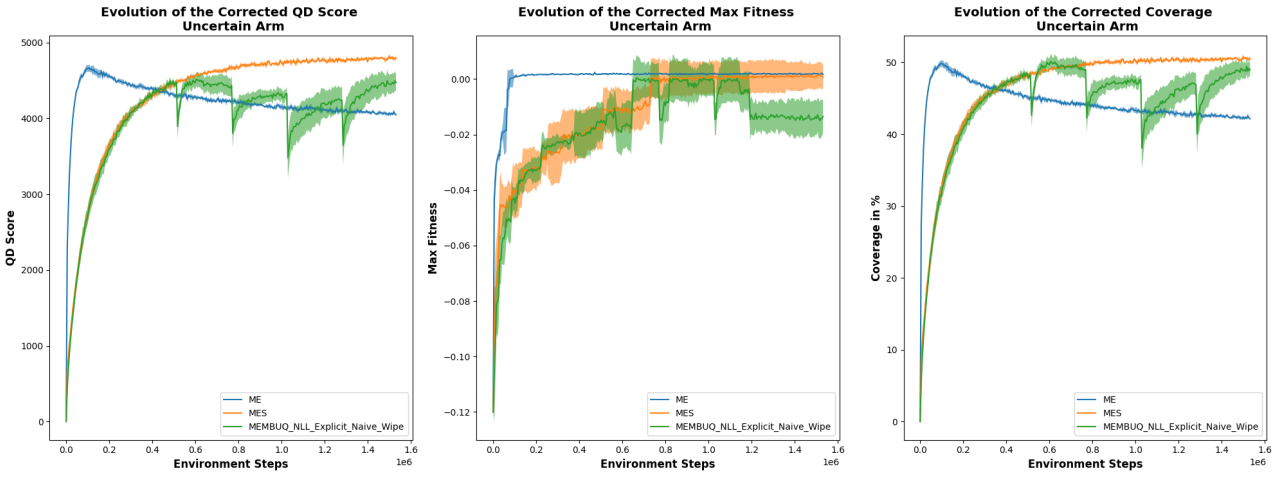
Whether concerning ME or MES, the results indicate that **the difference in QD score is mainly explained by the quality of the solutions** through the estimation of the set of fitnesses rather than by the number of solutions through the set of BDs and therefore the coverage. Indeed, MBMEUQW Explicit's coverage is close to MES's, yet the gap between the QD scores is greater. It demonstrates that although MBMEUQW Explicit's model proposes several solutions added to the repertoire close to MES's, **the latter proposes a lower overall fitness estimate for the set of solutions identified than MES**. Similarly, MBMEUQW Explicit's coverage is much higher than ME's, but the gap between the QD scores is not as high, showing that although the model adds many more solutions to the repertoire, **the quality of these solutions is not necessarily as good as those proposed by ME**. The repertoires in figure 7b clearly illustrate this difference, particularly with ME and MES, where **the golden heart shape is visually identifiable, unlike the MBMEUQW Explicit repertoire**, which is much less marked visually by this shape, but more by its space-covering aspect.
The problem of estimating fitness can also be seen in the max fitness metric. Even though the latter only considers the maximum value and therefore only represents a single point in a set, we note that MBMEUQW Explicit has a maximum value that is very different from ME and MES. This suggests that **the problem of estimating fitness applies not to a single solution but to a large population of solutions in the repertoire**, confirming the previous analysis of the QD score and the repertoire obtained.

Concerning the new metrics, UFS and UBS, it's important to remember that **we're trying to minimize them**. If one looks at the table E.1.1, it can be seen that MBMEUQW Explicit offers lower metrics than MES. However, one must be careful, as **these metrics take into account both the uncertainty estimate and the number of solutions in the repertoire**, in the same way as the QD score. MBMEUQW Explicit offers **fewer solutions than MES**, due to its slightly lower coverage, which **may explain the new metrics with lower values**. It's therefore natural to think that, for an equivalent number of solutions, MBMEUQW Explicit might not be as good in terms of these new metrics, and would therefore have higher values. However, it is important to note that **the model seems to work well** at first sight in terms of uncertainty quantification since **the behaviour and values obtained are very close to the values derived from MES** and the scoring function for this uncertain environment.

In conclusion, the MBMEUQW Explicit approach outperforms the first ME baseline but struggles with the second MES baseline. Overall, the model seems to learn well from the various jumps and the growth in coverage and QD score over 3000 generations. The model seems to perform better at estimating a solution through its BD than at estimating its quality through fitness.
This is where a more **in-depth analysis of the model** and its behaviour comes in, **to refute or confirm the various interpretations** established through this analysis.

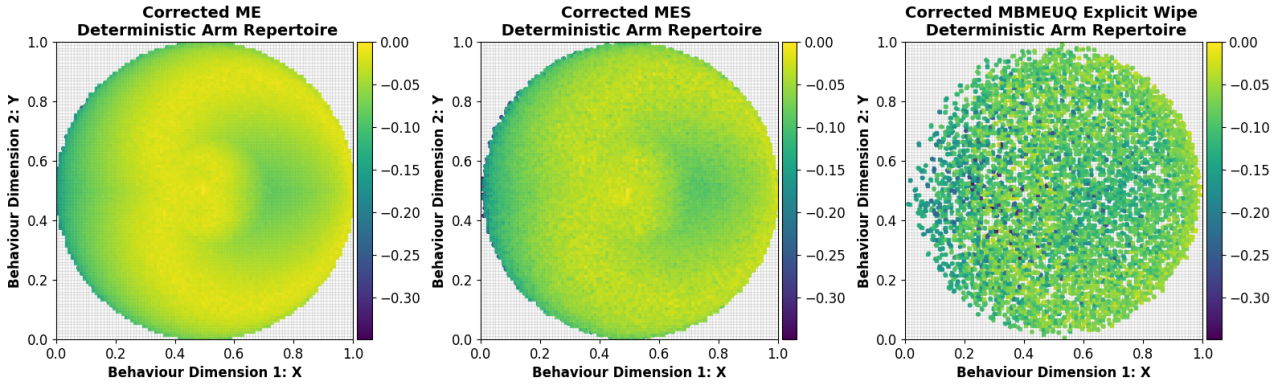


(a) Comparison plot of the corrected metrics with the **deterministic arm** task.

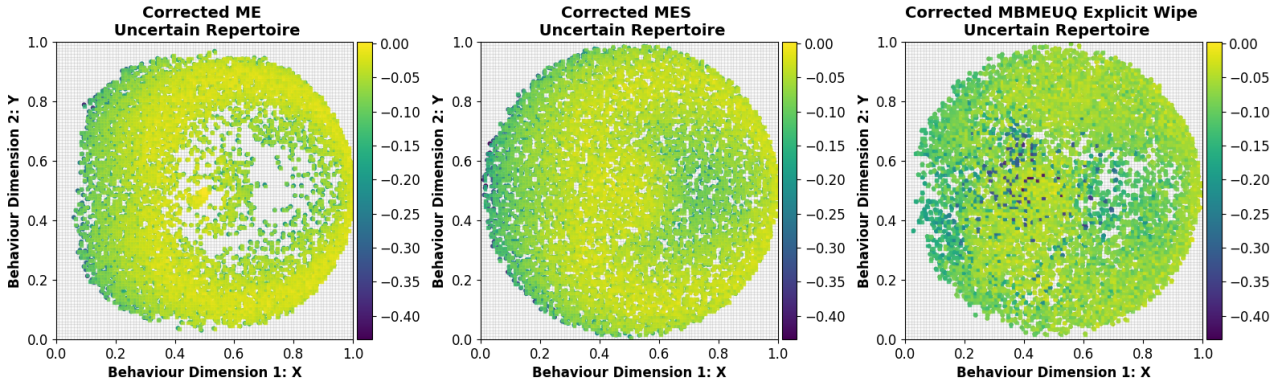


(b) Comparison plot of the corrected metrics with the **uncertain arm** task.

Figure 6: Comparison plot of **QD score** (left), **max fitness** (middle) and **coverage** (right). This comparison is made between the 2 reference algorithms **ME**, **MES**, and the best contribution **MB-MEUQW Explicit**. Results are obtained via **5 replications**, shaded areas correspond to **standard deviations** and dark lines correspond to **medians**.



(a) Comparison plot of the corrected repertoires with the **deterministic arm** task.



(b) Comparison plot of the corrected repertoires with the **uncertain arm** task.

Figure 7: The comparison is made between the 2 reference algorithms **ME** (left), **MES** (middle) and the best contribution **MBMEUQW Explicit** (right). Each repertoire corresponds to **the simulation with the highest QD score** among the 5 replications. The **colour scale is standardized** across the various figures.

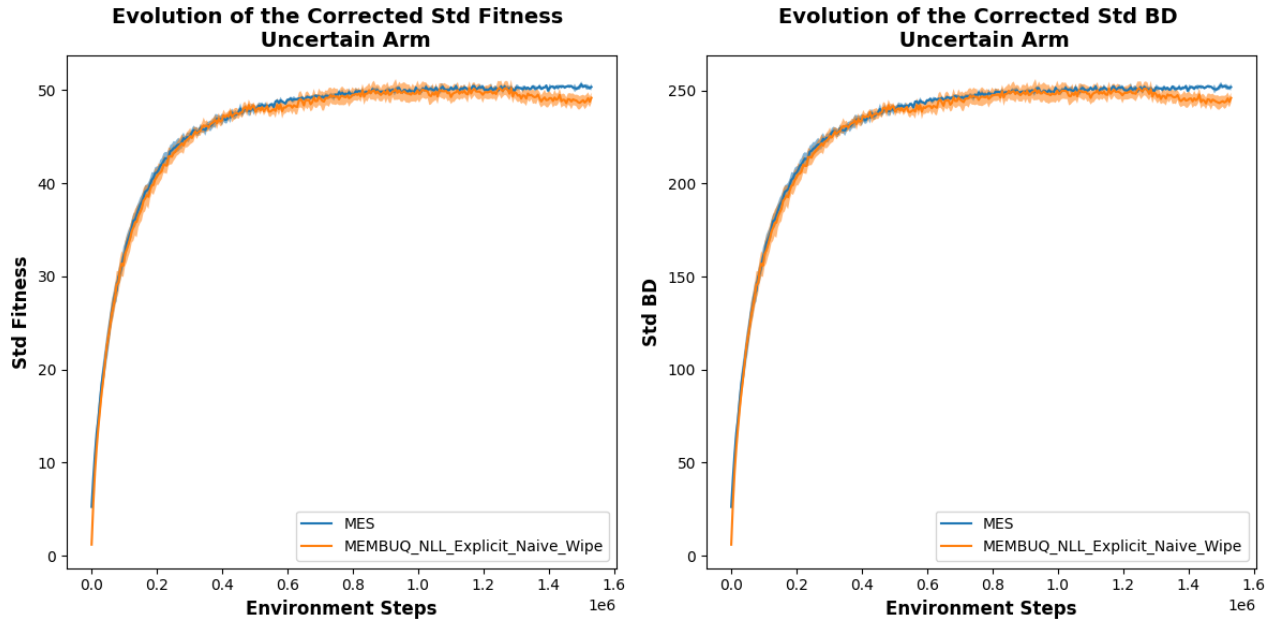


Figure 8: Comparison plot of **2 new corrected metrics**: (1) **UFS** (left) and (2) **UBS** (right) with the **uncertain arm** task. The comparison is made between (1) **MES** and (2) **MEMBUQW Explicit**. Results are obtained via **5 replications**, shaded areas correspond to **standard deviations** and dark lines correspond to **medians**.

E.1.2 In-Depth Model Analysis

To confirm or refute the analysis carried out in section E.1.1, it is important to explore other results. For this reason, the performance of the model associated with the MBMEUQW Explicit algorithm is studied in greater detail, using the criteria set out in section C.3.

Before interpreting other results, it is important to check the model's learning. Although the model seems to be learning correctly according to the jumps and growth in the metrics figure 6b, a double confirmation is essential. For this purpose, **the learning curves** are shown in figure 9.

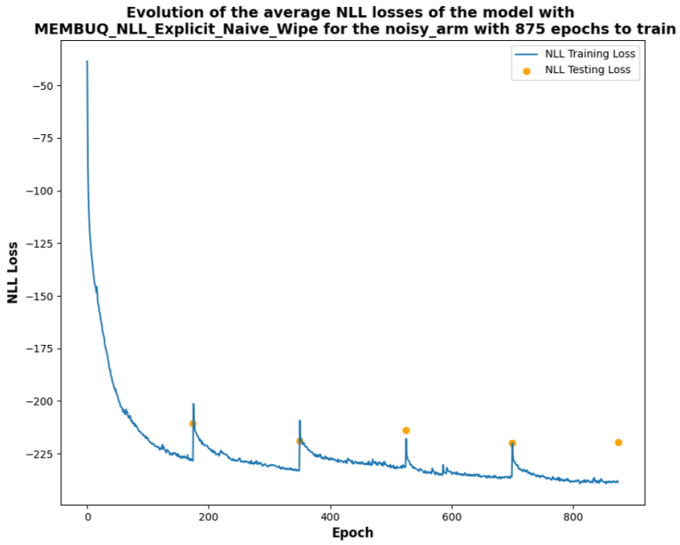


Figure 9: Graph showing **the evolution of the learning curves of the model** for the uncertain arm task with the MBMEUQW Explicit algorithm. In total, the model trained on 875 epochs spread over 5 training sessions.

