Better, Faster Optimisation

Ethan Maxwell

Discovering input parameters that yield optimal outputs from black-box functions poses a challenge in various domains, including machine learning and robotics applications. These challenges stem from the complex relationships among input parameters and between inputs and outputs, relationships that are unknown to the search algorithm. This means conventional mathematical techniques like gradient descent and differentiation are inapplicable, instead necessitating a systematic trial-and-error exploration of inputs. Numerous algorithms have been developed to address this issue; however, their performance falls significantly short of perfection. Recognising the potential for improvement, the objective of this project has been to design, implement, and evaluate novel algorithms aimed at addressing limitations within existing ones and surpassing their performance. This evaluation necessitated the creation of a testing environment to facilitate robust comparisons between different algorithms. Emphasis has been placed on stochastic methods that harness probability distributions to guide the exploration of potential optimal inputs. Within this scope, CMA-ES and Bayesian Optimisation have both demonstrated success through different techniques, but they also exhibit significant shortcomings. As such, the project explores concepts that leverage the successful aspects of both algorithms to address their flaws and enhance performance. The research has produced two innovative enhancements to these existing algorithms and demonstrates their potential to surpass current performance.

I. INTRODUCTION

Finding which values to input into a complex continuous black box function to yield the global optimum of the function is an important task in various fields, including machine learning and robotic applications [1], [2]. However, since these functions are often derived from complex problems they lack a clear underlying formula, instead having an ill-conditioned and rugged output space [3]. Traditional mathematical techniques such as differentiation and regression are unable to directly optimise these functions due to this complexity [4]. Therefore, an alternative methodology is required to search the space of all potential solutions through trial and error in which points are selected and evaluated to find the global optima [5]. Searching such a complex space through trial and error is a challenging task, which is greatly exacerbated by the curse of dimensionality [4]. The curse of dimensionality means that as the dimension of the problem increases, the solution space expands exponentially, making it increasingly more difficult to search. Additionally, the computational cost of evaluating a single potential point in these complex functions can be prohibitively high, making an exhaustive search impractical [6]. Hence, an algorithm is required that is capable of effectively searching the large, rugged, and ill conditioned solution space for the global optimum, while minimising the number of point evaluations needed.

A. Motivation

Numerous algorithms have been developed to tackle the challenges of high-dimensional black box optimisation. These algorithms employ various techniques, such as stochastic evolutionary methods, derivative free optimisation, and others [3]. However, given the complexity of this problem and the objective of minimising the number of points evaluated, it is challenging to make highly effective algorithms. This leaves significant room for further improvement over existing algorithms. Therefore, the goal of this project is to explore novel ideas for algorithms that can more effectively address the challenges of high-dimensional optimisation. The ultimate objective is to design and implement a new algorithm that surpasses existing approaches in terms of efficiency and effectiveness, contributing to advancements in the field of high-dimensional optimisation algorithms.

Given the multitude of diverse existing approaches to high-dimensional optimisation, the scope of this project becomes broad. This makes it necessary to impose a restriction upon potential new algorithms to establish a clearer research direction. This restriction will be that stochastic methods that utilise probability distributions will be the only algorithms explored in this project. The probability distributions will model predictions of which regions of the solution space are most promising to explore next. This direction includes existing algorithms such as Bayesian optimisation and CMA-ES, which select points for evaluation based on a probabilistic model that predicts lucrative regions of space [6], [7]. The success of these functions at optimisation black box functions highlights how the use of probability distributions can be an effective solution to such problems. So, the algorithms explored within this project utilise a probabilistic model for selecting points to evaluate next, adhering to the defined scope restrictions.

B. Solution

To develop a new algorithm for effective high-dimensional optimization, it was essential to first understand existing algorithms and their approaches to address the challenges at hand. Then, using the concepts learned, new ideas could be

This project was supervised by Marcus Frean (primary) and Andrew Lensen.

created that had the potential to optimize more efficiently. Building upon this knowledge allowed partially or entirely new algorithms to be designed and implemented. These new algorithms could then be evaluated in comparison to existing ones to see if they offered significant improvement.

The algorithms developed in this project draw significant inspiration from both Bayesian Optimization and CMA-ES. Specifically, this project explores concepts that blend the strengths of both algorithms, combining them in a synergistic way to leverage their successful aspects. This involves addressing the shortcomings in one algorithm with the strengths of the other, with the aim of enhancing overall performance.

To allow for the evaluation of new algorithms in comparison to existing algorithms, a testing environment was required that could run and evaluate them as needed. The three metrics each algorithm was assessed on were: 1) How good was the optimum produced? 2) How many points had to be evaluated to find an optimum? 3) How much computation was required to find an optimum? For each algorithm, all three metrics were measured for different optimization problems over a range of dimensions, varying from low numbers (under 20) to high numbers (multiple hundreds). Additionally, each situation had to be run multiple times so that the algorithm's average performance could be calculated, and its consistency could be determined.

For all the data produced, tables and graphics were generated that could be analysed to assess whether any new algorithms could beat the existing algorithms. To successfully solve the problem, any new algorithm created had to be able to consistently beat existing algorithms for either metric 1 or 2 for at least some functions and dimensionality ranges. Additionally, metric 3 had to remain low enough that measuring the values of metrics 1 and 2 remained feasible.

II. Related Work

A. Curse of Dimensionality

One of the key challenges in high-dimensionality scenarios is the curse of dimensionality, which is how spaces become considerably more challenging to handle as their dimensionality rises [8]. The curse of dimensionality poses significant challenges in black box optimisation problems, necessitating effective strategies to prevent rapid degradation of algorithm performance with increasing dimensions [9]. Firstly, as the dimensionality increases, data points become more sparsely distributed across the space, meaning there is a substantial gap between data points. This sparsity makes it difficult to accurately predict the values of the black box function due to an insufficient evaluated points for modelling the function's behaviour. Additionally, as the dimensionality of the problem increases, the number of parameters any model must learn increases substantially to account for the increased dimensions [9]. This means expanding dimensionality creates a challenge, demanding more complex models trained on more sparse data. Furthermore, this increasing dimensionality renders distance values less meaningful, as the distances between data points distributed across the space become more uniform [8]. This poses challenges when performing tasks such as comparing distances, as the compared distances will be far more similar in value, thus providing less information [8]. This places value on techniques that do not rely on distances to make inferences about high-dimensionality spaces.

B. Bayesian Optimisation

One such solution for continuous black box optimisation is Bayesian Optimisation [6]. Bayesian Optimisation employs a Bayesian statistical model, typically based on Gaussian process regression, to predict the behaviour of a complex black box function using all available evaluated data points. Gaussian process regression utilises the evaluated data points to fit a multivariate Gaussian distribution, which can then be used to predict the function's values across the entire space [10]. This predicted distribution is then used to form the acquisition function, that quantifies the likelihood of a given point being the quality optimum within the search space [6]. It combines the predicted values and uncertainties from the Bayesian statistical model, with higher expected values and higher uncertainties suggesting regions that are potentially better than the current optima. Next, the maximum of the acquisition function needs to be located to give the point which has the highest expected improvement over the current optima. Now the point with the highest expected improvement can be evaluated. This allows for the Bayesian statistical model to be updated with the new information, so the new point with the highest expected improvement can be found. This process is repeated until a stopping criterion is met. This produces a model that always attempts to evaluate the point next with the highest expected improvement of the current optimum.

C. Bayesian Optimisation with a Gaussian process regression, and the Curse of Dimensionality

Unfortunately, Bayesian optimisation is highly affected by the curse of dimensionality, rendering it ineffective above approximately twenty dimensions [6]. One of the challenges with Bayesian optimisation is in modelling of predicted values for the black box function across the entire search space [9]. Modelling the entire search space becomes massively more challenging as the dimensionality of the problem increases, requiring a far more complex model to be used. Moreover, Gaussian process regression, commonly used in this process, relies on distances to nearby data points for predicting the black box function. However, information regarding distance to nearby data points degrades significantly at higher dimensionality [9]. This is because distances between data points become relatively larger and more similar as dimensionality increases, making it more challenging to extract meaningful information about these distances.

