Quality Diversity Optimization: A Modular Framework and Continuous Density Estimation

By

David Haolong Lee

Submitted to the University of Southern California in partial fulfillment

of graduation requirements for honors degree

Viterbi School of Engineering

Thomas Lord Department of Computer Science

Spring 2024

Research Advisor: Prof. Stefanos Nikolaidis

Signature: _____

Honors Advisor: Prof. Sandeep Gupta

Signature: _____

Contents

1 Abstract **4**

2 Introduction **5**

2.1 Quality Diversity Optimization 5

2.2 Framework for Quality Diversity Optimization 9

2.3 Leveraging Density Estimation in Diversity Optimization 10

3 Problem Definitions **13**

4 Existing Algorithms **14**

5 pyribs: A Bare-Bones Python Library for Quality Diversity Optimization **18**

5.1 Design Principles 18

5.2 The RIBS Framework 20

5.3 Composing Algorithms in RIBS 23

5.4 Comparison to Existing QD Libraries 26

5.5 Summary 30

6 Density Descent in Diversity Optimization **31**

6.1 Density Estimation Methods 31

6.2 Algorithmic Details 33

6.3 Connection between Novelty Search and Kernel Density Estimates 35

6.3.1 Stability Under Non-stationarity 36

6.3.2 Equivalence of NS and DDS-KDE 38

6.4 Reservoir Sampling for Maintaining a Representative Buffer 38

6.5 Experiments 39

6.5.1 Experimental Design 40

6.5.2 Domain Details 41
6.5.3 Statistical Analysis 43
6.6 Bandwidth Selection for DDS-KDE 45
6.7 Summary of Results 47

7 Conclusion **48**

A Tukey's HSD Pairwise Comparisons **49**

B Proofs **50**

List of Figures

1 Adapted from [8]. 7

1a The layout in the deceptive maze domain. The goal is to find a neural network that navigates from the start (the bigger circle) to the end (the smaller circle). Red line traces a path through the maze. 7

1b Final positions reached by the robot when directly minimizing the distance to the goal location. 7

1c Final positions reached by the robot when applying diversity optimization (Novelty Search) to attempt to search for neural networks that reach every position in the maze. Notice that, despite ignoring the proximity of the robot to the goal location, more robots ends up finding the goal location. . . 7

2 The trajectories of the lunar rover approaching from three different directions, recreated from videos in pyribs tutorials [10]. The objective here is to minimize the impact velocity. The feature of each rover is determined by where they first contact the ground. 7

2a Lunar rover approaching from the **left** of the landing site. 7

2b Lunar rover approaching from **above** the landing site. 7

2c Lunar rover approaching from the **right** of the landing site. 7

3 Result of latent space illumination for objective “A photo of Beyonce” and for features “A small child” and “A woman with long blonde hair.” The axes values indicate the score returned by the CLIP model, where lower score indicates a better match between the text and the image. Adapted from [11]. 8

4 Pyribs implements the RIBS framework for QD optimization. The user first `ask()`'s for solutions from a *scheduler*. The scheduler selects *emitters* to `ask()` for solutions and returns the solutions to the user. After evaluating the solutions, the user `tell()`'s the results to the scheduler. The scheduler `add()`'s the solutions to the *archive* and receives information that it `tell()`'s to the emitters, enabling the emitters to update their internal search state. 19

5 Pyribs visualization tools. This figures show example 2D heatmaps, where the axes correspond to the feature values, and the color of each archive cell indicates its objective value. In `SlidingBoundariesArchive`, the points show the locations of solutions in feature space, and the lines show the grid boundaries. We also show a *parallel axes plot* which can visualize an archive of any dimensionality. In this plot, a single solution's features are plotted as a line connecting the feature $\phi_1 \dots \phi_k$, and the line is colored according to the solution's objective value. Adapted from [1]. 27

6 Tutorials enable pyribs users to quickly learn about the library and experiment with problems from the QD literature. Adapted from [1]. 28

7 The effect of the bandwidth h on a one-dimensional KDE. Red dots represent the data, and black lines depict the KDE. When h is too small, the KDE reveals many misleading local maxima. When h is too large, the KDE conceals modes from the underlying distribution. Figure adapted from [2] 32

8 Overview of density descent search (DDS) for solving diversity optimization (DO) problems. DDS first draws solutions from a Gaussian $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. After computing the solution features (in this case, the final position of the robot in a maze), DDS ranks solutions by density. This density ranking is passed to CMA-ES, which updates the search distribution to sample solutions with lower density on the next iteration. Concurrently, solutions are stored in a buffer that forms the basis for density estimates, and in a passive archive that tracks all discovered solutions. Adapted from [2]. 33

9 An example of an arm configuration for the Arm Repertoire domain. Note that while this arm has only 12 degrees of freedom, the arms in the experiments of this thesis has 100 degrees of freedom. Figure adapted from [81]. 42

10 Coverage and cross-entropy (CE) after 5,000 iterations of each algorithm in all domains. We report the mean over 10 trials, with error bars showing the standard error of the mean. Higher coverage and lower cross-entropy are better. 43

11 Coverage and cross-entropy (CE) after 5,000 iterations of DDS-KDE in all domains for each normalized bandwidth h_0. We report the mean over 10 trials (3 trials for Deceptive Maze) with error bars showing the standard error of the mean. Higher coverage and lower cross-entropy are better. Highlighted cells are results from the main experiments in Fig. 10. The plots show the normalized bandwidth on the x-axis 46

List of Tables

1 By selecting different components in the RIBS framework, we can compose a variety of recent algorithms from the QD literature and test them in pyribs. Furthermore, we can identify combinations of components which may lead to new algorithms. Adapted from [1]. 23

2 One-way ANOVA results in each domain. All p-values are less than 0.001. 43

3 Pairwise comparisons for **coverage** in each domain. Each entry compares the method in the row to the method in the column. For instance, we can see that DDS-KDE had significantly higher coverage than NS in LP. Comparisons were performed with Tukey's HSD test. $>$ indicates significantly greater, $<$ indicates significantly less, and $-$ indicates no significant difference. $\varnothing$ indicates an invalid comparison. 49

4 Pairwise comparisons for **cross-entropy** in each domain. Note that lower cross-entropy is better, so significantly less ($<$) indicates that a method is significantly better. 49

1 Abstract

Quality diversity optimization seeks to discover a set of high-quality solutions that elicit diverse features. My thesis contributes to this field by introducing a modular framework to understand quality diversity algorithms. I utilized the framework to develop an improved algorithm incorporating insight from density estimation techniques. Researchers have developed many quality diversity algorithms incorporating insights from diverse fields such as reinforcement learning, architecture, human-robot interaction, and scenario generation. To account for the growing array of quality diversity algorithms, I developed a flexible modular framework that encapsulates this diverse array of algorithms. Since its conception, this framework has been utilized in at least 20 research papers. Furthermore, I proposed a novel algorithm (which conforms to the modular framework) based on density estimation techniques that outperforms existing baselines algorithms. I compare my algorithm, Density Descent Search (DDS), with existing quality diversity algorithms in empirical experiments and theoretical analysis. The empirical experiments reveal the superior performance of DDS, and the theoretical analysis helps explain the improved performance of DDS over other algorithms.

2 Introduction

This thesis presents research on quality diversity optimization (QD) and diversity optimization (DO) completed at the Interactive and Collaborative Autonomous Robotics (ICAROS) lab at University of Southern California. This thesis describes research related to

(a) *conceptualizing an algorithmic framework for quality diversity* to illustrate how the framework addresses the problem of a growing numbers of quality diversity algorithms and how the framework informed the development of a software library, pyribs, to support implementing novel quality diversity algorithms for both users and researchers [1].

(b) *conceptualizing an algorithm for diversity optimization, Density Descent Search,* to illustrate the crucial role that continuous density estimation plays in efficiently exploring the feature space [2].

The algorithmic framework (RIBS) and software library (pyribs) [1] played an important role in designing and prototyping Density Descent Search [2]. The algorithmic structure of Density Descent Search closely resembles that of the model proposed in the algorithmic framework. Moreover, the algorithm itself and all baselines algorithms that it was compared to are all implemented with the pyribs library.

2.1 Quality Diversity Optimization

Quality diversity optimization is a generalization of single-objective optimization. Single-objective optimization is an ubiquitous problem solving technique that has applications spanning from path-finding in a maze [3] to landing a lunar rover [4] to generating realistic images of celebrities [5]. In single-objective optimization problems, the goal is to find a single solution that optimizes an objective function $f(\cdot)$. Instead of seeking the single optimal solution, QD searches for a set of behaviorally diverse solutions where each solution is optimal among the solutions with the same behaviors. In other words, QD searches of a set of *high-quality* solutions that exhibits *diverse*

(a) The layout in the deceptive maze domain. The goal is to find a neural network that navigates from the start (the bigger circle) to the end (the smaller circle). Red line traces a path through the maze.

(b) Final positions reached by the robot when directly minimizing the distance to the goal location.

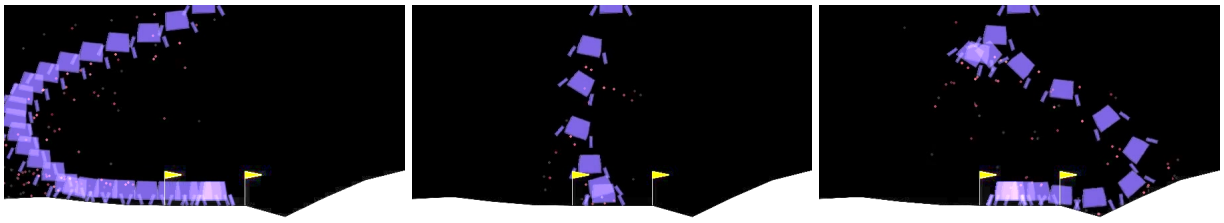
(c) Final positions reached by the robot when applying diversity optimization (Novelty Search) to attempt to search for neural networks that reach every position in the maze. Notice that, despite ignoring the proximity of the robot to the goal location, more robots ends up finding the goal location.

Figure 1: Adapted from [8].

features.

The problem of finding a range of solutions that are diverse with respect to pre-specified features is referred to as diversity optimization (DO). *DO can be characterized as a special instance of QD where the objective value of all solutions is constant [6].* It is important to note the close connection between QD and DO algorithms: "When the objective value of all solutions is constant, [a QD algorithm] will be indifferent to the quality of the solutions and therefore only optimize for diversity" [2]. Thus, some researchers consider QD or DO individually and then leverage the insights on one problem to solve the other one more effectively. An example of this is Novelty Search [3], a DO algorithm, and Novelty Search with Local Competition [7], a QD algorithm based on Novelty Search. A more formal description of QD and DO are presented in Sec. 3.

QD presents many advantages over single-objective optimization. QD can mitigate convergence to local optima, enhance robustness of the algorithm, and improve control over the output of the algorithm. I will illustrate the benefits of QD by extending the aforementioned single-objective optimization problems — path-finding in a maze, landing a lunar rover, and generating realistic images of celebrities — to QD problems.



(a) Lunar rover approaching from the **left** of the landing site.

(b) Lunar rover approaching from **above** the landing site.

(c) Lunar rover approaching from the **right** of the landing site.

Figure 2: The trajectories of the lunar rover approaching from three different directions, recreated from videos in pyribs tutorials [10]. The objective here is to minimize the impact velocity. The feature of each rover is determined by where they first contact the ground.

Convergence to local optima present many challenges in single-objective optimization. In contrast, searching directly for behavioral diversity may results in finding sub-optimal solutions that act as "stepping stones," mitigating convergence to local optima [9]. A classic example is the problem of training an agent (such as a neural network) to reach a target position in a *deceptive maze* [3]. An example of such maze is shown in Fig. 1a. Here, a single-objective optimizer that is directly minimizing the distance between the agent's final position and a target goal causes the agent to get stuck. This behavior is depicted in Fig. 1b: the majority of the robots ends up on the bottom-left and middle-left of the maze, as these regions are *deceptively* close to the goal location. The problem can be solved by ignoring the objective of directly reaching the target location and instead attempting to find a diverse range of agents, each of which reaches a different region of the maze (Fig. 1c).

QD algorithms are more robust than single-objective algorithms, as they can adapt to unexpected changes in the environment. This is a reasonable consideration for real world models/algorithms that are trained on simulations, since the real world may introduce variables or factors that are unaccounted for in the simulation. For instance, consider the task of landing a lunar rover, where the objective is to minimize the impact velocity of the rover subject to the hard constraint that the rover's final position must be with in the boundary of the landing site. A single-objective algorithm will only find one trajectory that lands the rover. In contrast, QD algorithms can search for a diverse set of trajectories that approach the landing site from different directions, in addition to minimizing the impact velocity of these trajectories (Fig. 2). This is useful when certain sudden changes in the

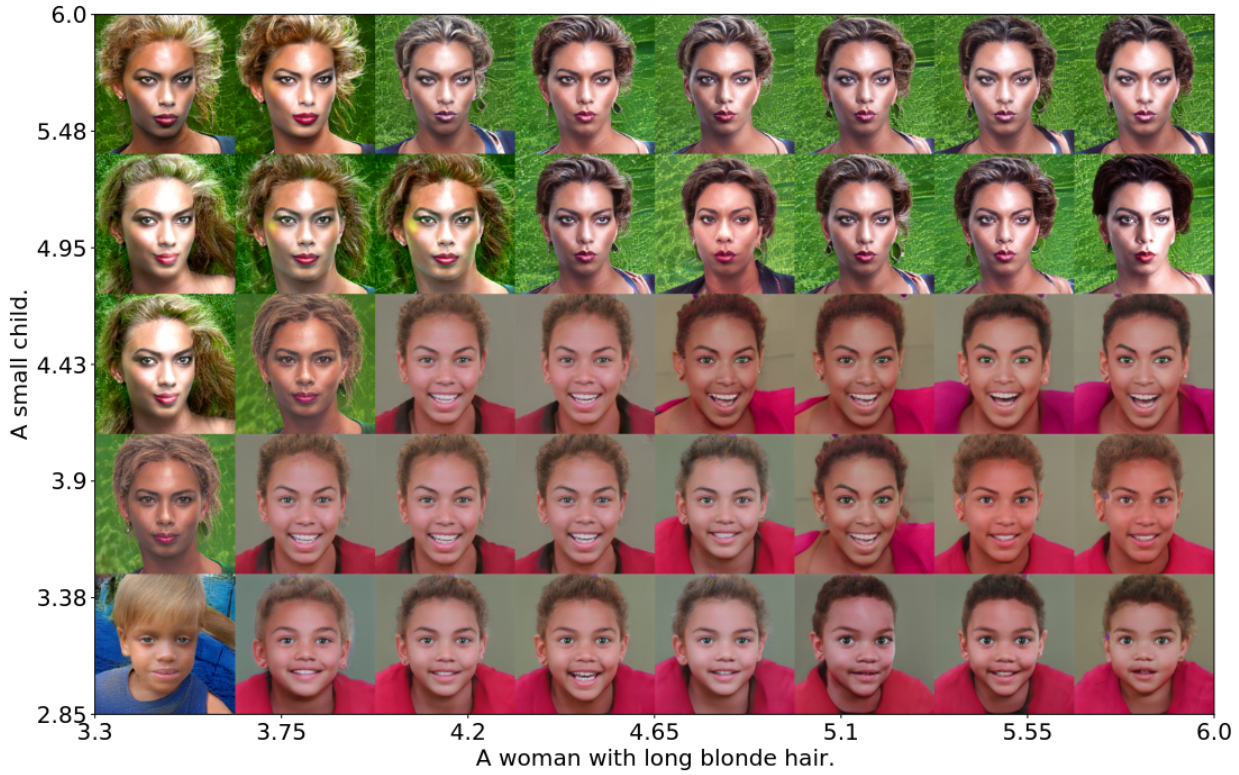


Figure 3: Result of latent space illumination for objective "A photo of Beyonce" and for features "A small child" and "A woman with long blonde hair." The axes values indicate the score returned by the CLIP model, where lower score indicates a better match between the text and the image. Adapted from [11].

environment makes approaching from certain directions less desirable.

QD algorithm can be combined with existing single-objective algorithms directly to improve control over the output of the algorithm. The most prominent example of this is Latent Space Illumination [11], in which QD algorithm explores the latent space of a generative model to produce images exhibiting speicifc features. For example, in Fig. 3, the objective function to maximize is the text-to-image similarity between the generated image and the text prompt "A photo of Beyonce" as returned by CLIP (Contrastive Language-Image Pre-Training) [12]. The features are the text-to-image similarity between the generated image and "A woman with long blode hair" and "A small child," respectively.

These examples are implemented in pyribs [1], a Python library for implementing QD algorithms and experiments. Executable code and a detailed walkthrough of the lunar lander and image

generation example can be found on pyribs's website (at [10] and [13], respectively).

The three applications of QD presented here demonstrates the benefits that QD can bring to versatile fields of research. Indeed, QD has grown to become a general-purpose optimization paradigm with applications in a number of diverse areas [1]. As of writing, there are at least 238 papers on the topic [14], spanning areas such as reinforcement learning [15–20], robot manipulation [21, 22], human-robot interaction [23–25], video game level generation [26, 27], agent testing [28], urban planning [29], design [30], internet congestion control [31], and drug compound discovery [32]. QD has also moved outside of publications and into more popular forms of media like blog posts [33–39] and conference tutorials [40–43].

