



Experience-based Computation:  
**Learning to Optimise**

**Project Number: 766186**

**Project Acronym: ECOLE**

**Project title: Experienced-based Computation: Learning to Optimise**

## **Deliverable D2.1**

**Experience-based high dimensional & big data assisted optimisation**

**Authors:**

**Jiawen Kong, Sibghat Ullah, Wojtek Kowalczyk, Thomas Bäck – Universiteit Leiden  
Stephen Friess, Gan Ruan – University of Birmingham**

**Project Coordinator: Professor Xin Yao, University of Birmingham**

**Beneficiaries: Universiteit Leiden, Honda Research Institute Europe, NEC Laboratories  
Europe**

**H2020 MSCA-ITN**

**Date of the report: 31.03.2021**



## Contents

|  |    |
|--|----|
| Executive summary .....                                      | 3  |
| Major Achievements .....                                     | 3  |
| 1. Introduction .....  | 5  |
| 2. Background.....   | 6  |
| 3. Instance-based Transfer Approaches .....                  | 7  |
| 4. Model-based Transfer Approaches.....                      | 9  |
| 5. Inductive Biases in Search-based Optimization .....       | 11 |
| 5.1. Analysis of Operators and Algorithms.....               | 11 |
| 5.2. Extracting Knowledge from Evolutionary Searches .....   | 12 |
| 5.3. Extension of Candidate Algorithms .....                 | 13 |
| 5.4. Techniques for Constructing Search Operators .....      | 15 |
| 5.5. Properties of the Retrieved Distributions .....         | 17 |
| 5.6. Hyperparameter Optimization and Model Selection.....    | 18 |
| 5.7. Effectiveness of the Learned Operators .....            | 19 |
| 6. High Dimensional Surrogate-Assisted Optimisation .....    | 23 |
| 6.1. Comparison of Dimensionality Reduction Techniques ..... | 24 |
| 7. Summary and Outlook.....                                  | 32 |
| Bibliography .....   | 33 |

## Executive summary

The objective of WP2.1 is to study experience-based high-dimensional and big-data assisted optimisation. The status of this work package within the ECOLE project is presented in this report. Our study on the representation of experience in the form of inductive biases for evolutionary search algorithms is first introduced (in Section 5). Particularly, this is done by means of investigating methods to design operators tailored for specific continuous single-objective optimization problems. The existing work in the literature which has been proposed so far only considers experience in the form of previously found solutions, but completely neglects a procedural view. The results show that operators can be indeed designed for low-dimensional and regular structured problems. Apart from this, our study on the issue of high dimensionality in *Surrogate-Assisted Optimization* is also presented (in Section 6). In particular, some of the most common dimensionality reduction techniques are compared against each other to efficiently construct the low dimensional surrogate models which are applicable for expensive to evaluate computer simulations. The results in this section demonstrate the efficacy of Principal Component Analysis and Autoencoders for dimensionality reduction in surrogate modeling. This is since the surrogate models based on Autoencoders achieve highest modeling accuracy in 132/720 test cases, whereas the surrogates based on Principal Component Analysis achieve the best optimal function value in 82/720 cases. In most of the remaining cases, the performance of the dimensionality reduction techniques is analogous.

## Major Achievements

Major scientific achievements concerning the research invested in this deliverable are presented. In particular, short answers to the most important research questions – practical issues – are described:

| Research Questions  | Discussion  |
|---|---|
| What is the state of literature and what are major knowledge gaps existing in existing research on experience-based optimization? | The state of the literature is comparably sparse and existing work offers vastly different notions of experience. Especially in more recent literature the procedural view has been completely neglected.   |
| What are properties of existing algorithm designs and which methods can be used to extend and improve them?                       | In most existing algorithms, so called operators draw random variates from symmetrical and isotropic distributions. We could show that by keeping statistics about their inner mechanisms we design improved operators tailored for a specific problem. |

|  |   |
|--|---|
| How can the discovered knowledge gaps be related to concepts and efforts in similar research fields?   | The effort of designing improved operators can be seen as an application of machine learning to construct inductive biases for search-based optimization algorithms.  |
| What are potential ways of relating them to application scenarios?   | The most practical scenarios at hand are improved initializations for estimation of distribution and CMA-ES algorithms. Potentially also in regards to hybridization with dimensionality reduction techniques and similar scenarios in combinatorial optimization.            |
| How does high dimensionality affect the practical applicability of surrogate modeling?   | For expensive to evaluate objective functions e.g., computer simulations, high dimensionality can severely affect the applicability of surrogate modeling since the associated computational cost becomes prohibitively high (Table III).                                     |
| Which dimensionality reduction technique is best suited to efficiently construct the low dimensional surrogate models?   | Our findings demonstrate the effectiveness of Principal Component Analysis and Autoencoders for efficiently constructing the low dimensional surrogate models (Figure 11 - Figure 16).  |
| How does the size of the latent dimensionality (in the dimensionality reduction algorithm) affect the accuracy of the corresponding low dimensional surrogate model? | In our findings, the size of the latent dimensionality only has a nominal impact on the accuracy of the low dimensional surrogate model (Figure 11 - Figure 12). Problem landscape, on the other hand, has the biggest impact on the accuracy of the corresponding surrogate. |
| Does the choice of modeling technique, e.g., Kriging, Polynomials, significantly affect the quality of the low dimensional surrogate models?                         | For the commonly employed modeling techniques – Kriging and Polynomials – our findings do not indicate that (Figure 11 - Figure 16). Note, however, that further research must be invested to validate this on more complex cases.  |

## 1. Introduction

In the Experience-based Computation: Learning to Optimise (ECOLE) project, we aim to address several challenges in the automotive setting, including *Optimisation using Natural Computation*, *Multi-objective optimisation* and *System Engineering and Big Data Analytics*. The project has been further divided into several subprojects, where each early stage researcher (ESR) is responsible for each subproject. The research aims of ECOLE include shortening the product cycle, reducing the resource consumption during the complete process, and creating more balanced and innovative products. Instead of just developing technologies to solve a given problem, it takes a bold step forward and propose to use knowledge automatically across different problem domains. Referring to knowledge, skill, and practice derived from problem solving processes in time, the goal is to automatically learn and transfer the experience of optimizing one product or process for solving other optimization problems.

Due to the lack of consideration of robustness, multi-objectivity and nonlinear constraints, current techniques are unable to meet the industrial needs for system-level optimisation. In Work Package 2 (WP2) of the ECOLE project, the research focus is to deal with multiple conflicting objectives in optimisation for learning in dynamic and uncertain environment. WP2.1, as an individual part in WP2, focuses on the development of methods and foundations for high-dimensional non-linear optimisation by making use of experiences derived from previous optimisation.

The remainder of this report is organized as follows. In Section 2, the background knowledge of three different aspects of experience are presented. The details on our achievements are introduced in Section 3, Section 4, Section 5 and Section 6 respectively. Section 7 concludes the report and outlines further research.



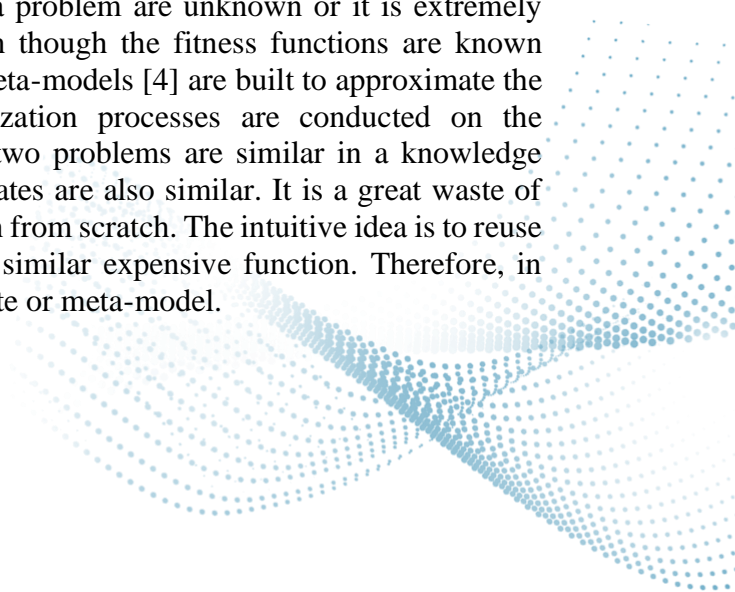
## 2. Background

Nowadays, as product complexity increases, with the application of advanced sensors and the booming of computer storage, the dimensionality and volume of collected data have been becoming more and more complex, which puts enormous burdens and challenges on existing data analyses techniques and optimization methods. This situation requires the proposal of more advanced systematical methodologies. Most existing optimization techniques always solve a new problem from scratch without assuming any prior knowledge of the problem, which may show ineffectiveness when dealing with more complicated problems due to the involved high-dimension and highly-complex data. Nevertheless, real-world problems rarely exist in isolation. Specifically, many industrial systems in the real-world are designed with a lifetime and expected to resolve a large number of practical problems before the shutdown. Those problems, within a specific domain, may either be similar or even repetitive. Thus, the idea of using experience to tackle a newly emerging while related task is naturally motivated.