These challenges of the curse of dimensionality are notably a prominent when optimising the acquisition function [9]. Firstly, due to the sparsity of the data points used to construct the acquisition function, and the increase in similarity of distance between them, the acquisition function becomes increasingly less accurate at higher dimensionality. As a result, significant proportions of the acquisition function become flat plains of

uniform value, making optimisation of it challenging with traditional techniques like DIRECT and gradient descent [5]. Additionally, since the acquisition function incorporates uncertainty, the large flat regions will have high uncertainty due to their high distance from evaluated data points. Consequently, maximising the acquisition function will often result in selecting points in the furthest regions of the search space, as they exhibit the highest uncertainty [9]. This promotes a blind search of far regions of the search space rather than converging towards an optimum.

In summary, the curse of dimensionality has a significant impact on multiple steps in Bayesian optimisation [5]. Consequently, it is crucial to employ techniques to address these challenges if Bayesian optimisation is to remain applicable at high dimensionality. Fortunately, there are approaches available to alleviate these issues. One such approach is dimensionality reduction, for example selecting the most influential dimensions for optimisation or embedding the high-dimensional space into a lower-dimensional one [9]. Even more advanced techniques taking advantage of ideas such as low dimensional feature spaces and sparse axis-aligned subspaces, also focused on reducing the dimensions of the problem [5], [11]. These techniques make assumptions about the data, allowing for a reduction in dimensionality and mitigating the effects of the curse of dimensionality. It is important to note that these dimensionality reduction techniques are applicable to various optimisation techniques, as they do not specifically target the shortcomings of Bayesian optimisation.

D. CMA-ES

A more resent technique that has proven successful in performing high dimensional optimisation is Covariance Matrix Adaptation Evolutionary Strategy (CMA-ES) [7]. CMA-ES has been successfully applied in various fields, including neural networks, thermo-mechanical processing, neural topologies, molecular alignment, and many others [12]. CMA-ES achieves this by maintaining a probability distribution that models the likelihood of a point being the global optimum. It performs this using a multivariate Gaussian distribution. The distribution is represented as:

$$m + \sigma N(0, C) \quad (1)$$

In this equation, m represents the mean of the Gaussian distribution, which indicates the point currently considered most likely to be the global optimum. σ is the step size parameter that affects the spread or proximity of the distribution around the mean [7]. C represents the covariance matrix of the distribution, which contains the predicted variances and covariances between dimensions.

At each step of the CMA-ES algorithm, a small selection of points is sampled from the distribution and evaluated [7]. This evaluation allows for the update of m, σ and C using the newly sampled points. Firstly, to update these values, the new points are ranked based on their evaluated values. Next each point receives a weight in accordance with their rank position, with higher weights being given to points with better evaluations.

Next, m is updated to the weighted average of the new data points. This update functionally shifts the mean, and thus the whole distribution, towards the new highest ranked points to facilitate exploration towards potentially better solutions. Covariance matrix C is updated by comparing the weighted data points to m and produces a new covariance matrix specific for the new data points. This new covariance matrix is then combined with C with linear combination to obtain the updated value of C. This allows the covariance matrix C to capture the directions around the mean m that exhibit the most variance in high-quality evaluated data points.

CMA-ES also incorporates the concept of momentum in the covariance update. Moreover, the step size parameter σ is updated based on the observed movement of the mean m. This means if the mean m is constantly moving a significant amount in a specific direction, the step size will increase to exploit this trend to find the optima more efficiently. Conversely, if the mean is not moving constantly in a specific direction, the step size will be decreased to focus on exploration around the vicinity of the mean's current location.

Overall, this allows CMA-ES to estimate potential locations of the global optima using only a multivariate Gaussian distribution as a model [7]. Additionally, CMA-ES can use small sets of evaluated points to adapt this representation during the optimisation process to converge towards the optimum.

E. CMA-ES and the Curse of Dimensionality

CMA-ES has shown success at in high dimensionality problems as it is less affected by the curse of dimensionality than Bayesian optimisation. Firstly, it does not aim to model the entire search space, instead estimating the location of the global optima using only a multivariate Gaussian distribution. This means CMA-ES uses a simple distribution that still allows for stability, and finite predictions of the variances in the data. Additionally, this allows CMA-ES to make minimal assumptions about the data, avoiding making assumptions about independence or relationships, instead learning them as it goes. This allows CMA-ES to avoid overly complex models, such as the acquisition function required in Bayesian optimisation. Moreover, CMA-ES circumvents directly comparing lengths, which can pose challenges at high-dimensionality [8]. Instead, it employs processes such as ranking points along with measuring their mean, variance, and covariance, which are far less susceptible to the adverse effect of the curse of dimensionality.

Unfortunately, CMA-ES is not immune to the challenges imposed by the curse of dimensionality [13]. Firstly, it is affected by the inherent challenge of increasingly large search spaces. In addition, CMA-ES needs to learn the n×n covariance matrix for a problem with n dimensions [14]. This is a substantial $(n^2 + n)/2$ parameters that need to be learnt, which equates to 500500 parameters for a 1000-dimensional problem. This means massive quantities of data are required to learn these many parameters. This abundance of parameters to learn becomes significant hindrance holding back CMA-ES at hundreds of dimensions.

One solution to the large number of parameters in CMA-ES

is proposed by SEP-CMA-ES [14]. This approach suggests not learning covariances, instead focusing only on learning variances, thereby reducing the parameters to learn down to n for a n-dimensional problem. This reduction in parameters can help mitigate the impact of the curse of dimensionality [13]. However, this simplification assumes that all variables are independent, which may lead to significant degradation of performance when optimising non-separable functions. Since most real world high-dimensional optimisation problems are non-separable functions, the applications of this solution are limited [14].

Additionally, the simplification to a Gaussian distribution does come with limitations. One such limitation is that CMA-ES primarily focuses on convergence towards a single point, specifically the mean of its distribution. This removes its ability to switch to an entirely new point if exploitation of the current point has led to an optimum. In contrast, other optimisation techniques like Bayesian optimisation can perform this switch back to exploration when the expected improvement of the currently explored optimum becomes low. This ability to switch back to exploration is important for exploring the entirety of the search space and to remaining catch in a local optimum. But CMA-ES is instead must restart the learning process after it converges to an optimum if it wishes to see further improvement, disregarding a significant amount of information about evaluated points in the process. This limitation ultimately decreases the quality of CMA-ES in terms of exploration adaptability.

F. Recent CMA-ES Techniques

New techniques aimed at improving CMA-ES are continuously being proposed, one of which involves using a mixture model [15]. In this approach, the traditional Gaussian distribution in CMA-ES is replaced with a collection of small mutation vectors. These mutation vectors together form a mixture model that approximates a Gaussian distribution. This reduces the number of parameters that CMA-ES is required to learn to be reduced by leveraging the simplicity of the small mutation vectors. This helps mitigate the need to learn an entire covariance matrix, reducing what the algorithm is required to learn.

Another promising technique is combining CMA-ES with Monte Carlo Tree Search, a technique with application in complex optimisation problems [16]. This technique entails decomposing CMA-ES into a tree structure, where each node represents a region of space with its own mean and covariance derived from its children. By utilising this tree structure, CMA-ES can effectively search multiple promising regions of space simultaneously as the tree decomposes the space down into promising subregions. Importantly, since the nodes are constructed from their children, this technique does not require significantly more parameters to be learned compared to standard CMA-ES. The success of this technique demonstrates the potential benefits of searching multiple means within CMA-ES.

III. Design

A. Decision CMA-ES

When a run of CMA-ES stops finding better optima, indicating convergence to a local optimum, in the usual approach it restarts, and an alternative solution is sought. An alternative to this iterative technique involves parallelising individual searches. In this parallel approach, multiple CMA-ES searches start from the first generation, and in each successive generation, one CMA-ES algorithm is selected to advance by a generation. The simplest implementation of this parallel algorithm advances all searches uniformly, lacking interaction between them. However, under these conditions, if a predetermined number of searches are executed, the result remains the same as in the iterative approach. In this scenario, the predetermined number of searches is performed without interaction among them, resulting in the same outcomes.