2.2 Framework for Quality Diversity Optimization

To account for the the growing number of QD algorithms and publication, my colleagues at the ICAROS lab and I conceptualized a modular framework, RIBS,[1] to represent the field's growing array of algorithms. Moreover, we developed pyribs [1], a highly modular library built on RIBS, that facilitates implementation of QD algorithms under RIBS.

To unify the many algorithms proposed by these works, the RIBS framework abstracts all QD algorithms into three components: the *archive*, the *emitters*, and the *scheduler* [1]. An additional advantage of this approach is that it facilitates the mixing of various components from different algorithms to create new algorithms.

Our work was published in "pyribs: A Bare-Bones Python Library for Quality Diversity Optimization" at The Genetic and Evolutionary Computation Conference (GECCO) 2023 [1]. The discussion of the details of the framework and relevant implementation choices will draw heavily from the published paper.

[1]The name "RIBS" stems from the title of [44], "Covariance Matrix Adaptation for the **R**apid **I**llumination of **B**ehavior **S**pace," which introduced the notions of emitters and schedulers.

2.3 Leveraging Density Estimation in Diversity Optimization

Continuous, stable approximations of the density in feature space can empower DO algorithms to explore the feature space more efficiently. To this end, my colleagues at the ICAROS lab and I proposed a new algorithm, density descent search (DDS), incorporating continuous density estimates. We realized that existing DO algorithms implicitly search the feature space according to the solution density. However, current approaches applies *discrete* or *unstable* density estimates, which hinders efficient exploration in the feature space. Thus, DDS leverages density estimation methods such as kernel density estimation (KDE) and continuous normalizing flow (CNF).

One common approach to DO problems is the Novelty Search (NS) algorithm [3]. NS retains an *archive* of previously found solutions, and aims to expand the archive by finding solutions that are far away in the feature space from existing solutions in the archive. Specifically, NS optimizes for solutions with a high *novelty score*, which is calculated as the average distance in feature space from a solution to its k-nearest neighbors in the archive.

Although novelty score is originally defined in terms of distance, it can be interpreted as an approximation of *density* [45]. Density is directly proportional to the number of solutions in a region of feature space. Generally, A high novelty score indicates that a solution's features are far from the features of its k-nearest neighbors in the archive, i.e., it is located in an area of the feature space with low density.

Another approach to DO is to apply QD algorithms and let all solutions in the search space have the same quality. Notably, A state-of-the-art QD algorithm is Covariance Matrix Adaptation MAP-Annealing (CMA-MAE) [46], which optimizes for archive improvement with the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [47] optimizer. CMA-MAE maintains a discrete archive (usually a grid) of solutions in feature space. When the solution qualities are identical, this archive becomes a histogram that represents the distribution of solutions in feature space [46]. Each cell in the archive becomes a bin of the histogram, with lower values of the bins indicating lower density of solutions. CMA-MAE then performs *density descent* on this histogram, where it continually

seeks to fill the areas of low density.

To efficiently explore feature space, both NS and CMA-MAE leverage density estimates of solutions in feature space — NS is guided by novelty score [3], while CMA-MAE performs density descent on its histogram [46]. However, both of these density estimates have their own drawbacks:

(a) The novelty score in NS is *continuous*, but provides a *weaker* stability guarantee, meaning that its value can change arbitrarily when new features are discovered.

(b) The histogram in CMA-MAE provides a *stronger* stability guarantee, but utilizes a *discrete* approximation that gives flat gradient signals on its bins.

To this end, I propose to utilize continuous and stable density estimation techniques such as kernel density estimation.

[46] also discusses the challenge of distortion in QD, which describes the situation where many solutions exhibit similar or identical features. With strong distortion, the optimizer does not receive enough signal to reach edge of the feature space since most solutions exhibit similar features. This is analogous to the challenge of performing gradient descent on a function that has equal objective value for most solutions. [46] mitigated this problem by using a more fine-grained grid to allow a much higher level of precision when measuring the diversity between two features. Our empirical results indicate that DDS is very resilient against distortion Sec. 6.5.3.

Our empirical experiment and theoretical analysis are published in the paper "Density Descent for Diversity Optimization" and will be appearing at The Genetic and Evolutionary Computation Conference (GECCO) 2024 [2]. The discussion of the details and performance of DDS will draw heavily from the published work.

3 Problem Definitions

Our problem definitions follows that of prior work [2, 3, 44].

Quality Diversity Optimization (QD). For solution $\boldsymbol{\theta} \in \mathbb{R}^n$, QD assumes an *objective* function $f : \mathbb{R}^n \to \mathbb{R}$ and m *feature* functions, jointly represented as a vector-valued function $\boldsymbol{\phi} : \mathbb{R}^n \to \mathbb{R}^m$. We refer to the image of $\boldsymbol{\phi}$ as the *feature space* S. The goal of QD is to find, for each feature $\boldsymbol{s} \in S$, a solution $\boldsymbol{\theta}$ where $f(\boldsymbol{\theta})$ is maximized and $\boldsymbol{\phi}(\boldsymbol{\theta}) = \boldsymbol{s}$ (i.e. the solution $\boldsymbol{\theta}$ exhibits feature $\boldsymbol{s}$).

Diversity Optimization (DO). DO is a special case of QD where the objective is constant, i.e., $f(\boldsymbol{\theta}) = C$. The goal of DO is to find, for each $\boldsymbol{s} \in S$, a solution $\boldsymbol{\theta}$ where $\boldsymbol{\phi}(\boldsymbol{\theta}) = \boldsymbol{s}$.

4 Existing Algorithms

Various methods address the QD and DO problems by relaxing them to the problem of finding an *archive* (i.e., a set) $\mathcal{A}$ of representative solutions. The structure of the archive defines two major families of algorithms: those based on Novelty Search and those based on MAP-Elites.

The literature review in this section draws heavily from the discussion of QD algorithms and non-stationarity outlined in [2].

Novelty Search (NS). The key insight of NS [3] is to discover diverse solutions in feature space by optimizing for solutions that are "novel" with respect to a current set of solutions. Given a set of features $\mathcal{X} \subseteq \mathbb{R}^m$, the novelty of a solution $\boldsymbol{\theta} \in \mathbb{R}^n$ and its features $\boldsymbol{y} = \boldsymbol{\phi}(\boldsymbol{\theta})$ relative to $\mathcal{X}$ is encapsulated in its *novelty score*, denoted $\rho(\boldsymbol{y}; \mathcal{X})$:

$$\rho(\boldsymbol{y}; \mathcal{X}) = \frac{1}{k} \sum_{\boldsymbol{y}' \in N_k(\boldsymbol{y}; \mathcal{X})} \mathrm{dist}(\boldsymbol{y}, \boldsymbol{y}') \tag{1}$$

where $N_k(\boldsymbol{y}; \mathcal{X})$ is the set of k-nearest neighbors to $\boldsymbol{y}$ in $\mathcal{X}$, and dist is a distance function.

NS gradually adds novel solutions to the archive $\mathcal{A}$ and explores the feature space. NS maintains the archive $\mathcal{A}$ of unbounded size and generates solutions with an underlying optimizer, traditionally a genetic algorithm. For each solution $\boldsymbol{\theta}$ produced by the optimizer, NS computes the novelty score with respect to an archive $\mathcal{A}$, i.e., $\rho(\boldsymbol{y}; \mathcal{A})$. If $\rho(\boldsymbol{y}; \mathcal{A})$ exceeds an *acceptance threshold*, then $(\boldsymbol{\theta}, \boldsymbol{y})$ is added to the archive.

Importantly, the novelty score is non-stationary, meaning that the novelty of a solution $\boldsymbol{\theta}$ changes as the archive $\mathcal{A}$ is updated. Thus, the shape of the novelty score function changes significantly throughout the QD search, which is not ideal for adaptive optimizers that adapts according to the shape of the function.

Multi-dimensional Archive of Phenotypic Elites (MAP-Elites). While NS was developed for DO, MAP-Elites [48] was developed for the general QD setting. Compared to the unstructured archive

of NS, MAP-Elites divides the feature space into a predefined tessellation $T : \mathbb{R}^m \to \{1, \ldots, l\}$, where $e \in \{1, \ldots, l\}$ is a *cell* in the tessellation and l is the total number of cells. Given a cell e, the MAP-Elites archive associates a solution $\boldsymbol{\theta}$ with cell e if and only if the solution's features are contained in e, i.e., $T(\boldsymbol{\phi}(\boldsymbol{\theta})) = e$. Moreover, MAP-Elites stores at most one solution for every cell in the tessellation.

During a QD search, MAP-Elites gradually collects high-performing solutions that have diverse features. MAP-Elites first draws solutions from a predefined distribution, e.g., a Gaussian, and inserts the solutions into the archive. Subsequently, MAP-Elites generates and inserts new solutions by mutating existing archive solutions with a genetic operator, such as the Iso+LineDD operator [49]:

$$\boldsymbol{\theta}' := \boldsymbol{\theta}_i + \sigma_1 \mathcal{N}(\mathbf{0}, \boldsymbol{I}) + \sigma_2 \mathcal{N}(0, 1)(\boldsymbol{\theta}_i - \boldsymbol{\theta}_j) \tag{2}$$

Importantly, when solutions inserted into the archive land in the same tessellation cell, MAP-Elites only retains the solution with the greatest objective value.

The choice of tessellation in MAP-Elites can significantly impact scalability. The most common tessellation is a grid tessellation [48], which divides the feature space into equally-sized hyperrectangles. In high-dimensional feature spaces, grid tessellations require exponentially more memory due to the curse of dimensionality. For example, a grid tessellation in a 10D feature space, with 5 cells along each dimension, requires 5^{10} cells. Thus, a common alternative is the centroidal Voronoi tessellation (CVT) [50], which divides the feature space into l evenly-sized polytopes.

Covariance Matrix Adaptation Evolution Strategy (CMA-ES). One recent line of QD algorithms has combined CMA-ES [47] with MAP-Elites. CMA-ES is a state-of-the-art single-objective optimizer that represents a population of solutions with a multivariate Gaussian $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Each iteration, CMA-ES draws λ solutions from the Gaussian. Based on the *rankings* (rather than the raw objectives) of the solutions, CMA-ES adapts the covariance matrix $\boldsymbol{\Sigma}$ to regions of higher-performing solutions. While CMA-ES is a derivative-free optimizer, it has been shown to approximate a natural gradient [51].

Covariance Matrix Adaptation MAP-Elites (CMA-ME). The first work to integrate CMA-ES with MAP-Elites was CMA-ME [44]. The key idea of CMA-ME is to optimize for archive improvement with CMA-ES. Namely, in addition to a MAP-Elites-style archive, CMA-ME maintains instances of CMA-ES. Each CMA-ES instance samples solutions from a Gaussian distribution, and the solutions are ranked based on how much they improve the archive, e.g., solutions that found new cells in the archive are ranked high, while those that were not added at all are ranked low. With this *improvement ranking*, CMA-ES adapts the Gaussian to sample solutions that will further improve the archive.

Mathematically, the ranking is a function $\pi : \{1, \dots, \lambda\} \to \{1, \dots, \lambda\}$ such that $\pi(i) \geq \pi(j)$ if and only if $\Delta_i \geq \Delta_j$ where Δ_i is the improvement of the archive by solution $\boldsymbol{\theta}_i$. Δ_i is defined as $f(\boldsymbol{\theta}_i)$ if the cell e corresponding to $\boldsymbol{\theta}_i$ is empty and $f(\boldsymbol{\theta}_i) - f(\boldsymbol{\theta}_e)$ is t_e is occupied by an incumbent $\boldsymbol{\theta}_e$. In DO settings, since all solutions have equal objective value, the improvement ranking simply places solutions that fills a new cell ahead of solutions that did not add a new cell. In other words, solutions are simply ranked based on whether or not they fill an empty e: $\Delta_i = C$ for a $\boldsymbol{\theta_i}$ such that $T(\boldsymbol{\phi}(\boldsymbol{\theta_i})) = \{\}$ (e is empty), and $\Delta_i = 0$ otherwise.

Covariance Matrix Adaptation MAP-Annealing (CMA-MAE). One limitation of CMA-ME is that it focuses too much on exploring for new cells in feature space rather than optimizing the objective [17]. CMA-MAE [46] addresses this problem by introducing an *archive learning rate* α. When $\alpha = 1$, CMA-MAE maintains the same exploration behavior as CMA-ME, but when $\alpha = 0$, CMA-MAE focuses solely on optimizing the objective, like CMA-ES. For $0 < \alpha < 1$, CMA-MAE blends between these two extremes, enabling it to both explore feature space and optimize the objective.

In DO settings, where the objective is constant ($f(\cdot) = C$), CMA-MAE naturally performs *density descent* [46]. According to [46], the archive becomes a histogram that models how many solutions have been found in each region of the feature space. A historgram is composed of many bins in a grid pattern; lower values in a bin indicate lower *density* of solutions, i.e., if a histogram bin has

a low value, few solutions have been discovered in that region of feature space. CMA-MAE then seeks to *descend* the histogram by searching for solutions that fill the low-valued histogram bins.

5 pyribs: A Bare-Bones Python Library for Quality Diversity Optimization

As a growing field of research in computer science, QD has had many libraries that attempted to encapsulate QD algorithms under a general framework. These libraries are important as they support ongoing research in the field, and help lower the barrier of entry for new members of the QD community. For instance, Sferes$_{v2}$ [52], QDpy [53], and QDax [54] have all been used in QD literature to implement and describe novel QD algorithms. However, these libraries each came short on at least one of the three criteria that our lab holds to be the core principles in designing a fundamental library for QD. These criteria are as follows: (i) sufficiently *simple* for new users and for modification, (ii) *accessible* to a wide audience, and (iii) sufficiently *flexible* to support concurrent and future research.

To this end, the first version of the RIBS framework was outlined in [44], and the first version of pyribs was released about a year later. After many iterations of development, the RIBS framework was fully conceptualized and implemented in pyribs [1].

I was responsible for implementing the fundamental modules that supported the RIBS framework and modeling QD algorithms under the RIBS framwork. I will describe the design principle that guided the development of pyribs (Sec. 5.1), expound on the RIBS framework in detail (Sec. 5.2), and comparing pyribs to existing QD libraries (Sec. 5.4).

5.1 Design Principles

Simple. pyribs was designed to be "bare-bones," maintaining only the core components that are required for a QD algorithm to optimize in a continuous feature space [1]. The simplicity of the design makes the library easier for new users to adopt, while the focus on continuous optimization problems reduces implementation complexity and makes the defined search space less abstract.

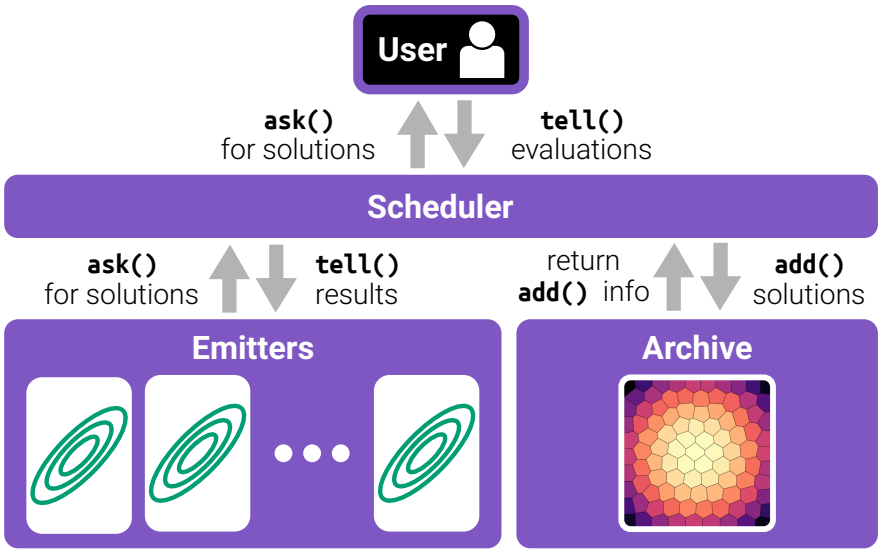


Figure 4: Pyribs implements the RIBS framework for QD optimization. The user first `ask()`'s for solutions from a *scheduler*. The scheduler selects *emitters* to `ask()` for solutions and returns the solutions to the user. After evaluating the solutions, the user `tell()`'s the results to the scheduler. The scheduler `add()`'s the solutions to the *archive* and receives information that it `tell()`'s to the emitters, enabling the emitters to update their internal search state.

Flexible. pyribs is also "bare-bones" in the sense that the core components of the library — archives, emitters, and schedulers — are all exposed to the user [1]. This allows users to easily exchange components of the QD algorithm. For example, by exchanging components, users can implement any of the algorithms listed in Tab. 1. Moreover, the design provides a foundation to implement future QD algorithms discovered by researchers, as the researchers can easily extend or rewirte the functionalities of individual components to achieve the outcome that they desire.