In experience-based optimization, it is important to figure out what is the experience, i.e. what is the representation of the experience in terms of optimization. There are three different aspects of representing the experience: (optimal) solutions, search operators and surrogates, which are discussed in the literature [1] [2] [3] [4]. When an optimization problem is to be resolved, one or more optimal solutions are expected to be obtained. Therefore, the most intuitive representative way of using past experience is using (optimal) solutions. For example, in [1], several transfer learning methods are implemented by transferring a number of good individuals or sub-individuals from the source problem to the target one for Genetic Programming.

Another type of the representation of experience is the search operator [2] [3]. In real-world problems, experienced experts are always aware of the rough direction to get the optimal solutions when given a similar problem or task, which is the real experience of the expert. In evolutionary optimization, good solutions of an objective function are always sequentially found by genetic operators and selection processes. The knowledge in the search operators can be extracted in the form of empirical and improved distributions, which can be transferred to help solving similar problem instances.

In most cases of the real-world, fitness functions of a problem are unknown or it is extremely computationally expensive to evaluate solutions even though the fitness functions are known beforehand. To tackle those problems, surrogates or meta-models [4] are built to approximate the real fitness functions and then the normal optimization processes are conducted on the approximated expensive functions. Considering that two problems are similar in a knowledge specific domain, it is very likely that two built surrogates are also similar. It is a great waste of time to rebuild the surrogate for another fitness function from scratch. The intuitive idea is to reuse one established surrogate for another different while similar expensive function. Therefore, in those problems, the notion of experience is the surrogate or meta-model.



### 3. Instance-based Transfer Approaches

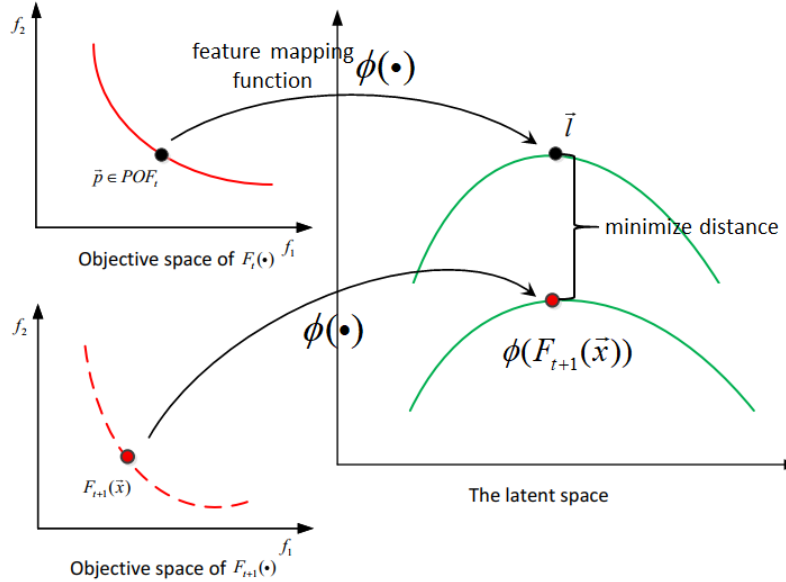


Figure 1. The instance-based approach from [5] as investigated by ESR7.

Instance-based transfer approaches are due to their simplicity some of the earliest ones which have been proposed within the literature on knowledge transfer in search-based optimization. Noteworthy approaches are CIGAR [6] and GTL [7], which both have been originally proposed for optimization problems solved using the binary formulation of genetic algorithms. The framework of Case-Injected Genetic Algorithms (CIGAR) [6] from 2004 may be considered as one of the seminal works and also shares high similarity with the Genetic Transfer Learning (GTL) [7] framework from 2010. Central to CIGAR is a case-base, in which every case represents a solution from a previously solved problem. Similar to GTL, when a problem is tackled, CIGAR extracts intermediate solutions at every generation and stores them into a case-base. When tackling a new optimization problem, CIGAR queries the case-base to find the ones with the most similar of previously tackled problems and uses the solutions stored in these to partly initialize the start population for the optimization algorithm on the new problem. However, the authors openly acknowledge, that in many scenarios problem similarity measures may not be trivially definable. Therefore, they suggest as a way to cope with this scenario, to reflect problem similarity by means of solution similarity. Explicitly so, by repeatedly querying the case-base during the algorithm run for solutions similar to the current best one within the solution population. The found solutions from the previously solved tasks are then subsequently used to replace the worst performing solutions in the current population. Both CIGAR and GTL are based on binary genetic algorithms. While the former explicitly consider combinatorial single-objective problems arising in application scenarios, the latter consider solely 2D synthetic continuous single-objective problems.



Variations of the instance-based approach have more recently emerged within the scenario of continuous multi-objective optimization [8]. Similar to CIGAR in solution similarity mode, these approaches repeatedly inject solutions from previously solved optimization problems into the population while solving a new target task, but do so by repeatedly constructing a linear mapping between ranked intermediate solutions of a task, and then subsequently either map final or current best solution of a past or concurrent related task into the population.

Particular interesting approaches within this scenario have been developed in the domain of dynamic multi-objective optimization [5], [9], [10]. These explicitly use machine learning techniques to construct a mapping such that it reduces the distances between the Pareto-fronts between source and target tasks in a low-dimensional feature space. For example, work on the so called Transfer Dynamic Multi-Objective Evolutionary Algorithm (Tr-DMOEA) [5] minimizes distances in the latent space based upon the MMD measure using the TCA algorithm [11]. Within the ECOLE project, the work of ESR7 has dealt with investigating this instance-based approach from [5]. Closer scrutiny has shown that Tr-DMOEA is computationally expensive [12] and therefore might not be considered feasible for real-world problems. Deliverable D3.4 deals more in-depth with the analysis discussing the results of this study. More recent work based upon manifold learning has shown to be able to realize significant performance improvements [9]. However, in conclusion it is quite noteworthy that none of the former nor recent works further explores the possibility of exploiting knowledge about problem characteristics.





## 4. Model-based Transfer Approaches

Model-based approaches on the other hand explicitly concern techniques to build predictive models from data generated during optimization runs. There are in principle two ways on how knowledge transfer research has found its interest in these techniques. In multi-objective optimization, this has taken the form of using density estimation techniques to model distributions of candidate solutions.

Particularly noteworthy is the proposed Adaptive Model-based Transfer Evolutionary Algorithm (AMTEA) from [13]. Within their work, a Gaussian distribution is fitted to each final population obtained from previously solved multi-objective optimization problems and the obtained parameters are stored within a database. Thus, when encountering new and previously unseen problems, the algorithm halts at regular intervals of  $\Delta$  iterations and performs a stacked density estimation of the current solution population using the Gaussian distributions stored within the database constructed from the previously solved problems. Effectively, the obtained mixture weights subsequently encode the solution similarity between the single target and the multiple source problems. Using this mixture model, the algorithm subsequently can sample candidate solutions from previously solved optimization problems based upon their similarity to the target distribution. This approach has the advantage that it circumvents the problem of negative transfer, i.e. the uncontrolled transfer of knowledge into the optimization procedure which deteriorates algorithm performance. This effect has shown up in previous work [8] and can be considered as a potential key weakness and danger of any proposed knowledge transfer framework. In a similar fashion to AMTEA, also the work from [14] has employed Gaussian distributions. However, in their proposed multisource selective transfer framework, the similarity between solution distributions of the current target problem and the ones stored from the source problems in the repository are explicitly calculated through the Wasserstein distance. Based upon the calculated similarities, different source selection strategies can subsequently be employed.

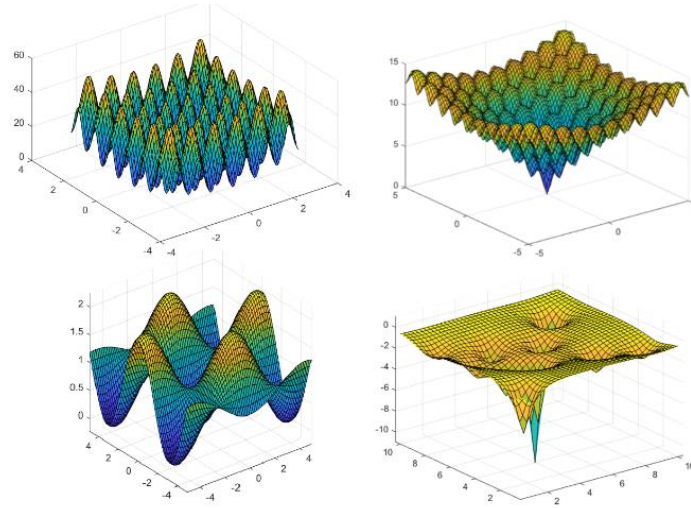
The second line of research concerning knowledge transfer through model-based approaches has concerned surrogate-assisted and Bayesian optimization. In surrogate modelling, also referred to as meta-modelling, one attempts to build a regression model of the objective function of the considered problem. There are in principle many different scenarios in which building such a model can be of interest [15] [16]. However, in most cases one wants to build the regression model such that one can spare expensive function evaluations. Bayesian optimization on the other side, attempts to directly guide the search for optimal solutions using the metamodel of the objective function. In principle, two works can be identified on both approaches [17] [18]. Within the ECOLE project, the work of ESR3 in Section 6 as well as deliverable D2.3 is concerned with the hybridization of dimensionality reduction with surrogate-assisted optimization to this matter.