In the parallel approach, there is flexibility in selecting which search to advance in each generation as they do not need to be advanced uniformly. This presents an opportunity to incorporate the Bayesian principle of expected improvement into CMA-ES. Expected improvement can guide the selection of which search to advance in the next generation based on its current performance, producing a decision CMA-ES algorithm. This allows the best-performing search to be favoured for advancement. Then as a search converges to a local optimum and its performance stagnates, its expected improvement diminishes and should be selected less for advancement. This will then allow for the opportunity of another search to be advanced which have not yet converged to an optimum to be selected. This produces a dynamic system where the search with the highest expected improvement is advanced each generation.

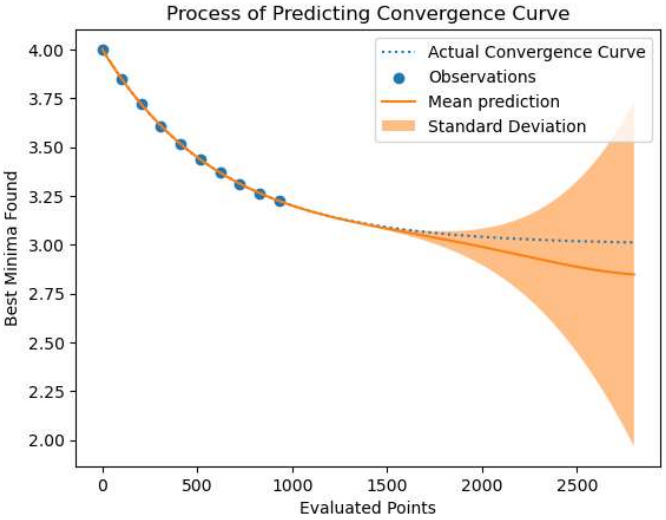

Fig. 1. The process of predicting the convergence curve of a minimisation problem.

To calculate the expected improvement for each individual search in this context, a crucial shift from the Bayesian optimisation paradigm is required. In Bayesian optimization, expected improvement is typically computed using a prediction

of the objective function being optimized. However, in our case, we are predicting the convergence curve itself, as demonstrated in Fig. 1.

This shift in focus means the objective differs from Bayesian Optimisation, where the primary aim is to identify the maximum of an acquisition function. In our context, the goal is to choose the predicted convergence curve that has the best expected improvement. This selection process involves pinpointing a specific point within the predicted convergence curves to assess and determine which one demonstrates the most significant expected improvement.

If we consider the expected improvement of advancing just one generation, the algorithm will invariably favour the search with the best optima observed so far. This is because, in the immediate next generation, the predicted convergence curve for the search with the best optima is likely to be the lowest. However, when looking ahead by more than one generation, searches that predict greater improvements, even if their current optima are suboptimal, will be favoured. This effect can be seen in fig. 1.

To strike a balance between exploring poorly explored regions and exploiting areas with known quality optima, we must look ahead by an appropriate number of generations. This concept appears in other optimization algorithms, which often start by favouring exploration and gradually shift toward exploitation as they progress [17]. It would then logically follow that decision CMA-ES would benefit from dynamically adjusting how many generations ahead is based on the known quantity of points that can be evaluated. When the algorithm is expected to run for many generations, considering several generations ahead is appropriate. Conversely, when the algorithm has only a few more generations to run, a more immediate comparison of expected improvements is warranted.

In Bayesian optimization, Gaussian Processes are typically employed to model and predict the solution space, enabling the computation of expected improvement. In this context, these techniques can predict the convergence curve. However, a challenge arises when using Gaussian processes in this manner, because the function produced converges to a preset value when far the observed values in the convergence curve. While this value is traditionally set to the mean of the observed values, this assumption is incorrect, as convergence curves do not converge toward their own average. Instead, they tend to converge to a value equal to or lower than their current value. To account for this, the preset value would need to be set lower than the current optima for the search, becoming the assumed asymptotic convergence point. This, however, would require predicting what the convergence curve converges to through another method, potentially introducing inaccurate assumptions. Furthermore, when considering a sufficiently large number of generations ahead, all searches are assumed to converge toward their respective assumed convergence point. In such a scenario, the potentially incorrect assumed convergence points significantly influence the expected improvement calculations. For instance, if the same convergence point were selected for all searches, then, when predicting far enough ahead, all searches would yield nearly identical expected improvements. This renders the expected improvement metric essentially worthless as a predictor.

Essentially what is needed is a model capable of producing predicted mean and variance values, so that expected improvement can be calculated. Other, regression models can be employed to predict the convergence curve. The simplest approach involves using Bayesian linear regression, but this assumes linearity, which may not hold true for convergence curves. To mitigate the effects of this assumption, only the most recent generations from the convergence curve can be used for linear regression, thereby assuming the search's optima will continue to improve at its recent rate.

Alternatively, other regression models, such as exponential decay models, might be used to incorporate behaviour akin to that of convergence functions and offer more flexibility regarding curve shape. However, these models are more complex to fit the linear regression while still making significant assumptions about the shape of the convergence curve [18].

B. Mixture CMA-ES

In the parallel CMA-ES structure, each search is viewed as an isolated process. However, this assumption of isolation likely incorrect, as the other searches running in parallel are within the same search space and potentially overlap with each other. This is notably the case when the algorithm is performing exploration of the search space, or multiple searches have converged to the same region of the search space. This means data from within one search could hold useful information for updating other distributions during both exploration and exploitation.

The independent distributions might be view as a single amalgamated distribution. Given that the individual distributions are Gaussian, this amalgamation creates a Gaussian mixture model that encompasses the entire search space. This unified distribution facilitates the sampling of data points for evaluation, and these evaluated data points can be employed to update the entire model as required, as opposed to updating just a single Gaussian distribution.

The updating technique for CMA-ES distributions is specific to individual Gaussian distributions within this mixture model. Each model should be updated with relevant data from within its own distribution, as some data points may lie far outside of it and would not be helpful in the updates. For example, if all distributions were to consider all data points relevant then all the distributions will immediately become identical, as they are using the same process to update their means and covariances. This immediate convergence of all means together when all data is share can be seen in Fig. 2.

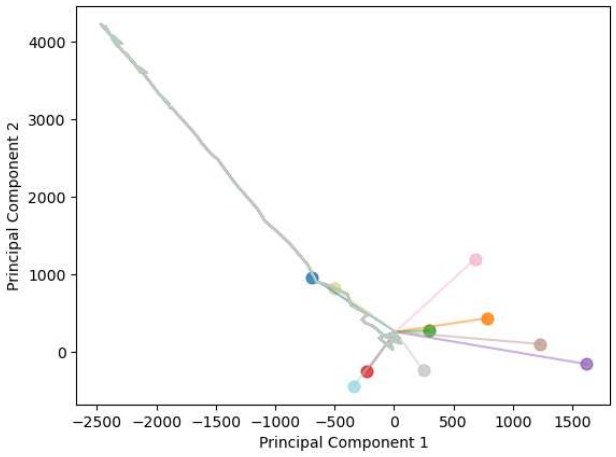

Fig. 2. Mean's converging together when data is shared, dimensionality has been reduced to 2 dimensions with PCA.

To determine the appropriate data points to update each search, we need to establish the responsibility of each distribution for each data point, to weight how much each distribution should be updated with each data point.

From a Bayesian perspective, our goal is to calculate the posterior probabilities, which represent the updated probabilities of each data point's association with the distributions after observing the data. This process involves two key components: the likelihood and the prior. The likelihood measures how well a data point aligns with each distribution and is represented by the Probability Density Function (PDF) values for each distribution evaluated at the data point. This step assesses the compatibility of the data point with each distribution.

In the absence of prior information or prior knowledge about the origin of the data point, we adopt a uniform prior, assigning equal probability to all distributions before data observation. This choice maintains a neutral standpoint and prevents bias in favour of any specific distribution. We apply Bayes' theorem to combine the likelihood and the prior to determine the posterior probabilities in the standard way, as implemented in Gaussian mixture models [19].