Accessible. From beginners to experienced researchers, pyribs is accessible to everyone. pyribs publishes readable source code, is easy to install, and has a comprehensive documentation website defining its usage [1]. The detailed documentation facilitates experienced researchers to extend the library or tamper with the source code. Moreover, pyribs's choices of simple dependency ensure that installing pyribs and studying the tutorials is straightforward and frictionless even for beginners with limited computational resources and/or basic hardwares.

5.2 The RIBS Framework

As shown in Fig. 4, a QD algorithm in RIBS is comprised of three components: (1) an *archive* to store solutions generated by the QD algorithm, (2) one or more *emitters* to generate new solutions, and (3) a *scheduler* to manage the interaction of the archive and emitters. Algorithm 1 shows the standard execution loop for combining these components. As shown in Sec. 5.3, this execution loop is flexible and not limited to a single call to the ask-tell interface.

Archive. The archive is a data structure that stores the solutions generated by the QD algorithm, along with any information relevant to solutions, such as objective and feature values. The primary archive method is `add()`, which takes in multiple solutions with their objective and feature values, attempts to add them to the collection of solutions, and returns information about the addition. Examples of such information include "status" (whether the solution found a new cell in the archive, improved an existing cell, or was not added at all [44]), "novelty" (the average distance in feature space from the solution to its k-nearest neighbors in the archive [3]), and "improvement value" (the difference between the solution's objective value and that of the solution which it replaced [46]). Archives may support additional functionality, such as methods for sampling solutions and retrieving solutions with given feature values.

An important choice in the implementation of `add()` is the order of inserting solutions. The simplest choice is to insert solutions *sequentially*, i.e., one after another. pyribs offers sequential addition but defaults to the alternative of inserting all solutions simultaneously as a *batch*.

Batching the additions has the following benefits:

- Some metrics depend on the order in which solutions are inserted. For example, if two solutions $\boldsymbol{\theta}_a$ and $\boldsymbol{\theta}_b$ have similar features, then $\boldsymbol{\theta}_a$ may be inserted with high novelty, while $\boldsymbol{\theta}_b$ is subsequently inserted with low novelty because $\boldsymbol{\theta}_a$ is already in the archive. Batching overcomes this issue by "freezing" the archive, then computing the metrics of all solutions with respect to the frozen archive.

- Batching the additions enhances the performance of the operation, as libraries like NumPy [55] (used in pyribs [1]) and JAX [56] (used in QDax [54]) are designed to operate on batches of data.

Emitters. QD algorithms in RIBS instantiate one or more emitters. Emitters generate solutions and adapt its internal states based on the objective, features, and archive insertion feedback of those solutions. Since sometimes not all solutions are inserted to the archive, an emitter may adapt is parameters to avoid generating solutions similar to those that have already been rejected by the archive. Emitters in RIBS must implement two methods:

- The `ask()` method generates candidate solutions.
- The `tell()` method updates the internal states of the emitter based on the objective and feature values of the generated solutions and any information gained from adding the solutions to the archive.

Simple QD algorithms may implement emitters that do not require any internal states. For instance, when `ask()` is called, MAP-Elites generates new solutions by sampling existing archive solutions and perturbing them with isotropic Gaussian noise. Since there are no parameters to update for this Gaussian noise mutation, the `tell()` method does not perform any operation.

Another example of a more complicated emitter is the CMA-ES [47] emitter from CMA-ME [44]. Here, the `ask()` method samples solutions from the multivariate Gaussian distribution maintained by CMA-ES, and the `tell()` method updates the Gaussian distribution and internal CMA-ES parameters as specified by the CMA-ES algorithm.

Scheduler. The scheduler facilitates the interaction between the archive and the population of emitters. The scheduler adds the generated solutions from the emitters to the archive and passes the results of the evaluation and archive insertion to the emitters. Moreover, the scheduler selects which emitters generate new solutions on each iteration of the algorithm. Schedulers make decisions on active emitters based on how well each emitter performs in previous iterations. While most

QD algorithms implements a basic scheduler that always activates all the emitters, Multi-Emitter MAP-Elites [57] schedules the emitters using a multi-armed bandit algorithm. Ultimately, the scheduler is the primary user interface in the RIBS framework.

Algorithm 1: Standard Execution Loop in RIBS

1 **QD Algorithm** *(n_e, n_it)*:
  Input: Number of emitters n_e, number of iterations n_it, parameters for `Archive`, `Emitters`, and `Scheduler`
  Result: Generates solutions to optimize the QD objective, stored in an `Archive`
2 `Archive` ← init_archive()
3 $[\texttt{Emitter}_1 \ldots \texttt{Emitter}_{n_\text{e}}]$ ← init_emitters(`Archive`)
4 `Scheduler` ← init_scheduler(`Archive`, $[\texttt{Emitter}_1..\texttt{Emitter}_{n_\text{e}}]$)
5 **for** $itr \leftarrow 1$ **to** n_it **do**
6 $L \leftarrow$ `Scheduler`.ask()
7 *User* computes `Evals` = $[f(\boldsymbol{\theta}), \boldsymbol{\phi}(\boldsymbol{\theta})$ for $\boldsymbol{\theta}$ in $L]$
8 `Scheduler`.tell(`Evals`)
9 **return** `Archive`
10 **Scheduler.ask** *()*:
  Result: Returns a list of solutions L generated by the emitters.
11 $L \leftarrow []$ `// Empty list`
12 **for** $i \leftarrow 1$ **to** n_e **do**
13 **if** $\texttt{Emitter}_i$ should generate solutions **then**
14 $L_i \leftarrow \texttt{Emitter}_i$.ask()
15 $L \leftarrow LL_i$ `// Concatenate` L_i `to` L
16 **return** L
17 **Scheduler.tell** *(`Evals`)*:
  Input: Objective and measure function evaluations of the list of solutions L.
  Result: Inserts solutions into `Archive` and updates `Emitters`.
18 `add_info` ← `Archive`.add(L, `Evals`)
19 **for** $i \leftarrow 1$ **to** n_e **do**
20 **if** $\texttt{Emitter}_i$ generated solutions **then**
21 Retrieve solutions L_i generated by $\texttt{Emitter}_i$
22 Retrieve $\texttt{Evals}_i$ corresponding to L_i
23 Retrieve $\texttt{add\_info}_i$ corresponding to L_i
24 $\texttt{Emitter}_i$.tell(L_i, $\texttt{Evals}_i$, $\texttt{add\_info}_i$)

Schedulers utilizes an ask-tell interface as shown in Algorithm 1. When `ask()` is called (line 10), the scheduler selects one or more emitters and calls each emitter's `ask()` method to generate solutions. When `tell()` is called (line 17), the scheduler takes in the objective and feature function evaluations of these solutions and `add()`'s the solutions to the archive. Then, the scheduler passes

Table 1: By selecting different components in the RIBS framework, we can compose a variety of recent algorithms from the QD literature and test them in pyribs. Furthermore, we can identify combinations of components which may lead to new algorithms. Adapted from [1].

	Archive				Emitters					Scheduler	
	Grid	CVT	Sliding Boundaries	Unstructured	Gaussian	Iso+LineDD	CMA-ES	Genetic Algorithm	Gradient Arborescence	Basic	Bandit
MAP-Elites [48]	×				×					×	
CVT-MAP-Elites [50]		×			×					×	
Iso+LineDD MAP-Elites [49]		×				×				×	
MESB [58]			×		×					×	
NSLC [7]				×				×		×	
CMA-ME [44]	×						×			×	
CMA-MAE [46]	×						×			×	
ME-MAP-Elites [57]	×					×	×				×
CMA-MEGA [11]	×								×	×	
CMA-MAEGA [46]	×								×	×	

the solutions, evaluations, and archive addition information to the emitters via each emitter's `tell()` method.

In the original emitter implementation [44], emitters directly called `add()` to insert solutions into the archive. However, allowing emitters to modify the archive meant that feedback from `add()` depended on the order in which emitters were called. This behavior was revised during the development of pyribs. Now, although the emitters may read data from the archive (e.g., when sampling solutions), only the scheduler calls `add()` and passes the returned information to the emitters through their `tell()` method. Consequently, all emitters receive feedback from the same version of the archive, making the algorithms invariant to the order in which the emitters operate.

5.3 Composing Algorithms in RIBS

Algorithm 1 shows a standard execution loop in RIBS. First, the user configures the core components. Then, in the main loop (line 5), the user calls the scheduler's ask-tell interface and evaluates solutions in between the calls. Importantly, the RIBS components (archive, emitters, and scheduler) in this loop are interchangeable, and the execution loop can be customized to support new QD algorithms. Replacing components or modifying the execution loop enables RIBS to support a variety of QD algorithms [1].

First consider algorithms that replace components of RIBS without modifying the standard execution loop outlined in Algorithm 1. Tab. 1 summarizes the components required for each algorithm.

Throughout this section, components listed in Tab. 1 are italicized.

We begin with MAP-Elites [48], which has a *Grid Archive* that tessellates the feature space into a grid. MAP-Elites incorporates a single emitter that randomly selects solutions from the archive and applies mutations. One kind of mutation is to add Gaussian noise; in this case, we call the emitter *Gaussian Emitter*. As is common in many versions of MAP-Elites, *Gaussian Emitter* can also sample directly in the solution space on initial calls to `ask()`. Since this emitter has no adaptive components, its `tell()` method does nothing. Finally, MAP-Elites has a *Basic Scheduler* that simply selects this emitter on every iteration.

Replacing components creates different MAP-Elites variants. Substituting the Gaussian emitter with the *IsoLine+DD Emitter*, which applies the Iso+LineDD operation [49], results in MAP-Elites (Line). We can also replace the archive with a *CVT Archive* or *Sliding Boundaries Archive* to obtain CVT-MAP-Elites [50] and MAP-Elites with Sliding Boundaries [58].

We can also consider methods based on Novelty Search like NSLC [7]. Here, the *Unstructured Archive* adds solutions if they are far away from their k nearest neighbors in the archive. Meanwhile, the *Genetic Algorithm Emitter* contains a genetic algorithm such as NEAT [59] that manages a population of solutions. In contrast to *Gaussian Emitter* and *IsoLine+DD Emitters*, *Genetic Algorithm Emitter*'s `tell()` updates its internal population. The scheduler remains the same as in MAP-Elites.

CMA-ME [44] and CMA-MAE [46] are more complicated because they require managing multiple instances of CMA-ES in parallel. In this case, we create multiple *CMA-ES Emitters*, each with their own CMA-ES instance. Calling `ask()` on each emitter samples solutions from CMA-ES's multivariate Gaussian distribution, and calling `tell()` updates the distribution parameters and the internal CMA-ES parameters. We combine these emitters with the *Grid Archive* and *Basic Scheduler* from MAP-Elites.

Multi-Emitter MAP-Elites (ME-MAP-Elites) [57] provides an example of an algorithm that requires a different scheduler. The default ME-MAP-Elites includes *CMA-ES* and *Iso+LineDD Emitters*. Its

Algorithm 2: QD Algorithm with Surrogate Model in RIBS

1 **QD Algorithm with Surrogate Model** $(n_e, n_{inner}, n_{outer})$**:**
  Input: Number of emitters n_e, inner loop iterations n_{inner}, outer loop iterations n_{outer}, parameters for $Archive$, $Emitters$, $Scheduler$, and $Model$
  Result: Generates solutions to optimize the QD objective, stored in an $Archive$
2 $Archive \leftarrow$ init_archive()
3 $Model \leftarrow$ init_surrogate_model()
4 $\mathcal{D} \leftarrow \{\}$ `// Dataset of evaluated solutions`
5 **for** $itr \leftarrow 1..n_{outer}$ **do**
6 `// Construct surrogate archive.`
7 $Archive' \leftarrow$ init_archive()
8 $[Emitter'_1..Emitter'_{n_e}] \leftarrow$ init_emitters($Archive'$)
9 $Scheduler' \leftarrow$ init_scheduler($Archive'$,
10 $[Emitter'_1..Emitter'_{n_e}]$)
11 **for** $iter \leftarrow 1..n_{inner}$ **do**
12 $L \leftarrow Scheduler'$.ask()
13 $Evals' \leftarrow [Model.f(\boldsymbol{\theta}), Model.\boldsymbol{m}(\boldsymbol{\theta})$ for $\boldsymbol{\theta}$ in $L]$
14 $Scheduler'$.tell($Evals'$)
15 `// Record true evaluations of solutions.`
16 $L \leftarrow$ all solutions in $Archive'$
17 $User$ computes $Evals = [f(\boldsymbol{\theta}), \boldsymbol{\phi}(\boldsymbol{\theta})$ for $\boldsymbol{\theta}$ in $L]$
18 $Archive$.add($L, Evals$)
19 `// Update model.`
20 $\mathcal{D} \leftarrow \mathcal{D} \cup (L, Evals)$
21 Train $Model$ on data in $\mathcal{D}$
22 **return** $Archive$

Bandit Scheduler applies a multi-armed bandit algorithm to select a subset of these emitters based on whether they have previously generated solutions that were inserted into the archive.

In addition to replacing components of RIBS, some algorithms also modifies the RIBS execution loop. For instance, CMA-MEGA [11] and CMA-MAEGA [46] both leverages the *gradient arborescence emitter*, which constructs solutions by branching from a solution point based on the objective and feature gradients. This branching requires calling `ask()` and `tell()` twice. The first time to evaluate the gradients at a particular point, and a second time to generate solutions according to the gradients. To support this behavior, add another set of calls to `ask()` and `tell()` in the loop on line 5, with appropriate arguments to handle passing gradients back to `tell()`.

A number of recent works [28, 30, 60, 61] integrate surrogate models with QD algorithms in domains where evaluations are expensive. Surrogate-assisted QD algorithms construct an archive based on evaluations predicted by a surrogate model and then select candidate solutions for ground-truth evaluations [1]. Such models are beneficial when solution evaluations are expensive.

Algorithm 2 demonstrates a general layout for surrogate-assisted QD algorithms. In addition to the standard QD ask-tell, the surrogate-assisted QD stores the solutions inside a *ground-truth archive* (line 2). Then, during an "outer loop" (line 5), it performs three phases. First, it constructs a *surrogate archive* in an "inner loop" (line 11) based on solutions evaluated by the model (line 13). Second, the user evaluates the candidate solutions from the surrogate archive, and the evaluated solutions are added into the ground-truth archive (line 18). Finally, the algorithm trains the model to improve its predictions (line 21).

5.4 Comparison to Existing QD Libraries

It is important to compare and contrast pyribs to existing QD libraries to understand its advantages and disadvantages. In this section, I will review existing QD libraries by outlining their main features, strength, and weaknesses, and compare them to that of pyribs.

Sferes$_{v2}$. Sferes$_{v2}$ [52] is primarily a `C++` framework for evolutionary computation, but it also supports QD algorithms. Sferes$_{v2}$ is primarily designed for high performance, leveraging template-based meta-programming to provide an efficient object-oriented interface and offering multi-core parallel execution through Intel TBB and MPI. While the template-based structure results in significant performance benefits, it significantly limits the readability of the code for non-expert users, which hinders its accessibility.

In comparison, pyribs focuses solely on QD algorithms rather than on general evolutionary computation. This means that the design and implementation decision of pyribs is more oriented towards QD users. For instance, in addition to the algorithms supported by Sferes$_{v2}$, pyribs supports CMA-ME [44], CMA-MEGA [11], CMA-MAE [46], and Multi-Emitter MAP-Elites [57] (Tab. 1).

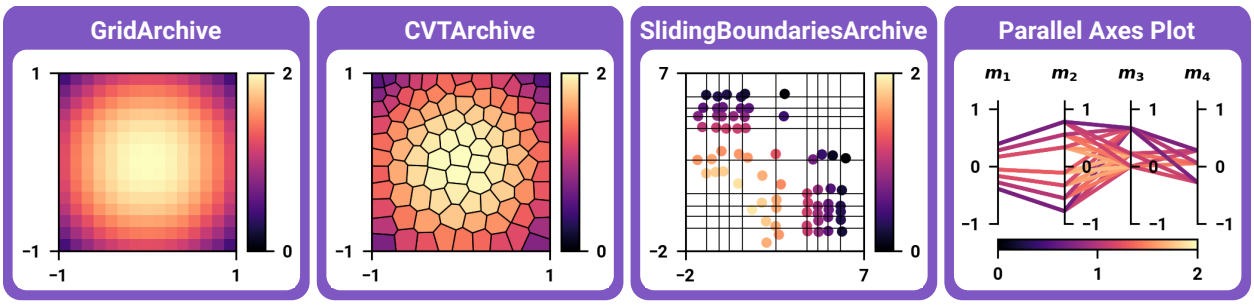


Figure 5: Pyribs visualization tools. This figures show example 2D heatmaps, where the axes correspond to the feature values, and the color of each archive cell indicates its objective value. In `SlidingBoundariesArchive`, the points show the locations of solutions in feature space, and the lines show the grid boundaries. We also show a *parallel axes plot* which can visualize an archive of any dimensionality. In this plot, a single solution's features are plotted as a line connecting the feature $\phi_1 \ldots \phi_k$, and the line is colored according to the solution's objective value. Adapted from [1].

pyribs is a Python library that emphasizes accessibility over performance. While Python itself is slower than lower-level languages like `C++`, Python libraries like NumPy [55] reduces the proformance gap between Python and `C++` significantly by providing efficient numerical computation routines. Moreover, beyond being a beginner-friendly language, it has a flourishing ecosystem, with package repositories like the Python Package Index (PyPI) [62] and Anaconda [63] providing easy access to many useful libraries. Thus, implementing the RIBS framework in Python and distributing pyribs on PyPI and Anaconda makes pyribs *accessible*, as users can easily install and learn to use the library.