Interestingly, aside from these mentioned ones, barely any of the recent literature tries to learn across problems explicitly by means of internal sampling models for evolutionary algorithms. Quite intuitively, the interplay between algorithm and optimization problem should enforce characteristic search strategies and behaviors. Modern model-based algorithms such EDAs [19]

and the CMA-ES [20] acknowledge this by adapting a distribution online during the optimization run. However, they do not attempt to memorize these in a more rough and abstract way, such that these can be reused on similar problems. In many ways, this perspective might be also the only meaningful notion to realize knowledge transfer in the continuous single-objective optimization scenario. The work of ESR6 within Section 5 therefore particularly investigates this research question.



## 5. Inductive Biases in Search-based Optimization



*Figure 2. Examples of differently structured fitness landscape from different continuous optimization problems.*

One of the key questions concerning the notion of experience in search-based optimization algorithms, is the question for what can be considered to constitute any form of learnable knowledge and how is it represented in the first place. Many modern algorithms such EDAs [19] and the CMA-ES [20] have by default isotropy assumptions build into their design. However, these design assumptions can be considered to become violated when specific problems are considered. As they might possess special structures or certain forms of irregularities superimposed on top of them (c.f. Figure 2).

The project of ESR6 therefore concerns the idea of using various forms of analytics as a way to explicitly learn and predict so called inductive biases for search-based optimization algorithms. Inductive biases pose the set assumptions a learning algorithm needs to form from tackling training problems and apply to unseen situations [21]. Thus, they are a necessary prerequisite for any form of generalization to occur.

The algorithm components we pay a justified focus within our study are in our case search operators, and research on obtaining these can be considered as making a contribution towards understanding and improving the way “how” an optimization algorithm solves problems.

### 5.1. Analysis of Operators and Algorithms

Essentially, one of the key questions which needs to be solved in the first place is in regards to

finding a way of extracting knowledge from optimization algorithms by means of finding a way of modifying them to produce reasonable and abundant statistics about their inner mechanisms.

The basic outline of an evolutionary search algorithm is elaborated in the following: Given a specific problem, the algorithm is initialized with a start population which is randomly initialized on the search space. The start population can be considered to represent the practitioner's initial guesses about possible solutions for a given problem. Given this start population, subsequently so called variation and selection operators are applied to iteratively modify the given solutions. In analogy to evolutionary biology each iteration is referred to as „generation“. The goal of an evolutionary search routine in the usual global single-optimization scenario therefore is to iteratively modify and select the set of candidate solutions in a way such that high-performing solutions are retrieved according to a given pre-defined criteria, or after a given predefined number of maximum iterations.

- *Variation Operators* can in principle be either unary or n-ary. Meaning, they are either applied on each solution individually, or in groups of n pre-selected solutions. The most common operators used in evolutionary search routines are unary „mutation“ operators and binary „crossover“ operators. Both considered in a direct analogy to evolutionary biology. The meaning of mutation is essentially to introduce noise into a solution, such that new areas of the search space are explored. On the other hand, crossover can be considered as a form of ‚mating‘ known solutions to create new ones. From a mathematical point of view, it can be considered as an attempt in finding new well performing regions of a partially known solution distribution by interpolating from known samples. However, a key understanding in evolutionary biology is that only through mutations novelty can emerge [22]. In fact, the emergence of the latter ‚crossover‘ mechanism can be still considered to be a kind of puzzle within evolutionary biology. However, it is argued that it aids a rapid adaption in the scenario of small populations sizes.
- *Selection Operators* have their use in evolutionary search routines as a way of maintaining a fixed size of candidate solutions. As often within an iteration, more solutions are produced than can be effectively maintained, the selection operator therefore decides on which ones are allowed to proceed into the next phase. There are in principle different ways of implementing this operation, motivated by different perspectives and understandings of natural selection processes. However, we spare a further discussion of them in favor of a mere functional understanding.

## 5.2. Extracting Knowledge from Evolutionary Searches

All three operators have an impact on the procedural performance of an evolutionary search routine. In principle, they introduce various options for us on how we may modify them such that their performance can be tuned, measured and improved. However, it is clear from the prior discussion that the unary mutation operator is most important within the search process, as it is the prime source of variation. Thus, it allows the optimization algorithm to generate solutions which would be otherwise unobtainable through cross-over operations alone.

In the usual evolutionary search routine, given a start population of solutions with known fitness values, crossover is conducted first, followed by mutation, recalculation of fitness values, and

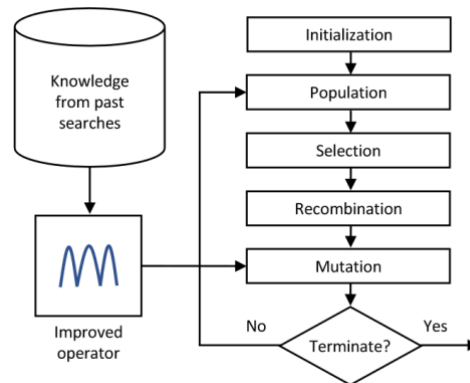
subsequent selection to form the new population. Thus, the direct effect of cross-over and mutation operators on obtaining good solutions and ensuring fast convergence is rather obscured.

Mechanically, unary mutation operators in their most basic form can be considered to be comparably simple. This is because they rely simply upon adding random variates on solutions, sampled from symmetric distributions which are shared by all solutions, independently from their position within the search space. Binary cross-over operators on the flip-side are particularly more sensitive to specific chosen solution pair, as the offspring created from them directly depends upon their position in the search space. Selection operators can in principle introduce further stochasticity, however it has been suggested that so called  $(\mu, \lambda)$  and  $(\mu + \lambda)$  selection schemes lead to performance boosts in mutation-based algorithms [23], where in the former all the progenitor solutions are discarded and in the latter only new solutions are accepted into the successor generation, which have notably improved the fitness in comparison to their progenitors.

Considering the simplicity and importance of unary mutation operators, as well as the nature of the  $(\mu + \lambda)$  selection scheme, it is particularly tempting to consider an algorithmic framework solely based upon these two operations. In principle, it is obvious that within such a framework only sampled random variates are useful which improve the solution quality while any worsening ones are discarded by the selection scheme. Thus, one is inclined to keep statistics about the quality of the performed mutations during algorithm runs, and construct from these inductive biases. These can be represented in the form of improved operators, such that they can be used to realize performance improvements on re-runs of similar problems.

Even though such a framework may spare out many of the complexities of modern algorithms, it can still serve as an effective model to understand the pitfalls and opportunities in regards to constructing experience-based approaches from a procedural view.

### 5.3. Extension of Candidate Algorithms



*Figure 3. Outline of the basic framework studied within our work.*

In the ECOLE project setting, we consider continuous single-objective optimization problems of the form  $f: \mathcal{S} \subseteq \mathbb{R}^d \rightarrow \mathbb{R}$ , where  $\mathcal{S}$  is the search space and  $d$  its associated dimension. We further use for our investigation a basic continuous evolutionary algorithm, such that solutions are directly represented in the search space by vectors

$$\mathbf{x}(j) = (x_1(j), x_2(j), \dots, x_d(j)),$$

where the variable  $j$  simply indicates the  $j$ -th solution. In the following, mutations are considered to be drawn from a multivariate Gaussian mutation operator

$$\Delta \mathbf{x} \sim N(\mathbf{0}, \Sigma),$$

with spherical covariance  $\Sigma = \mathbf{1} \cdot \sigma^{-2}$  and fixed sampling width  $\sigma$ , which upon mutation shifts solutions such that

$$\mathbf{x}' = \mathbf{x} + \Delta \mathbf{x}$$

To learn inductive biases, the developed framework keeps track of mutations performed. The necessary modifications to the evolutionary algorithm are illustrated in Figure 3. In principle, the standard architecture is only extended by a repository, which is filled with copies of pairs of fitness values  $f$  and solution positions  $\mathbf{x}$  from before and after application of the mutation operator. This mutation tracking allows in the following to further distinguish between *improving*

$$f(\mathbf{x}(j)_{before}^i) - f(\mathbf{x}(j)_{after}^i) \geq 0$$

and *worsening mutations*

$$f(\mathbf{x}(j)_{before}^i) - f(\mathbf{x}(j)_{after}^i) < 0.$$

As the  $(\mu + \lambda)$  selection scheme is used and therefore only mutations are accepted which are improving, we thus can filter them according to their quality and consider the set of improving mutations to represent the inductive bias for a given optimization problem. To harness it, we can use density estimation methods to explicitly construct new search operators.





## 5.4. Techniques for Constructing Search Operators

### 5.4.1 Histograms

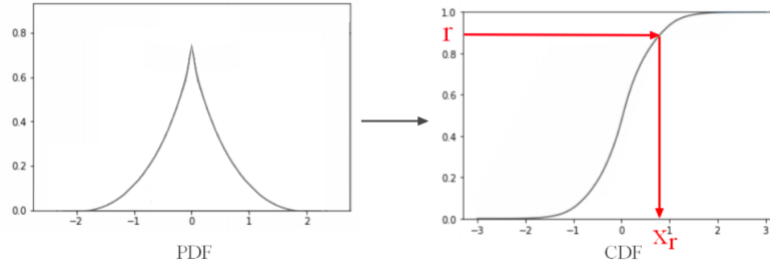


Figure 4. Illustration of the inverse transform sampling technique.