The Bayesian inference process involves updating our initial beliefs, represented by the uniform prior, with the information obtained from the likelihood, which reflects the observed data. To derive meaningful probabilities for each distribution, we utilize the softmax function as a normalization step. The softmax function transforms the PDF values (indicating likelihood) into probabilities (known as responsibilities). Crucially, this normalization ensures that the probabilities sum to one across all distributions, providing a coherent representation of the likelihood that each data point belongs to each distribution.

Unfortunately, this technique is susceptible to a problem from the high dimensionality of the input domains to which the algorithm is applied. Notably, the probability density of one standard deviation away from the mean at a given dimensionality can be modelled as follows:

$$P = \frac{1}{(2\pi)^{\frac{d}{2}}} e^{-\frac{d}{2}} \quad [20] \ (2)$$

Where P is the probability density one standard deviation from the mean, and d is the number of dimensions. This equation exhibits exponential decay, and by the time 200 dimensions are reached, the probability density at one standard deviation is extremely small, about 5.66×10^{-124}. These minuscule values fall outside the range of floating-point precision used by Python, the language in which the algorithm is implemented. Moreover, they cannot be used in the softmax function because exponentiating such values would demand even greater precision. To circumvent this issue, the use of log PDF values is preferred. Logarithmic values operate on a more stable logarithmic scale, preserving numerical precision while maintaining relative relationships between PDF values.

However, transitioning to the log scale alone is insufficient, especially at high dimensionality. For example, the log PDF value one standard deviation away from the mean in 200 dimensions is still log(5.66 × 10^-124), which equals -281. Due to the magnitude of these probabilities and the exponential nature of the softmax function, small differences in log PDF values can lead to significant disparities in the calculated responsibilities. As a result, it's highly likely that all responsibilities, except for one, will be rounded to 0, *making data points become unambiguously associated with a particular Gaussian*. Such responsibility values are not useful for effectively distributing data, as each data point will contribute to only one Gaussian distribution. Therefore, it becomes necessary to scale these probabilities toward 0. A natural way to achieve the required softening of probability values is to make and adjustment to the softmax as follows:

$$\sigma(z)_i = \frac{e^{\beta z_i}}{\sum_{j=1}^{K} e^{\beta z_j}} \quad (3)$$

Where σ is the softmax function, z represents the log probability density value, K is the number of algorithms, and β is the base used in the calculations. β must be within the range of $0 \leq \beta \leq 1$ to scale the probabilities effectively. Decreasing the value of β concentrates probability distributions around the positions with the smallest input values. Setting β to 1 maintains the same softmax function as before, which is unlikely to yield useful responsibilities. Setting β to 0 results in equal usage of data by all distributions, causing full information sharing and the convergence of all CMA-ES searches after one generation. Determining an appropriate β value is essential to balance data sharing in the softmax equation. This introduces an additional hyperparameter into the multi-CMA-ES algorithm, and dynamically setting this hyperparameter poses significant challenges. These challenges stem from the remarkable differences in PDF values produced by different problems and dimensionalities, which must be appropriately reduced to achieve correct data sharing tailored to each specific problem.

Another possible approach to determine responsibility, like how CMA-ES assigns weights to higher fitness points, involves adopting a ranking-based system. In this method, each data point is assigned a responsibility value based on its rank within the distribution. The point with the highest probability density

receives the highest weight, and this ranking-based approach is followed in descending order for the remaining points. This means that the magnitude of the probability values would not be considered; rather, it focuses on whether values are higher or lower than each other. Consequently, it eliminates the need for a scaling parameter like β.

However, this technique comes at a cost – it results in the loss of information about the actual magnitude of probability density values between different points. For instance, two points may have significantly different probability density values, but if there are probability density values between them, they will be assigned similar ranks and, consequently, similar responsibilities. This loss could have a detrimental impact on the calculation of accurate responsibilities. In this ranking-based system, points may be incorrectly assigned large or small responsibilities solely based on the quantity of points with higher or lower probability densities than themselves.

Now that the responsibilities have been found; they can be used to dictate what data is to be used to update each search. Fortunately, CMA-ES already incorporates point weighting based on fitness, so we can apply additional weighting to each point based on responsibilities directly. This can be done as:

$$w^*_i = \frac{w_i \cdot p_i}{\sum_{j=1}^{N} w_j \cdot p_j} \quad (4)$$

Using the original fitness-based weight values of w, and the responsibility values of p, the new weight values w^*, can be found for point i, for the N different data points. This allows each CMA-ES search favour point that when its own distribution when updating its distribution.

C. Testing Environment

To facilitate the proper evaluation of the designed algorithms, a testing environment required establishment. The testing environment must allow for many existing and designed algorithms to be run on a variety of toy problems to see which preforms better and to highlight intricacies of how the algorithm works. More specifically this includes visualisations of how the different convergence curves compare across multiple runs, and how the one or more Gaussian distributions are changed over time. It must also document computation times of the algorithms to ensure they maintain computational feasibility.

IV. Implementation

A. Testing Environment

The programming language for the testing environment is exclusively python [21]. It has been selected as it is the standard programming language used in machine learning tasks, such as this project, thus has the best libraries and other tools available for utilisation within this project. Such libraries that will be used extensively include NumPy, scikit-learn, pandas and matplotlib, as they are standard libraries for use in machine learning tasks. Much of the results are presented in Jupyter Notebooks [22]. Jupyter Notebooks are an ideal platform for enhancing data visualisation and results presentation in the testing environment due to their interactive and versatile nature, facilitating the seamless integration of code, visualisations, and explanatory text in a single document.

The testing environment itself will heavily utilise the pymoo library [23]. Pymoo was selected as it is a library designed for performing black box optimisation tasks such as those within the project. This means it can be used as the basis for the testing environment, removing the requirement to create code for executing newly created algorithms, as well as large quantities of pre implemented optimisation algorithms and problems that are invaluable for evaluation of the new algorithms.

The evaluation of implemented algorithms requires test problems that replicate the high-dimensional, multi-optima structure found in the black-box functions these algorithms are designed to optimise. These test problems should also have the advantage of being computationally inexpensive to evaluate, ensuring efficient testing. To meet these criteria, the Schwefel, Rastrigin, Ackley, and Griewank functions have been selected [24]. They offer a diversity of problems with many optima and can be scaled to higher dimensions.

B. Ensuring Fair Testing

The testing environment must prioritize fairness in its comparisons to enable meaningful conclusions to be drawn from the results. Firstly, all convergence curves will be generated from many runs of the algorithm being tested to ensure that the performance is accurately represented, and robustness is assessed across multiple runs. Furthermore, the quantity of evaluated points as the common x-axis for all performance comparisons. This approach aligns with the primary objective of high-dimensionality optimization algorithms, which is to minimize the number of evaluated points, a critical and costly step, thereby ensuring meaningful comparisons among algorithms.

Additionally, all algorithms use an identical implementation of CMA-ES that is based on the techniques presented in [7]. Additionally, all initial populations are generated with Latin Hypercube Sampling, and simple and effective way of generating diverse samples a high dimensionality [25]. To further ensure the CMA-ES searches are the same, identical hyperparameters used in all, see Table I for details the values.

Table I

Hyperparameters used by all CMA-ES algorithms.

Hyperparameter	**Set value**
μ/λ Populations size	200
Initial σ value, Starting standard deviation of sampled points	Problem Range
C_σ Decay rate for the cumulation path for the step-size control	0.047
c_c Decay rate for cumulation path for the rank-one update of the covariance matrix	0.17
c_1 Learning rate for the rank-one update of the	0.096

covariance matrix update	
c_μ Learning rate for the rank-μ update of the covariance matrix update	0.082
d_σ Damping parameter for step-size update	9.27

The selected hyperparameter show in Table I allow for the are suitable to effectively preform optimisation on the problems used in the evolution of the algorithms.