Fortunately, the model's learning curves testify to **correct learning**. **The NLL decreases and is minimized** as training progresses. The model shows a slight case of overfitting, but this remains negligible given the training, especially as it **can be explained by the PODR hyperparameter**, which retains a certain percentage of the old data for the next training and model test. Finally, **this slight overfitting may also confirm the fact that the model learns correctly**, due to the design of the algorithm, although this may seem counter-intuitive.

Now that we know that the model is correctly trained, let's take a look at its predictions and compare them with the environment's evaluations. Figures 10 to 13 are complementary.
The errors in figure 12 confirm that **the model performs less well in predicting fitness than in predicting BD**. This is shown by **a progressive decrease in the error** between the model and the scoring function **for both BD and the associated uncertainty**. On the contrary, the error seems **to stagnate for fitness predictions** and **to increase slightly for fitness uncertainty predictions**. Thus, as the model trains, it becomes better and more confident in the predictions of the BD, whereas it stagnates but becomes slightly less confident about fitness.
Another important point is the distribution of these errors, shown in figure 13. In addition to the fact that the BD error seems to be decreasing, unlike the fitness error, it is important **to compare this error with the scale of predictions and possibilities**. As a result of normalization, BD varies between 0 and 1, and fitness is mostly contained between -0.10 and 0. For BD, we find a median error of 0.128, while for fitness we find a median error of 0.019, which when brought back to the expected scale represents **a median error percentage of** 12.8% and 19% respectively. **These percentages confirm once again that the model is worse at predicting fitness than BD**, explaining the results and confirming the analysis in section E.1.1.

Figure 11 shows a comparison of the model's predictions with the environment's scoring function. The model seems **to cover a large area of the expected space**, although some spaces are less accessible, namely the extremities of the circle. However, it's important to note that these results are obtained after removing the outliers. The number of outliers for fitness and BD predictions respectively is 7.76% and 13.41% for a total of 127680 predictions. However, this difference in the proportion of outliers can be explained by (1) **the possibility set with a wider domain for the BD** and also by (2) **the fact that the prediction error associated with the BD is still quite high and dispersed before the second training** of the model, as shown in figure 12a.
Outliers can be explained by the fact that the model (1) **is wrong** or (2) **has not been able to train on similar data**. The first case corresponds to poor training, while the second to no training on the data. Given the previous results, metrics as well as errors and learning curves, **case (2) seems to be the most predominant**. This is explained by the design of the model with an **MTF** of 500. Over 500 generations, **it is likely that the model will need to evaluate several solutions that it has never yet encountered through training**, due to this low training frequency. To limit this number of outliers, it may be worthwhile to force an algorithm design **where the model trains with a higher frequency** while adjusting the other hyperparameters.

Finally, even if these results **show that it is possible to improve the model** mainly via its fitness predictions but also from the BD where outliers and error nevertheless remain high, they **also confirm the analysis proposed in section** E.1.1.

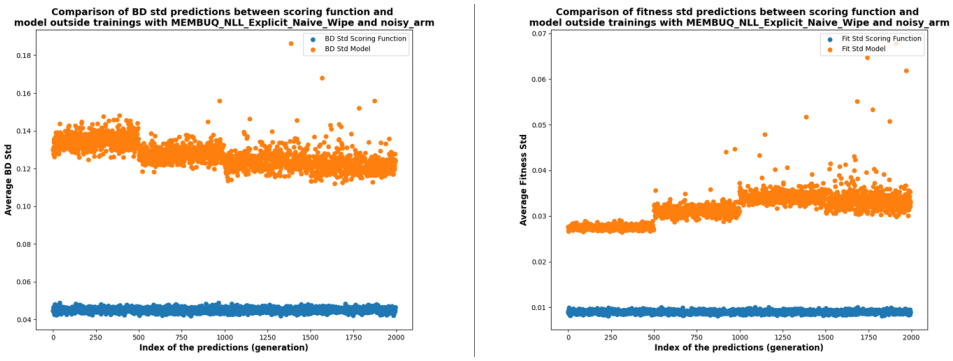


Figure 10: **Comparison plots of the uncertainty BD predictions** (left) and **uncertainty fitness predictions** (right) between MBMEUQW Explicit model predictions and scoring function evaluations with **the uncertain arm** task. These graphs are obtained via the predictions **at each generation when the model is not trained**.

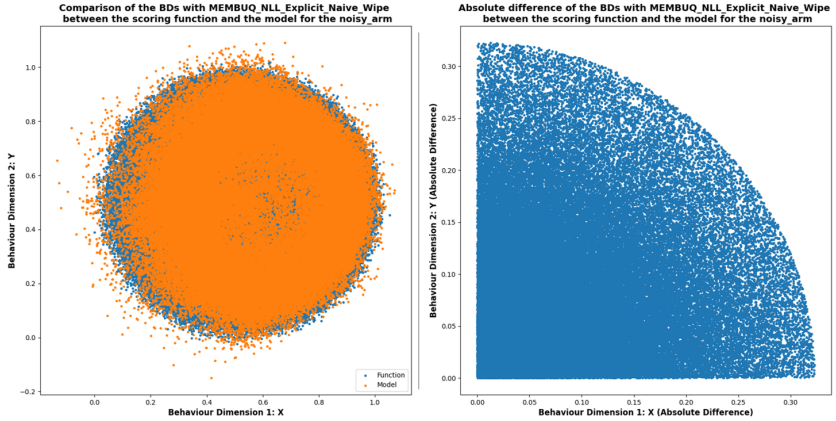
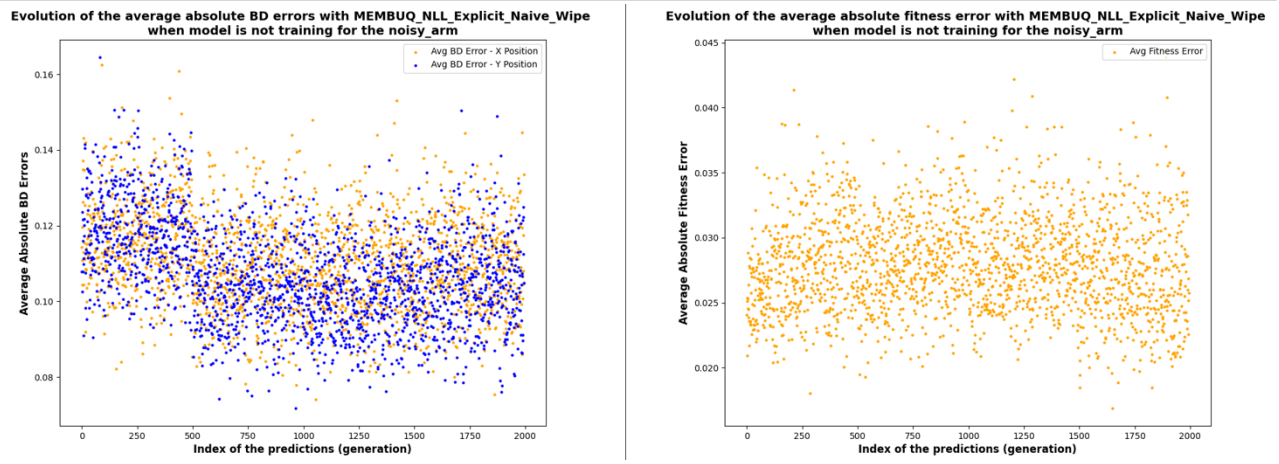
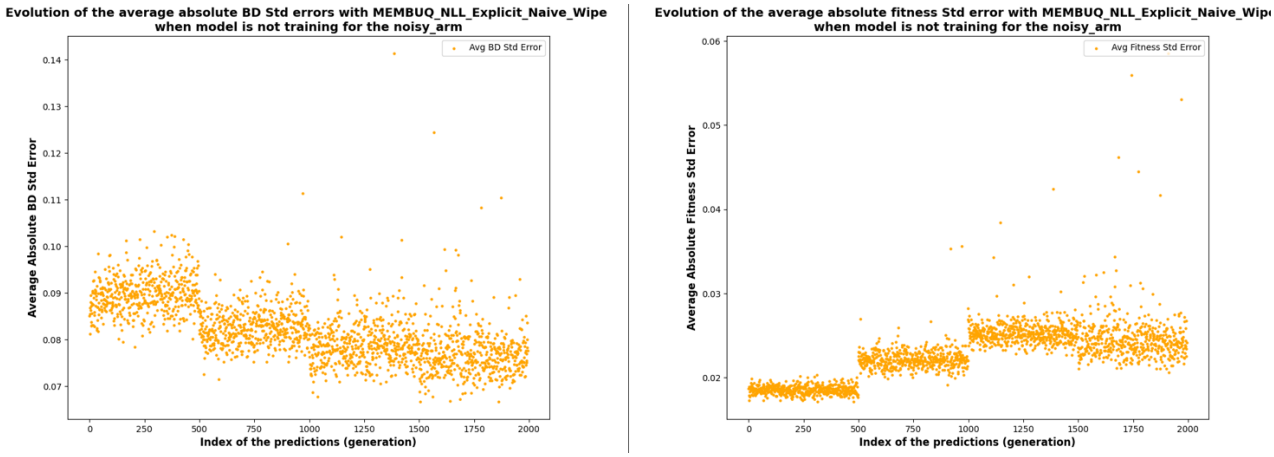


Figure 11: **Comparison plot** (left) and **difference plot** (right) between MBMEUQW Explicit model and scoring function evaluations for the BD with **the uncertain arm** task. These graphs are obtained via the predictions **at each generation without model training** and **after removing outliers**.



(a) Prediction errors of fitness and BD by the MBMEUQW Explicit model.



(b) Prediction errors of fitness and BD uncertainty quantification by the MBMEUQW Explicit model.

Figure 12: Average errors in predictions related to **BD** (left) and **fitness** (right) between the MBMEUQW Explicit model and the scoring function in the case of the **uncertain arm**. These errors are calculated for **each generation when the model is not trained**.

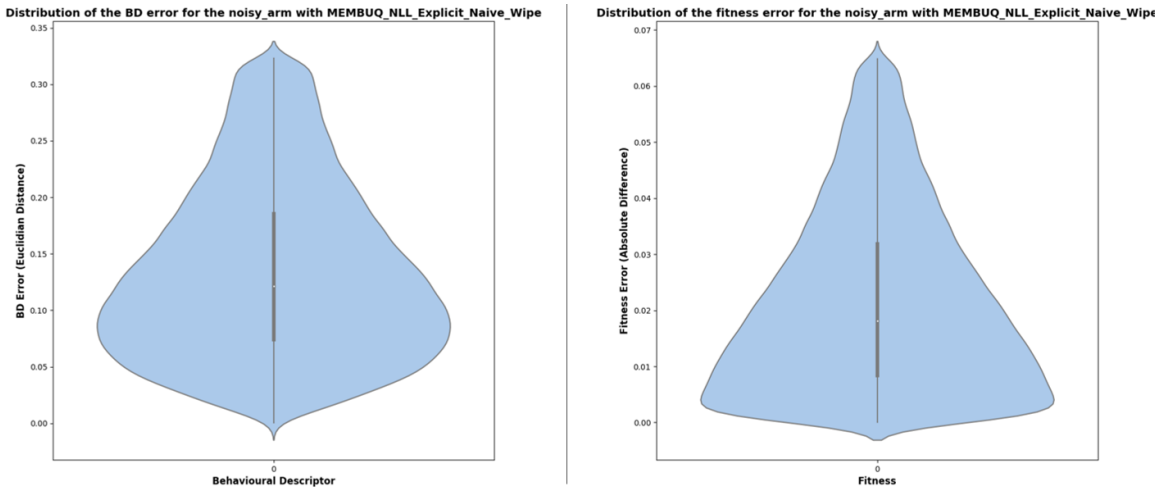


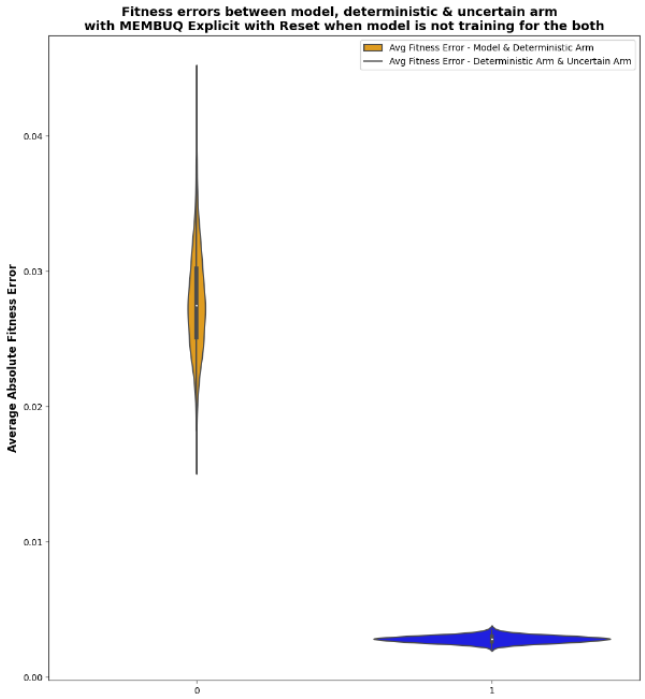
Figure 13: **Violin diagrams of BD error** (left) and **fitness error** (right) between MBMEUQW Explicit model predictions and scoring function evaluations with **the uncertain arm** task. These graphs represent the distribution of errors obtained **at each generation when the model is not trained** and **after removing outliers**. The width of the figure indicates the proportion of elements for each error value, and the centre bar shows both the median and interquartile ranges.