These are among one the most intuitive techniques to be used. Given a data-set  $D = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$  for for  $\mathbf{x}_i \in \mathbb{R}^d$ , one partitions the data space into a number of equisized bins  $B = \{b_1, \dots, b_n\}$ . Subsequently, for every bin  $b_i \in B$  the number of data points  $c_i$  for which  $\mathbf{x}_j \in b_i$  is counted and probabilities are calculated based on these. Based upon the probabilities one can easily calculate pseudo-random numbers from the modeled distribution using the inverse transform sampling technique [24] as illustrated in Figure 4. For example, in the case of a discrete distribution one first samples a random number  $r \in [0,1]$ . Subsequently, we can calculate the cumulative density function  $CFD(j) = \sum_{i=1}^j p_i$ . To generate a pseudo-random number we choose  $j$  such that  $CFD(j) < r \leq CFD(j+1)$  and based upon the associated part of the data space defined by bin  $b_j$  we generate the random number. E.g. for  $b_j = [a, b]$ , we uniformly sample a random number from  $[a, b]$ . While histograms are at first glance intuitive and therefor tempting, their use on the flip-side is rather cumbersome due to them being parametric. Thus, requiring the explicit definition of bin positions and widths. And often times Further, pre-processing steps such as the cropping of the data-space to ensure that those parts are covered at good resolution which reveal the essential structure the modeled distribution.

### 5.4.2 Kernel Density Estimation

These techniques, which are also known in signal processing as *Parzen windows*, are among the most accurate methods one can use to model a data distribution. Essentially, for a given data-set  $D$  we place a kernel function  $K_{\mathbf{H}}(\mathbf{x}, \mathbf{x}_i)$  on every data point  $\mathbf{x}_i \in D \subseteq \mathbb{R}^d$ , where  $\mathbf{H} \in \mathbb{R}^{d \times d}$  is a parameter or bandwidth matrix shared by all data points and controlling the shape of the kernel function. The full data distribution can then be modeled as  $(\mathbf{x}) = \frac{1}{N} \sum_{i=1}^N K_{\mathbf{H}}(\mathbf{x}, \mathbf{x}_i)$ . For the kernel function different choices can be made. A popular one is the use of the Gaussian kernel. A key drawback of kernel density estimation method, however, is the determination of the bandwidth matrix  $\mathbf{H}$ . While in principle one can do this by maximizing the log-likelihood in a cross-validation



training scheme, this approach is known to underfit. Likewise, bandwidth selection rules can be employed but suffer from the same issue [25]. The mathematically correct way to determine  $\mathbf{H}$  would be by minimizing the mean integrated squared error of  $f$  to the true distribution  $f^*$  using a bandwidth selection method. However, the former distribution is in most cases not known. For sampling from a kernel density estimate, one simply selects a point  $\mathbf{x}_r \in D$  from the dataset at random and subsequently samples from the kernel function  $K_{\mathbf{H}}(\mathbf{x}, \mathbf{x}_r)$ . While accurate, sampling within kernel density estimates can be intensive in regards to memory requirements.

### 5.4.3 Gaussian Mixture Model

These are among the more popular and well-studied approaches for density estimation. The Gaussian mixture model is in some way similar to the kernel density estimation technique with Gaussian kernel, however is more sparing in its complexity. In the Gaussian mixture model, the data distribution is modeled as  $f(\mathbf{x}) = \sum_{i=1}^K \pi_i N(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ , where  $\pi_k$  are the so called mixture coefficients and  $K \ll |D|$  is the number of mixture coefficients, which naturally is chosen such that it is much smaller than the total size of the data set  $D$ . The parameters  $\pi_i$ ,  $\boldsymbol{\mu}_i$  and  $\boldsymbol{\Sigma}_i$  are determined using the expectation-maximization (EM) algorithm. The number of mixture components has to be chosen by the practitioner first and is in principle likewise to kernel density estimation a non-trivial topic. However, one can similarly reside from a pragmatic point of view to maximizing the log-likelihood using a cross-validation training scheme. The mixture coefficients are determined such that  $\sum_{i=1}^K \pi_i = 1$ , thus each coefficient  $\pi_i$  can be interpreted as a probability for choosing the specific component  $i$ . In this framework, pseudo random numbers can be generated using ancestral sampling. Meaning that for a given distribution  $p(\mathbf{x}|\mathbf{z}) p(\mathbf{z})$ , we first sample  $\mathbf{z}_r \sim p(\mathbf{z})$  and based upon it sample  $\mathbf{x}_r \sim p(\mathbf{x}|\mathbf{z}_r)$ . For the Gaussian mixture model this would equate to  $p(\{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}|i) p(i)$ , where  $p(i) = \{\pi_1, \dots, \pi_N\}$  and  $p(\{\boldsymbol{\mu}, \boldsymbol{\Sigma}\}|i) = N(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$ . Where  $p(i)$  can be sampled using the previously mentioned inverse transform sampling technique and the normal distribution  $N(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i)$  using readily available methods.



## 5.5. Properties of the Retrieved Distributions

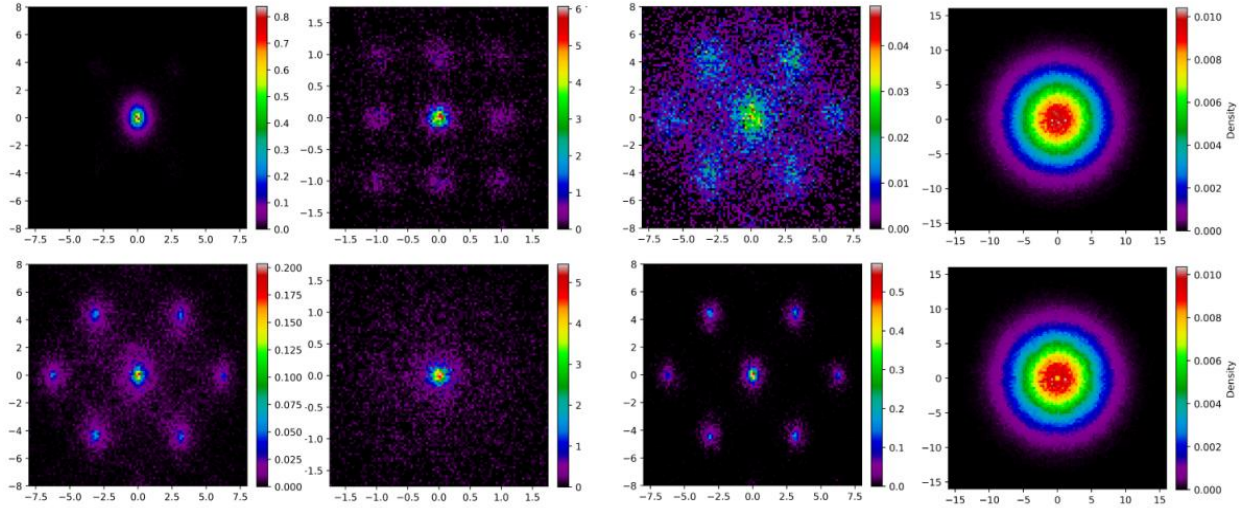


Figure 5. Different mutation distributions retrieved from running an evolutionary search routine under variable settings: Different sampling widths (Column 1), different problem parameters (Column 2), different generational intervals (Column 3), as well as all mutations and only worsening mutations (Column 4, top to bottom).

We start our first series of experiments with an investigation into the algorithm-problem interaction, based upon the expectation that the interplay between algorithms and problems should lead to notable differences in the statistical distributions. Experiments have been performed to investigate this using our extended evolutionary algorithm with the configuration as detailed previously and a mutation rate of 1.

In the first experiment we consider Griewank's benchmark function with  $\chi = [-600, 600]^d$ . However, we keep the problem parameters constant and only vary the sampling width for mutations from  $\sigma=1.5$  to 4. In the second one we keep the algorithm configuration constant and consider exclusively Ackley's benchmark function [26] with  $\chi = [-32.768, 32.768]^d$  and the depth parameter being usually defined as  $a = 20$ . However, in the following we vary  $a$  in the range from 1 to 20, thus varying the depth and steepness of the funnel while essentially keeping the positions of local extrema the same. In the third experiment, we simply visualize different generational intervals from 0 and 100, as well as 100 to 1000. While in the last experiment, we simply show the distribution of all mutations, as well as the worsening ones for Griewank's function for comparison. Histogram representations of all retrieved distributions are illustrated from first to fourth subfigures of Figure 5.

We find on Griewank's function [26] where we only vary the sampling width  $\sigma$ , that for  $\sigma = 1.5$ , the algorithm only retrieves a Gaussian multivariate distribution. After significantly enhancing the sampling to  $\sigma = 4$ , we can however retrieve a neighborhood structure of peaks arranged in a hexagonal grid akin to the pattern in the fitness landscape in Figure 2. Note, that we can interpret

the recovered distribution as consisting out of a central part for local improvements and an outer part for long-range exploration of the neighborhood.

On Ackley's function for the parameter  $a=20$ , the retrieved distribution resembles a simple multivariate normal distribution and does not seem to encode any problem specific information. Setting the parameter to  $a=1$ , the retrieved distribution strongly differs from a Gaussian bell shape by having further peaks akin to grid points in a Moore neighborhood. The algorithm configuration thus is able to resolve notable problem-specific information. In the third experiment on Griewank's function again, we find that in the initial generations the retrieved distribution strongly resembles a multivariate normal distribution, and only in the latter phase, the minima structure becomes very prominent. Note, that in the 4<sup>th</sup> experiment, the full distribution is Gaussian as expected, however the distribution of worsening mutations has likewise appearance. Thus, building operators by suppressing mutations may not be considered a viable strategy.