C. *Ensuring Fair Testing in Decision CMA-ES*

For the evaluation the algorithms that uses expected improvement to select CMA-ES to advance next, a few additional things need to be ensured to make the tests are fair. Notably evaluations will be based on comparison to basic iterative and parallel CMA-ES algorithm shown in Fig. 3. This means that the evaluation will be able to conclude if using expected improvement offers significant improvement when running CMA-ES in parallel. Furthermore, it will allow for the evaluation of if the changes are enough to out preform the standard iterative approach.

These tests will need to be run for enough evaluate points so that at least three interactive CMA-ES searches can converge to an optimum so that full effects of running the CMA-ES Searches multiple times can be observed.

D. *Ensuring Fair Testing in Mixture CMA-ES*

The goal of the Mixture CMA-ES algorithm is to assess if running training multiple searches with the same data pools an improvement only a running a single CMA-ES algorithm. As such performance comparisons should be made to a standard implementation of CMA-ES.

E. *Algorithm Implementation Details*

The Decision CMA-ES implementation relies on Bayesian linear regression for expected improvement calculations. This was chosen for its simplicity to avoid overly complex computations and minimize the number of assumptions required, as discussed.

In the case of Mixture CMA-ES, the implementation employs the softmax function on log probability density values, ensuring accurate calculation of responsibility values. The only assumption in this approach is the selection of a β value for scaling responsibilities. A specific β value of 5.62×10^{-5} has been consistently chosen for all tests. This selection ensures that Mixture CMA-ES can be applied effectively to a range of problems without the responsibility values losing their meaningful information, as discussed.

V. Evaluation

A. *Decision CMA-ES*

Before the Decision CMA-ES algorithm can be evaluated, first the simple parallel algorithm and iterative algorithms used as benchmarks need to be investigated.

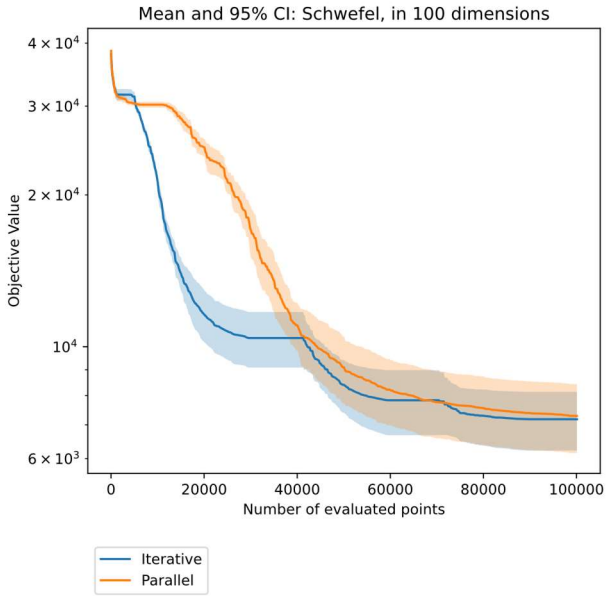

Fig. 3, Running three CMA-ES searches parallelly and iteratively.

Fig. 3 shows running CMA-ES searches in parallel does not provide significant advantages when compared to running them iteratively. The same quality minima are achieved only after 100,000 evaluated points, because the mean and confidence interval of the two algorithms become the same at that point. Unfortunately, a notably worse results occur at fewer than 40,000 evaluated points were the confidence interval for parallel CMA-ES is fully above iterative. So, Running the basic CMA-ES search in parallel, while not leading to noticeable improvements, maintains a consistent outcome, albeit with a slightly slower convergence rate.

But is the parallel CMA-ES is expanded to use the expected improvement by implementing the decision CMA-ES algorithm. If the expected improvement in incorporated into the algorithm's performance notable changes.

Parallel CMA-ES vs Decision CMA-ES

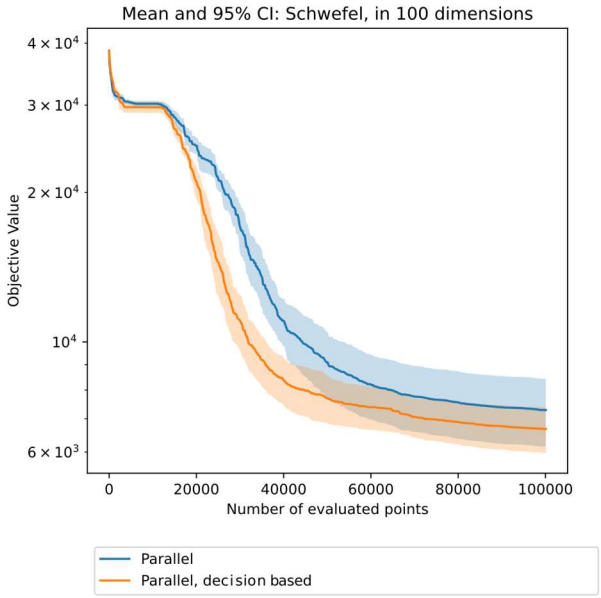

Fig. 4. Convergence of basic Parallel CMA-ES compared to Decision CMA-ES in 100 dimensions across 10 runs on the Schwefel problem.

Fig. 4 illustrates that during the initial phase of the algorithm's execution, Decision CMA-ES consistently identifies lower optima than the basic Parallel CMA-ES alternative. This difference becomes particularly evident between 20,000 and 40,000 evaluated points, as the confidence intervals of the two algorithms do not overlap in this region. This suggests that Decision CMA-ES exhibits a clear advantage in performance during the early stages of the optimization process.

However, beyond 40,000 evaluated points, an interesting shift occurs. The confidence intervals begin to overlap, and by the time 60,000 evaluated points is reached, the means of both algorithms fall within each other's 95% confidence intervals. This indicates that, in the latter half of the algorithm's execution, Decision CMA-ES, on average, continues to perform better than Basic Parallel CMA-ES but not to a statistically significant extent.

If the dimensionality of the problem is scaled from 100 to 200 dimensions, the result remains remarkably similar.

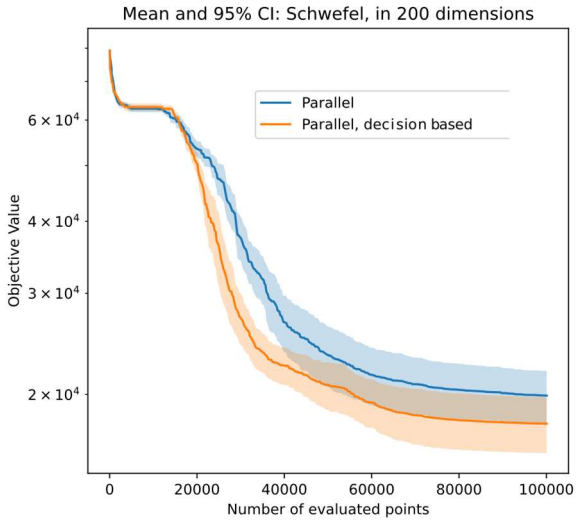

Fig. 5. Convergence of basic Parallel CMA-ES compared to Decision CMA-ES in 200 dimensions across 10 runs on the Schwefel problem.

In Fig. 5, it becomes evident that even when the dimensionality of the problem is significantly increased, Decision CMA-ES outperforms Parallel CMA-ES significantly. However, after all distributions begin to converge to a local optimum, the difference between the two algorithms shrinks, with only a slight favour towards Decision CMA-ES. This behaviour is notably consistent with what was observed in the 100-dimensional case, highlighting that dimensionality has a limited impact on the differences between these algorithms.

If the Decision CMA-ES algorithm can be additionally applied to other problems to see how it performs on problems significantly different to the Schwefel problem.