E.1.3 Comparison Between Model and Scoring Functions

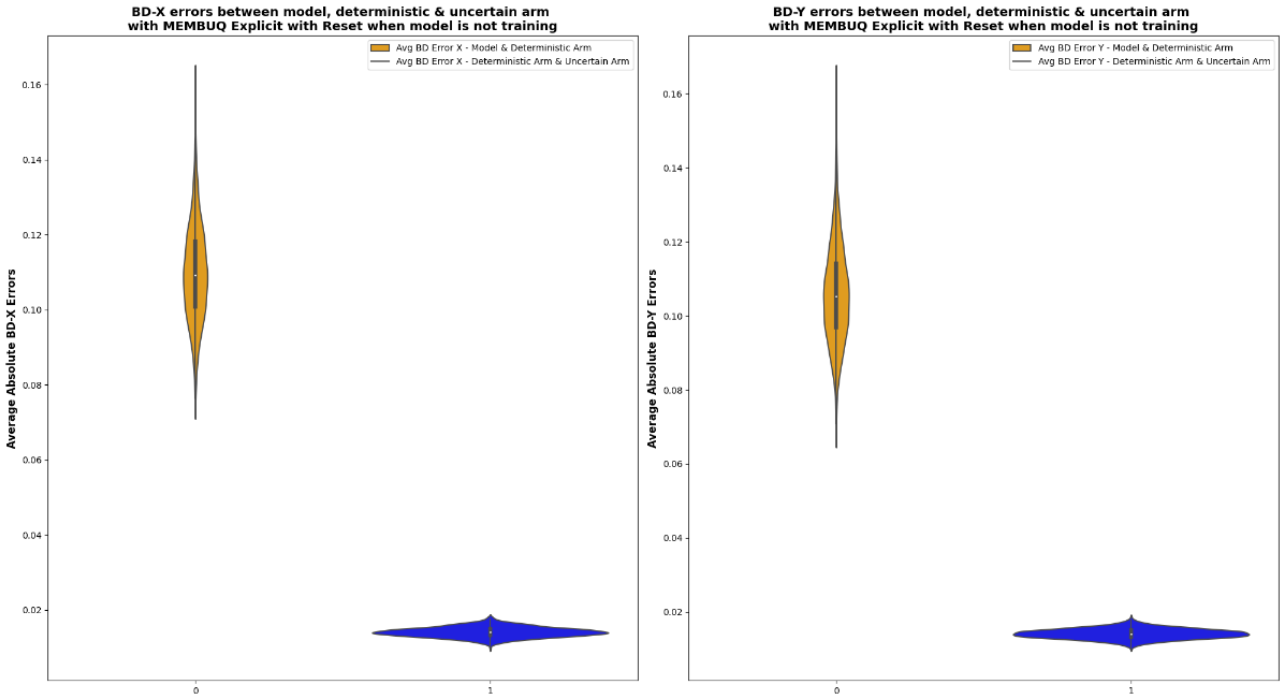
Sections E.1.1 and E.1.2 focus on the behaviour of the MBMEUQW Explicit algorithm and its results, mainly in an uncertain environment. Although the results are not as good as the second baseline, MES, they are **still promising and interesting to explore**. For further analysis, this section compares **the deterministic scoring function** with (1) **the model used in the uncertain environment** with MBMEUQW Explicit algorithm and (2) **the non-deterministic scoring function**.
The principle is simple: (1) the uncertain environment is used, (2) the model is trained in the same way as for all the results obtained in sections E.1.1 and E.1.2, and (3) the solutions are also evaluated by the scoring function associated with the deterministic environment.
Thus, **a new scoring function is introduced**. It performs exactly the same operations as the old one, with the added benefit of **evaluating solutions through its deterministic version even in an uncertain environment**. The results can be seen in figure 14.

These results add another dimension to the analysis of the model's performance and interest. The results show that the model underperforms the scoring function for the uncertain environment. This lack of performance is reflected in (1) **a higher error** and (2) **a more dispersed distribution**. At present, the model is less interesting than the simple scoring function, but these results **demonstrate the value of using a model**.
Indeed, even if the model performs less well individually, **its strength lies in the fact that it can evaluate more solutions at a lower cost**. As a reminder, this corresponds to the hyperparameter **number of subdivisions**, introduced in section C.2. Although the median error associated with the model is around 6 times greater than that associated with the uncertain scoring function, for both BD and fitness, **the metrics do not show such a discrepancy**, as shown in figure 6b or the repertoires in figure 7b. Using the model via MBMEUQW Explicit "only" represents a drop of 3.17% and 6.79% respectively in coverage and QD score as shown in table E.1.1 compared to using the scoring function via the MES baseline.

In conclusion, while the model still shows high errors compared to the uncertain scoring function, the metrics do not show such a discrepancy. This shows the value of using a model. **For the moment, the model is less interesting, but the results are promising**. They suggest that if a model were more properly trained, probably with **a slightly adapted design and more appropriate hyperparameters**, then the error associated with predictions would decrease all the more. As a result, **the model would identify a higher diversity of solutions, as well as a higher quality of the latter**, probably enabling it to beat the second baseline.



(a) Comparison of the **average fitness errors**.



(b) Comparison of the **average BD errors**. The BD associated with the X position is on the left, and the BD associated with the Y position is on the right.

Figure 14: Comparison of prediction errors between (1) **the model from MBMEUQW Explicit and the deterministic scoring function** (orange), and (2) **the deterministic and uncertain scoring functions** (blue).

E.2 Ablation Study

Now that the best algorithm has been presented, along with its performance against baselines, let's look at the other contributions. The comparison of the other contributions is done through **an ablation study**. For this purpose, 3 mechanisms will be studied, namely (1) **the repertoire reset mechanism**, (2) **the choice of model** quantifying or not the uncertainty and finally (3) **the sampling approach** used to train the model, respectively in sections E.2.1, E.2.2 and E.2.3.
Analysis and interpretation are provided by the results shown in figures 15 to 19.

E.2.1 Reset Mechanism

An important point to identify right away is the behaviour of the different contributions **without the repertoire reset**. The corresponding metrics are shown in figure 16a and the repertoires in figure 18b. The different metrics seem to show a **similar behaviour between MBME and MBMEUQ Explicit** and a second **similar behaviour between MBME and MBMEUQ Implicit**. According to these metrics, the explicit sampling approach seems more interesting than the implicit sampling approach, regardless of the model design. However, if we look at the visualization of results through repertoires, we can see that **the design of these algorithms without reset is misleading**.

First of all, **repertoires are built in a similar way** both between MBME and MBMEUQ Explicit and between MBME and MBMEUQ Implicit. If we look at the behaviour of these same algorithms, but this time **with the repertoire reset**, it is clear that **the behaviour is no longer at all similar between the algorithms**, either from the point of view of the repertoires or of the available metrics figure 16b. As stated earlier, this is **due to a misleading or poorly adapted design of the MBQD approaches without reset**. According to the design introduced in chapter C, all the new contributions work as follows: (1) use the scoring function to feed both the buffer and the repertoire, (2) use the model to feed the repertoire and the scoring function to feed the buffer from the FTG introduced in section C.2. The value of this FTG given in table C.2.1 for the different algorithms shows that it represents $1/6$ or $1/3$ of the simulation for algorithms with the implicit and explicit approaches respectively. So the scoring function is used for $1/6$ or $1/3$ of the simulation, i.e. 0.256×10^6 or 0.512×10^6 environment steps, and from figure 16b where these instants are easily identifiable thanks to the reset, we can see that (1) the QD score is higher at this instant than at the final value for the implicit cases and that (2) the QD score has reached around 98% of its final value for the explicit cases.
This behaviour shows that **the model plays little or no part in identifying relevant solutions and adding them to the repertoire**, which is **the opposite of the expected behaviour** that prompted the implementation of these MBQD approaches. This is justified by the fact that the solutions identified by the scoring function, during its period of use to populate the repertoire, are either **lucky or over-perform compared to the majority of solutions predicted by the model** despite its multiple trainings, preventing the latter from being truly effective.

Finally, this first ablation of the reset mechanism shows that it is essential **to consider the impact of the model** through the proposed design. Consequently, the 4 **contributions without this reset will not be analyzed further**, since **they are misleading** and the results obtained **depend mainly on the scoring function and not on the model**. Similarly, this justifies the superiority of approaches with explicit sampling irrespective of the model design, since the 1000-generation explicit approach is better in this uncertain environment than the 500-generation implicit approach, particularly regarding the curves of the benchmark algorithms ME and MES.

E.2.2 Model Design

Following the repertoire reset analysis in section E.2.1, only 4 of the 8 contributions are analyzed to understand the algorithm design in more detail. The 4 contributions are **MBMEW and MBMEUQW Explicit** as well as **MBMEW and MBMEUQW Implicit**.
To provide a quantifiable comparison between the different algorithms, a summary of the results is given in table E.2.1, in the same way as table E.1.1. This time, the improvement defined in equation E.1 is not made regarding the two baselines, but **from one algorithm to its equivalent with the other model choice**.

Metrics \ Algorithms	MBMEW Exp	MBMEUQW Exp	MBMEW Imp	MBMEUQW Imp
QD Score	2539.05	4472.36	331.8895	4397.39
		+76.14%		+1224.96%
	−43.23%		−92.45%	
Max fitness	0.00089	−0.01346	0.00061	0.00066
Coverage	31.05	48.94	4.73	50.12
		+57.62%		+959.62%
	−36.55%		−90.56%	
UFS		48.79		49.97
UBS		243.95		249.82

Table E.2.1: Comparison of the different metrics between MBMEW and MBMEUQW algorithms with **the uncertain arm** task. The white cells correspond to **the final value** (first line), the slightly shaded cells correspond to **the improvement over non-UQ algorithms** (second line) and the darker cells correspond to **the improvement over UQ algorithms** (3rd line). Black cells correspond to **undefined results**.

The repertoire reset gives an idea of the true performance of the proposed model concerning its predictions and its performance in identifying good solutions and filling the repertoire. Unlike the results without this reset, where performance seemed independent of the chosen model, as shown in figure 16a, table E.2.1 and figure 16b show **a real split in model behaviour and performance**. Whether for the explicit or implicit approach, there is **a performance gap when using repertoire reset, between a deterministic model that doesn't quantify uncertainty and a probabilistic model that quantifies the uncertainty** associated with each of the scores.
Indeed, the probabilistic model proposing a representation of the distribution of scores performs much better than the deterministic model predicting only the score of solutions. In the explicit case, the QD score shows an improvement of 76.14%, and in the implicit case, the improvement is even greater at 1224.96%. The improvement in this QD score is **closely linked to the improvement in coverage**. This means that the major difference in model design is reflected mainly through **an improvement in BD predictions rather than fitness**, although the repertoires in figure 18b also show a great improvement in fitness estimation for MBMEUQ algorithms.
Despite correct learning of **the deterministic models not modelling the distribution** as shown by the learning curves in appendix C, results reflect that **this design is not adapted to an uncertain environment**. On the other hand, in the deterministic case, the opposite behaviour is observed, with models performing better when they don't quantify uncertainty, as shown in figure 15.

In conclusion, this ablation of model selection demonstrates **the importance of selecting a probabilistic model that represents the distribution of scores** and therefore quantifies their uncertainty in uncertain environments. Consequently, algorithms offering neither uncertainty quantification nor repertoire reset will not be studied in section E.2.3.

E.2.3 Sampling Approach

As a result of the two studies proposed in sections E.2.1 and E.2.2, only 2 of the 8 contributions are analyzed, namely **MBMEUQW Explicit** and **MBMEUQW Implicit**, to measure and observe differences in behaviour according to the sampling approach chosen.
Once again, the results are presented and quantified in tabular form, E.2.2.

Metrics \ Algorithms	MBMEUQW Explicit	MBMEUQW Implicit
QD Score	4472.36	4397.39
		−1.68%
	+1.70%	
Max fitness	−0.01346	0.00066
Coverage	48.94	50.12
		+2.41%
	−2.35%	
UFS	48.79	49.97
UBS	243.95	249.82

Table E.2.2: Comparison of the different metrics between (1) MBMEUQW Explicit and (2) MBMEUQW Implicit with **the uncertain arm** task. The white cells correspond to **the final value** (first line), the slightly shaded cells correspond to **the improvement over explicit sampling approach** (second line) and the darker cells correspond to **the improvement over implicit sampling approach** (3rd line). Black cells correspond to **undefined results**.