## 5.6. Hyperparameter Optimization and Model Selection

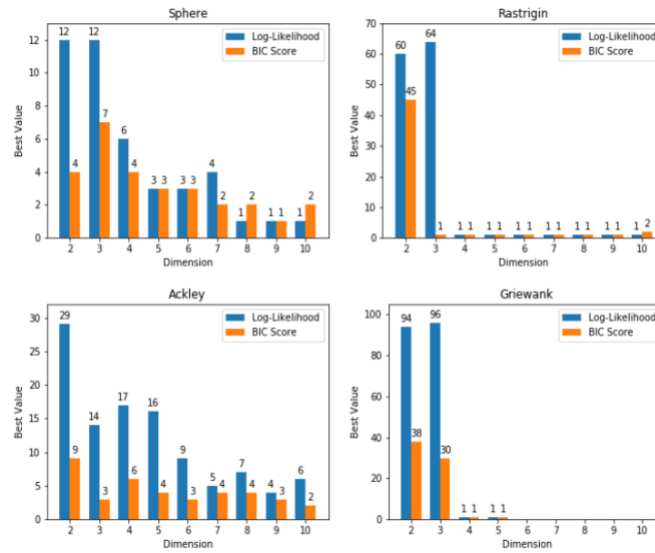


Figure 6. Comparison of the number of components retrieved using the Log-Likelihood and BIC Score on variable benchmark functions of differing dimensionality.

In principle, when designing search operators through density modeling, one could use different techniques. As previously mentioned, kernel density estimation is the most accurate technique, however may at times suffer from the problem that it can be memory intensive. The Gaussian mixture model also offers a viable alternative in which the dataset is reduced to a set of  $n$  descriptive clusters each modeled by a normal distribution. Sampling in this framework can be done using the previously mentioned ancestral sampling technique. While Gaussian mixture models are known to be an efficient method for density estimation and clustering, determining the best number of components  $n$  in an automatic fashion is a non-trivial topic. In principle, this is also

highly dependent upon the particular application of choice [27]. The off-the-shelf approach is to directly maximize the log-likelihood given by

$$l(\mathbf{X}|\boldsymbol{\theta}, n) := \sum_{i=1}^k \ln \{ \sum_{j=1}^n \pi_j N(\mathbf{x}_i | \boldsymbol{\theta}_j) \}$$

with respect to the number of components  $n$ , where  $\boldsymbol{\theta}$  are the parameters determined by EM algorithm. Alternatively, to prevent overfitting one may instead consider the so called Bayesian Information Criteria (BIC)

$$\mathbf{BIC} = -2 l(\mathbf{X}|\boldsymbol{\theta}, n) + v_n \ln(k),$$

which modifies the log-likelihood through an additional term which explicitly penalizes more complex models through the factor  $v_n = n(1 + d + d(d + 1)/2) + 1$ , scaling in the dimension  $d$  of the dataspace, as well as size  $k$  of the fitted dataset. The ideal model  $n$  is then the one which minimizes the respective BIC value.

We compare the found model complexity (c.f. Figure 6) by maximizing the log-likelihood in a 10-fold cross validation training scheme with the one suggested by the BIC score over a range of four different benchmark functions of varying dimensionality. Overall, we can confirm the expectation that the BIC score suggests the usage of low-complexity models. However, both methods show a tendency to break down for more than three dimensions, thus suggesting the use of the lowest complexity model with only a single component. Alternatively, one may reside to the view that the goal of density modeling should merely lie in reducing the complexity of the dataset. Thus, we could by this willfully neglect any model selection procedure and force the mixture model to have a fixed number of  $n$  components. While in principle this has the danger of overfitting, one might acknowledge that this still a much more desirable property than underfitting with low-complexity model. Likewise, model selection procedures such as the BIC score assume by design that the underlying distribution is generated by a finite set of normal distributions in the first place, which can be considered an unrealistic assumption for the empirical distributions and at best only valid in the case of infinitely many components.

## 5.7. Effectiveness of the Learned Operators

In the following we conduct experiments over a range of 9 different optimization problems listed in Table I. We group these into unimodal and valley-shaped problems (1st-3rd row), multimodal problems with single global optimum and high regularity (4th-6th row) and multimodal problems with single global optimum and high irregularity (7th-9th row).

All experiments are conducted with a population size of  $\mu = 10$  and we generate at each generation  $\lambda = 10$  offspring members by randomly selecting individuals and mutating them. In all experiments, the population is initialized randomly within the entire search space, where we use additionally a penalization for the difficult multimodal problems by means of rejecting mutations crossing the search space boundaries. This is necessary, as otherwise in these problems lower optima could be reached in the outer areas. Sampling widths are initialized such that  $\sigma = 4$

for the problems in row 1-6 in Table I. For the difficult multimodal functions we re-adjust the widths where we use for Schaffer  $\sigma = 100$ , Schwefel's function  $\sigma = 220$ , for Eggholder  $\sigma = 320$ . Experiments are conducted over 1000 generations and we accumulate data per experiment from 100 runs. Problem dimension is kept at  $d=2$  in all experiments, as this still allows the interpretation of the retrieved distributions and lifts problems of data sparsity arising with more degrees of freedom. The mixture model is constructed for the first 6 functions by explicitly tuning the number of components using the log-likelihood in Bayesian optimization routine. For the latter functions in row 7-9 we use explicitly the kernel density estimation technique, as otherwise the performance of the operator depends too heavily upon the retrieved mixture.

*Table I. Definitions of the benchmark functions used within the study.*

| Benchmark   | Search Space          | Function Definition   |
|-------------|-----------------------|---|
| Sphere      | $[-5.12, 5.12]^d$     | $f(\mathbf{x}) = \sum_{i=1}^d x_i^2$  |
| Bohachevsky | $[-100, 100]^d$       | $f(\mathbf{x}) = \sum_{i=1}^{d-1} [x_i^2 + 2x_{i+1}^2 - 0.3 \cos(3\pi x_i) - 0.4 \cos(4\pi x_{i+1}) + 0.7]$   |
| Rosenbrock  | $[-5, 10]^d$          | $f(\mathbf{x}) = \sum_{i=1}^d [100 (x_{i+1} - x_i^2)^2 + (x_i - 1)^2]$  |
| Rastrigin   | $[-5.12, 5.12]^d$     | $f(\mathbf{x}) = 10d + \sum_{i=1}^d [x_i^2 - 10 \cos(2\pi x_i)]$  |
| Ackley      | $[-32.768, 32.768]^d$ | $f(\mathbf{x}) = -a \exp \left[ 0.2 (d^{-1} \sum_{i=1}^d x_i^2)^{0.5} \right] + \exp \left[ -1/d \sum_{i=1}^d \cos(2\pi x_i) \right] + a + e$         |
| Griewank    | $[-600, 600]^d$       | $f(\mathbf{x}) = 1 + 1/4000 \sum_{i=1}^d x_i^2 + 1/4000 \prod_{i=1}^d \cos(x_i/\sqrt{i})$   |
| Schaffer    | $[-100, 100]^d$       | $f(\mathbf{x}) = \sum_{i=1}^{d-1} (x_i^2 + x_{i+1}^2)^{0.25} [\sin^2(50 \cdot (x_i + x_{i+1})^{0.10}) + 1.0 (\sqrt{ x_i })]$                          |
| Schwefel    | $[-500, 500]^d$       | $f(\mathbf{x}) = 418.9829 d - \sum_{i=1}^d x_i \sin(\sqrt{ x_i })$  |
| Eggholder   | $[-512, 512]^d$       | $f(\mathbf{x}) = x_1 \sin(\sqrt{ x_2 + 1 - x_1 }) \cos(\sqrt{ x_1 + x_2 + 1 }) + (x_2 + 1) \cos(\sqrt{ x_2 + 1 - x_1 }) \sin(\sqrt{ x_1 + x_2 + 1 })$ |

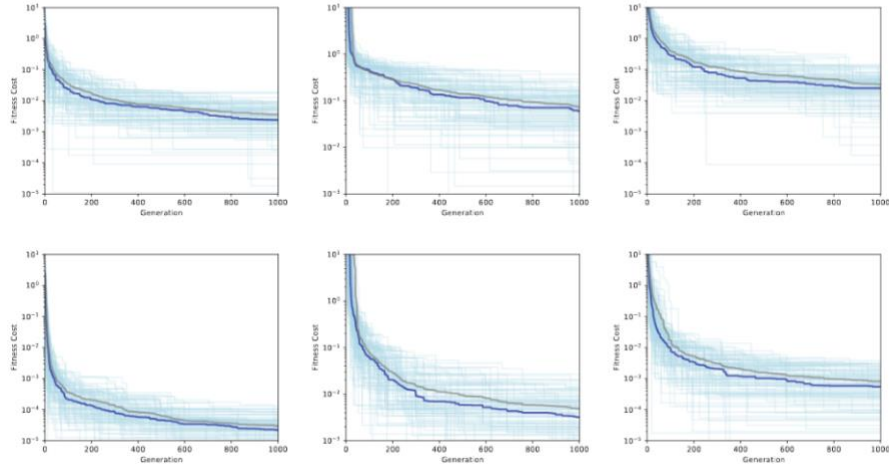
Resulting minimum fitness curves per generation of the optimization runs are plotted per problem group in Figure 7, Figure 8 and Figure 9. Here, top rows are the runs using default mutation distributions, and the lower rows are runs which use distributions of beneficial mutations. Further, median (dark blue), mean (grey) and individual runs (light blue) are plotted. Quite notably, across all considered problems the improved operators significantly improve the search behavior. Particularly, it reduces late convergences by acting in a regularizing fashion. However, also evidently in Figure 8 for Rastrigin's function, where an increased density of runs halts at a fitness costs of about 1, eliminates premature convergence to local optima. Likewise the same observations can be made for Griewank's function. The approach can even be shown to work on difficult multimodal functions in Figure 9. However, we openly admit that further precautions have to be taken for these experiments to work. In particular, for all three we had the sampling widths to the previously mentioned values such that we achieved good convergence behavior in the runs with default sampling. Without taking these precautions, we were not able to achieve any improvements using the new operators. In fact, the retrieved operators were in this case even detrimental to the optimization and encouraged premature convergence into local optima.