QDpy. QDpy [53] is designed to be a feature-rich Python library for QD optimization. Besides supporting ready-to-go implementations of algorithms such as MAP-Elites and CMA-ME, QDpy provides building blocks that can be assembled into new algorithms. To run a QD algorithm, a QDpy user instantiates a *container* (i.e., an archive) and passes it to an *algorithm* object. The user then defines an evaluation function and passes the function to the QDpy system to optimize. QDpy also provides logging and plotting utilities and tools to run the evaluation function on distributed computation.

QDpy's flexibility is limited by the requirement that users pass in an evaluation function. It is important to note that after passing in the evaluation function, QDpy completely takes over the evaluation and optimization process, with little user intervention possible. Thus, while passing

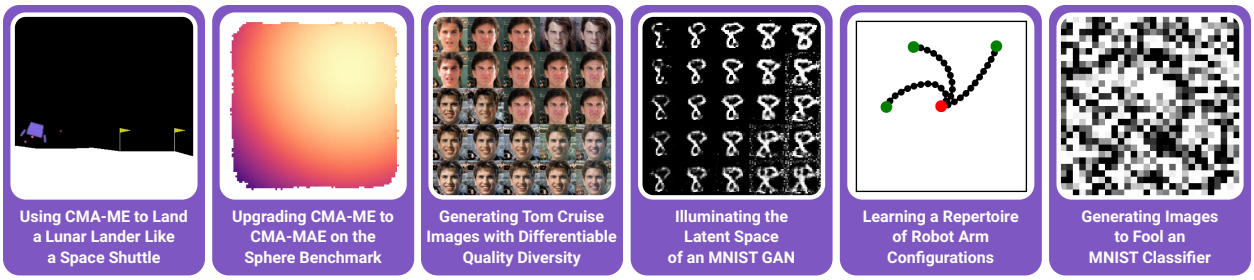


Figure 6: Tutorials enable pyribs users to quickly learn about the library and experiment with problems from the QD literature. Adapted from [1].

in this function allows users to leverage QDpy's various utilities, this requirement also makes it difficult for users to modify the evaluation loop. In contrast, pyribs provides an `ask-tell` interface where users handle evaluations on their own. Essentially, pyribs focuses on components necessary for running QD algorithms, allowing users to integrate tools and frameworks with which they are already familiar.

To make pyribs more *accessible*, pyribs implemented visualization methods for archives following QDpy [1]. As shown in Fig. 5, pyribs can visualize two-dimensional archives as heatmaps and display parallel axes plots that visualize archives of any dimensionality. The easily accessible visualization not only helps researchers debug and understand their algorithm but also fosters a more thorough understanding of QD among learners.

To furether increase *accessibility*, pyribs maintains extensive documentation and tutorials. Every pyribs component is documented in detail, and pyribs has an array of tutorials (Fig. 6). These tutorials teach users about the library and introduce them to common QD problems, such as latent space illumination [26], the arm repertoire benchmark [49, 64], and the sphere linear projection benchmark [44]. The documentation and tutorials are easily accessible online on the website for pyribs.

I was the primary author on the tutorial "Upgrading CMA-ME to CMA-MAE on the Sphere Benchmark" [65]. This tutorial introduced the readers to the sphere linear projection benchmark, and demonstrates how the changes between CMA-ME [44] and CMA-MAE [46] greatly improves the performance of the algorithm on the sphere benchmark. Importantly, this tutorial teaches the

users how to use pyribs to implement CMA-MAE, which is currently one of the most effective general-purpose QD algorithm.

QDax. QDax [54] is a recent library that was developed after the initial release of pyribs. The library focuses on efficient QD, reinforcement learning (RL), and evolutionary algorithm implementations for hardware accelerators such as GPUs and TPUs, taking advantage of the parallel nature of these methods. QDax specializes in reinforcement learning and robotics domains, where evaluation remains an expensive bottleneck. Many experiments that took hours or days on a CPU cluster take only minutes with GPU acceleration in QDax. To leverage accelerators in both function evaluation and algorithm implementation, QDax builds on the JAX library [56] and provides a JAX-based API.

In the spirit of *simplicity* and *accessibility*, pyribs only runs single-threaded on a single CPU [1], despite QDax providing parallelization utilities for both function evaluation and algorithm implemention. Nevertheless, single-threaded libraries has the advantage of being able running on any hardware ranging from laptops to high-performance clusters, since it does not require heavy computation resources. This make pyribs more accessible to people who have limited computation resources.

While being single-threaded may limit the performance of the algorithms implemented in pyribs, the runtime in many QD problems, such as Deceptive Maze, is dominated by the user's evaluation of solutions, rather than by the runtime of the QD algorithm in pyribs. Since evaluations are independent from the algorithm, the evaluations may be parallelized individually. For instance, a pyribs tutorial "Using CMA-ME to Land a Lunar Lander Like a Space Shuttle" parallelizes evaluations with a few lines of changes with Python's `Dask` library [10]. Thus, pyribs maintains its stance as a *simple* library that only supports what it needs.

5.5 Summary

In the future, I will continue to develop pyribs with my colleagues at the ICAROS lab. In the long run, our goal is for pyribs to become a library that supports a wide range of users in the

QD community. To this end, we will continue to maintain our documentation and tutorials and incorporate user feedback in our implementation decisions. Our vision is that pyribs will become an entry point into QD for many researchers.

We also aim to serve the needs of more experienced researchers by further expanding the capabilities of pyribs. We plan to expand pyribs to implement components from other recent and popular QD algorithms. For example, although pyribs currently centers on the MAP-Elites family, potential additions outside this family include the unstructured archive from Novelty Search, the archive with learned measures from AURORA [66], and emitters and schedulers from NS-ES [20] and SERENE [67]. Our future plan for pyribs will support these algorithm.

Beyond directly supporting practitioners, the lessons learned from the development of pyribs can inform the design and development of future QD libraries. I am excited about the role that pyribs can play in expanding the QD community and growing QD into a widely adopted discipline.

6 Density Descent in Diversity Optimization

I propose Density Descent Search (DDS), a DO algorithm that efficiently explores the feature space by leveraging continuous density estimates. *The key insight is to overcome the drawbacks of current density estimation methods in DO by introducing continuous, stable approximations of the solution density in feature space* [2]. Through both empirical (Sec. 6.5) and theoretical analysis (Sec. 6.3), I demonstrated that:

- When combined with a ranking-based optimizer like CMA-ES, NS reduces to a special case of DDS.
- KDE provides stronger algorithmic stability guarantees than novelty score.
- DDS algorithms outperform prior work on 3 out of 4 domains.
- DDS perform well on multi-dimensional feature spaces, which currently present significant challenges to QD and DO algorithms.

6.1 Density Estimation Methods

DDS utilizes two density estimation methods: kernel density estimation and continuous normalizing flow. In contrast to histograms, these two methods of density estimation are continuous. In the case of kernel density estimation, it also provides a stronger uniform stability bound than novelty search.

In general, DDS can be combined with any density estimation method. According to our analysis, any continuous density estimation method should achieve notable performance when combined with DDS [2].

Kernel Density Estimation (KDE). KDE is a non-parametric density estimation method, meaning that it does not make any assumptions about the underlying probability density distribution from which the samples are drawn [68]. The benefit of a non-parametric method is that it can be utilized in settings where nothing about the underlying distribution is known. Given a set of features

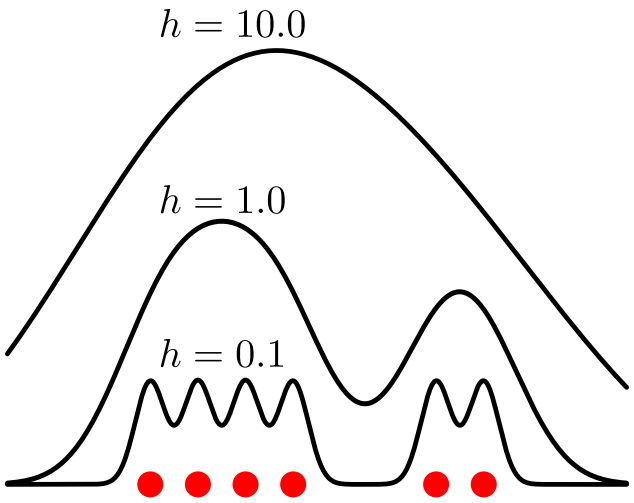


Figure 7: The effect of the bandwidth h on a one-dimensional KDE. Red dots represent the data, and black lines depict the KDE. When h is too small, the KDE reveals many misleading local maxima. When h is too large, the KDE conceals modes from the underlying distribution. Figure adapted from [2]

$\mathcal{X} \subseteq \mathbb{R}^m$, bandwidth parameter h, and kernel function $K(\cdot)$, KDE computes the density function $\hat{D} : \mathbb{R}^m \to \mathbb{R}$ for a given feature $\boldsymbol{y} \in \mathbb{R}^m$ as:

$$\hat{D}_h(\boldsymbol{y}; \mathcal{X}) = \frac{1}{|\mathcal{X}|h} \sum_{\boldsymbol{y}' \in \mathcal{X}} K\left(\frac{\|\boldsymbol{y} - \boldsymbol{y}'\|}{h}\right) \tag{3}$$

Prior work has thoroughly studied the problem of selecting bandwidth that accurately estimate the underlying density function [69–71]. The effect that the bandwidth has on the density estimate is visualized in Fig. 7. For this work, I discovered that DDS works better when the bandwidth is slightly over-smoothed.

When optimizing a density estimate (e.g., with gradient descent), KDEs have several advantages over histograms. First, the shape of a histogram depends on its bin size [72]. Second, the binning procedure leads to a discontinuous optimization landscape and flat gradient signals inside each bin [72]. In contrast, KDE produces a density function that is smooth and continuous based on the location of the samples in the underlying distribution [68].

Continuous Normalizing Flow (CNF). CNF [73] is a generative modeling method that constructs a diffusion path between a simple, usually parameterized distribution (e.g., a Gaussian distribution)

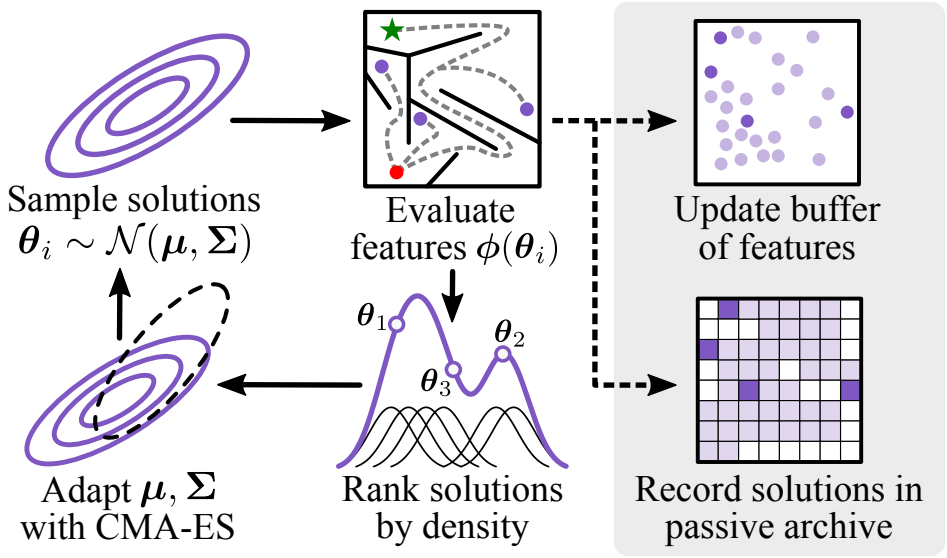


Figure 8: Overview of density descent search (DDS) for solving diversity optimization (DO) problems. DDS first draws solutions from a Gaussian $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. After computing the solution features (in this case, the final position of the robot in a maze), DDS ranks solutions by density. This density ranking is passed to CMA-ES, which updates the search distribution to sample solutions with lower density on the next iteration. Concurrently, solutions are stored in a buffer that forms the basis for density estimates, and in a passive archive that tracks all discovered solutions. Adapted from [2].

and an unknown complex distribution. The diffusion path describes a mapping from each point on the simple distribution to a corresponding point on the unknown distribution such that their probability densities (computed via the Wasserstein metric) are roughly equal. CNF utilizes the diffusion path to transform samples from the simple distribution to a sample from the complex distribution. This enable sampling from probability distributions where direct sampling is difficult (e.g., distributions of images).

Although CNF is has been developed for generative artificial intelligence, in this work [2], I only utilize CNF as a method to estimate the probability density of the feature space.

6.2 Algorithmic Details

DDS maintains a buffer of features $\mathcal{B}$ that represents the distribution of features discovered so far. Based on this buffer, DDS models a density estimation of the feature distribution, and by querying this density model, DDS guides an adaptive optimizer to discover solutions in less-dense areas of feature space. As it searches for these solutions, DDS expands the discovered set of features. The

intuition for DDS is that by always greedily seeking to explore areas that has not been explored very well, which is indicated by its low density, the algorithm will eventually explore everywhere in the space. An overview of the algorithm is illustrated in Fig. 8.

Density Estimation. The core of DDS is its representation of feature space density. We propose two variants of DDS that differ in their density representation. The first variant, DDS-KDE, represents density with KDE. With KDE, we compute density via Eq. 3, with $\mathcal{X}$ set to the buffer of features $\mathcal{B}$.

The second variant, DDS-CNF, estimates the density in feature space by learning a CNF between the standard normal distribution $\mathcal{N}(\mathbf{0}, \boldsymbol{I})$ and the observed feature space distribution. Similar to DDS-KDE, the feature space distribution is represented by the buffer $\mathcal{B}$. We compute a density estimate at any location in the feature space by integrating an ordinary differential equation [73]. Applying techniques from prior work [73], we represent the CNF with a neural network, and on each iteration, we fine-tune the network on the new distribution of features contained in the buffer.

KDE and CNF differ in how easily their smoothness can be controlled. On one hand, the shape of the KDE can be controlled with the bandwidth hyperparameter, where higher bandwidth leads to a smoother density estimate but conceals modes of the ground-truth density distribution, as illustrated in Fig. 7. On the other hand, while CNF does not require selecting a bandwidth hyperparameter, the smoothness of the density estimation cannot be easily controlled, which undermines the performance of the algorithm (Sec. 6.5.3). The effect of the bandwidth parameter on the performance of DDS-KDE is discussed more thoroughly in Sec. 6.3.

Feature Buffer. To provide the basis for density estimates in feature space, DDS maintains a buffer $\mathcal{B}$ that stores the features of sampled solutions (line 4). This buffer represents the *observed* distribution of the feature space and is updated every time new solutions and features are discovered. In theory, the buffer can have infinite capacity, storing every feature ever encountered. In practice, the buffer can only retain a finite number of features due to computation and memory limitations. To decide which features to retain in the buffer, we manage the buffer with an optimal reservoir sampling algorithm (Sec. 6.4). This algorithm updates the buffer with online samples that accurately

represent the distribution of features discovered so far by DDS (line 11).

Optimizer. DDS guides an adaptive optimizer to discover solutions in low-density regions of the feature space. Examples of such optimizers include xNES [74] and Adam [75]. I selected CMA-ES as the underlying optimizer due to its reputation as a state-of-the-art optimizer [47] and its high performance in existing QD algorithms [11, 44, 46].

Passive Archive. Following prior work [76], to evaluate how much of the feature space has been explored, DDS inserts all discovered solutions and features into a passive MAP-Elites-style (Sec. 4) archive $\mathcal{A}$ (line 3, line 10). While DDS itself only uses the archive to record solutions and features, the archive is useful when computing metrics in experiments and when comparing archive-based algorithms with archive-less algorithms (Sec. 6.5.1).

Summary. Algorithm 3 shows how the components of DDS come together. DDS begins by initializing its various components (line 2-4). During the main loop (line 5), DDS samples solutions with the underlying optimizer (line 7). Since our optimizer is CMA-ES, sampling consists of drawing from a multivariate Gaussian $\mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$. Subsequently, DDS computes the features of the solutions (line 8) and estimates their feature space density (line 9). After sampling solutions, DDS updates its various components. For instance, in DDS-KDE, the density update (line 12) consists of replacing the feature buffer, and in DDS-CNF, the update involves fine-tuning the neural network on the feature buffer. Furthermore, solutions with lower density are ranked first on line 13, causing CMA-ES to adapt (line 14) towards less-dense regions of the feature space.