The two algorithms produce almost identical results. The main difference comes from the max fitness metric, which is not the most representative.
However, it is interesting to note that despite **a higher QD score** of 1.70%, the Explicit approach has **a lower coverage** of 2.35%. This shows that **the explicit approach is better in terms of fitness estimation than the implicit approach**, which performs better in estimating BD. This interpretation is visually identifiable in figure 18b with the **golden heart shape more distinguishable with the explicit case**, while the implicit case offers more solutions but of lower quality.
From the point of view of uncertainty quantification, the two algorithms appear similar in terms of performance. As indicated in section E.1.1, it is not obvious which algorithm is better for uncertainty quantification, via UFS and UBS, given that they are proportional to coverage and that we aim to minimize them. However, we note that the orders of magnitude are also very similar concerning the coverage obtained, **confirming similar behaviour with respect to uncertainty quantification** despite the slight differences in value.
If we disregard the final results and **focus on the behaviour of each algorithm** during simulation, the MBMEUQW Explicit algorithm shows a real progression as the model is trained, whereas the MBMEUQW Implicit algorithm shows mostly stagnation. According to figure 16b, MBMEUQW Implicit achieves higher performance faster than MBMEUQW Explicit, but this performance remains more or less constant once achieved. In contrast, the MBMEUQW Explicit algorithm demonstrates **real added value over the generations following each model** training, **outperforming the main metric** for comparing the algorithms, namely the QD score obtained by MBMEUQW Implicit at the end of the simulation. Thus, implicit sampling seems to saturate to a sub-optimal point, while explicit sampling exceeds it, leading to **a favouring of MBMEUQW Explicit behaviour**.

In conclusion, **the explicit sampling approach is more interesting** and enables the algorithms **to perform better and behave more promisingly** than the implicit sampling approach in an uncertain environment. Consequently, we can deduce that in our case the model performs better if **it trains on better quality data, even if this means losing quantity**.

E.2.4 Global Comparison and Summary with Baselines

To summarize this ablation study, let's look at the 3 mechanisms, their results and key interpretations.

1. The repertoire reset mechanism

   With the design proposed for these 8 new contributions, this is the most essential tool, not in terms of performance, but **to enable a fair comparison of the different algorithms**. Indeed, MBQD approaches aim to use a model to identify solutions and add them to the repertoire. **Without this mechanism, the algorithms contribute to the repertoire mainly through the scoring function** and not through the model, **establishing a false point of comparison** about the performance of the 8 algorithms. This comparison is all the more unequal following the analysis indicating that the model proposes a higher error in its predictions than the scoring function of the environment.

2. The model design

   Two model choices were presented in this project, with one deterministic model seeking to predict the exact scores associated with a solution, and a probabilistic model seeking instead to identify the distribution associated with the scores of a solution, while quantifying the uncertainty. It is this latter choice of model that has **the greatest impact on the performance** obtained with the different algorithms. Indeed, in the case we're interested in, namely **an uncertain environment**, it is **the model that quantifies uncertainty that clearly stands out**, whereas in the deterministic case, it is the deterministic one.

3. The sampling approach

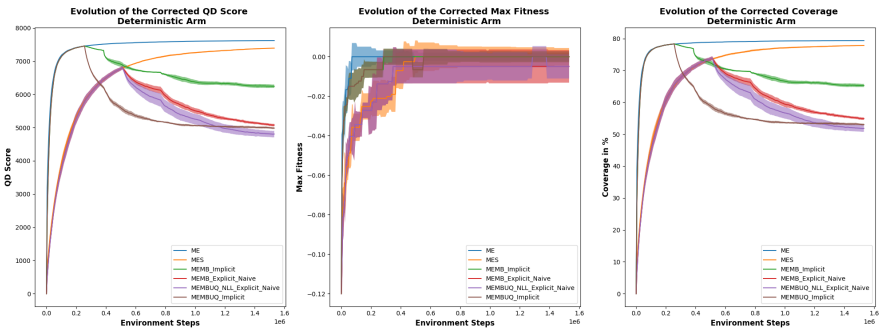
   This last criterion defines the type of data the model is trained on, thus **defining the quantity and quality of the data** supplied to it. Contrary to the choice of model, this last criterion does not have a significant impact on the model's final performance, but it does have **an impact on its evolution and behaviour during the simulation**. Indeed, **the explicit approach** not only delivers **higher results but also proposes more promising behaviour** than the implicit one.

It's important to point out that the hyperparameter **number of subdivisions** has been selected from the set $[5, 30, 50, 80, 100]$ **to maximize the QD score**. Even if the selection established in table C.2.1 **gives the impression of being unfair** between the different contributions, on the contrary, **it reinforces the analysis of the results** established so far and **demonstrates the interest of** (1) **using the reset** to limit elitism and (2) **using a probabilistic model** to have better predictions and lower errors.

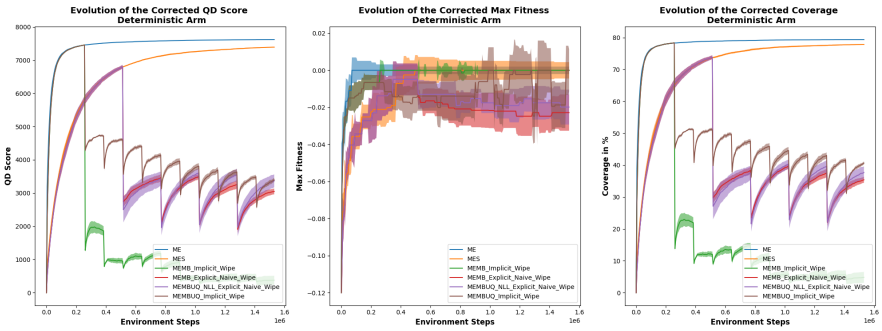
In addition to the comparison through metrics and repertoires, a new analysis is proposed through loss metrics and more particularly the **QD score losses** proposed in figure 20. As a reminder, this new metric, presented in section B.3.1 and equation B.8, represents **the extent to which an estimator is wrong in its prediction between actual and predicted performance**, so between corrected and uncorrected predictions. The aim is **to minimize this drop**.
We can see that the 8 contributions have **a median loss lower than ME's** 54.55%, but **higher than MES's** 37.84%. We need to be careful and also consider the previous results, particularly the metrics since this graph might lead us to believe that the MEMBW Explicit contribution is the most interesting after MES. According to this loss alone, this is the case, except that if we go back to the metrics in figure 16, we remember that this is clearly not the best-performing contribution, unlike MEMBUQW Explicit. Therefore, we're only interested in the two best contributions, which are **the MEMBUQW algorithms with the two sampling approaches**.

As well as having **a higher QD score**, the explicit approach has a median QD score loss of 45.74%, **lower than that of the implicit approach**, which is 50.64%. A final point of interest is that the MEMBUQW Explicit algorithm offers some **smaller overestimates than MES throughout a simulation**, as shown by the boxplot extremity at around 20%, although the median loss remains higher. Once again, **this confirms the interest and promising behaviour of using a model** through the MBMEUQW Explicit algorithm.

In conclusion, the approaches presented demonstrate the interest of the 3 proposed mechanisms. They have different purposes and **impact the algorithms** at different levels: **comparison, performance and behaviour**. Moreover, even if the second MES baseline isn't beaten by the best contribution, MBMEUQW Explicit, **the results of the latter remain close and its behaviour is also promising**.

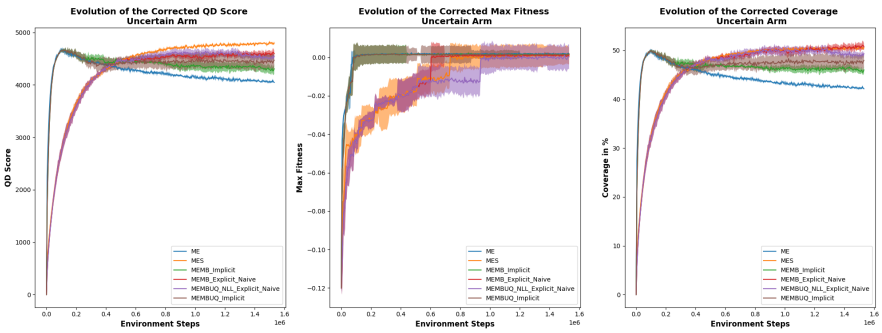


(a) Comparison plot between the 2 baselines and the 4 MBME(UQ) versions **without repertoire reset**.

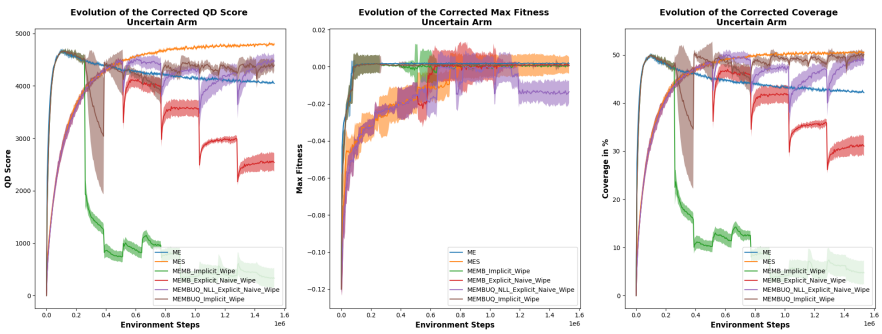


(b) Comparison plot between the 2 baselines and the 4 MBME(UQ) versions **with repertoire reset**.

Figure 15: Comparison plot of **3 corrected metrics**: **QD score** (left), **max fitness** (middle) and **coverage** (right) with the **deterministic arm** task and 10 algorithms. Results are obtained via **5 replications**, shaded areas correspond to **standard deviations** and dark lines correspond to **medians**.

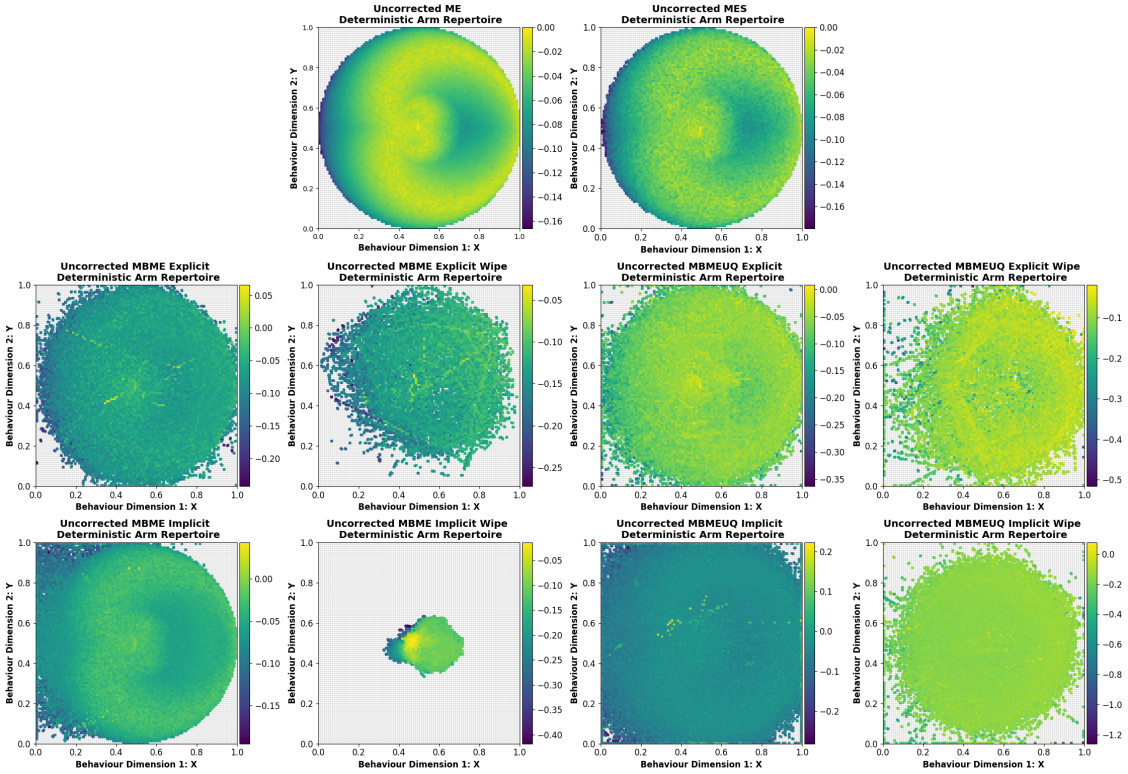


(a) Comparison plot between the 2 baselines and the 4 MBME(UQ) versions **without repertoire reset**.