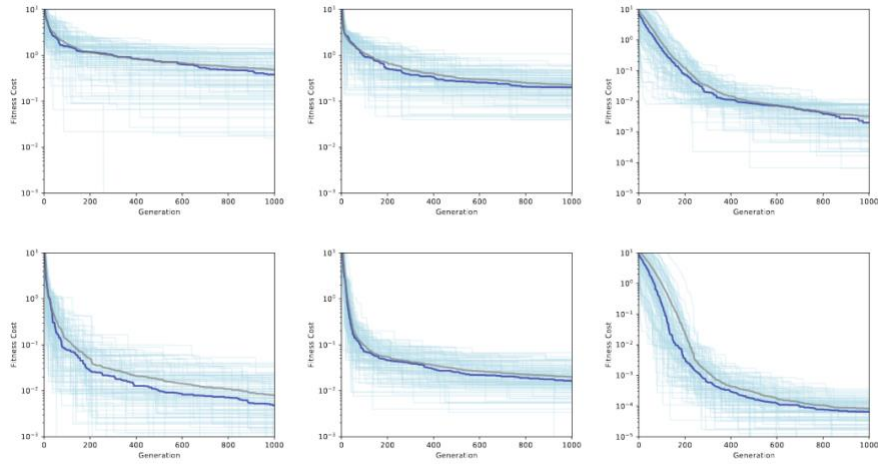
*Table II. Performance values for the default operator ( $N$ ) and improved operator ( $M$ ). Particularly, minimum median and mean fitness after 1000 generations, as well as variance and a statistical significance comparison of both operators using the  $p$ -Value.*

| Benchmark   | $\tilde{f}_{\min}(N)$ | $\bar{f}_{\min}(N)$ | $\sigma_{\min}(N)$ | $\tilde{f}_{\min}(M)$ | $\bar{f}_{\min}(M)$ | $\sigma_{\min}(M)$ | p-value   |
|-------------|-----------------------|---------------------|--------------------|-----------------------|---------------------|--------------------|-----------|
| Sphere      | 2.423e-3              | 3.631e-3            | 3.634e-3           | 2.218e-5              | 2.922e-5            | 2.731e-5           | 3.721e-32 |
| Bohachevsky | 6.000e-2              | 7.318e-2            | 6.329e-2           | 3.200e-3              | 4.896e-3            | 5.168e-3           | 3.328e-26 |
| Rosenbrock  | 2.549e-2              | 3.357e-2            | 3.078e-2           | 5.442e-4              | 8.128e-4            | 8.372e-4           | 1.779e-30 |
| Rastrigin   | 3.835e-1              | 4.880e-1            | 3.914e-1           | 4.806e-3              | 7.991e-3            | 8.622e-3           | 3.126e-32 |
| Ackley      | 2.014e-1              | 2.295e-1            | 1.562e-1           | 1.613e-1              | 1.986e-1            | 1.220e-1           | 9.992e-34 |
| Griewank    | 2.031e-3              | 3.239e-3            | 2.749e-3           | 6.543e-5              | 8.354e-5            | 7.211e-5           | 1.464e-32 |
| Schaffer    | 1.463e+0              | 1.434e+0            | 3.454e-1           | 6.654e-1              | 6.439e-1            | 1.772e-1           | 8.518e-31 |
| Schwefel    | 2.227e+0              | 4.497e+1            | 6.056e+1           | 1.720e-2              | 2.734e-2            | 3.002e-2           | 4.268e-33 |
| Eggholder   | 3.483e-3              | 1.444e-2            | 2.470e-2           | 5.740e-5              | 1.713e+1            | 3.458e+1           | 5.286e-8  |

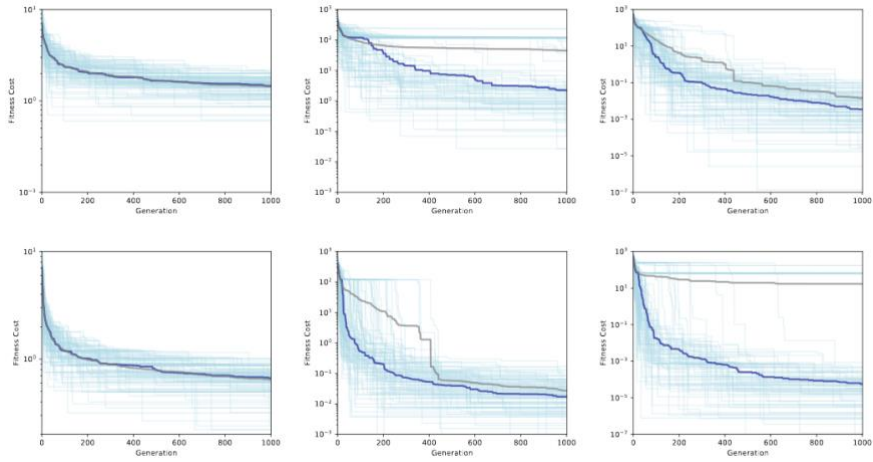
We further list performance values of our experiments, as well as results from a statistical Wilcoxon rank sum test under normal approximation Table II. The results indicate that for a significance level of  $\alpha = 0.05$ , the null hypothesis can be rejected in all experiments.



*Figure 7. Column 1-3: Fitness curves (light blue) for the unimodal Sphere, Bohachevsky's and Rosenbrock's function from 100 runs, as well as median (dark blue) and mean (dark grey) curves. Top row: With default sampling. Bottom row: With improved operators.*



*Figure 8. Column 1-3: Fitness curves (light blue) for the multimodal Rastrigin's, Ackley's and Griewank's function from 100 runs, as well as median (dark blue) and mean (dark grey) curves. Top row: With default sampling. Bottom row: With improved operators.*



*Figure 9. Column 1-3: Fitness curves (light blue) for difficult multimodal functions. In particular, Schaffer's Schwefel's and Eggholder function from 100 runs, as well as median (dark blue) and mean (dark grey) curves with adjusted sampling width to ensure convergence.*



## 6. High Dimensional Surrogate-Assisted Optimisation

Surrogate-Assisted Optimisation (SAO) refers to solving the optimisation problem with the help of a (surrogate) model, which replaces the actual function evaluations by the model prediction. The surrogate model approximates the true values of the objective function under consideration. This is desirable if the objective function is too complex and/or costly. The abstraction provided by the surrogate model is useful in a variety of situations. For instance, it simplifies the task to a great extent in simulation-based modeling and optimisation by providing the opportunity to evaluate the fitness function indirectly if the exact computation is intractable. Surrogate models can also provide practically useful insights, e.g., space visualization and comprehension. Despite the advantages, SAO faces many limitations in constraint handling, dynamic optimisation, multi-objective optimisation and high dimensional optimisation.

Modeling high dimensional optimisation problems with SAO is challenging due to two main reasons. Firstly, more training data is required to achieve a comparable level of modeling accuracy as the dimensionality increases. Secondly, training time complexity often increases rapidly with respect to the dimensionality and the number of training data points. Consequently, constructing the surrogate model becomes costlier. To highlight this issue, upper bounds on the time complexities of the most common surrogate models are presented in Table III. Note that in Table III,  $D$ ,  $N$ ,  $N_{\text{trees}}$ ,  $N_{\text{sv}}$  and  $K$  stand for the dimensionality, the number of training data points, the number of trees in Random Forest (RF), the number of support vectors in Support Vector Machines (SVMs) and the number of neighbors in K-Nearest Neighbors (KNNs) respectively. From Table III, it can be argued that higher dimensionality can severely affect the computational budget in SAO in two ways: directly, i.e., by a higher value of  $D$ , and indirectly, i.e., by a higher value of  $N$ ,  $N_{\text{trees}}$ ,  $N_{\text{sv}}$  and  $K$ .