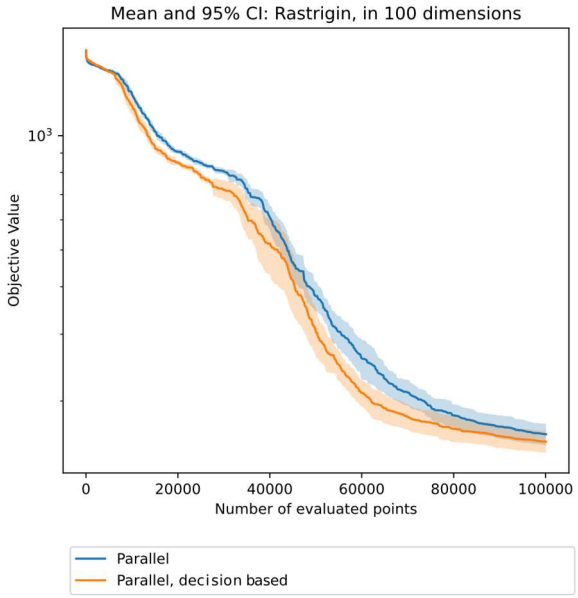

Fig. 6. Convergence of basic Parallel CMA-ES compared to Decision CMA-ES in 100 dimensions across 10 runs on the Rastrigin Problem.

As shown in Fig. 6, the Decision CMA-ES algorithm consistently outperforms the basic parallel alternative on the Rastrigin problem for the first 60,000 evaluated points. However, as the number of data points increases, the performance of the two algorithms becomes more similar, with Decision CMA-ES maintaining a slight edge in terms of average performance but not to a statistically significant extent. This observation highlights that the Decision CMA-ES algorithm is not overly problem-specific and can be applied to a broader range of optimization problems.

The slight improvements seen in the performance of Decision CMA-ES can be attributed to its expected improvement decision process as it is the only different between these algorithms. This decision process can be visualised to illustrate what sets Decision CMA-ES apart.

Convergence of Decision CMA-ES algorithm:

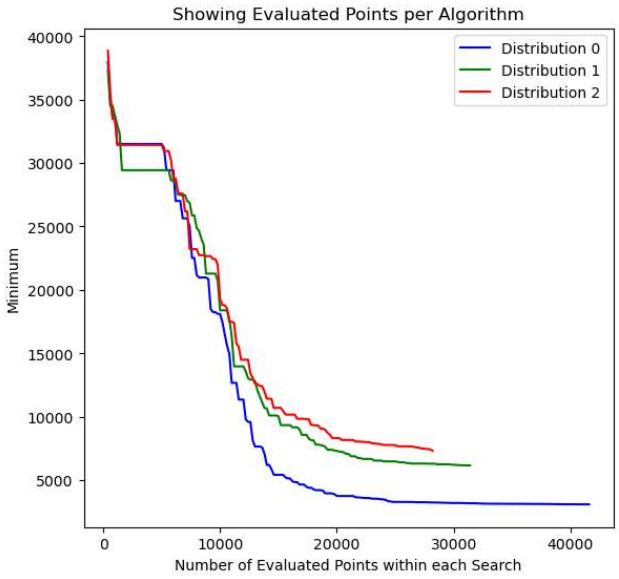

Fig. 7. The convergence of the three searches within one run of Decision CMA-ES shown in respects to number of evaluated points with the search.

As depicted in Fig. 7, there is an uneven distribution of evaluated points among the three searches. Search 0 has received over 40,000 of the total 100,000 evaluated points, while Search 2 has only received fewer than 30,000. Notably, Search 0 converged to the lowest value and had the most evaluated points, while Search 2 converged to the highest value and had the fewest evaluated points. This observation underscores the fact that the searches converging to lower and more optimal minima have more points evaluated within them. This behaviour is to be expected when following expected improvement as having better optima yields a greater expected improvement.

Convergence of Decision CMA-ES algorithm:

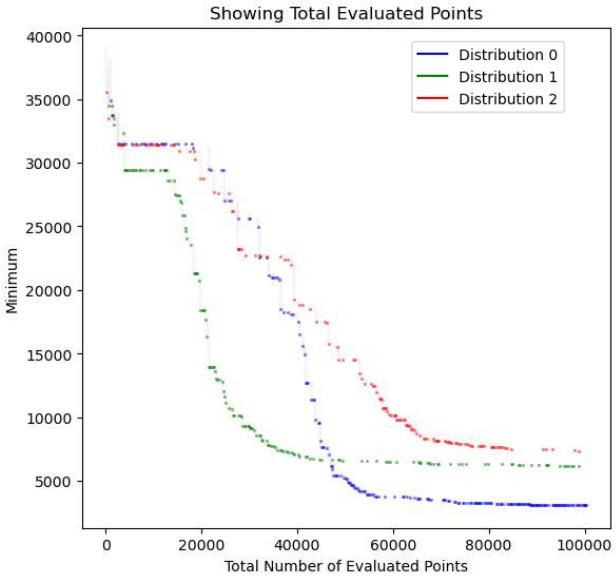

Fig. 8. The convergence of the three searches within one run of Decision CMA-ES in respect to the total number of evaluated points, instead of just points within its own search. Darker regions represent point being evaluated with the given distribution.

When the data shown in Fig. 7 is transformed to display the convergence of each search concerning the total evaluated points, as shown in Fig. 8, a different pattern emerges. In Fig. 8, it becomes evident that search 1 is favoured early in the process, indicated by its darker curve (show it has a higher number of evaluated points), causing it to exhibit the lowest minimum until about 40,000 points.

The preference for evaluating points in search 1 in early generation can be attributed to its achievement of the lowest minimum after evaluating 5,000 data points per search, as demonstrated in Fig. 7. In Fig. 8, after 40,000 total evaluated points, searches 0 and 2 start to be favoured, as indicated by their darkening lines and their minima catching up to that of search 1. However, after 90,000 evaluated points, only search 0 is favoured, leading it to have the highest number of evaluated points, as shown in Fig. 7.

This behaviour is also closely linked to the expected improvement criterion. Early on, search 1 had the highest expected improvement since it showed the most improvement. As it began to converge to a local optimum, searches 0 and 2 displayed higher expected improvements so explored more. Finally, all searches had converged to local optima and search 0 was favoured as it possessed the highest expected improvement.

The computational time taken for both algorithms also requires consideration.

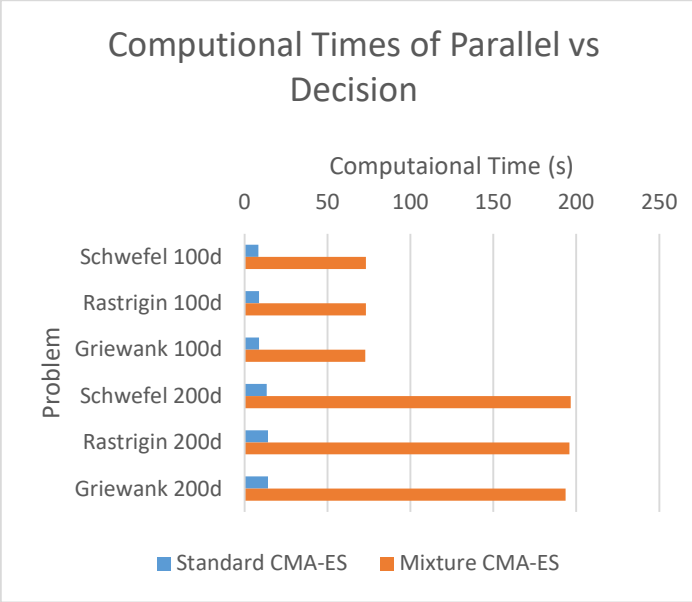

Fig. 9. Average computational time on Parallel and Decision CMA-ES with 100,000 evaluated points

As shown in Fig. 9, Decision CMA-ES consistently takes slightly more computational time than Parallel CMA-ES. This outcome is expected because both algorithms evaluate the same number of points and perform the same number of searches. However, Decision CMA-ES involves an additional step of calculating expected improvement in every generation, which demands slightly more computation. This additional computation step appears to take approximately one second for all problems. The small increase in computation time, which seems to be minimally affected by dimensionality, does not impose a significant computational cost. Therefore, Decision CMA-ES remains as computationally efficient as its basic parallel counterpart.