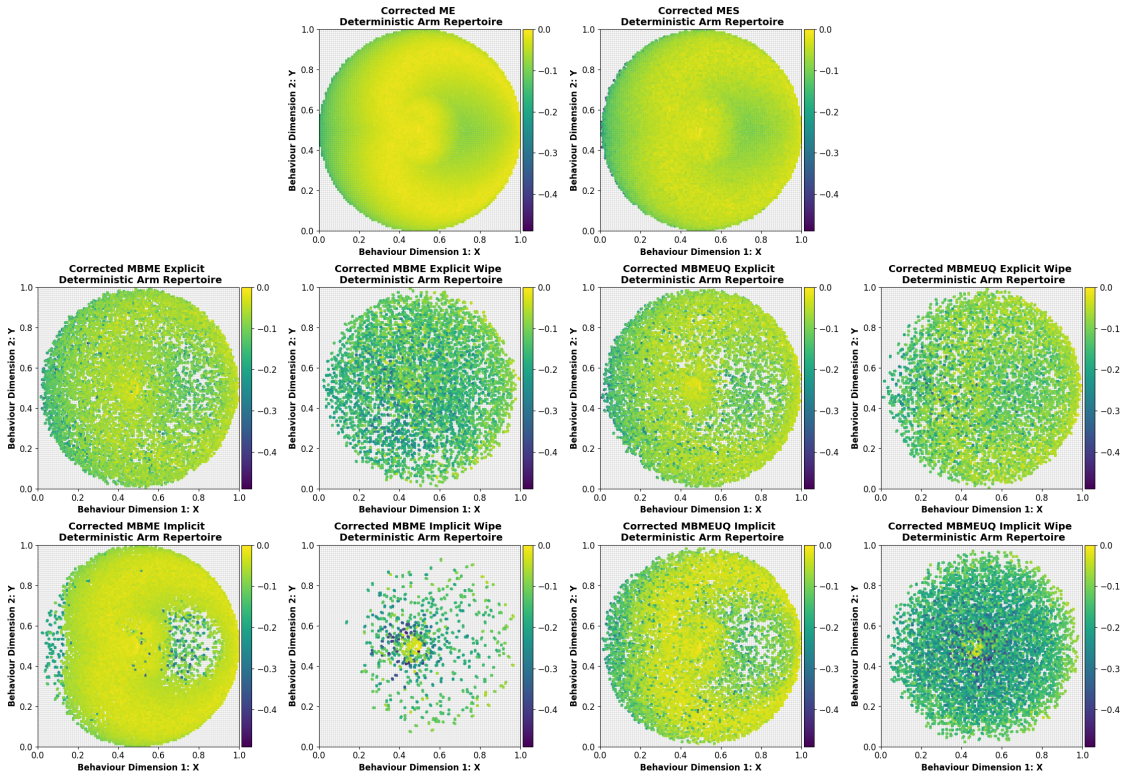


(b) Comparison plot between the 2 baselines and the 4 MBME(UQ) versions **with repertoire reset**.

Figure 16: Comparison plot of **3 corrected metrics**: **QD score** (left), **max fitness** (middle) and **coverage** (right) with the **uncertain arm** task and 10 algorithms. Results are obtained via **5 replications**, shaded areas correspond to **standard deviations** and dark lines correspond to **medians**.

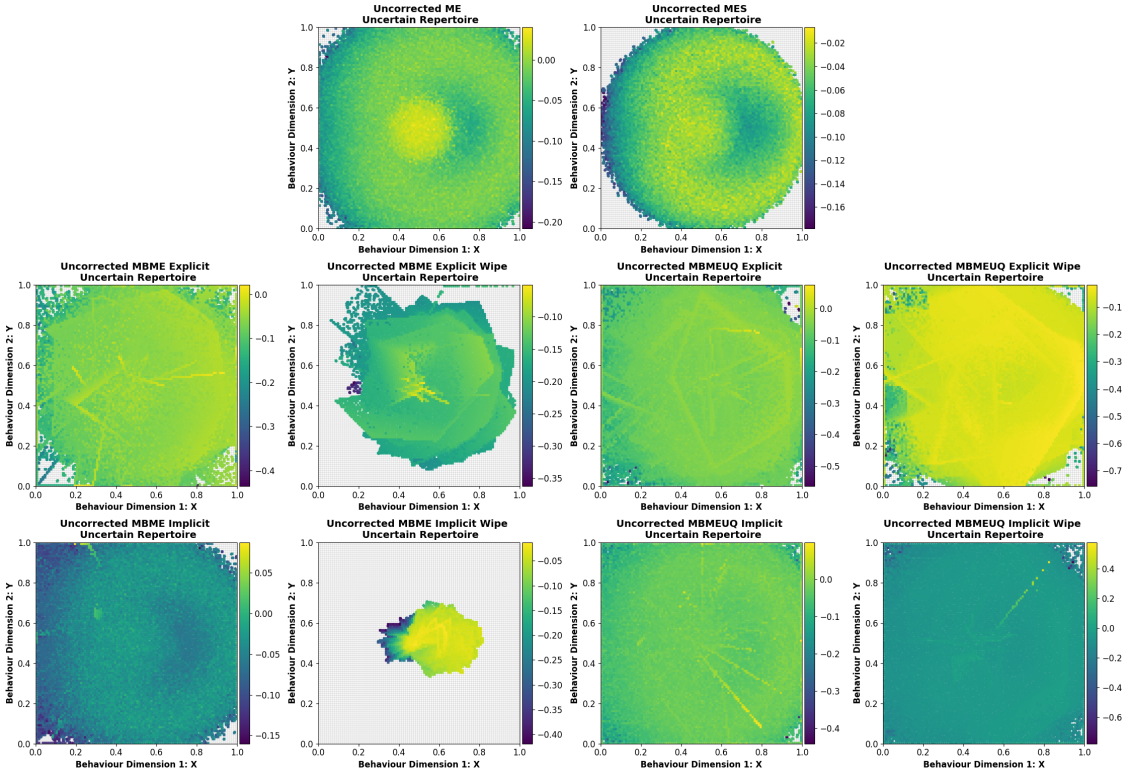


(a) Comparison plot of **10 uncorrected repertoires** with the **deterministic arm** task. The **colour scale is not standardized** across the various figures.

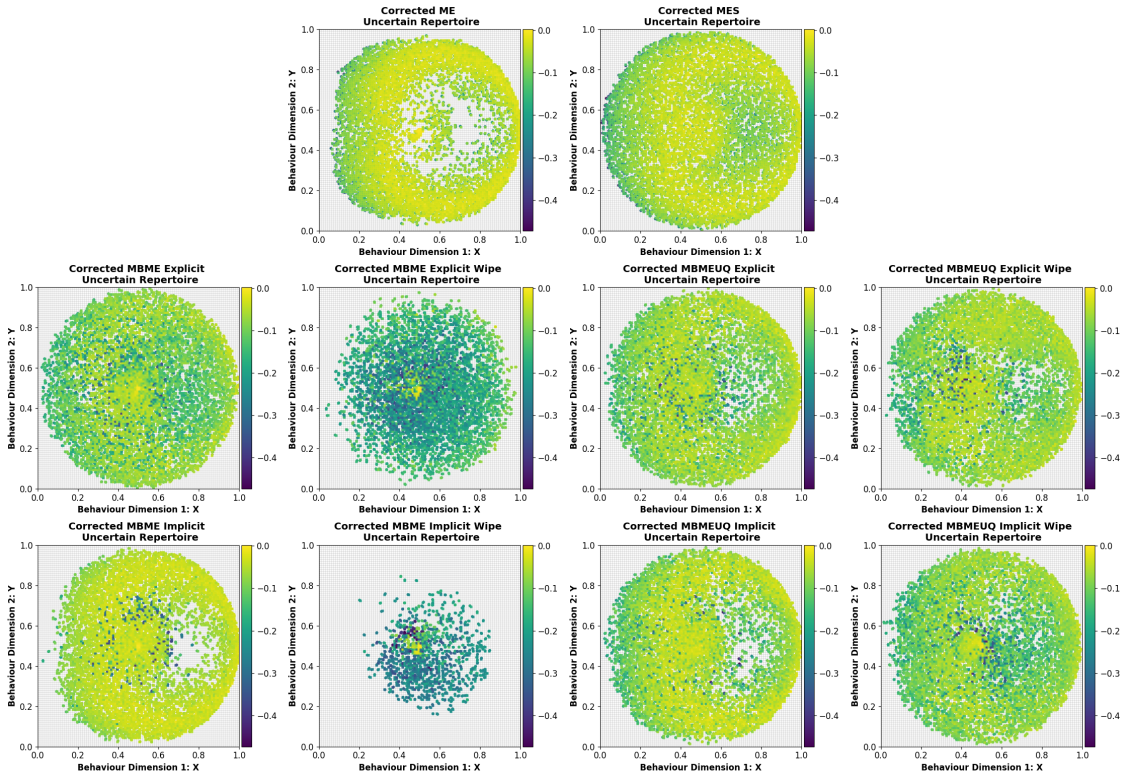


(b) Comparison plot of **10 corrected repertoires** with the **deterministic arm** task. The **colour scale is standardized** across the various figures.

Figure 17: The comparison is made between the 2 reference algorithms **ME** and **MES at the top**, **the 4 versions of MBME Explicit in the middle**, and **the 4 versions of MBME Implicit at the bottom**. A version corresponds (1) to the choice of model, with or without UQ, and (2) to the use or non-use of repertoire reset. Each repertoire corresponds to **the simulation with the highest QD score** among the 5 replications.



(a) Comparison plot of **10 uncorrected repertoires** with the **uncertain arm** task. The **colour scale is not standardized** across the various figures.



(b) Comparison plot of **10 corrected repertoires** with the **uncertain arm** task. The **colour scale is standardized** across the various figures.

Figure 18: The comparison is made between the 2 reference algorithms **ME** and **MES at the top**, **the 4 versions of MBME Explicit in the middle**, and **the 4 versions of MBME Implicit at the bottom**. A version corresponds (1) to the choice of model, with or without UQ, and (2) to the use or non-use of repertoire reset. Each repertoire corresponds to **the simulation with the highest QD score** among the 5 replications.

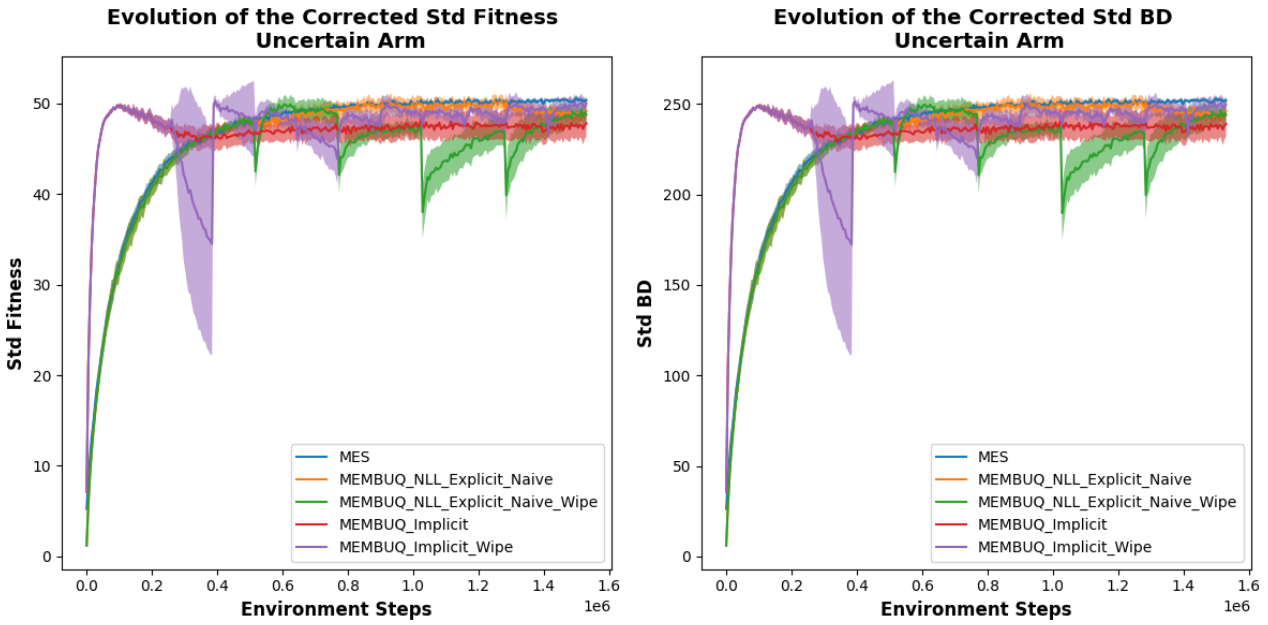


Figure 19: Comparison plot of **2 new corrected metrics**: (1) **UFS** (left) and (2) **UBS** (right) with the **uncertain arm** task. The comparison is made between (1) **MES** and (2) **the 4 MBMEUQ versions**. Results are obtained via **5 replications**, shaded areas correspond to **standard deviations** and dark lines correspond to **medians**.