6.3 Connection between Novelty Search and Kernel Density Estimates

I further provide theoretical insight into the connection between NS and DDS-KDE and delineate the advantage of KDE over novelty score. KDE and novelty score are non-stationary since they change as more solutions are discovered by their respective optimizers. However, the magnitude of non-stationarity is different in KDE and novelty score (Theorem 6.2, Theorem 6.3). Furthermore, when $k \geq |\mathcal{B}|$ in the k-nearest neighbor calculation for novelty score, novelty search becomes a

Algorithm 3: Density Descent Search (DDS)

DDS *($\phi(\cdot)$, `density`, $b, N, \lambda, \boldsymbol{\mu}_0, \sigma$)*:
- **Input:** Feature function $\phi(\cdot)$, density estimator object `density`, buffer size b, number of iterations N, batch size λ, initial solution $\boldsymbol{\mu}_0$, and initial step size σ
- **Result:** Generates $N \cdot \lambda$ solutions, storing a representative subset of them in a passive archive $\mathcal{A}$.

Initialize CMA-ES search point $\boldsymbol{\mu} := \boldsymbol{\mu}_0$, search direction $\boldsymbol{\Sigma} = \sigma \boldsymbol{I}$, and internal parameters $\boldsymbol{p}$
Initialize empty archive $\mathcal{A}$
Initialize empty buffer $\mathcal{B}$ of size b
for $itr \leftarrow 1$ **to** N **do**
  for $i \leftarrow 1$ **to** λ **do**
    $\boldsymbol{\theta}_i \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\Sigma})$
    $\boldsymbol{\phi}_i \leftarrow \boldsymbol{\phi}(\boldsymbol{\theta}_i)$
    $D_i \leftarrow \texttt{density}(\boldsymbol{\phi}_i)$
    Add $(\boldsymbol{\theta}_i, \boldsymbol{\phi}_i)$ to archive $\mathcal{A}$
  Update buffer $\mathcal{B}$ with $\boldsymbol{\phi}_{1..\lambda}$ via reservoir sampling
  Update `density` with the new buffer $\mathcal{B}$
  Rank $\boldsymbol{\theta}_i$ in *ascending* order by D_i
  Adapt CMA-ES parameters $\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{p}$ based on density ranking D_i

special case of DDS-KDE (Theorem 6.4). All theorems and proofs can be found in Appendix B.

6.3.1 Stability Under Non-stationarity

DO algorithms gradually learn a non-stationary density representation (e.g., KDE or histogram) as they explore the feature space [2]. This means that they maintain a density function over the feature space that changes every time a new solution is discovered. However, drastic changes in the representation may present significant challenges for adaptive optimizers such as Adam [75] and CMA-ES [47]. This is because adaptive optimizers gradually adapts their parameters according to the density function that the DO algorithm maintains. Thus, if the density function changes too drastically, then the adaptive optimizer will not be able to adapt fast enough to account for the changes in the optimization landscape.

To characterize the extent of change in the density estimate, I appeal to the notion of *uniform stability* [77], defined as follows:

Definition 6.1. *Let a function $F(\boldsymbol{x}; \mathcal{B}) : \mathbb{R}^d \to \mathbb{R}$ be parameterized by a set $\mathcal{B} \in \mathbb{R}^d$. We say that F is ϵ-uniformly stable if for all $\mathcal{B}, \mathcal{B}' \subseteq \mathbb{R}^d$, where $\mathcal{B}$ and $\mathcal{B}'$ differ by at most one element, we have*

$$\sup_{\boldsymbol{x} \in \mathbb{R}^d} |F(\boldsymbol{x}; \mathcal{B}) - F(\boldsymbol{x}; \mathcal{B}')| \leq \epsilon \tag{4}$$

I proved that KDE has uniform stability directly proportional to the size of its feature buffer and bandwidth:

Theorem 6.2. *A kernel density estimate $\hat{D}_h(\boldsymbol{x}; \mathcal{B})$ managed with reservoir sampling, such that features in the buffer are exchanged one at a time, is $\frac{1}{|\mathcal{B}|h}$-uniformly stable, where h is the bandwidth.*

Higher bandwidth makes the function more stationary and thus more suited for adaptive optimizers. However, higher bandwidth also leads to *over-smoothing* of the density estimate, which conceals modes of the true density function [68] (Fig. 7). Thus, selecting an optimal bandwidth for DDS requires a fine balance between the accuracy and uniform stability of the KDE.

In contrast, novelty score becomes less uniformly stable as the diameter of the feature space W increases:

Theorem 6.3. *Novelty score $\rho(\boldsymbol{x}; \mathcal{B})$ is $\frac{W}{k}$-uniformly stable, where k is the nearest neighbors parameter in novelty score and $W = \max_{\boldsymbol{s}_1, \boldsymbol{s}_2 \in S} \|\boldsymbol{s}_1 - \boldsymbol{s}_2\|$ is the diameter of the feature space S.*

Therefore, for unbounded feature spaces, the uniform stability of novelty score is also unbounded, and for bounded feature spaces, novelty score has uniform stability directly proportional to k.

When the feature space is bounded, as in the experiments, KDE has a stronger stability guarantee than novelty score for bandwidth $h \geq \frac{k}{|\mathcal{B}|}$. Following this insight, we select a bandwidth satisfying this inequality with the bandwidth selection mechanism in Sec. 6.6. Our theoretical results are corroborated by our experiments in Sec. 6.5, which demonstrate that DDS-KDE outperforms NS in all domains on all metrics.

6.3.2 Equivalence of NS and DDS-KDE

When $h = 1$ and as $k \to |\mathcal{B}|$, the uniform stability upper-bound of novelty score approaches that of KDE. I show in Theorem 6.4 that when all points are considered in the novelty score computation (i.e., let $k = \infty$), NS is a special case of DDS-KDE under ranking-based optimizers such as CMA-ES. Specifically, under these conditions, the ranking of solutions based on their novelty score is identical to the ranking based on their kernel density estimate.

Theorem 6.4. *Let $\mathcal{B} \subseteq \mathbb{R}^m$ be a set of features. Consider the rankings π_{NS} and π_{KDE} on another set of features $\{\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_m\} \subseteq \mathbb{R}^m$ where $\boldsymbol{\phi}_i = \boldsymbol{\phi}(\boldsymbol{\theta}_i)$. We define the rankings as follows: $\pi_{\text{NS}}(i) \geq \pi_{\text{NS}}(j) \iff \rho(\boldsymbol{\phi}_i; \mathcal{B}) \geq \rho(\boldsymbol{\phi}_j; \mathcal{B})$ and $\pi_{\text{KDE}}(i) \geq \pi_{\text{KDE}}(j) \iff \hat{D}(\boldsymbol{\phi}_i; \mathcal{B}) \leq \hat{D}(\boldsymbol{\phi}_j; \mathcal{B})$. We show that $\pi_{NS} = \pi_{KDE}$, when NS has $k = \infty$ (or, equivalently, $k = |\mathcal{B}|$) and KDE has triangular kernel $K(u) = 1 - |u|$ with support over the entire feature space S.*

6.4 Reservoir Sampling for Maintaining a Representative Buffer

As outlined in Sec. 6, DDS maintains a buffer $\mathcal{B}$ that stores the features of sampled solutions. While it would be ideal to maintain an infinite buffer that stores all features ever encountered, due to runtime and memory constraints, DDS instead maintains a finite-sized buffer $\mathcal{B}$ of size $|\mathcal{B}| = b$. Moreover, the subset of features retained by $\mathcal{B}$ must accurately represent all the features ever encountered. This can be described as the *reservoir sampling* problem.

Problem 6.1 (Reservoir Sampling [78])**.** *Suppose there is a finite population of unknown size N. Derive an algorithm that draws a simple random sample of size b from this population by scanning it exactly once, with the memory constraint such that the algorithm can remember at most b samples at a time.*

The population consists of the features that are discovered by DDS throughout its entire execution. Since the buffer $\mathcal{B}$ is a simple random sample of all features, the density function estimated on $\mathcal{B}$ will be close to the density function of all features. Thus, with reservoir sampling, we are only

required to maintain a subset of all discovered features without severely altering the shape of the density estimation of the feature space. In DDS, this is achieved by leveraging algorithm L, the optimal reservoir sampling algorithm [78].

I originally used a naïve algorithm that constructed a representative buffer B_b by sampling b solutions from B_N without replacement. This algorithm permutes B_N uniformly at random and selects the first b solutions. Eq. 5 calculates $P(a_i) = \frac{b}{N}$, the probability that solution $\boldsymbol{\theta}_i$ is added to buffer B_b.

$$P(a_i) = \frac{\binom{N-1}{b-1}}{\binom{N}{b}} = \frac{\frac{(N-1)!}{(b-1)!(N-b)!}}{\frac{N!}{b!(N-b)!}} = \frac{b!(N-1)!(N-b)!}{N!(b-1)!(N-b)!} = \frac{b}{N} \tag{5}$$

However, soon I realized that this approach is not feasible, as this requires that the algorithm stores every single solution it comes across in the global buffer B_N. Therefore, I switched to using the reservoir sampling algorithm in [78].

6.5 Experiments

I compared the performance of DDS-KDE and DDS-CNF with the QD algorithms MAP-Elites (line),[2] CMA-ME, and CMA-MAE, and with NS using CMA-ES as the underlying optimizer [2]. All algorithms and domains are implemented with pyribs [1].

My experiments include canonical benchmark domains from QD and DO: Linear Projection [44], Arm Repertoire [49], and Deceptive Maze [3]. As discussed in Sec. 3, we convert QD domains into DO domains by setting the objective to be constant, effectively removing the importance of solution quality. The Deceptive Maze domain is implemented with the Kheperax library [79]. Furthermore, to address the challenges posed by high-dimensional feature spaces, I introduced a new domain, Multi-feature Linear Projection, that generalizes Linear Projection to high-dimensional feature spaces [2].

I completed the experimental design, domain and algorithm implementation, data collection, and

[2]MAP-Elites with the Iso+LineDD operator.

analysis. The description of the experimental design, domain details, and results is based on my paper [2]. All experiments run on a 128-core workstation with an NVIDIA RTX A6000 GPU. As pyribs is a single-threaded library (Sec. 5.4), we only use the GPU for training the CNF in DDS-CNF and for evaluating the Deceptive Maze domain.

6.5.1 Experimental Design

Independent Variable. In each domain, I conducted a between-groups study with the algorithm as the independent variable.

Dependent Variables. I computed the archive coverage as the number of occupied cells in the archive divided by the total number of cells. To compare the coverage of archive-based algorithms (CMA-MAE, CMA-ME, and MAP-Elites) with that of archive-less algorithms (DDS-KDE, DDS-CNF, and NS), the coverage of all algorithms are tracked on a passive archive $\mathcal{A}$ tessellated by a 100×100 grid.

For the high-dimensional feature space experiment, the coverage is computed with a centroidal Voronoi tessellation with 10,000 cells [50]. This is because when the number of cells in a grid scales exponentially with the dimension of the feature space (curse of dimensionality) [50]. Thus, it is unfeasible to compute a solution for every single cell on a grid in high-dimensional feature space. Centroidal Voronoi tessellation can fix the total number of cells, so the maximum number of solutions stored in the archive will be reasonable. However, this does mean that every cell in the centroidal Voronoi tessellation takes up more volume in the feature space than a grid cell.

Following prior work [80], we also assess the ability of each algorithm to uniformly explore the feature space. We measure the cross-entropy between a uniform distribution and the distribution of sampled features. Let N_e be the frequency that a solution were discovered in cell $e \in \{1, \ldots, l\}$ in the passive archive, and $N_{\text{total}} = \sum_{e=1}^{l} N_e$. The cross-entropy score is defined as:

$$\text{CE} = -\sum_{e=1}^{l} \frac{1}{l} \log\left(\frac{N_e}{N_{\text{total}}}\right) \tag{6}$$

CE achieves its minimum value when N_e is uniformly distributed for all cells e. For all passive archives in our experiments, this minimum is 9.21 for $l = 10{,}000$ cells.

6.5.2 Domain Details

Linear Projection (LP). LP is a QD domain where $\boldsymbol{\phi}$ induces high distortion by mapping an n-dimensional solution space to a 2D feature space [44]. A harsh penalty is applied outside the bounds of the feature space to hinder exploration near the bounds. The feature function $\boldsymbol{\phi} : \mathbb{R}^n \to \left[-5.12 \cdot \frac{n}{2}, 5.12 \cdot \frac{n}{2}\right]^2$ is defined as:

$$\boldsymbol{\phi}(\boldsymbol{\theta}) = \left(\sum_{i=1}^{\frac{n}{2}} \text{clip}(\theta_i), \sum_{i=\frac{n}{2}+1}^{n} \text{clip}(\theta_i) \right) \tag{7}$$

$$\text{clip}(\theta_i) = \begin{cases} \theta_i & \text{if } |\theta_i| \leq 5.12 \\ 5.12/\theta_i & \text{otherwise} \end{cases} \tag{8}$$

where θ_i is the ith component of $\boldsymbol{\theta}$, and we assume that n is divisible by 2. $\boldsymbol{\phi}(\boldsymbol{\theta})$ applies $\text{clip}(\cdot)$ to each θ_i and sums the two halves of $\boldsymbol{\theta}$. Since $\text{clip}(\theta_i)$ restricts θ_i to the interval $[-5.12, 5.12]$, $\boldsymbol{\phi}(\boldsymbol{\theta})$ is bounded by the closed interval $\left[-5.12 \cdot \frac{n}{2}, 5.12 \cdot \frac{n}{2}\right]^2$.

For DO experiments, we set the objective function $f(\boldsymbol{\theta}) = 1$. Following prior work [11, 46], we let the solution dimension be $n = 100$.

Arm Repertoire. The goal of Arm Repertoire [49, 64] is to search for a diverse collection of arm positions for a planar robotic arm with n revolving joints. In this domain, $\boldsymbol{\theta} \in [-\pi, \pi]^n$ represents the angles of the n joints, and $\boldsymbol{\phi}(\boldsymbol{\theta})$ computes the (x, y) position of the arm's end-effector using forward kinematics. While all other domains in this paper have a maximum of 100% coverage, Arm Repertoire only has a maximum coverage of 80.24% when using a 100×100 grid archive [46], since the arm can only move in a circle of radius n. This is because while the passive archive covers a square area in the feature space, the arm can only reach a portion of that area. Similar to LP, we set $f(\boldsymbol{\theta}) = 1$ and $n = 100$.

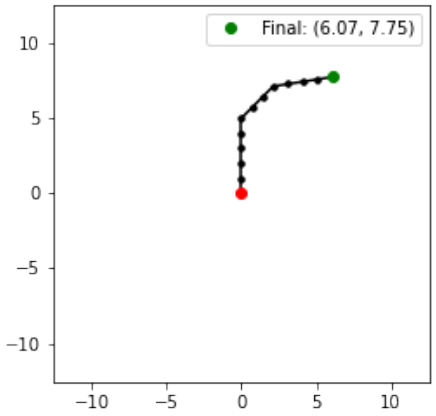


Figure 9: An example of an arm configuration for the Arm Repertoire domain. Note that while this arm has only 12 degrees of freedom, the arms in the experiments of this thesis has 100 degrees of freedom. Figure adapted from [81].

Deceptive Maze. Deceptive Maze is a DO domain that challenges the algorithm to discover a diverse set of final positions for robots navigating a maze (Fig. 1a) [3]. In this domain, $\boldsymbol{\theta}$ parameterizes the robot's neural network controller. $\boldsymbol{\phi}(\boldsymbol{\theta})$ is the final position of the robot after evaluating its path in the maze. As this is a DO domain, this has no objective function. In our experiments, the neural network is a MLP with $n = 66$ parameters.

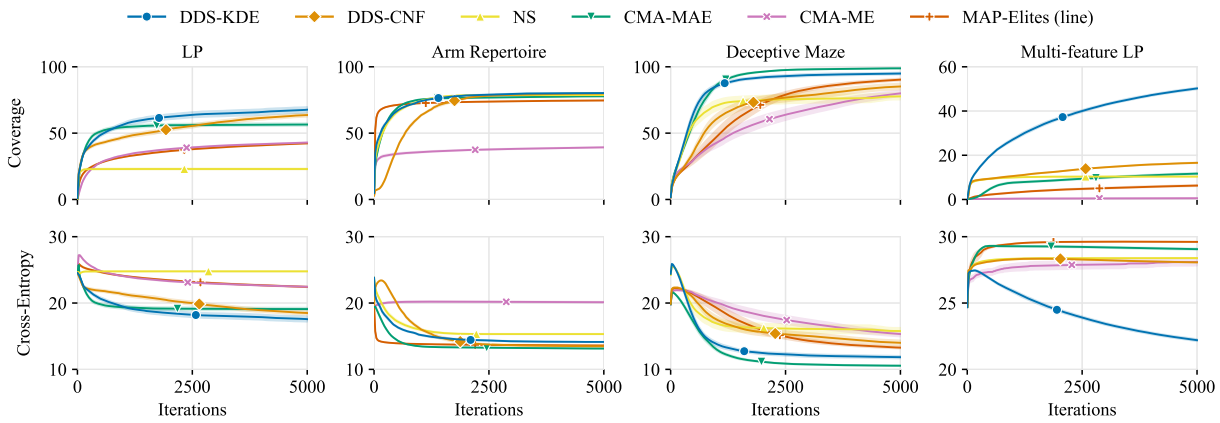
Multi-feature Linear Projection (Multi-feature LP). We introduce a generalized version of LP that scales to m-dimensional feature spaces. The feature function $\boldsymbol{\phi} : \mathbb{R}^n \to \left[-5.12 \cdot \frac{n}{m}, 5.12 \cdot \frac{n}{m}\right]^m$ is defined as

$$\boldsymbol{\phi}(\boldsymbol{\theta}) = \left(\sum_{i=\frac{jn}{m}+1}^{\frac{(j+1)n}{m}} \text{clip}(\theta_i) : j \in \{0, \ldots, m\} \right) \tag{9}$$

where we assume that n is divisible by m. When $m = 2$, this domain turns into the classical LP domain, as evident by Eq. 7. The experiments selects $n = 100$ and $m = 10$ for this domain.