In conclusion, it is evident that while parallel CMA-ES may not yield performance improvements over the iterative approach, by the end of the algorithm's execution, they converge to similar results within the same number of evaluations. However, Decision CMA-ES demonstrates a noteworthy improvement over parallel CMA-ES during the early generations and maintains a slight advantage in the later stages, all without incurring a substantial increase in computational cost. As such, Decision CMA-ES shows promise in enhancing the performance of CMA-ES searches, especially when enough points are evaluated.

B. Mixture CMA-ES

If Mixture CMA-ES is taken and compared to a standard implementation of CMA-ES, we can assess how well it performs.

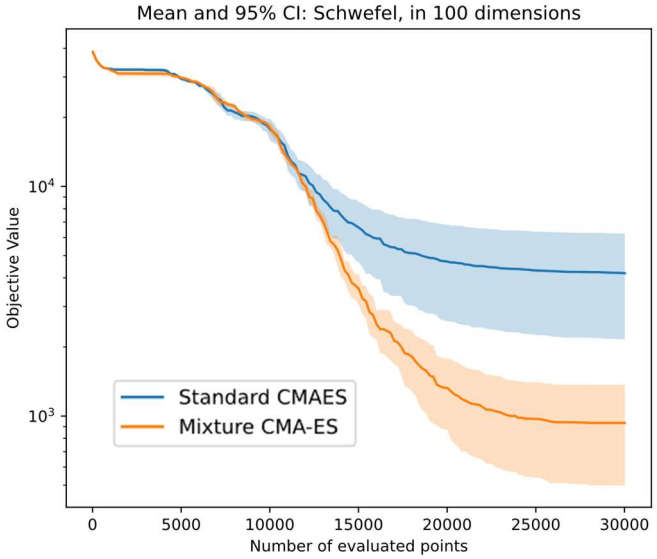

Fig. 10. Convergence of standard CMA-ES compared to Mixture CMA-ES with 20 Gaussian Distributions in 100 dimensions across 10 runs on the Schwefel problem.

As shown in Fig. 10, mixture CMA-ES significantly outperforms a standard implementation of CMA-ES, with the confidence interval not overlapping past 13,000 evaluated points and consistently finding lower minima. This highlights that despite being required to learn additional distributions, it can converge at a similar rate to standard CMA-ES in the early generations. This suggests that Mixture CMA-ES offers significant improvements over standard CMA-ES on the 100-dimensional Schwefel problem.

Unfortunately, the same success does not appear in all problems.

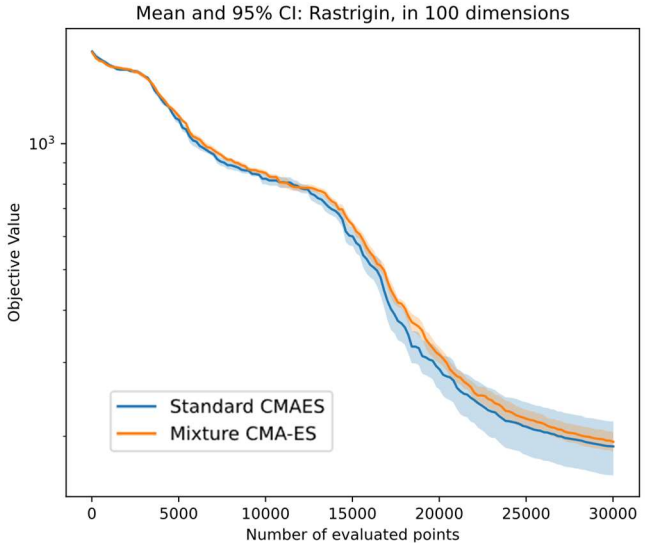

Fig. 11. Convergence of standard CMA-ES compared to Mixture CMA-ES with 20 Gaussian Distributions in 100 dimensions across 10 runs on the Rastrigin problem.

As shown in Fig. 11, Mixture CMA-ES is no able to out

preform Standard CMA-ES on the Rastrigin problem, with both algorithms appearing to find similar optima at all points, show by their confidence intervals normally overlapping partially, or even fully. Notably, after 30,000 evaluated points mixture CMA-ES is, on average, finding slightly higher minima. This all makes it apparent that Mixture CMA-ES is not able to offer significant improvements to standard CMA-ES on the 100-dimentional Rastrigin Problem.

The computational time taken for both algorithms also requires consideration.

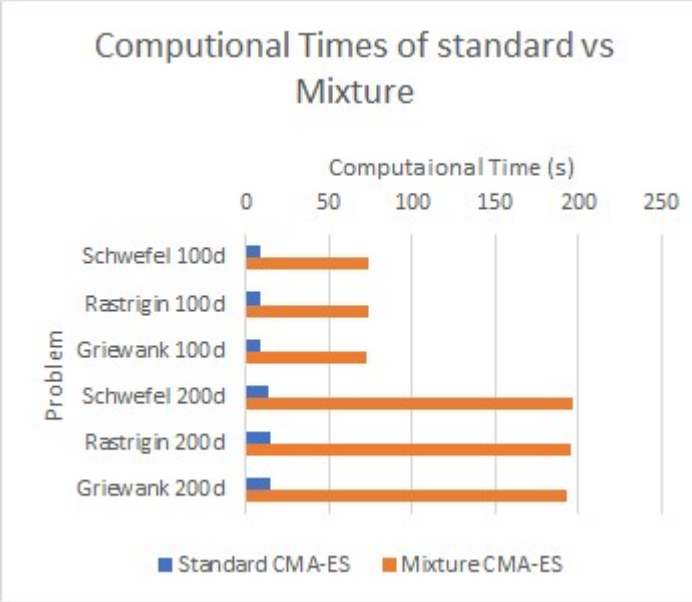

Fig. 12. Average computational time on standard and Mixture (20 searches) CMA-ES, with 30,000 evaluated points.

As depicted in Fig. 12, updating the Gaussian mixture model in the Mixture CMA-ES algorithm imposes significantly higher computational demands. Notably, this process takes approximately 9 times longer in 100 dimensions and 14 times longer in 200 dimensions compared to the standard CMA-ES algorithm. The increased computational requirements are expected due to the need to update a more complex model. However, it's important to note that these requirements do not render Mixture CMA-ES computationally infeasible. The algorithm remains executable, with Mixture CMA-ES completing 30,000 evaluations in under 4 minutes even in 200 dimensions. Moreover, when compared to the overall cost associated with evaluating such many points, this additional time requirement is likely insignificant.

In summary, the Mixture CMA-ES algorithm can achieve better optima within the same number of evaluated data points as the standard CMA-ES on select problem. It accomplishes this improvement without sacrificing performance on other problems or becoming computationally infeasible. Therefore, Mixture CMA-ES appears to offer a significant enhancement over the standard CMA-ES for the tested problems.

VI. Future Research

A. Combining Algorithm

Both the Decision and Mixture CMA-ES algorithms introduce modifications to different aspects of a basic parallel CMA-ES algorithm. The first aspect determines which search should be favoured, while the second governs how information is shared among searches. Importantly, these two techniques can be integrated into a single algorithm. This integration involves applying the distribution favouring mechanism of decision CMA-ES to the Gaussian Mixture model of multi-CMA-ES. As a result, Gaussians from searches with higher expected improvements are more likely to be sampled from within the mixture model. This will be increasing their representation in the sampling process, allow higher expected improvement searches to be favoured. However, it's important to note that the expected improvement calculation needs adjustments, as points are not explicitly assigned to a distribution but rather have responsibilities towards each distribution. Furthermore, the initial assumption that a data point is equally likely to be sampled from all distributions must be refined to accommodate the preference given to select distributions.

B. Dynamic Number of Distributions

In the algorithms implemented, all CMA-ES searches were initialized at the beginning of the search process. However, initializing all searches at the start necessitates determining the number of searches to be used before running the algorithm. Ideally, you'd want to initialize enough searches to explore various promising regions, but not so many that the algorithm struggles to converge to a good solution within the available point evaluations. Unfortunately, without sufficient knowledge of the problem and the number of point evaluations required for CMA-ES to converge, determining the appropriate number of searches can be challenging.