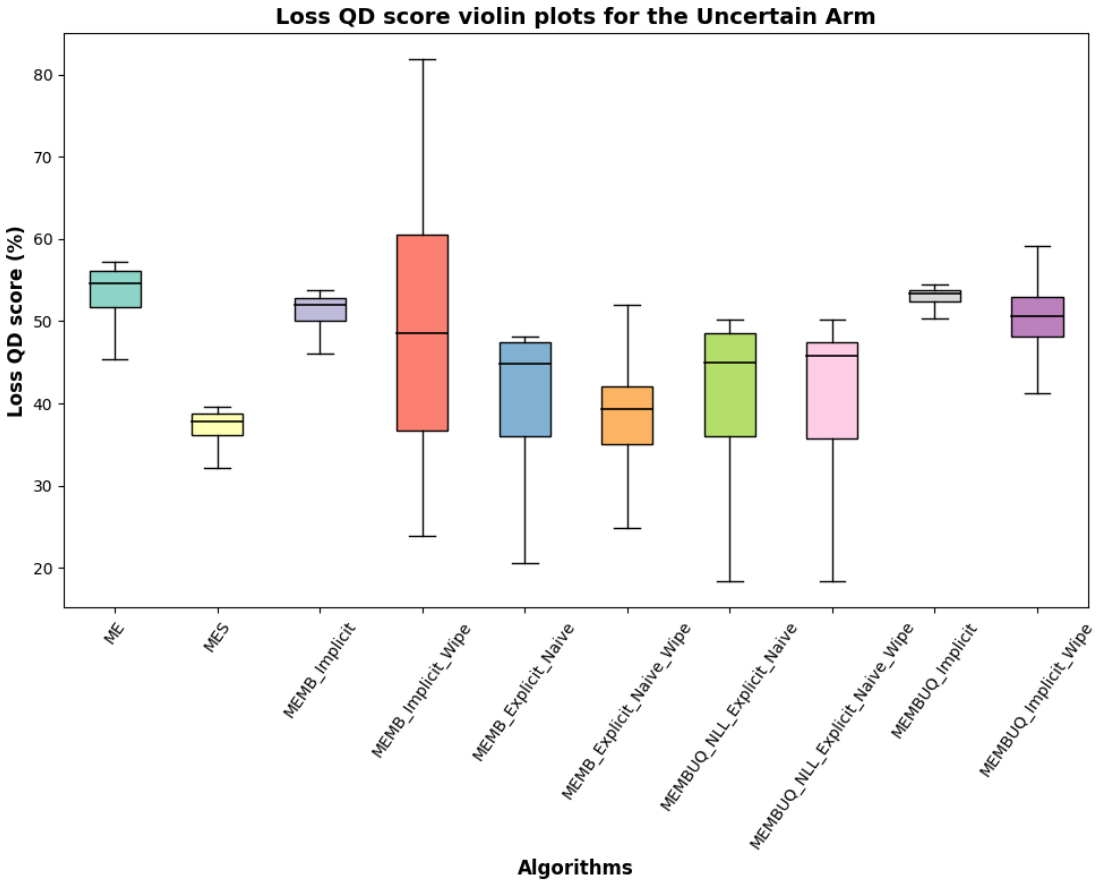


Figure 20: Boxplots of **the QD score loss** between the 2 benchmark algorithms and the 8 new contributions. Losses and distributions are calculated across the 5 replications for the entire simulation rather than just the final value.

Chapter F

Conclusion

To conclude this research project, let's recall **the various key points** as well as **the associated progress and remarks**, before looking at **future perspectives** to pursue the development and research proposed through this project. These points are discussed in sections F.1 and F.2 respectively.

F.1 Progress and Discussion

In this research project, we proposed and compared several **model-based approaches associated with a QD algorithm**, MAP-Elites. These approaches combine **3 mechanisms**, namely (1) **the ability to reset and re-evaluate a repertoire** through the model, (2) **the model design to quantify or not the uncertainty** of solutions, and (3) **2 sampling methods** to modify the model's training data. The first aim of such model-based approaches is **to capture the relationship between inputs and outputs**, i.e. genotype and scores respectively, following the evaluation of a solution, **despite noise** and therefore in non-deterministic environments. The second aim of such approaches is **to improve the data-efficiency of QD algorithms**, in particular by achieving better or similar performance to the scoring function, but with a reduced number of evaluations.

Due to the design of the contributions and the objectives of this project, **new hyperparameters and metrics have been introduced**. These include **the number of subdivisions** hyperparameter, indicating the number of additional batches the model can evaluate, to be fair in terms of resources to the scoring function's evaluation of a single batch. Two metrics are also proposed, **UFS and UBS**, quantifying the uncertainty associated with the scores of the solutions identified by the model.
The results show that the use of **the repertoire reset is essential for a fair comparison** of algorithms. In addition, this repertoire reset (1) **corrects model errors** that accumulate from generation to generation after each model training and (2) **limits the elitism of MAP-Elites**. The results also show that for an uncertain environment, **the probabilistic model that quantifies uncertainty is the most efficient and suitable**. Finally, **the explicit sampling approach shows better results and better model behaviour** within the simulation. The algorithm resulting from the combination of these three criteria, **Model-Based MAP-Elites Uncertainty Quantification Wipe Explicit**, manages to **beat the first MAP-Elites baseline but performs slightly less well than the second MAP-Elites-Sampling baseline**. Although the model performs less well than the scoring function in its predictions, its "free" evaluation feature enables it to partially make up for this shortcoming. As a result, the model identifies a large collection of solutions, almost identical to the second baseline, but with slightly poorer performance estimates.

This project highlights **the challenge of designing MBQD algorithms and adjusting the set of hyperparameters** to enable the model to learn correctly and regularly about the operation of **an uncertain environment**. Even if the second baseline is not beaten, the difference between the metrics of the proposed algorithm and the second baseline remains small, especially for coverage. Moreover, the evolution of the model shows **promising behaviour and results**. Thus, it is assumed that if the hyperparameters related to model training and algorithm design were better tuned, then the results would be all the more convincing and potentially beat this second baseline.

F.2 Prospects for Future Research

This research project, which combines QD algorithms, and MB approaches with uncertainty quantification and sampling approaches, suggests the emergence of new areas of research and new considerations.

First of all, it is important to note that **MES is a good baseline for the task under consideration**. Indeed, keeping the mean over numerous reevaluations is an efficient method, given that the noise applied to the scores follows a Gaussian distribution. On top of that, **the environment has no dynamics**, since there are no transitions or state-action spaces. Solution scores are obtained directly after genotype evaluation, with **no intermediate process between input and output**. These two considerations are linked to the structure of (1) the noise and (2) the environment, and **they amplify the efficiency of MES while reducing that of MBQD algorithms**.
Therefore, it could be interesting to compare the 2 baselines and the 8 contributions through **new environments featuring dynamics** such as Halfcheetah from the Brax open-source library introduced by Freeman et al. [64]. The interest of the model is to train and accumulate information as the simulation progresses. In theory, the presence of dynamics will demonstrate the under-efficiency of MES and its explicit sampling which does not consider past information unlike the model, and therefore independently re-evaluates a solution based on previous observations. Proposing comparisons of baselines and contributions with **new distributions of noise** could also be of interest.

This project is also interested in the quantification of uncertainty but does not use it explicitly. However, the quantified uncertainty is mixed between aleatoric and epistemic uncertainty so it is important **to distinguish one from the other**. To this end, it may be worth proposing an approach based on **an ensemble of models**. This ensemble would **quantify epistemic uncertainty** from its different predictions of each model for the same solution, then **aleatoric uncertainty would be deduced** from the total uncertainty.
Following this quantification, it could be interesting to propose conditions on the addition, reevaluation or emission of solutions, **taking into account the level of epistemic uncertainty**.

Finally, the contributions proposed in this project use the scoring function and therefore the environment for a number of generations, rather than using the model to feed the directory all the time. Firstly, it might be more relevant to move to **a full MBQD approach by using only the model to feed the repertoire**, even from the first generations where the model has not necessarily been trained. On the one hand, this would make the MBQD approach **fairer**, and on the other, it would enable us to (1) **compare all contributions**, even those without repertoire reset, and (2) realize and potentially **quantify the impact of repertoire reset**, which limits the elitism of MAP-Elites.
In addition to this modification linked to the use of the model, it could be interesting **to modify the buffer feed** by identifying the most promising solutions and allowing the model to train on them. To do this, as soon as a solution is added to the repertoire, **a new score** must be associated with it: **the repertoire improvement** (RI). This score comes from the emitter introduced by Fontaine et al. [33]. It is described in equation F.1.

$$\text{Repertoire Improvement: RI} = \left| \text{Fitness}_{\text{Old Value}} - \text{Fitness}_{\text{New Value}} \right| \tag{F.1}$$

Finally, the operation of this **adapted MBME algorithm** is shown in figure 21.
The solutions added to the buffer correspond to those with the highest RI. This mechanism **explicitly considers fitness** through the RI expression, and **implicitly considers BD**, since only solutions added to the repertoire at each generation are taken into account.

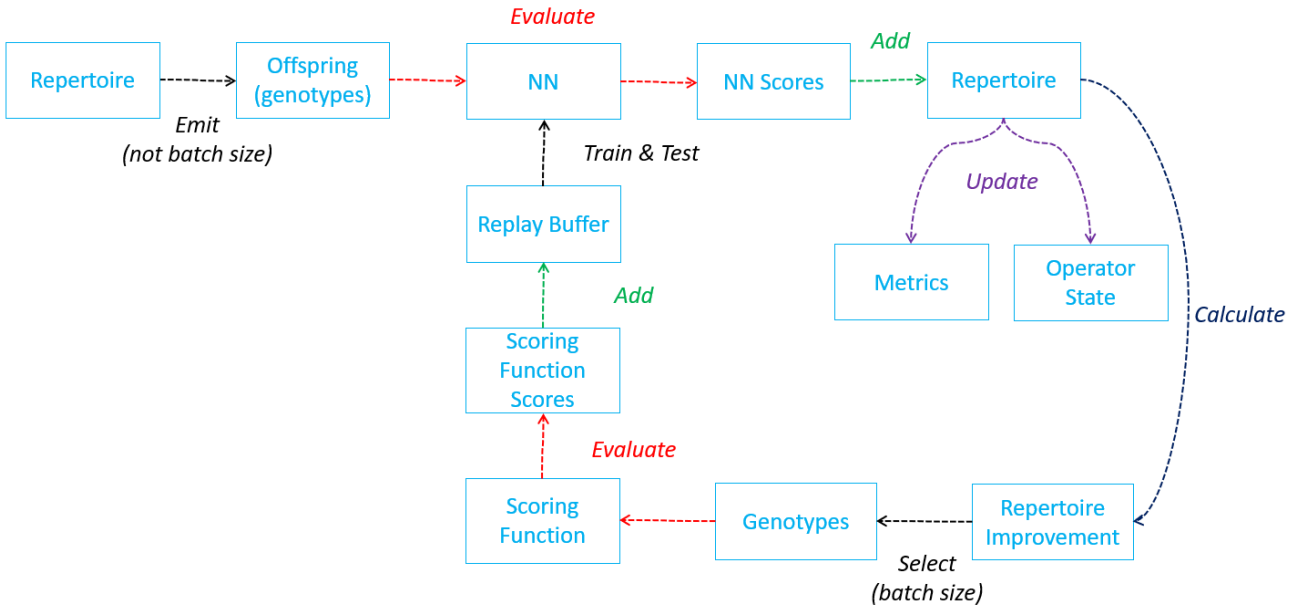


Figure 21: This diagram depicts an adapted version of MBME(UQ) algorithms after initialization. Contrary to the proposed contributions in this project, the idea is to always use the model to feed the repertoire. In addition, to populate the buffer, solutions are selected based on a new score: repertoire improvement (RI). The solutions with the highest RI, which are added to the repertoire with each generation, are re-evaluated by the scoring function to fill the buffer $\mathscr{D}$, which serves as a training and test set for the model. In addition, the number of solutions the model evaluates at each generation depends on the hyperparameter number of subdivisions, and the 3 mechanisms proposed throughout this project are also applicable to this new version.

References

1. Berthier L, Lim B, Manon F, and Cully A. Model-Based Uncertainty Quantification in the context of Reinforcement Learning and Quality-Diversity. 2023
2. Cully A and Demiris Y. Quality and Diversity Optimization: A Unifying Modular Framework. 2017 May 12. arXiv: 1708.09251[cs]. Available from: http://arxiv.org/abs/1708.09251 [Accessed on: 2023 Jan 8]
3. Chatzilygeroudis K, Cully A, Vassiliades V, and Mouret JB. Quality-Diversity Optimization: a novel branch of stochastic optimization. 2020 Dec 16. arXiv: 2012.04322[cs,math,stat]. Available from: http://arxiv.org/abs/2012.04322 [Accessed on: 2023 Jan 8]
4. Fu C, Sinou JJ, Zhu W, Lu K, and Yang Y. A state-of-the-art review on uncertainty analysis of rotor systems. Mechanical Systems and Signal Processing 2023; 183:109619. DOI: https://doi.org/10.1016/j.ymssp.2022.109619. Available from: https://www.sciencedirect.com/science/article/pii/S0888327022007087
5. Matott L, Babendreier J, and Purucker S. Evaluating Uncertainty in Integrated Environmental Models: A Review of Concepts and Tools, Supporting Information. Water Resources Research 2009 Jun; 45. DOI: 10.1029/2008WR007301
6. Flageat M and Cully A. Fast and stable MAP-Elites in noisy domains using deep grids. CoRR 2020; abs/2006.14253. arXiv: 2006.14253. Available from: https://arxiv.org/abs/2006.14253
7. Busch M, Schnoes F, Elsharkawy A, and Zaeh M. Methodology for model-based uncertainty quantification of the vibrational properties of machining robots. Robotics and Computer-Integrated Manufacturing 2022 Feb; 73:102243. DOI: 10.1016/j.rcim.2021.102243
8. Laue V, Schmidt O, Dreger H, Xie X, Röder F, Schenkendorf R, Kwade A, and Krewer U. Model-Based Uncertainty Quantification for the Product Properties of Lithium-Ion Batteries. Energy Technology 2020 Feb. DOI: 10.1002/ente.201900201
9. Alvarez A, Dahlskog S, Font JM, and Togelius J. Empowering Quality Diversity in Dungeon Design with Interactive Constrained MAP-Elites. CoRR 2019; abs/1906.05175. arXiv: 1906.05175. Available from: http://arxiv.org/abs/1906.05175
10. Guerrero-Romero C, Lucas S, and Perez Liebana D. Beyond Playing to Win: Creating a Team of Agents with Distinct Behaviours for Automated Gameplay. IEEE Transactions on Games 2023 Jan; PP:1–14. DOI: 10.1109/TG.2023.3241864
11. Gallotta R, Arulkumaran K, and Soros LB. Preference-Learning Emitters for Mixed-Initiative Quality-Diversity Algorithms. IEEE Transactions on Games 2023 :1–14. DOI: 10.1109/tg.2023.3264457. Available from: https://doi.org/10.1109%2Ftg.2023.3264457
12. Cully A, Clune J, and Mouret J. Robots that can adapt like natural animals. CoRR 2014; abs/1407.3501. arXiv: 1407.3501. Available from: http://arxiv.org/abs/1407.3501
13. Allard M, Smith SC, Chatzilygeroudis K, and Cully A. Hierarchical quality-diversity for online damage recovery. *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2022 Jul. DOI: 10.1145/3512290.3528751. Available from: https://doi.org/10.1145%2F3512290.3528751