6.5.3 Statistical Analysis

Fig. 10 shows the mean coverage and cross-entropy over 10 trials. In each domain, we conducted a one-way ANOVA to check if the algorithms differed in their coverage and cross-entropy. Since all ANOVAs were significant ($p < 0.001$), as shown in Tab. 2, I followed up with pairwise comparisons



	LP		Arm Repertoire		Deceptive Maze		Multi-feature LP	
	Coverage	Cross-Entropy	Coverage	Cross-Entropy	Coverage	Cross-Entropy	Coverage	Cross-Entropy
DDS-KDE	**67.67 ±2.13%**	**17.57 ±0.41**	**80.22 ±0.01%**	14.14 ±0.02	94.93 ±0.84%	11.84 ±0.21	**50.22 ±0.45%**	**22.20 ±0.09**
DDS-CNF	63.65 ±1.38%	18.50 ±0.29	79.82 ±0.19%	13.48 ±0.12	85.17 ±2.28%	14.01 ±0.38	16.57 ±0.22%	28.07 ±0.03
NS	22.96 ±1.59%	24.78 ±0.06	78.67 ±0.23%	15.32 ±0.07	77.70 ±2.47%	15.78 ±0.50	10.36 ±0.16%	28.39 ±0.03
CMA-MAE	56.47 ±1.06%	19.09 ±0.23	77.77 ±0.10%	**13.14 ±0.01**	**98.83 ±0.18%**	**10.55 ±0.06**	11.68 ±0.37%	29.06 ±0.06
CMA-ME	42.90 ±0.25%	22.45 ±0.05	39.29 ±0.24%	20.13 ±0.09	79.97 ±2.17%	15.33 ±0.39	0.56 ±0.14%	28.10 ±0.28
MAP-Elites (line)	42.30 ±0.31%	22.43 ±0.07	74.62 ±0.07%	13.60 ±0.01	90.32 ±0.87%	13.28 ±0.16	6.28 ±0.34%	29.61 ±0.02

Figure 10: Coverage and cross-entropy (CE) after 5,000 iterations of each algorithm in all domains. We report the mean over 10 trials, with error bars showing the standard error of the mean. Higher coverage and lower cross-entropy are better.

via Tukey's HSD test (Appendix A).

Table 2: One-way ANOVA results in each domain. All p-values are less than 0.001.

	Coverage	Cross-Entropy
LP	$F(5, 54) = 206.61$	$F(5, 54) = 156.04$
Arm Repertoire	$F(5, 54) = 9691.05$	$F(5, 54) = 1523.35$
Deceptive Maze	$F(5, 54) = 23.89$	$F(5, 54) = 39.33$
Multi-feature LP	$F(5, 54) = 3350$	$F(5, 54) = 466.49$

Coverage. DDS-KDE and DDS-CNF outperform all baselines on LP, Arm Repertoire, and Multi-feature LP. They exhibit no statistical difference in performance with the best-performing algorithm on Deceptive Maze (CMA-MAE). Notably, DDS-KDE solves Arm Repertoire nearly perfectly, as the maximum coverage in the domain is 80.24%.

The high coverage of DDS-KDE and DDS-CNF on these domains can be attributed to the continuity of their density estimate, which prevents DDS-KDE from converging prematurely. For example, on LP, our algorithms discover more solutions near the edges of the feature space than CMA-MAE [2]. The passive archive maintained by CMA-MAE converges as all the solutions fall into previously discovered cells [44]. In contrast, the continuity of our density estimate and the online buffer updates

facilitate DDS algorithms to achieve slow but continual progress in exploring the feature space (Fig. 10, LP and Arm Repertoire). This is because the shape of the continuous density estimate always changes slightly after each iteration of the algorithm. Therefore, the adaptive optimizer CMA-ES always gets some signal to move away from the dense regions in feature space.

For multi-feature LP, we observe that algorithms leveraging continuous representations of the feature space, i.e., DDS and NS, explore the feature space much faster than other algorithms driven by discrete feature space representations, e.g., CMA-MAE (Fig. 10). This is because CMA-MAE is optimizing on a centroidal Voronoi tessellation with 10,000 cells, where each cell has significantly more volume compared to that of a 100×100 grid due to the increased dimensionality of the feature space. Hence, more solutions map to the same cells, making it more difficult to find solutions outside of previously explored cells.

A similar phenomenon was observed in prior work when increasing the dimensionality of the solution space in the LP domain [46]. Increased solution dimensionality more heavily distorts the feature mapping and, similarly, causes most solutions to fall in the same cells of the feature space. Consequently, the LP domain becomes exceptionally challenging for QD algorithms working with a tessellation (like CMA-MAE), as there is insufficient signal to drive the algorithm towards the boundaries of the feature space.

DDS-KDE overcomes this drawback of utilizing tessellations with its continuous density estimation of the feature space. While in tesselation-based algorithms, solutions will fall into the same cell, DDS-KDE will retain the solutions in its buffer, which generates signal to drive the search towards the feature space boundaries. Thus, DDS-KDE is more resilient to the distortions of the feature mapping and can better scale with the dimensionality of the feature space.

Cross-Entropy. DDS-KDE and DDS-CNF outperform[3] all baselines in LP, with the exception of DDS-CNF, whose performance is not significantly different from CMA-MAE. On Arm Repertoire and Deceptive Maze, DDS-KDE and DDS-CNF outperform NS. However, while DDS-KDE is on

[3]Recall that lower CE is better as it indicates a more uniform exploration of the feature space (Sec. 6.5.1)

par with CMA-MAE on Arm Repertoire , CMA-MAE outperforms DDS-KDE on Deceptive Maze and DDS-CNF on both Arm Repertoire and Deceptive Maze. Finally, DDS-KDE outperforms all algorithms on Multi-feature LP, while DDS-CNF outperforms CMA-MAE and MAP-Elites (line).

We attribute the improved performance of CMA-MAE to the nature of the cross-entropy metric. Cross-entropy is intended to approximate the distributional similarity between the exploration of the feature space and the uniform distribution. CMA-MAE directly optimizes a passive archive with uniform tessellation, unlike DDS and NS, which are agnostic to the passive archive. This experimental setup naturally favors CMA-MAE with respect to the cross-entropy metric.

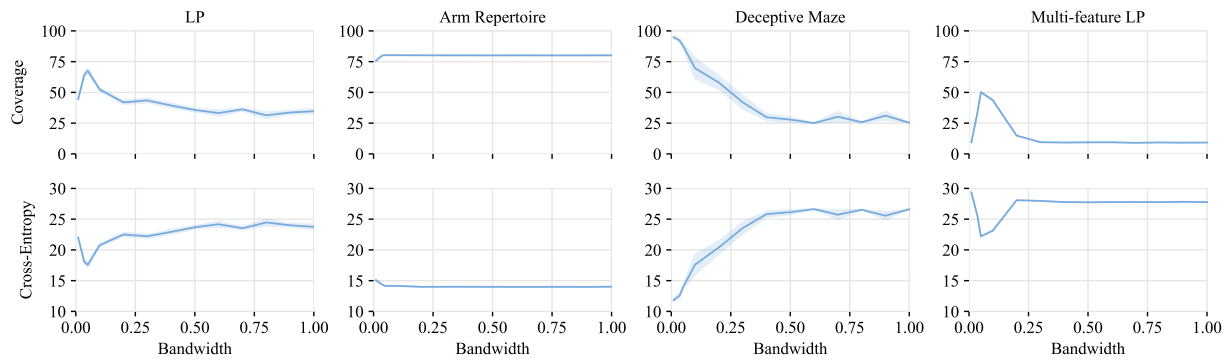
DDS-KDE performs as well as DDS-CNF in terms of coverage and cross-entropy across most domains. However, on Multi-feature LP, DDS-KDE outperforms DDS-CNF on both metrics; on LP and Deceptive Maze, DDS-KDE outperforms DDS-CNF in terms of cross-entropy.

We attribute the difference between the performance of DDS-KDE and DDS-CNF to the bandwidth parameter in KDE, which allows us to adjust the smoothness of the KDE (Fig. 7). On the other hand, CNF lacks explicit control over its smoothness. Thus, while we adjust the smoothness of the KDE to improve performance for DDS-KDE (Sec. 6.3), we can not apply the same techniques to boost the performance of DDS-CNF.

6.6 Bandwidth Selection for DDS-KDE

The bandwidth parameter h in KDE controls the smoothness of the estimation (Fig. 7). While prior work [69–71] has developed bandwidth selection methods for accurate density estimations, our theoretical results show that the optimal bandwidth for DDS-KDE may require an over-smoothing of the KDE (Sec. 6.3).

To examine the effect of the bandwidth parameter on the performance of DDS-KDE, we perform ablation experiments on LP, Arm Repertoire, and Deceptive Maze. While applying the same hyperparameters as the experiments in Sec. 6.5, I vary the bandwidth h in the interval $(0, W]$, where W is the bound of the feature space defined for each domain (Sec. 6.5.2). For instance, in the LP



	LP		Arm Repertoire		Deceptive Maze		Multi-feature LP	
	Coverage	Cross-Entropy	Coverage	Cross-Entropy	Coverage	Cross-Entropy	Coverage	Cross-Entropy
$h_0 = 0.01$	44.57 ±0.75%	21.99 ±0.14	75.75 ±1.06%	15.07 ±0.22	**94.93 ±0.84%**	**11.84 ±0.21**	9.50 ±0.15%	29.33 ±0.02
$h_0 = 0.035$	64.00 ±1.82%	18.10 ±0.37	79.84 ±0.20%	14.39 ±0.09	92.22 ±1.44%	12.56 ±0.37	33.17 ±0.26%	25.57 ±0.05
$h_0 = 0.05$	**67.67 ±2.13%**	**17.57 ±0.41**	**80.22 ±0.01%**	14.14 ±0.02	87.70 ±1.60%	13.98 ±0.35	**50.22 ±0.45%**	**22.20 ±0.09**
$h_0 = 0.1$	52.41 ±1.18%	20.75 ±0.22	80.20 ±0.01%	14.14 ±0.04	69.63 ±8.59%	17.60 ±1.69	43.61 ±0.81%	23.13 ±0.15
$h_0 = 0.2$	41.90 ±1.69%	22.50 ±0.33	80.15 ±0.00%	13.99 ±0.02	57.95 ±5.43%	20.41 ±1.03	14.88 ±0.18%	28.08 ±0.02
$h_0 = 0.3$	43.48 ±1.79%	22.22 ±0.32	80.10 ±0.01%	14.01 ±0.02	41.98 ±5.51%	23.54 ±0.98	9.53 ±0.12%	27.97 ±0.03
$h_0 = 0.4$	39.37 ±2.00%	22.92 ±0.40	80.10 ±0.01%	14.00 ±0.01	29.77 ±2.80%	25.81 ±0.47	9.24 ±0.09%	27.78 ±0.03
$h_0 = 0.5$	35.74 ±1.49%	23.68 ±0.28	80.09 ±0.00%	13.99 ±0.02	27.90 ±2.12%	26.14 ±0.37	9.39 ±0.19%	27.74 ±0.04
$h_0 = 0.6$	33.19 ±2.39%	24.17 ±0.44	80.10 ±0.01%	13.99 ±0.01	25.04 ±0.24%	26.63 ±0.03	9.42 ±0.11%	27.78 ±0.02
$h_0 = 0.7$	36.20 ±1.31%	23.52 ±0.28	80.09 ±0.01%	13.99 ±0.01	30.23 ±4.19%	25.73 ±0.77	9.02 ±0.16%	27.78 ±0.03
$h_0 = 0.8$	31.35 ±2.65%	24.46 ±0.54	80.09 ±0.00%	14.00 ±0.01	25.74 ±0.32%	26.53 ±0.06	9.29 ±0.17%	27.77 ±0.04
$h_0 = 0.9$	33.66 ±1.61%	24.00 ±0.30	80.10 ±0.00%	**13.98 ±0.01**	31.08 ±3.55%	25.56 ±0.64	9.13 ±0.18%	27.82 ±0.04
$h_0 = 1$	34.69 ±2.13%	23.75 ±0.41	80.10 ±0.00%	14.02 ±0.01	25.34 ±0.27%	26.61 ±0.04	9.23 ±0.13%	27.77 ±0.03

Figure 11: Coverage and cross-entropy (CE) after 5,000 iterations of DDS-KDE in all domains for each normalized bandwidth h_0. We report the mean over 10 trials (3 trials for Deceptive Maze) with error bars showing the standard error of the mean. Higher coverage and lower cross-entropy are better. Highlighted cells are results from the main experiments in Fig. 10. The plots show the normalized bandwidth on the x-axis

domain, $W_{\text{LP}} = 5.12 \cdot \frac{n}{2}$ where n is the dimensionality of the solution space. For conciseness and consistency, I report the results in terms of the *normalized bandwidth* $h_0 = \frac{h}{W_{\text{domain}}}$ such that $h_0 \in (0, 1]$, where W_{domain} denotes the feature space bound for the particular domain.

A performance peak can be observed across all domains around $h_0 = 0.05$ (Fig. 11). For LP, Deceptive Maze, and Multi-feature LP, DDS-KDE's performance exhibits diminishes as the bandwidth increases for $h_0 \geq 0.05$. For Arm Repertoire, DDS-KDE achieves near-optimal coverage for all $h_0 \geq 0.05$.

6.7 Summary of Results

My experimental results demonstrate the both DDS variants excel at discovering diverse solutions, establishing a strong connection between continuous density estimation and diversity optimization. This relation is further explored and justified with theoretical analysis.

The overcoming strong distortion — the covergence of the archive when many solution map to the same cell — is an open problem discussed in [46]. I introduced a new domain to QD by generalizing the canon sphere benchmark to higher-dimensional feature spaces. In this domain, high feature dimension correlates with strong distortion. This domain presented significant challenges for baseline algorithms such as CMA-MAE and NS Sec. 6.5, but DDS-KDE only suffered minor performance loss on 10 dimensional feature space, outperforming all baselines at exploring on the feature spaces. This is another step closer to overcoming distortion caused by the feature mapping.

While DDS is a DO algorithm, the underlying insight of DDS — leveraging continuous density representations to search the feature space — can be applied to improve the exploration power of general QD algorithms, especially in high-dimensional feature spaces. Currently, even on QD domains high-dimensional feature spaces, CMA-MAE obtains poor results (ongoing work with ICAROS lab). However, by incorporating quality in addition to the density estimation, one may be able to overcome the distortions induced by high-dimensional feature spaces.

7 Conclusion

My work on pyribs will continue support conception of new QD algorithms and bring more people into the field of QD. According to Google Scholar, pyribs has already been cited at least 30 times. In fact, my work on DDS, from prototype to final experiments, was all completed using pyribs. My other ongoing research projects are also utilizing pyribs to facilitate algorithm prototyping.

My work on DDS makes significant progress towards resolving the open problem of performing QD under feature spaces with strong distortion. Continuity is the key ingredient here. Although my algorithm is only designed for DO settings, the concrete connection it estalishes between continuous density estimation and diversity optimization through both empirical experiments and theoretical analysis is also applicable in QD settings. One just need to find a proper way to incorporate quality into DDS, or incorporate DDS into QD algorithms. DDS also offers improvement on classical two-dimensional DO domains, outperforming the baseline algorithms on three out of the four domains.

8 References

[1] B. Tjanaka *et al.*, "Pyribs: A bare-bones python library for quality diversity optimization," in *Proceedings of the Genetic and Evolutionary Computation Conference*, ser. GECCO '23, New York, NY, USA: Association for Computing Machinery, 2023, pp. 220–229, ISBN: 9798400701191. DOI: 10.1145/3583131.3590374. [Online]. Available: https://doi.org/10.1145/3583131.3590374.

[2] D. H. Lee, A. V. Palaparthi, M. C. Fontaine, B. Tjanaka, and S. Nikolaidis, "Density descent for diversity optimization," in *Proceedings of the Genetic and Evolutionary Computation Conference*, ser. GECCO '24, New York, NY, USA: Association for Computing Machinery, 2024.

[3] J. Lehman and K. O. Stanley, "Abandoning objectives: Evolution through the search for novelty alone," *Evolutionary computation*, vol. 19, no. 2, pp. 189–223, 2011.