*Table III. Training and prediction time complexities of the most common surrogate models. Notation  $N_{\text{trees}}$ : is the number of trees in Random Forest,  $N_{\text{sv}}$  is the number of support vectors, and  $K$  the number of neighbours in K-Nearest Neighbours.*

| Model                   | Training                  | Prediction             |
|-------------------------|---------------------------|------------------------|
| Quadratic Regression    | $O(D^4N + D^9 + D^2)$     | $O(D^2)$               |
| Random Forest           | $O(N^2DN_{\text{trees}})$ | $O(NN_{\text{trees}})$ |
| Support Vector Machines | $O(N^2D + N^3)$           | $O(DN_{\text{sv}})$    |
| K-Nearest Neighbors     | $O(1)$                    | $O(KD)$                |
| Kriging                 | $O(N^3D)$                 | $O(ND)$                |

Various methodologies have been proposed to deal with the issue of high dimensionality in SAO including divide-and-conquer, variable screening and mapping the data to a lower dimensional space using dimensionality reduction techniques (DRTs). One of the most common DRTs is the Principal Component Analysis (PCA). PCA can be defined as the orthogonal projection of the data onto a lower dimensional linear space, known as the “principal subspace”, such that the variance of the projected data is maximized. Various generalized extensions of PCA have been established in the literature such as Kernel PCA, Probabilistic PCA and Bayesian PCA. On the other hand,

Autoencoders (AEs) have been contemplated as feed-forward neural networks (FFNNs) trained to attempt to copy their input to their output, so as to learn the useful low dimensional encoding of the data. Like PCA, AEs have also been extended over the years by generalized frameworks such as Sparse Autoencoders, Denoising Autoencoders, Contractive Autoencoders and Variational Autoencoders (VAEs). Besides PCA and AEs, other important DRTs include Isomap, Locally-Linear Embedding, Laplacian Eigenmaps, Curvilinear component analysis and t-distributed stochastic neighbor embedding.

This section of the report summarizes the key results of the ECOLE project [28] on the comparison of DRTs for efficiently constructing the low dimensional surrogate models (LDSMs). To this end, PCA and AEs were chosen. Furthermore, Kernel PCA was incorporated due to the generalized non-linear extension of the classical PCA algorithm. Similarly, VAEs were also considered since they provide the non-linear stochastic encodings of the data space which can be utilized for constructing the surrogate models efficiently. The focal point of the research in ECOLE was to provide a novel perspective on the applicability of these DRTs in SAO. This was accomplished by performing an extensive quality assessment of the corresponding LDSMs on a diverse range of test cases. In the remainder of this section, the details on the experimental setup and the associated results on this research are provided.

## 6.1. Comparison of Dimensionality Reduction Techniques

### 6.1.1 Test Cases

In ECOLE, ten unconstrained, noiseless, single-objective optimisation problems were selected from the continuous benchmark function test-bed known as “Black-Box-Optimisation-Benchmarking” (BBOB) [29]. BBOB provides a total of twenty-four such functions divided in five different categories – “Separable Functions”, “Functions with low or moderate conditioning”, “Functions with high conditioning and unimodal”, “Multi-modal functions with adequate global structure”, and “Multi-modal functions with weak global structure”. Two functions from each of these categories were selected to diversify the landscape of the test cases. The selected functions were ***f2***, ***f3***, ***f7***, ***f9***, ***f10***, ***f13***, ***f15***, ***f16***, ***f20*** and ***f24***. Each of these test functions was evaluated on three different values of dimensionality – **50**, **100**, and **200**. Additionally, all test functions were subject to minimization.

### 6.1.2 Generating the Training Data

The details on the data generation and preprocessing are now shared. For the purpose of data generation, the choice of training sample size  $N$  is problem-dependent. The practical advice [30] however is to begin with  $N = \beta D$ , where  $\beta$  is usually a low valued scalar and  $D$  stands for the dimensionality of the problem. Therefore,  $\beta = 20$  was selected for the Design of Experiment (DoE). Choosing  $\beta = 20$  was based on previous empirical evidence [30] as this resulted in a training data set of moderate size, which was neither too small to train nor too big to hinder the computational efficiency. Additionally, the testing data set with size  $M = 0.2 \beta D$  was generated to evaluate the modeling accuracy of the LDSMs. Notably, it was made sure that the training and

testing data sets were completely disjoint – no data point was shared between the two sets. The sampling locations for both data sets were chosen using a maximum-distance Latin hyper-cube sampling scheme. The data preprocessing in this study was a rather straightforward task involving only the rescaling of the features between **0** and **1**.

### 6.1.3 Implementation Details

Four DRTs were employed – PCA, KPCA, AEs and VAEs. For each of these techniques, specifying the size of the latent dimensionality – denoted as  $L$  – was crucial since it could affect the quality of the corresponding LDSM. Therefore, for each distinct value of dimensionality  $D$ , three values of  $L$  were chosen –  $L \in \{0.7D, 0.4D, 0.1D\}$ . For instance,  $L \in \{35, 20, 5\}$  when  $D = 50$ . In AEs and VAEs, both the encoder and the decoder had four hidden layers each with hyperbolic tangent non-linearity. For PCA and KPCA, a linear transformation of the original features was computed before performing the dimensionality reduction. Two (surrogate) modeling techniques were chosen – Kriging and Polynomial Regression (degree=2 with elastic-net penalty). Notably, both sets of techniques – the DRTs and the modeling techniques, had some hyper-parameters. Therefore, it was crucial to tune these hyper-parameters to get the best quality surrogate models.

### 6.1.4 Hyper-Parameter Optimisation

At this stage, there were a total of **720** cases due to four DRTs, two modeling techniques, ten test functions, three values of original and latent dimensionality each –  $D$  and  $L$ . Therefore, performing Hyper-Parameter Optimisation (HPO) for each of these **720** cases was infeasible. Hence, the number of cases were reduced to a total of **72** by aggregating the performance of the LDSMs on all ten test functions. This implies that the hyper-parameters for each of the **72** cases were optimised, where the cases were defined on combinations of three values of the original dimensionality  $D$ , three values of  $L$ , two modeling techniques and four DRTs. In each of these **72** cases, the hyper-parameters for the DRTs and the modeling techniques were optimised together based on the aggregated quality of the corresponding LDSMs on all ten test functions. The quality assessment for an individual LDSM, i.e., for a particular test function such as  $f_2$ , was measured by taking the so-called relative mean absolute error:

$$\text{RMAE} = \frac{1}{M} \sum_{l=1}^M 100 \cdot \left( \frac{|y_l - \hat{y}_l|}{|y_l|} \right) \quad (1)$$

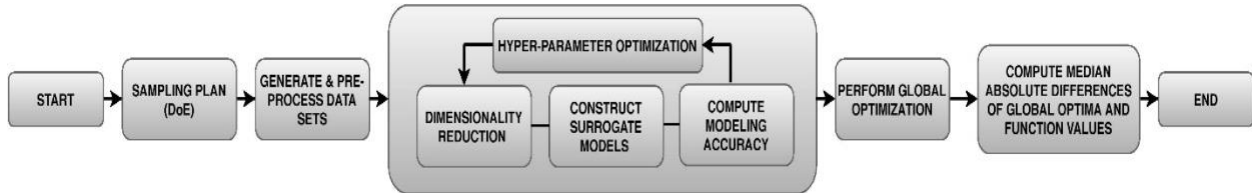
where  $M$  denotes the size of the testing data set. Similarly,  $y_l$  and  $\hat{y}_l$  stand for the true and approximated, i.e., predicted, function values. For HPO, this RMAE was measured for all ten test functions by specifying  $D$ ,  $L$ , the DRT and the modeling technique. After this, the median of the RMAE values on all ten test functions was taken. The goal of the HPO thus simplified to find out the best configuration of the hyper-parameters which minimized this median. This process was repeated for all **72** cases. Overall, this approach made the HPO feasible and ensured that the configuration of the hyper-parameters generalized well across all ten test functions. Tree Parzen Estimator (TPE)

algorithm was employed to perform the HPO for each of the **72** cases discussed above. The number of function evaluations were restricted to **150** for finding the best configuration of the hyper-parameters.

### 6.1.5 Evaluation Criteria

Two criteria were used to evaluate and compare the LDSMs. The first criterion was that of the modeling accuracy. For this criterion, the LDSMs were first constructed in all **720** cases after performing the HPO. Since the modeling technique, the landscape, the dimensionality and the size of LDSM were varied, it was possible to perform a comprehensive analysis of the modeling accuracy of the LDSMs based on a particular DRT. RMAE in Eq. (1) was employed as the performance measure for this criterion.