The parallel CMA-ES structure does present a solution to this challenge, and it doesn't require all searches to be present from the beginning of the algorithm. This means that if deemed necessary, new searches can be initialised during algorithm execution and added to the collection of parallel searches. In the decision CMA-ES, this could entail initialising a new search once the expected improvement of all current searches becomes too low. This change functionally allowing the algorithm to run indefinitely without getting stuck indefinably in local optima, as new searches can be initialised as required. In the Mixture CMA-ES algorithm, there's potential for a more advanced process where, if it's deemed that there are insufficient Gaussian distributions to produce a Gaussian mixture model that sufficiently captures the search space, more searches can be initialized to address this. Additionally, if there are more Gaussians than required as they largely or fully overlap with each other, searches could be removed to optimise the representation of the search space.

C. Automatic setting of β parameter from Mixture CMA-ES

The β parameter use in Mixture CMA-ES is, as discussed, is

a challenging parameter to set before executing of the algorithm. As such it would be best to adjust the responsibility calculations to avoid the requirement to scaling of the log probability density with a β value, or to find a way to dynamically set it. To preform task, further research into what responsibility values promote desire search patterns in a Gaussian mixture model. Once this insight in gained, the responsibility calculations need to be adjusted to produce these ideal responsibilities more consistently, without a preset β parameter.

VII. References

[1] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyper-parameter optimization," *Adv Neural Inf Process Syst*, vol. 24, 2011, [Online]. Available: https://proceedings.neurips.cc/paper/4443-algorithms-for-hyper-parameter-optimization

[2] R. Calandra, A. Seyfarth, J. Peters, and M. P. Deisenroth, "Bayesian optimization for learning gaits under uncertainty," *Ann Math Artif Intell*, vol. 76, no. 1–2, pp. 5–23, Feb. 2016, doi: 10.1007/s10472-015-9463-9.

[3] Y. Akimoto, A. Auger, and N. Hansen, "Continuous Optimization and CMA-ES," in *Proceedings of the Companion Publication of the 2015 Annual Conference on Genetic and Evolutionary Computation*, in GECCO Companion '15. New York, NY, USA: Association for Computing Machinery, Jun. 2015, pp. 313–344. doi: 10.1145/2739482.2756591.

[4] D. Eriksson and M. Jankowiak, "High-dimensional Bayesian optimization with sparse axis-aligned subspaces," in *Proceedings of the Thirty-Seventh Conference on Uncertainty in Artificial Intelligence*, C. de Campos and M. H. Maathuis, Eds., in Proceedings of Machine Learning Research, vol. 161. PMLR, 2021, pp. 493–503. [Online]. Available: https://proceedings.mlr.press/v161/eriksson21a.html

[5] S. Rana, C. Li, S. Gupta, V. Nguyen, and S. Venkatesh, "High Dimensional Bayesian Optimization with Elastic Gaussian Process," in *Proceedings of the 34th International Conference on Machine Learning*, D. Precup and Y. W. Teh, Eds., in Proceedings of Machine Learning Research, vol. 70. PMLR, 2017, pp. 2883–2891. [Online]. Available: https://proceedings.mlr.press/v70/rana17a.html

[6] P. I. Frazier, "A Tutorial on Bayesian Optimization," Jun. 2018.

[7] A. Auger and N. Hansen, "Tutorial CMA-ES: evolution strategies and covariance matrix adaptation," in Proceedings of the 14th annual conference companion on Genetic and evolutionary computation. 2012.

[8] M. Köppen, "The curse of dimensionality," in *5th online world conference on soft computing in industrial applications (WSC5)*, 2000, pp. 4–8. [Online]. Available: https://www.class-specific.com/csf/papers/hidim.pdf

[9] M. Binois and N. Wycoff, "A survey on high-dimensional Gaussian process modeling with application to Bayesian optimization," *ACM Transactions on Evolutionary Learning and Optimization*, vol. 2, no. 2, pp. 1–26, 2022.

[10] J. Görtler, R. Kehlbeck, and O. Deussen, "A visual exploration of gaussian processes," *Distill*, vol. 4, no. 4, p. e17, 2019.

[11] R. Moriconi, M. P. Deisenroth, and K. S. Sesh Kumar, "High-dimensional Bayesian optimization using low-dimensional feature spaces," *Mach Learn*, vol. 109, no. 9, pp. 1925–1943, Jun. 2020, doi: 10.1007/s10994-020-05899-z.

[12] P. I. C. Ryan and Others, "References to cma-es applications," *Strategies*, vol. 4527, no. 467, 2007, [Online]. Available: http://www.cmap.polytechnique.fr/~nikolaus.hansen/cmaapplications.pdf

[13] M. N. Omidvar and X. Li, "A Comparative Study of CMA-ES on Large Scale Global Optimisation," in *AI 2010: Advances in Artificial Intelligence*, Springer Berlin Heidelberg, 2011, pp. 303–312. doi: 10.1007/978-3-642-17432-2_31.

[14] R. Ros and N. Hansen, "A Simple Modification in CMA-ES Achieving Linear Time and Space Complexity," in *Parallel Problem Solving from Nature – PPSN X*, Springer Berlin Heidelberg, 2008, pp. 296–305. doi: 10.1007/978-3-540-87700-4_30.

[15] X. He, Z. Zheng, and Y. Zhou, "MMES: Mixture model-based evolution strategy for large-scale optimization," *IEEE Transactions on Evolutionary Computation*, vol. 25, no. 2, pp. 320–333, 2020.

[16] M. M. Drugan, "Efficient Real-Parameter Single Objective Optimizer Using Hierarchical CMA-ES Solvers," in *EVOLVE - A Bridge between Probability, Set Oriented Numerics, and Evolutionary Computation VI*, Springer International Publishing, 2018, pp. 131–145. doi: 10.1007/978-3-319-69710-9_10.

[17] A. K. Gupta, K. G. Smith, and C. E. Shalley, "The Interplay between Exploration and Exploitation," *Source: The Academy of Management Journal*, vol. 49, no. 4, pp. 693–706, 2006, Accessed: Oct. 15, 2023. [Online]. Available: https://www.jstor.org/stable/20159793?seq=1&cid=pdf-

[18] J. Frost, "Model Specification: Choosing the Best Regression Model - Statistics By Jim." Accessed: Oct. 15, 2023. [Online]. Available: https://statisticsbyjim.com/regression/model-

specification-variable-selection/

[19] E. P. Xing, A. Qiao, H. Zhang, and B. Liu, “8 : Learning Partially Observed GM: the EM algorithm,” pp. 10–708, 2015.

[20] S. L. Miller and D. Childers, “Probability and Random Processes: With Applications to Signal Processing and Communications: Second Edition,” *Probability and Random Processes: With Applications to Signal Processing and Communications: Second Edition*, pp. 1–611, 2012, doi: 10.1016/C2010-0-67611-5.

[21] S. Raschka, *Python Machine Learning*. Packt Publishing Ltd, 2015. [Online]. Available: https://play.google.com/store/books/details?id=GOVOCwAAQBAJ

[22] F. Pérez and B. E. Granger, “IPython: a System for Interactive Scientific Computing,” *Comput Sci Eng*, vol. 9, no. 3, pp. 21–29, May 2007, doi: 10.1109/MCSE.2007.53.

[23] J. Blank and K. Deb, “Pymoo: Multi-Objective Optimization in Python,” *IEEE Access*, vol. 8, pp. 89497–89509, 2020, doi: 10.1109/ACCESS.2020.2990567.

[24] S. Surjanovic and D. Bingham, “Virtual Library of Simulation Experiments: Test Functions and Datasets.” Accessed: Oct. 12, 2023. [Online]. Available: https://www.sfu.ca/~ssurjano/index.html

[25] A. Olsson, G. Sandberg, and O. Dahlblom, “On Latin hypercube sampling for structural reliability analysis,” *Structural Safety*, vol. 25, no. 1, pp. 47–68, Jan. 2003, doi: 10.1016/S0167-4730(02)00039-5.