[4] S. Huang *et al.*, "Open rl benchmark: Comprehensive tracked experiments for reinforcement learning," *arXiv preprint arXiv:2402.03046*, 2024.

[5] T. Karras, S. Laine, M. Aittala, J. Hellsten, J. Lehtinen, and T. Aila, "Analyzing and improving the image quality of stylegan," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 8110–8119.

[6] J. K. Pugh, L. B. Soros, and K. O. Stanley, "Quality diversity: A new frontier for evolutionary computation," *Frontiers in Robotics and AI*, vol. 3, p. 40, 2016, ISSN: 2296-9144. DOI: 10.3389/frobt.2016.00040. [Online]. Available: https://www.frontiersin.org/article/10.3389/frobt.2016.00040.

[7] J. Lehman and K. O. Stanley, "Evolving a diversity of virtual creatures through novelty search and local competition," in *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '11, Dublin, Ireland: Association for Computing Machinery, 2011, pp. 211–218, ISBN: 9781450305570. DOI: 10.1145/2001576.2001606. [Online]. Available: https://doi.org/10.1145/2001576.2001606.

[8] B. G. Woolley and K. O. Stanley, "A novel human-computer collaboration: Combining novelty search with interactive evolution," in *Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '14, Vancouver, BC, Canada: Association for Computing Machinery, 2014, pp. 233–240, ISBN: 9781450326629. DOI: 10.1145/2576768.2598353. [Online]. Available: https://doi-org.libproxy1.usc.edu/10.1145/2576768.2598353.

[9] A. Gaier, A. Asteroth, and J.-B. Mouret, "Are quality diversity algorithms better at generating stepping stones than objective-based search?" In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, ser. GECCO '19, Prague, Czech Republic: Association for Computing Machinery, 2019, pp. 115–116, ISBN: 9781450367486. DOI: 10.1145/3319619.3321897. [Online]. Available: https://doi.org/10.1145/3319619.3321897.

[10] B. Tjanaka, S. Sommerer, N. Klapsis, M. C. Fontaine, and S. Nikolaidis, "Using cma-me to land a lunar lander like a space shuttle," *pyribs.org*, 2021. [Online]. Available: https://docs.pyribs.org/en/stable/tutorials/lunar_lander.html.

[11] M. Fontaine and S. Nikolaidis, "Differentiable quality diversity," *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[12] A. Radford *et al.*, "Learning transferable visual models from natural language supervision," in *International conference on machine learning*, PMLR, 2021, pp. 8748–8763.

[13] N. R. Balam, B. Tjanaka, D. H. Lee, M. C. Fontaine, and S. Nikolaidis, "Generating tom cruise images with dqd algorithms," *pyribs.org*, 2023. [Online]. Available: https://docs.pyribs.org/en/stable/tutorials/tom_cruise_dqd.html.

[14] A. Cully, *Quality-diversity optimisation algorithms*, https://quality-diversity.github.io, Retrieved 2023-01-31.

[15] O. Nilsson and A. Cully, "Policy gradient assisted map-elites," in *Proceedings of the Genetic and Evolutionary Computation Conference*, ser. GECCO '21, Lille, France: Association for Computing Machinery, 2021, pp. 866–875, ISBN: 9781450383509. DOI: 10.1145/

3449639.3459304. [Online]. Available: https://doi.org/10.1145/3449639.3459304.

[16] C. Colas, V. Madhavan, J. Huizinga, and J. Clune, "Scaling map-elites to deep neuroevolution," in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, ser. GECCO '20, Cancún, Mexico: Association for Computing Machinery, 2020, pp. 67–75, ISBN: 9781450371285. DOI: 10.1145/3377930.3390217. [Online]. Available: https://doi.org/10.1145/3377930.3390217.

[17] B. Tjanaka, M. C. Fontaine, J. Togelius, and S. Nikolaidis, "Approximating gradients for differentiable quality diversity in reinforcement learning," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2022, pp. 1102–1111.

[18] T. Pierrot *et al.*, "Diversity policy gradient for sample efficient quality-diversity optimization," in *Proceedings of the Genetic and Evolutionary Computation Conference*, ser. GECCO '22, Boston, Massachusetts: Association for Computing Machinery, 2022, pp. 1075–1083, ISBN: 9781450392372. DOI: 10.1145/3512290.3528845. [Online]. Available: https://doi.org/10.1145/3512290.3528845.

[19] B. Tjanaka, M. C. Fontaine, A. Kalkar, and S. Nikolaidis, "Training diverse high-dimensional controllers by scaling covariance matrix adaptation map-annealing," *arXiv preprint arXiv:2210.02622*, 2022.

[20] E. Conti, V. Madhavan, F. Petroski Such, J. Lehman, K. Stanley, and J. Clune, "Improving exploration in evolution strategies for deep reinforcement learning via a population of novelty-seeking agents," in *Advances in Neural Information Processing Systems 31*, S. Bengio, H. Wallach, H. Larochelle, K. Grauman, N. Cesa-Bianchi, and R. Garnett, Eds., Curran Associates, Inc., 2018, pp. 5027–5038. [Online]. Available: http://papers.nips.cc/paper/7750-improving-exploration-in-evolution-strategies-for-deep-reinforcement-learning-via-a-population-of-novelty-seeking-agents.pdf.

[21] D. Morrison, P. Corke, and J. Leitner, "Egad! an evolved grasping analysis dataset for diversity and reproducibility in robotic manipulation," *IEEE Robotics and Automation Letters*, vol. 5, no. 3, pp. 4368–4375, 2020. DOI: 10.1109/LRA.2020.2992195.

[22] A. Morel, Y. Kunimoto, A. Coninx, and S. Doncieux, "Automatic acquisition of a repertoire of diverse grasping trajectories through behavior shaping and novelty search," in *2022 International Conference on Robotics and Automation (ICRA)*, 2022, pp. 755–761. DOI: 10.1109/ICRA46639.2022.9811837.

[23] M. Fontaine and S. Nikolaidis, "A quality diversity approach to automatically generating human-robot interaction scenarios in shared autonomy," *Proceedings of Robotics: Science and Systems 17*, 2020.

[24] M. C. Fontaine and S. Nikolaidis, "Evaluating human–robot interaction algorithms in shared autonomy via quality diversity scenario generation," *ACM Transactions on Human-Robot Interaction (THRI)*, vol. 11, no. 3, pp. 1–30, 2022.

[25] M. C. Fontaine, Y.-C. Hsu, Y. Zhang, B. Tjanaka, and S. Nikolaidis, "On the importance of environments in human-robot coordination," *Proceedings of Robotics: Science and Systems 17*, Jul. 2021. DOI: 10.15607/RSS.2021.XVII.038.

[26] M. C. Fontaine *et al.*, "Illuminating mario scenes in the latent space of a generative adversarial network," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 35, 2021, pp. 5922–5930.

[27] S. Earle, J. Snider, M. C. Fontaine, S. Nikolaidis, and J. Togelius, "Illuminating diverse neural cellular automata for level generation," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2022, pp. 68–76.

[28] V. Bhatt, B. Tjanaka, M. Fontaine, and S. Nikolaidis, "Deep surrogate assisted generation of environments," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35, Curran Associates, Inc., 2022, pp. 37 762–37 777. [Online]. Available: https://proceedings.neurips.cc/

paper_files/paper/2022/file/f649556471416b35e60ae0de7c1e3619-Paper-Conference.pdf.

[29] T. Galanos, A. Liapis, G. N. Yannakakis, and R. Koenig, "Arch-elites: Quality-diversity for urban design," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, ser. GECCO '21, Lille, France: Association for Computing Machinery, 2021, pp. 313–314, ISBN: 9781450383516. DOI: 10.1145/3449726.3459490. [Online]. Available: https://doi.org/10.1145/3449726.3459490.

[30] A. Gaier, A. Asteroth, and J.-B. Mouret, "Data-Efficient Design Exploration through Surrogate-Assisted Illumination," *Evolutionary Computation*, vol. 26, no. 3, pp. 381–410, Sep. 2018, ISSN: 1063-6560. DOI: 10.1162/evco_a_00231. eprint: https://direct.mit.edu/evco/article-pdf/26/3/381/1552355/evco_a_00231.pdf. [Online]. Available: https://doi.org/10.1162/evco%5C_a%5C_00231.

[31] T. Endo, H. Abe, and M. Oka, "Toward automatic generation of diverse congestion control algorithms through co-evolution with simulation environments," in *ALIFE 2022: The 2022 Conference on Artificial Life*, Jul. 2022. DOI: 10.1162/isal_a_00515. eprint: https://direct.mit.edu/isal/proceedings-pdf/isal/34/33/2035325/isal_a_00515.pdf. [Online]. Available: https://doi.org/10.1162/isal%5C_a%5C_00515.

[32] J. Verhellen and J. Van den Abeele, "Illuminating elite patches of chemical space," *Chem. Sci.*, vol. 11, pp. 11 485–11 491, 42 2020. DOI: 10.1039/D0SC03544K. [Online]. Available: http://dx.doi.org/10.1039/D0SC03544K.

[33] S. Zhao, *Cabbagecat's blogs*, https://szhaovas.github.io, Retrieved 2023-01-31.

[34] Institute of Digital Games, *Game ai - creative artificial evolution through quality diversity algorithms*, https://www.game.edu.mt/blog/game-ai-creative-artificial-evolution-through-quality-diversity-algorithms/, Retrieved 2023-01-31, Apr. 2019.

[35] K. Frans, *Quality diversity: Evolving ocean creatures*, https://kvfrans.com/quality-diversity-evolving-ocean-creatures/, Retrieved 2023-01-31, Dec. 2020.

[36] O. Mohamed, *Quality-diversity algorithms: Map-polar*, https://towardsdatascience.com/quality-diversity-algorithms-a-new-approach-based-on-map-elites-applied-to-robot-navigation-f51380deec5d, Retrieved 2023-01-31, Mar. 2021.

[37] M. Flageat and B. Lim, *Benchmarking quality-diversity algorithms on neuroevolution for reinforcement learning*, https://aihub.org/2022/12/14/benchmarking-quality-diversity-algorithms-on-neuroevolution-for-reinforcement-learning/, Retrieved 2023-01-31, Dec. 2022.

[38] M. Allard, *Quality-diversity algorithms*, https://maximeallard.lu/2021/03/24/quality-diversity-algorithms/, Retrieved 2023-01-31, Mar. 2021.

[39] D. Wolz, *Fcmaes - a python-3 derivative-free optimization library*, Available at https://github.com/dietmarwo/fast-cma-es, Python/C++ source code, with description and examples, 2022.

[40] A. Cully, J.-B. Mouret, and S. p. Doncieux, "Quality-diversity optimisation," in *Proceedings of the 2020 Genetic and Evolutionary Computation Conference Companion*, ser. GECCO '20, Cancún, Mexico: Association for Computing Machinery, 2020, pp. 701–723, ISBN: 9781450371278. DOI: 10.1145/3377929.3389852. [Online]. Available: https://doi.org/10.1145/3377929.3389852.

[41] A. Cully, J.-B. Mouret, and S. p. Doncieux, "Quality-diversity optimisation," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, ser. GECCO '21, Lille, France: Association for Computing Machinery, 2021, pp. 715–739, ISBN: 9781450383516. DOI: 10.1145/3449726.3461403. [Online]. Available: https://doi.org/10.1145/3449726.3461403.

[42] A. Cully, J.-B. Mouret, and S. p. Doncieux, "Quality-diversity optimisation," in *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, ser. GECCO '22, Boston, Massachusetts: Association for Computing Machinery, 2022, pp. 864–889, ISBN: 9781450392686. DOI: 10.1145/3520304.3533637. [Online]. Available: https://doi.org/10.1145/3520304.3533637.

[43] J. Clune, J. Lehman, and K. O. Stanley, *Recent advances in population-based search for deep neural networks*, ICML 2019 Tutorials, https://youtu.be/g6HiuEnbwJE, 2019.

[44] M. C. Fontaine, J. Togelius, S. Nikolaidis, and A. K. Hoover, "Covariance matrix adaptation for the rapid illumination of behavior space," in *Proceedings of the 2020 genetic and evolutionary computation conference*, 2020, pp. 94–102.

[45] A. Chenu, N. Perrin-Gilbert, S. Doncieux, and O. Sigaud, "Selection-expansion: A unifying framework for motion-planning and diversity search algorithms," in *Artificial Neural Networks and Machine Learning–ICANN 2021: 30th International Conference on Artificial Neural Networks, Bratislava , Slovakia, September 14–17, 2021, Proceedings, Part IV 30*, Springer, 2021, pp. 568–579.

[46] M. Fontaine and S. Nikolaidis, "Covariance matrix adaptation map-annealing," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2023, pp. 456–465.

[47] N. Hansen, "The cma evolution strategy: A tutorial," *arXiv preprint arXiv:1604.00772*, 2016.

[48] J.-B. Mouret and J. Clune, "Illuminating search spaces by mapping elites," *arXiv preprint arXiv:1504.04909*, 2015.

[49] V. Vassiliades and J.-B. Mouret, "Discovering the elite hypervolume by leveraging interspecies correlation," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2018, pp. 149–156.

[50] V. Vassiliades, K. Chatzilygeroudis, and J.-B. Mouret, "Using centroidal voronoi tessellations to scale up the multidimensional archive of phenotypic elites algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 4, pp. 623–630, 2018. DOI: 10.1109/TEVC.2017.2735550.

[51] Y. Akimoto, Y. Nagata, I. Ono, and S. Kobayashi, "Bidirectional relation between cma evolution strategies and natural evolution strategies," in *International Conference on Parallel Problem Solving from Nature*, Springer, 2010, pp. 154–163.

[52] J.-B. Mouret and S. Doncieux, "SFERESv2: Evolvin' in the multi-core world," in *Proc. of Congress on Evolutionary Computation (CEC)*, 2010, pp. 4079–4086.

[53] L. Cazenille, *Qdpy: A python framework for quality-diversity*, `https://gitlab.com/leo.cazenille/qdpy`, 2018.

[54] B. Lim, M. Allard, L. Grillotti, and A. Cully, "Accelerated quality-diversity for robotics through massive parallelism," *arXiv preprint arXiv:2202.01258*, 2022.

[55] C. R. Harris *et al.*, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. DOI: `10.1038/s41586-020-2649-2`. [Online]. Available: `https://doi.org/10.1038/s41586-020-2649-2`.

[56] J. Bradbury *et al.*, *JAX: Composable transformations of Python+NumPy programs*, `http://github.com/google/jax`, version 0.3.13, 2018. [Online]. Available: `%5Curl%7Bhttp://github.com/google/jax%7D`.

[57] A. Cully, "Multi-emitter MAP-elites," in *Proceedings of the Genetic and Evolutionary Computation Conference*, ACM, Jun. 2021. DOI: `10.1145/3449639.3459326`. [Online]. Available: `https://doi.org/10.1145%5C%2F3449639.3459326`.

[58] M. C. Fontaine, S. Lee, L. B. Soros, F. De Mesentier Silva, J. Togelius, and A. K. Hoover, "Mapping hearthstone deck spaces through map-elites with sliding boundaries," in *Proceedings of the Genetic and Evolutionary Computation Conference*, ser. GECCO '19, Prague, Czech Republic: Association for Computing Machinery, 2019, pp. 161–169, ISBN: 9781450361118. DOI: `10.1145/3321707.3321794`. [Online]. Available: `https://doi.org/10.1145/3321707.3321794`.

[59] K. O. Stanley and R. Miikkulainen, "Evolving neural networks through augmenting topologies," *Evolutionary computation*, vol. 10, no. 2, pp. 99–127, 2002.

[60] Y. Zhang, M. C. Fontaine, A. K. Hoover, and S. Nikolaidis, "Deep surrogate assisted map-elites for automated hearthstone deckbuilding," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2022, pp. 158–167.

[61] P. Kent and J. Branke, "Bop-elites, a bayesian optimisation algorithm for quality-diversity search," *arXiv preprint arXiv:2005.04320*, 2020.

[62] Python Software Foundation, *Python Package Index*, `https://pypi.org`.

[63] Anaconda, Inc., *Anaconda*, `https://anaconda.org`.

[64] A. Cully and Y. Demiris, "Quality and diversity optimization: A unifying modular framework," *IEEE Transactions on Evolutionary Computation*, vol. 22, no. 2, pp. 245–259, 2017.

[65] D. H. Lee, B. Tjanaka, M. C. Fontaine, and S. Nikolaidis, "Upgrading cma-me to cma-mae on the sphere benchmark," *pyribs.org*, 2022. [Online]. Available: `https://docs.pyribs.org/en/stable/tutorials/cma_mae.html`.

[66] A. Cully, "Autonomous skill discovery with quality-diversity and unsupervised descriptors," in *Proceedings of the Genetic and Evolutionary Computation Conference*, ser. GECCO '19, Prague, Czech Republic: Association for Computing Machinery, 2019, pp. 81–89, ISBN: 9781450361118. DOI: `10.1145/3321707.3321804`. [Online]. Available: `https://doi.org/10.1145/3321707.3321804`.