14. Wickman R, Poudel B, Villarreal M, Zhang X, and Li W. Efficient Quality-Diversity Optimization through Diverse Quality Species. 2023. arXiv: 2304.07425 [cs.LG]

15. Gaier A, Asteroth A, and Mouret JB. Aerodynamic Design Exploration through Surrogate-Assisted Illumination. 2017 Jun. DOI: 10.2514/6.2017-3330

16. Hagg A, Asteroth A, and Bäck T. Prototype Discovery using Quality-Diversity. 2018. arXiv: 1807.09488 [cs.NE]

17. Hagg A, Kliemank ML, Asteroth A, Wilde D, Bedrunka MC, Foysi H, and Reith D. Efficient Quality Diversity Optimization of 3D Buildings through 2D Pre-optimization. Evolutionary Computation 2023 Apr :1–21. DOI: 10.1162/evco_a_00326. eprint: https://direct.mit.edu/evco/article-pdf/doi/10.1162/evco_a_00326/2078507/evco_a_00326.pdf. Available from: https://doi.org/10.1162/evco%5C_a%5C_00326

18. Fontaine M and Nikolaidis S. A Quality Diversity Approach to Automatically Generating Human-Robot Interaction Scenarios in Shared Autonomy. 2021. arXiv: 2012.04283 [cs.RO]

19. Bhatt V, Nemlekar H, Fontaine MC, Tjanaka B, Zhang H, Hsu YC, and Nikolaidis S. Surrogate Assisted Generation of Human-Robot Interaction Scenarios. 2023. arXiv: 2304.13787 [cs.RO]

20. Masuda N and Saito D. Quality Diversity for Synthesizer Sound Matching. *2021 24th International Conference on Digital Audio Effects (DAFx)*. 2021 Sep :300–7. DOI: 10.23919/DAFx51585.2021.9768271

21. Justesen N, Risi S, and Mouret JB. MAP-Elites for Noisy Domains by Adaptive Sampling. *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. GECCO '19. Prague, Czech Republic: Association for Computing Machinery, 2019 :121–2. DOI: 10.1145/3319619.3321904. Available from: https://doi.org/10.1145/3319619.3321904

22. Flageat M, Chalumeau F, and Cully A. Empirical analysis of PGA-MAP-Elites for Neuroevolution in Uncertain Domains. 2022 Oct 24. arXiv: 2210.13156[cs]. Available from: http://arxiv.org/abs/2210.13156 [Accessed on: 2023 Jan 15]

23. Flageat M and Cully A. Uncertain Quality-Diversity: Evaluation methodology and new methods for Quality-Diversity in Uncertain Domains. 2023 Feb 1. arXiv: 2302.00463[cs]. Available from: http://arxiv.org/abs/2302.00463 [Accessed on: 2023 Feb 6]

24. Chua K, Calandra R, McAllister R, and Levine S. Deep Reinforcement Learning in a Handful of Trials using Probabilistic Dynamics Models. 2018 Nov 2. arXiv: 1805.12114[cs,stat]. Available from: http://arxiv.org/abs/1805.12114 [Accessed on: 2023 Jan 8]

25. Abdar M, Pourpanah F, Hussain S, Rezazadegan D, Liu L, Ghavamzadeh M, Fieguth P, Cao X, Khosravi A, Acharya UR, Makarenkov V, and Nahavandi S. A review of uncertainty quantification in deep learning: Techniques, applications and challenges. Information Fusion 2021; 76:243–97. DOI: https://doi.org/10.1016/j.inffus.2021.05.008. Available from: https://www.sciencedirect.com/science/article/pii/S1566253521001081

26. Sambyal AS, Krishnan NC, and Bathula DR. Towards Reducing Aleatoric Uncertainty for Medical Imaging Tasks. 2021. DOI: 10.48550/ARXIV.2110.11012. Available from: https://arxiv.org/abs/2110.11012

27. Tuna OF, Catak FO, and Eskil MT. Exploiting epistemic uncertainty of the deep learning models to generate adversarial samples. 2021. arXiv: 2102.04150 [cs.LG]

28. Mouret JB and Clune J. Illuminating search spaces by mapping elites. 2015 Apr 19. arXiv: 1504.04909[cs,q-bio]. Available from: http://arxiv.org/abs/1504.04909 [Accessed on: 2023 Feb 20]

29. Vassiliades V, Chatzilygeroudis K, and Mouret JB. Using Centroidal Voronoi Tessellations to Scale Up the Multi-dimensional Archive of Phenotypic Elites Algorithm. 2017. arXiv: 1610.05729 [cs.NE]

30. Goldberg DE and Deb K. A Comparative Analysis of Selection Schemes Used in Genetic Algorithms. Ed. by RAWLINS GJ. Vol. 1. Foundations of Genetic Algorithms. Elsevier, 1991 :69–93. DOI: https://doi.org/10.1016/B978-0-08-050684-5.50008-2. Available from: https://www.sciencedirect.com/science/article/pii/B9780080506845500082

31. Lehman J and Stanley KO. Evolving a Diversity of Virtual Creatures through Novelty Search and Local Competition. *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*. GECCO '11. Dublin, Ireland: Association for Computing Machinery, 2011 :211–8. DOI: 10.1145/2001576.2001606. Available from: https://doi.org/10.1145/2001576.2001606

32. Nilsson O and Cully A. Policy Gradient Assisted MAP-Elites. *Proceedings of the Genetic and Evolutionary Computation Conference*. GECCO '21. Lille, France: Association for Computing Machinery, 2021 :866–75. DOI: 10.1145/3449639.3459304. Available from: https://doi.org/10.1145/3449639.3459304

33. Fontaine MC, Togelius J, Nikolaidis S, and Hoover AK. Covariance matrix adaptation for the rapid illumination of behavior space. *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*. ACM, 2020 Jun. DOI: 10.1145/3377930.3390232. Available from: https://doi.org/10.1145%5C%2F3377930.3390232

34. Fontaine MC and Nikolaidis S. Differentiable Quality Diversity. 2021. arXiv: 2106.03894 [cs.AI]

35. Pierrot T, Richard G, Beguir K, and Cully A. Multi-objective quality diversity optimization. *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM, 2022 Jul. DOI: 10.1145/3512290.3528823. Available from: https://doi.org/10.1145%5C%2F3512290.3528823

36. Colas C, Madhavan V, Huizinga J, and Clune J. Scaling MAP-Elites to deep neuroevolution. *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*. ACM, 2020 Jun. DOI: 10.1145/3377930.3390217. Available from: https://doi.org/10.1145%5C%2F3377930.3390217

37. Pierrot T and Flajolet A. Evolving Populations of Diverse RL Agents with MAP-Elites. 2023. arXiv: 2303.12803 [cs.NE]

38. Flageat M, Lim B, Grillotti L, Allard M, Smith SC, and Cully A. Benchmarking Quality-Diversity Algorithms on Neuroevolution for Reinforcement Learning. 2022 Nov 3. arXiv: 2211.02193[cs]. Available from: http://arxiv.org/abs/2211.02193 [Accessed on: 2023 Jan 15]

39. Cantu-Paz E. Adaptive Sampling for Noisy Problems. *Genetic and Evolutionary Computation – GECCO 2004*. Vol. 3102. 2004 Jan :947–58. DOI: 10.1007/978-3-540-24854-5_95

40. Rakshit P, Konar A, and Das S. Noisy evolutionary optimization algorithms – A comprehensive survey. Swarm and Evolutionary Computation 2017; 33:18–45. DOI: https://doi.org/10.1016/j.swevo.2016.09.002. Available from: https://www.sciencedirect.com/science/article/pii/S221065021630308X

41. Swazinna P, Udluft S, Hein D, and Runkler T. Comparing Model-free and Model-based Algorithms for Offline Reinforcement Learning. 2022. arXiv: 2201.05433 [cs.LG]

42. Yao Y, Ge D, Yu J, and Xie M. Model-Based Deep Transfer Learning Method to Fault Detection and Diagnosis in Nuclear Power Plants. Frontiers in Energy Research 2022 Mar; 10. DOI: 10.3389/fenrg.2022.823395

43. Tabatabaei M, Hakanen J, Hartikainen M, Miettinen K, and Sindhya K. A survey on handling computationally expensive multiobjective optimization problems using surrogates: non-nature inspired methods. Structural and Multidisciplinary Optimization 2015 Jul; 52. DOI: 10.1007/s00158-015-1226-z

44. Jegorova M, Doncieux S, and Hospedales TM. Generative Adversarial Policy Networks for Behavioural Repertoire. CoRR 2018; abs/1811.02945. arXiv: 1811.02945. Available from: http://arxiv.org/abs/1811.02945

45. Kent P and Branke J. BOP-Elites, a Bayesian Optimisation algorithm for Quality-Diversity search. 2020. arXiv: 2005.04320 [math.OC]

46. Kim S, Coninx A, and Doncieux S. From exploration to control: Learning object manipulation skills through novelty search and local adaptation. Robotics and Autonomous Systems 2021; 136:103710. DOI: https://doi.org/10.1016/j.robot.2020.103710. Available from: https://www.sciencedirect.com/science/article/pii/S0921889020305509

47. Keller L, Tanneberg D, Stark S, and Peters J. Model-Based Quality-Diversity Search for Efficient Robot Learning. CoRR 2020; abs/2008.04589. arXiv: 2008.04589. Available from: https://arxiv.org/abs/2008.04589

48. Lim B, Grillotti L, Bernasconi L, and Cully A. Dynamics-Aware Quality-Diversity for Efficient Learning of Skill Repertoires. *2022 International Conference on Robotics and Automation (ICRA)*. 2022 :5360–6. DOI: 10.1109/ICRA46639.2022.9811559

49. Lim B, Flageat M, and Cully A. Efficient Exploration using Model-Based Quality-Diversity with Gradients. 2022. arXiv: 2211.12610 [cs.NE]

50. Valdenegro-Toro M and Saromo D. A Deeper Look into Aleatoric and Epistemic Uncertainty Disentanglement. 2022. arXiv: 2204.09308 [cs.LG]

51. Filos A, Vértes E, Marinho Z, Farquhar G, Borsa D, Friesen AL, Behbahani FMP, Schaul T, Barreto A, and Osindero S. Model-Value Inconsistency as a Signal for Epistemic Uncertainty. CoRR 2021; abs/2112.04153. arXiv: 2112.04153. Available from: https://arxiv.org/abs/2112.04153

52. Kwon Y, Won JH, Kim BJ, and Paik MC. Uncertainty quantification using Bayesian neural networks in classification: Application to biomedical image segmentation. Computational Statistics & Data Analysis 2020; 142:106816. DOI: https://doi.org/10.1016/j.csda.2019.106816. Available from: https://www.sciencedirect.com/science/article/pii/S016794731930163X

53. Kendall A and Gal Y. What Uncertainties Do We Need in Bayesian Deep Learning for Computer Vision? CoRR 2017; abs/1703.04977. arXiv: 1703.04977. Available from: http://arxiv.org/abs/1703.04977

54. Böhm V, Lanusse F, and Seljak U. Uncertainty Quantification with Generative Models. 2019. arXiv: 1910.10046 [stat.ML]

55. Sagar A. Uncertainty Quantification using Variational Inference for Biomedical Image Segmentation. 2021. arXiv: 2008.07588 [eess.IV]

56. Akrami H, Joshi A, Aydore S, and Leahy R. Quantile Regression for Uncertainty Estimation in VAEs with Applications to Brain Lesion Detection. Vol. 12729. 2021 Jun :689–700. DOI: 10.1007/978-3-030-78191-0_53

57. Curi S, Berkenkamp F, and Krause A. Efficient Model-Based Reinforcement Learning through Optimistic Policy Search and Planning. CoRR 2020; abs/2006.08684. arXiv: 2006.08684. Available from: https://arxiv.org/abs/2006.08684

58. Lim B, Allard M, Grillotti L, and Cully A. Accelerated Quality-Diversity for Robotics through Massive Parallelism. CoRR 2022; abs/2202.01258. arXiv: 2202.01258. Available from: https://arxiv.org/abs/2202.01258

59. LeNail A. NN-SVG: Publication-Ready Neural Network Architecture Schematics. Journal of Open Source Software 2019; 4:747. DOI: 10.21105/joss.00747. Available from: https://doi.org/10.21105/joss.00747