[67] G. Paolo, A. Coninx, S. Doncieux, and A. Laflaquière, "Sparse reward exploration via novelty search and emitters," in *Proceedings of the Genetic and Evolutionary Computation Conference*, ser. GECCO '21, Lille, France: Association for Computing Machinery, 2021, pp. 154–162, ISBN: 9781450383509. DOI: `10.1145/3449639.3459314`. [Online]. Available: `https://doi.org/10.1145/3449639.3459314`.

[68] Y.-C. Chen, "A tutorial on kernel density estimation and recent advances," *Biostatistics & Epidemiology*, vol. 1, no. 1, pp. 161–187, 2017.

[69] B. W. Silverman, *Density estimation for statistics and data analysis*. CRC press, 1986, vol. 26.

[70] M. Rudemo, "Empirical choice of histograms and kernel density estimators," *Scandinavian Journal of Statistics*, vol. 9, no. 2, pp. 65–78, 1982. [Online]. Available: http://www.jstor.org/stable/4615859.

[71] A. W. Bowman, "An alternative method of cross-validation for the smoothing of density estimates," *Biometrika*, vol. 71, no. 2, pp. 353–360, 1984. [Online]. Available: http://www.jstor.org/stable/2336252.

[72] S. Weglarczyk, "Kernel density estimation and its application," *ITM Web Conf.*, vol. 23, p. 00 037, 2018. DOI: 10.1051/itmconf/20182300037. [Online]. Available: https://doi.org/10.1051/itmconf/20182300037.

[73] Y. Lipman, R. T. Q. Chen, H. Ben-Hamu, M. Nickel, and M. Le, "Flow matching for generative modeling," in *The Eleventh International Conference on Learning Representations*, 2023. [Online]. Available: https://openreview.net/forum?id=PqvMRDCJT9t.

[74] T. Glasmachers, T. Schaul, S. Yi, D. Wierstra, and J. Schmidhuber, "Exponential natural evolution strategies," in *Proceedings of the 12th annual conference on Genetic and evolutionary computation*, 2010, pp. 393–400.

[75] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015*, Y. Bengio and Y. LeCun, Eds., 2015. [Online]. Available: http://arxiv.org/abs/1412.6980.

[76] J. K. Pugh, L. B. Soros, P. A. Szerlip, and K. O. Stanley, "Confronting the challenge of quality diversity," in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '15, Madrid, Spain: Association for Computing Machinery, 2015, pp. 967–974, ISBN: 9781450334723. DOI: 10.1145/2739480.2754664. [Online]. Available: https://doi.org/10.1145/2739480.2754664.

[77] M. Hardt, B. Recht, and Y. Singer, "Train faster, generalize better: Stability of stochastic gradient descent," in *International conference on machine learning*, PMLR, 2016, pp. 1225–1234.

[78] K.-H. Li, "Reservoir-sampling algorithms of time complexity o(n(1 + log(n/n)))," *ACM Trans. Math. Softw.*, vol. 20, no. 4, pp. 481–493, Dec. 1994, ISSN: 0098-3500. DOI: 10.1145/198429.198435. [Online]. Available: https://doi.org/10.1145/198429.198435.

[79] L. Grillotti and A. Cully, "Kheperax: A lightweight jax-based robot control environment for benchmarking quality-diversity algorithms," in *Proceedings of the Companion Conference on Genetic and Evolutionary Computation*, ser. GECCO '23 Companion, Lisbon, Portugal: Association for Computing Machinery, 2023, pp. 2163–2165, ISBN: 9798400701207. DOI: 10.1145/3583133.3596387. [Online]. Available: https://doi.org/10.1145/3583133.3596387.

[80] J. Gomes, P. Mariano, and A. L. Christensen, "Devising effective novelty search algorithms: A comprehensive empirical study," in *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, ser. GECCO '15, Madrid, Spain: Association for Computing Machinery, 2015, pp. 943–950, ISBN: 9781450334723. DOI: 10.1145/2739480.2754736. [Online]. Available: https://doi.org/10.1145/2739480.2754736.

[81] B. Tjanaka, M. C. Fontaine, and S. Nikolaidis, "Learning a repertoire of robot arm configurations," *pyribs.org*, 2021. [Online]. Available: https://docs.pyribs.org/en/stable/tutorials/arm_repertoire.html.

A Tukey's HSD Pairwise Comparisons

We include results from our statistical analysis in Sec. 6.5. We performed one-way ANOVAs in each domain, shown in Tab. 2. We also performed pairwise comparisons with Tukey's HSD, shown in Tab. 3 and Tab. 4.

Table 3: Pairwise comparisons for **coverage** in each domain. Each entry compares the method in the row to the method in the column. For instance, we can see that DDS-KDE had significantly higher coverage than NS in LP. Comparisons were performed with Tukey's HSD test. $>$ indicates significantly greater, $<$ indicates significantly less, and $-$ indicates no significant difference. $\varnothing$ indicates an invalid comparison.

	LP						Arm Repertoire						Deceptive Maze						Multi-dim LP					
	DDS-KDE	DDS-CNF	NS	CMA-MAE	CMA-ME	MAP-Elites (line)	DDS-KDE	DDS-CNF	NS	CMA-MAE	CMA-ME	MAP-Elites (line)	DDS-KDE	DDS-CNF	NS	CMA-MAE	CMA-ME	MAP-Elites (line)	DDS-KDE	DDS-CNF	NS	CMA-MAE	CMA-ME	MAP-Elites (line)
DDS-KDE	$\varnothing$	$-$	$-$	$>$	$>$	$>$	$-$	$<$	$<$	$<$	$<$	$<$	$\varnothing$	$<$	$<$	$<$	$<$	$-$	$\varnothing$	$<$	$<$	$<$	$<$	$<$
DDS-CNF	$-$	$\varnothing$	$-$	$>$	$>$	$>$	$>$	$\varnothing$	$>$	$>$	$>$	$>$	$>$	$\varnothing$	$>$	$>$	$>$	$>$	$>$	$\varnothing$	$>$	$>$	$>$	$>$
NS	$<$	$<$	$\varnothing$	$>$	$>$	$>$	$<$	$<$	$\varnothing$	$>$	$>$	$>$	$<$	$<$	$\varnothing$	$>$	$>$	$>$	$<$	$<$	$\varnothing$	$>$	$>$	$>$
CMA-MAE	$<$	$<$	$<$	$\varnothing$	$>$	$>$	$<$	$<$	$<$	$\varnothing$	$>$	$>$	$<$	$<$	$<$	$\varnothing$	$>$	$-$	$<$	$<$	$<$	$\varnothing$	$>$	$>$
CMA-ME	$<$	$<$	$<$	$<$	$\varnothing$	$>$	$<$	$<$	$<$	$<$	$\varnothing$	$>$	$>$	$>$	$>$	$<$	$\varnothing$	$>$	$<$	$<$	$<$	$<$	$\varnothing$	$>$
MAP-Elites (line)	$<$	$<$	$>$	$<$	$-$	$\varnothing$	$<$	$<$	$<$	$<$	$<$	$\varnothing$	$-$	$-$	$>$	$<$	$>$	$\varnothing$	$<$	$<$	$<$	$<$	$<$	$\varnothing$

Table 4: Pairwise comparisons for **cross-entropy** in each domain. Note that lower cross-entropy is better, so significantly less ($<$) indicates that a method is significantly better.

	LP						Arm Repertoire						Deceptive Maze						Multi-dim LP					
	DDS-KDE	DDS-CNF	NS	CMA-MAE	CMA-ME	MAP-Elites (line)	DDS-KDE	DDS-CNF	NS	CMA-MAE	CMA-ME	MAP-Elites (line)	DDS-KDE	DDS-CNF	NS	CMA-MAE	CMA-ME	MAP-Elites (line)	DDS-KDE	DDS-CNF	NS	CMA-MAE	CMA-ME	MAP-Elites (line)
DDS-KDE	$\varnothing$	$-$	$-$	$<$	$<$	$<$	$\varnothing$	$>$	$>$	$<$	$<$	$>$	$\varnothing$	$>$	$<$	$<$	$-$	$<$	$\varnothing$	$>$	$>$	$<$	$<$	$<$
DDS-CNF	$-$	$\varnothing$	$-$	$<$	$<$	$<$	$<$	$\varnothing$	$<$	$<$	$<$	$>$	$<$	$\varnothing$	$<$	$<$	$<$	$<$	$<$	$\varnothing$	$>$	$<$	$<$	$<$
NS	$>$	$>$	$\varnothing$	$<$	$<$	$<$	$>$	$>$	$\varnothing$	$<$	$<$	$>$	$>$	$>$	$\varnothing$	$<$	$<$	$<$	$>$	$>$	$\varnothing$	$<$	$<$	$<$
CMA-MAE	$>$	$>$	$>$	$\varnothing$	$-$	$<$	$>$	$>$	$>$	$\varnothing$	$<$	$>$	$>$	$>$	$>$	$\varnothing$	$>$	$<$	$>$	$>$	$>$	$\varnothing$	$<$	$<$
CMA-ME	$>$	$>$	$>$	$-$	$\varnothing$	$<$	$>$	$>$	$>$	$<$	$\varnothing$	$>$	$>$	$>$	$>$	$<$	$\varnothing$	$>$	$>$	$>$	$>$	$>$	$\varnothing$	$>$
MAP-Elites (line)	$>$	$>$	$<$	$<$	$<$	$\varnothing$	$<$	$<$	$<$	$<$	$<$	$\varnothing$	$>$	$>$	$<$	$<$	$<$	$\varnothing$	$>$	$>$	$<$	$<$	$<$	$\varnothing$

B Proofs

Theorem B.1. *A kernel density estimate $\hat{D}_h(\boldsymbol{x}; \mathcal{B})$ managed with reservoir sampling, such that features in the buffer are exchanged one at a time, is $\frac{1}{|\mathcal{B}|h}$-uniformly stable, where h is the bandwidth.*

Proof. Let buffer $\mathcal{B} \subseteq \mathbb{R}^m$ and $\mathcal{B}' = \mathcal{B} \setminus \{\boldsymbol{x}_i\} \cup \{\boldsymbol{x}_j\}$ for $\boldsymbol{x}_i \in \mathcal{B}$ and $\boldsymbol{x}_j \in \mathbb{R}^m$. Consider the maximum difference between $\hat{D}_h(\boldsymbol{x}; \mathcal{B})$ and $\hat{D}_h(\boldsymbol{x}; \mathcal{B}')$:

$$\sup_{\boldsymbol{x} \in \mathbb{R}^d} \left| \hat{D}_h(\boldsymbol{x}; \mathcal{B}) - \hat{D}_h(\boldsymbol{x}; \mathcal{B}') \right| = \sup_{\boldsymbol{x} \in \mathbb{R}^m} \left| \frac{1}{|\mathcal{B}|h} \sum_{\boldsymbol{a} \in \mathcal{B}} K\left(\frac{\|\boldsymbol{x} - \boldsymbol{a}\|}{h} \right) - \frac{1}{|\mathcal{B}|h} \sum_{\boldsymbol{a}' \in \mathcal{B}'} K\left(\frac{\|\boldsymbol{x} - \boldsymbol{a}'\|}{h} \right) \right| \tag{10}$$

$$= \frac{1}{|\mathcal{B}|h} \sup_{\boldsymbol{x} \in \mathbb{R}^m} \left| \sum_{\boldsymbol{a} \in \mathcal{B}} K\left(\frac{\|\boldsymbol{x} - \boldsymbol{a}\|}{h} \right) - \sum_{\boldsymbol{a}' \in \mathcal{B}'} K\left(\frac{\|\boldsymbol{x} - \boldsymbol{a}'\|}{h} \right) \right| \tag{11}$$

$$= \frac{1}{|\mathcal{B}|h} \sup_{\boldsymbol{x} \in \mathbb{R}^m} \left| K\left(\frac{\|\boldsymbol{x} - \boldsymbol{x}_i\|}{h} \right) - K\left(\frac{\|\boldsymbol{x} - \boldsymbol{x}_j\|}{h} \right) \right| \tag{12}$$

$$\leq \frac{1}{|\mathcal{B}|h} \tag{13}$$

Since $\mathcal{B}$ and $\mathcal{B}'$ differ by exactly one element, each term in the summation for Eq. 11 cancels except for $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$. Eq. 13 is due to the fact that any kernel $K(\cdot)$ is bounded between $[0, 1]$ by definition. □

Theorem B.2. *Novelty score $\rho(\boldsymbol{x}; \mathcal{B})$ is $\frac{W}{k}$-uniformly stable, where k is the nearest neighbors parameter in novelty score and $W = \max_{\boldsymbol{s}_1, \boldsymbol{s}_2 \in S} \|\boldsymbol{s}_1 - \boldsymbol{s}_2\|$ is the diameter of the feature space S.*

Proof. Let buffer $\mathcal{B} \subseteq \mathbb{R}^m$ and $\mathcal{B}' = \mathcal{B} \cup \{\boldsymbol{x}_j\}$ for $\boldsymbol{x}_j \in \mathbb{R}^m$. Since we constructed $\mathcal{B}'$ by adding a single element to $\mathcal{B}$, $N_k(\boldsymbol{x}, \mathcal{B})$ and $N_k(\boldsymbol{x}, \mathcal{B}')$ differ by at most one element $\boldsymbol{x}_i \in \mathcal{B}$ and $\boldsymbol{x}_j \in \mathcal{B}'$.

Hence,

$$\begin{aligned}
\sup_{\boldsymbol{x}\in\mathbb{R}^d} |\rho(\boldsymbol{x};\mathcal{B}) - \rho(\boldsymbol{x};\mathcal{B}')| &= \sup_{\boldsymbol{x}\in\mathbb{R}^d} \left| \frac{1}{k} \sum_{\boldsymbol{a}\in N_k(\boldsymbol{x};\mathcal{B})} \|\boldsymbol{x}-\boldsymbol{a}\| - \frac{1}{k} \sum_{\boldsymbol{a}\in N_k(\boldsymbol{x};\mathcal{B}')} \|\boldsymbol{x}-\boldsymbol{a}\| \right| \\
&= \frac{1}{k} \sup_{\boldsymbol{x}\in\mathbb{R}^d} \left| \|\boldsymbol{x}-\boldsymbol{x}_i\| - \|\boldsymbol{x}-\boldsymbol{x}_j\| \right| \\
&\leq \frac{1}{k} \sup_{\boldsymbol{x}\in\mathbb{R}^d} \|\boldsymbol{x}_i - \boldsymbol{x}_j\| \\
&\leq \frac{W}{k}
\end{aligned}$$

□

Theorem B.3. *Let $\mathcal{B} \subseteq \mathbb{R}^m$ be a set of features. Consider the rankings π_{NS} and π_{KDE} on another set of features $\{\boldsymbol{\phi}_1, \ldots, \boldsymbol{\phi}_m\} \subseteq \mathbb{R}^m$ where $\boldsymbol{\phi}_i = \boldsymbol{\phi}(\boldsymbol{\theta}_i)$. We define the rankings as follows: $\pi_{\mathrm{NS}}(i) \geq \pi_{\mathrm{NS}}(j) \iff \rho(\boldsymbol{\phi}_i;\mathcal{B}) \geq \rho(\boldsymbol{\phi}_j;\mathcal{B})$ and $\pi_{\mathrm{KDE}}(i) \geq \pi_{\mathrm{KDE}}(j) \iff \hat{D}(\boldsymbol{\phi}_i;\mathcal{B}) \leq \hat{D}(\boldsymbol{\phi}_j;\mathcal{B})$. We show that $\pi_{NS} = \pi_{KDE}$, when NS has $k = \infty$ (or, equivalently, $k = |\mathcal{B}|$) and KDE has triangular kernel $K(u) = 1 - |u|$ with support over the entire feature space S.*

Proof. Recall the novelty score function,

$$\rho(\boldsymbol{x};\mathcal{B}) = \frac{1}{|\mathcal{B}|} \sum_{\boldsymbol{b}\in\mathcal{B}} \|\boldsymbol{x}-\boldsymbol{b}\|$$

Consider the kernel density estimator with $h = 1$ and triangular kernel,

$$\begin{aligned}
\hat{D}(\boldsymbol{x};\mathcal{B}) &= \frac{1}{|\mathcal{B}|} \sum_{\boldsymbol{b}\in\mathcal{B}} K\left(\|\boldsymbol{x}-\boldsymbol{b}\|\right) \\
&= \frac{1}{|\mathcal{B}|} \sum_{\boldsymbol{b}\in\mathcal{B}} (1 - \|\boldsymbol{x}-\boldsymbol{b}\|) \\
&= 1 - \frac{1}{|\mathcal{B}|} \sum_{\boldsymbol{b}\in\mathcal{B}} \|\boldsymbol{x}-\boldsymbol{b}\| \\
&= 1 - \rho(\boldsymbol{x};\mathcal{B})
\end{aligned}$$

Thus, $\rho(\boldsymbol{\phi}_i; \mathcal{B}) \leq \rho(\boldsymbol{\phi}_j; \mathcal{B}) \iff \hat{D}(\boldsymbol{\phi}_i; \mathcal{B}) \geq \hat{D}(\boldsymbol{\phi}_j; \mathcal{B})$. $\square$