60. Smith SC, Lim B, Janmohamed H, and Cully A. Quality-Diversity Optimisation on a Physical Robot Through Dynamics-Aware and Reset-Free Learning. *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*. ACM, 2023 Jul. DOI: 10.1145/3583133.3590625. Available from: https://doi.org/10.1145%2F3583133.3590625

61. Kingma DP and Ba J. Adam: A Method for Stochastic Optimization. 2017. arXiv: 1412.6980 [cs.LG]

62. Cully A and Demiris Y. Hierarchical Behavioral Repertoires with Unsupervised Descriptors. CoRR 2018; abs/1804.07127. arXiv: 1804.07127. Available from: http://arxiv.org/abs/1804.07127

63. Kurtzer G, Sochat V, and Bauer M. Singularity: Scientific containers for mobility of compute. PLoS ONE 2017 May; 12. DOI: 10.1371/journal.pone.0177459

64. Freeman CD, Frey E, Raichuk A, Girgin S, Mordatch I, and Bachem O. Brax – A Differentiable Physics Engine for Large Scale Rigid Body Simulation. 2021. arXiv: 2106.13281 [cs.RO]

Appendices

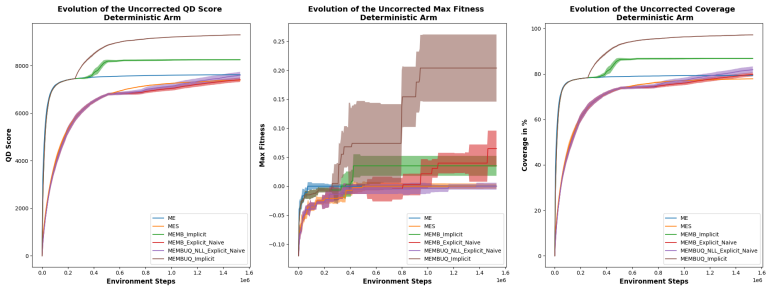
A Ethics Checklist

	Yes	No
Section 1: HUMAN EMBRYOS/FOETUSES		
Does your project involve Human Embryonic Stem Cells?		+
Does your project involve the use of human embryos?		+
Does your project involve the use of human foetal tissues / cells?		+
Section 2: HUMANS		
Does your project involve human participants?		+
Section 3: HUMAN CELLS / TISSUES		
Does your project involve human cells or tissues? (Other than from "Human Embryos/Foetuses" i.e. Section 1)?		+
Section 4: PROTECTION OF PERSONAL DATA		
Does your project involve personal data collection and/or processing?		+
Does it involve the collection and/or processing of sensitive personal data (e.g. health, sexual lifestyle, ethnicity, political opinion, religious or philosophical conviction)?		+
Does it involve processing of genetic information?		+
Does it involve tracking or observation of participants? It should be noted that this issue is not limited to surveillance or localization data. It also applies to Wan data such as IP address, MACs, cookies etc.		+
Does your project involve further processing of previously collected personal data (secondary use)? For example Does your project involve		+
Section 5: ANIMALS		
Does your project involve animals?		+
Section 6: DEVELOPING COUNTRIES		
Does your project involve developing countries?		+
If your project involves low and/or lower-middle income countries, are any benefit-sharing actions planned?		+
Could the situation in the country put the individuals taking part in the project at risk?		+
Section 7: ENVIRONMENTAL PROTECTION AND SAFETY		
Does your project involve the use of elements that may cause harm to the environment, animals or plants?		+
Does your project deal with endangered fauna and/or flora /protected		+
Does your project involve the use of elements that may cause harm to humans, including project staff?		+
Does your project involve other harmful materials or equipment, e.g. high-powered laser systems?		+
Section 8: DUAL USE		
Does your project have the potential for military applications?		+
Does your project have an exclusive civilian application focus?		+
Will your project use or produce goods or information that will require export licenses in accordance with legislation on dual use items?		+
Does your project affect current standards in military ethics – e.g., global ban on weapons of mass destruction, issues of proportionality, discrimination of combatants and accountability in drone and autonomous robotics developments, incendiary or laser weapons?		+

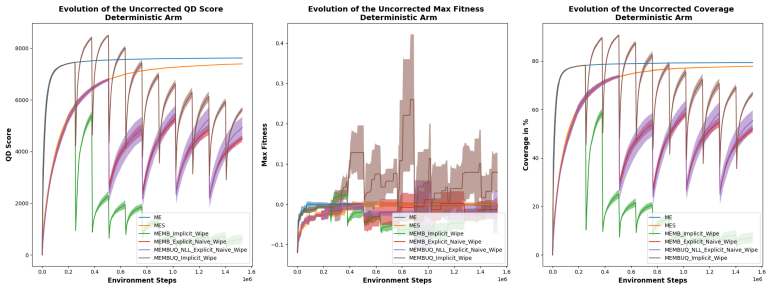
Section 9: MISUSE		
Does your project have the potential for malevolent/criminal/terrorist		+
Does your project involve information on/or the use of biological-, chemical-, nuclear/radiological-security sensitive materials and explosives, and means of their delivery?		+
Does your project involve the development of technologies or the creation of information that could have severe negative impacts on human rights standards (e.g. privacy, stigmatization, discrimination), if		+
Does your project have the potential for terrorist or criminal abuse e.g. infrastructural vulnerability studies, cybersecurity related project?		+
SECTION 10: LEGAL ISSUES		
Will your project use or produce software for which there are copyright licensing implications?		+
Will your project use or produce goods or information for which there are data protection, or other legal implications?		+
SECTION 11: OTHER ETHICS ISSUES		
Are there any other ethics issues that should be taken into		+

Figure 22: Questions and answers on the ethical and societal implications of this research project.

B Uncorrected Metrics

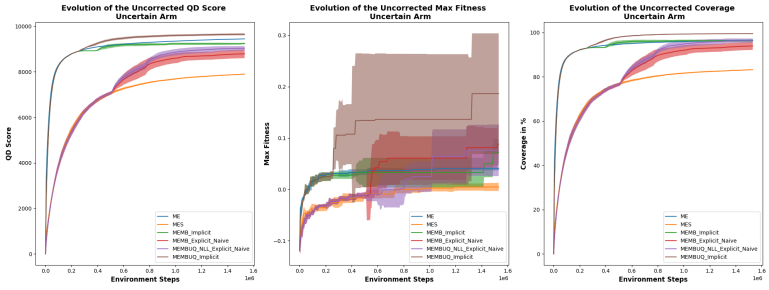


(a) Comparison plot between the 2 baselines and the 4 MBME(UQ) versions **without repertoire reset**.

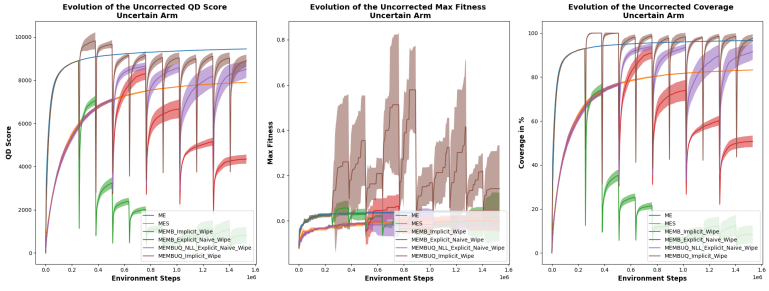


(b) Comparison plot between the 2 baselines and the 4 MBME(UQ) versions **with repertoire reset**.

Figure 23: Comparison plot of **3 uncorrected metrics**: **QD score** (left), **max fitness** (middle) and **coverage** (right) with the **deterministic arm** task and 10 algorithms. Results are obtained via **5 replications**, shaded areas correspond to **standard deviations** and dark lines correspond to **medians**.



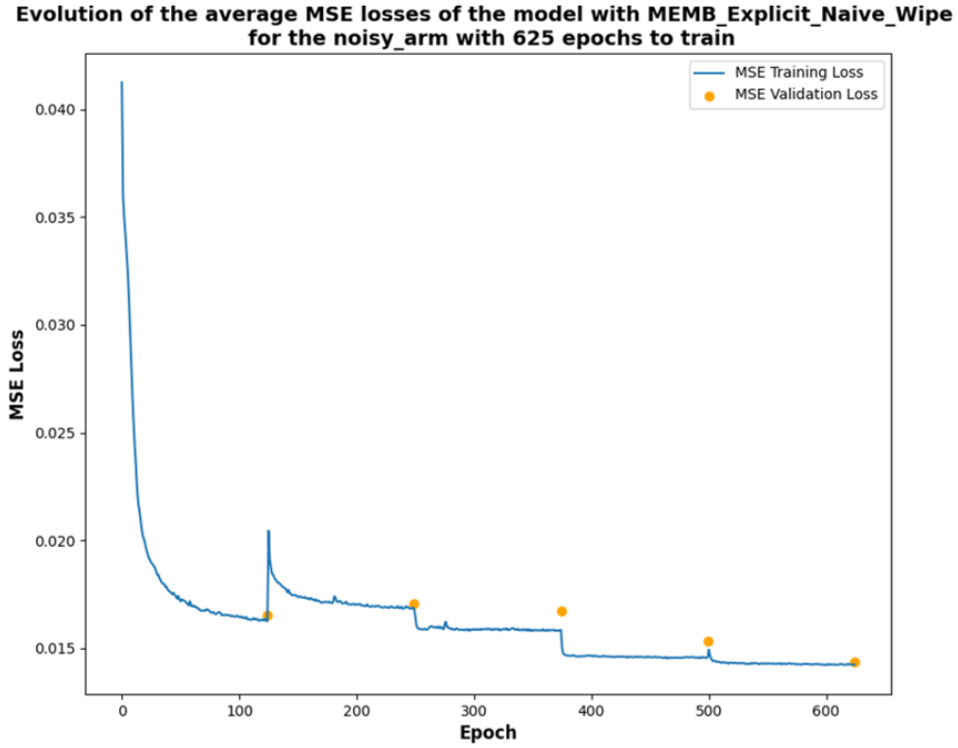
(a) Comparison plot between the 2 baselines and the 4 MBME(UQ) versions **without repertoire reset**.



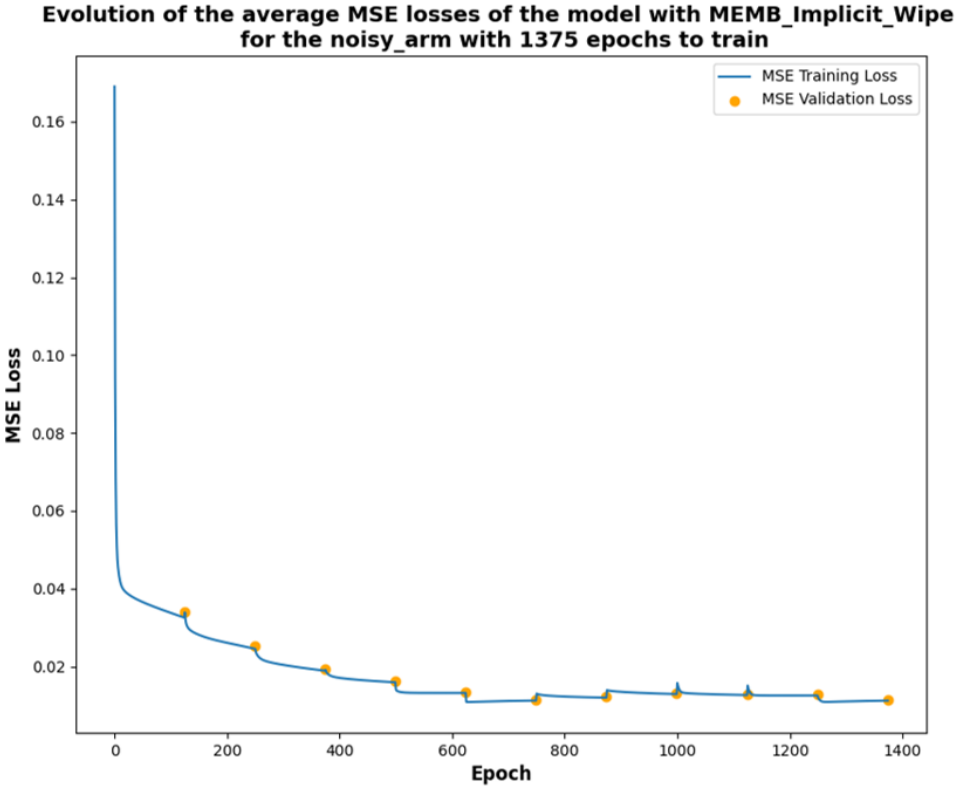
(b) Comparison plot between the 2 baselines and the 4 MBME(UQ) versions **with repertoire reset**.

Figure 24: Comparison plot of **3 uncorrected metrics**: **QD score** (left), **max fitness** (middle) and **coverage** (right) with the **uncertain arm** task and 10 algorithms. Results are obtained via **5 replications**, shaded areas correspond to **standard deviations** and dark lines correspond to **medians**.

C Learning Curves



(a) MBMEW Explicit learning curves.



(b) MBMEW Implicit learning curves.

Figure 25: Learning curves for the **uncertain arm** task with **MBMEW algorithms**.