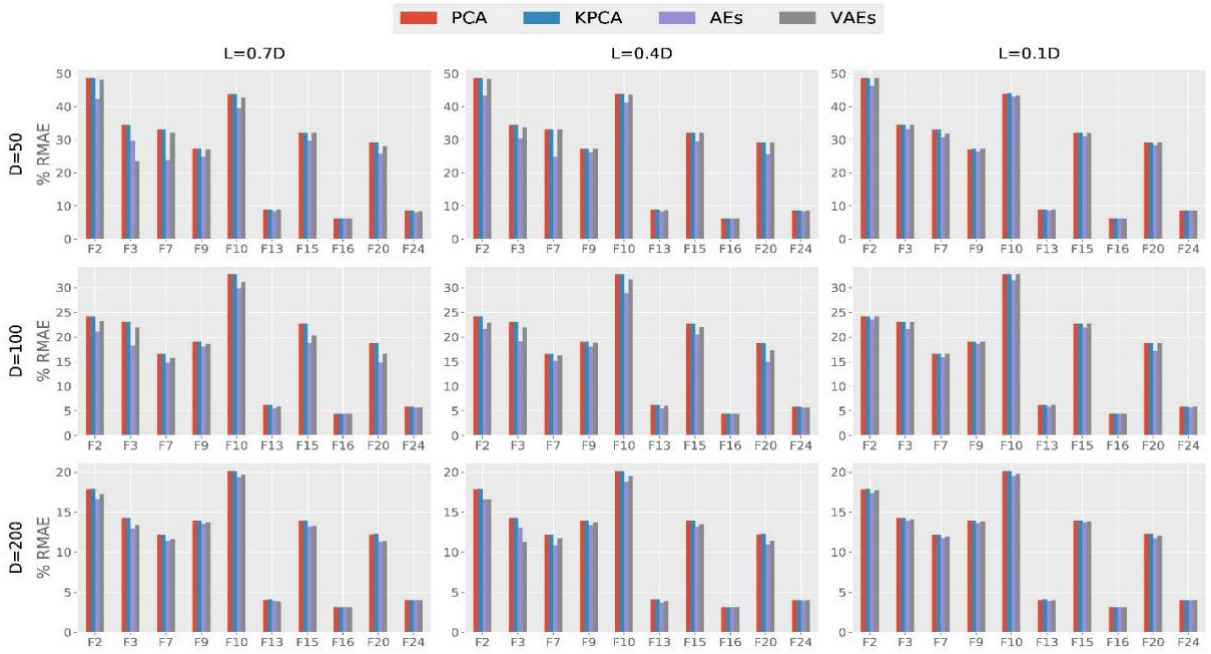
The second criterion to compare the LDSMs was the quality of the proposed optimal solution. Same experimental setup was repeated to compare the LDSMs for this criterion as well. This implies that the LDSMs in each of the **720** cases were constructed based on the best configuration of the hyper-parameters. Then, these LDSMs were utilized to substitute the exact function evaluations within the optimisation loop of the Limited-memory Broyden–Fletcher–Goldfarb–Shanno (L-BFGS) algorithm for global optimisation. For this, the maximum function evaluations were restricted to  $1000 \times D$ . A total of **30** runs of L-BFGS were performed for each LDSM by varying the starting position – initial guess. The resulting optimum in each of these **30** runs was plugged into the test function to achieve the optimal function value. After this procedure, the LDSMs were compared based on two aspects. Firstly, the median absolute difference of the globally optimal function value with the proposed optimal function values based on **30** runs of global optimisation. This was done for each of the **720** cases. The second aspect to compare the LDSMs was the median absolute difference of the proposed optimums with the global optimum based on **30** runs – the median absolute difference of the globally optimal point on the search space with the optimal points proposed by the LDSM. Together, these two aspects were utilized to make a comprehensive analysis on the performance of the LDSMs with respect to the criterion of global optimality. The flowchart of the entire experimental setup is provided in Figure 1 for clarification.



*Figure 10. Flowchart of the experimental setup. Each step of the process is shown in grey rectangles. The central rectangles indicate the hyper-parameter optimisation loop based on the modeling accuracy of the surrogates.*

### 6.1.6 Results

Graphs illustrating the modeling accuracy of the LDSMs are shared in Figure 11 and Figure 12. Both figures contain a total of nine subplots each based on three distinct values of  $D$  and  $L$ . Each subplot contains ten bar charts corresponding to the ten test functions. Furthermore, each bar chart shares the RMAE for the four LDSMs based on the DRTs. From Figure 11 and Figure 12, it was observed that the LDSMs based on AEs achieved the highest modeling accuracy, i.e., lowest RMAE values, in **132/720** cases. This was clearer to notice for  $D \in \{50, 100\}$ , and  $L \in \{0.7D, 0.4D\}$ . In most of the remaining cases, the RMAE values were analogous, though there were some exceptional cases where the LDSMs based on other DRTs performed better. From these figures, it was also observed that the LDSMs performed likewise for both modeling techniques.



*Figure 11. Modeling accuracy of the low dimensional Kriging surrogates for all test cases is presented. The test cases were defined on combinations of ten test functions, four dimensionality reduction techniques and three distinct values for  $D$  and  $L$  each.*



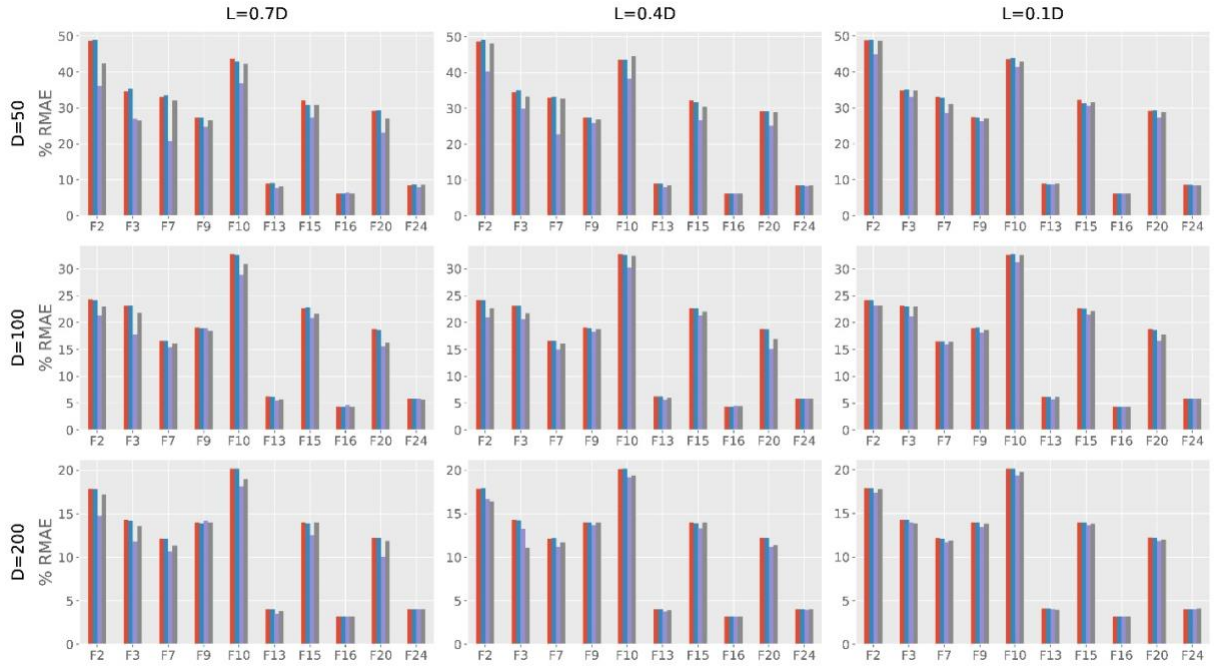


Figure 12. Modeling accuracy of the low dimensional Polynomial surrogates for all test cases is presented. The test cases were defined on combinations of ten test functions, four dimensionality reduction techniques and three distinct values for  $D$  and  $L$  each.

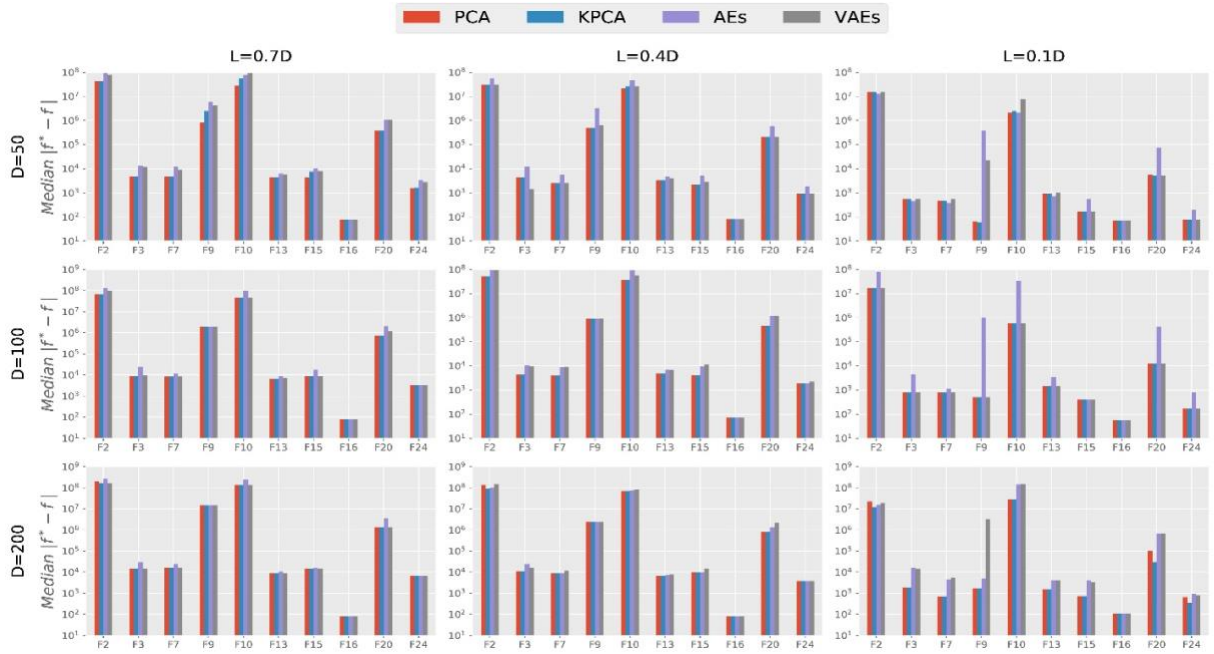


Figure 13. Median absolute difference of the globally optimal function values with the proposed optimal function values for all test cases based on Kriging surrogates is presented. The test cases were defined on combinations of ten test functions, four dimensionality reduction techniques and three distinct values for  $D$  and  $L$  each.

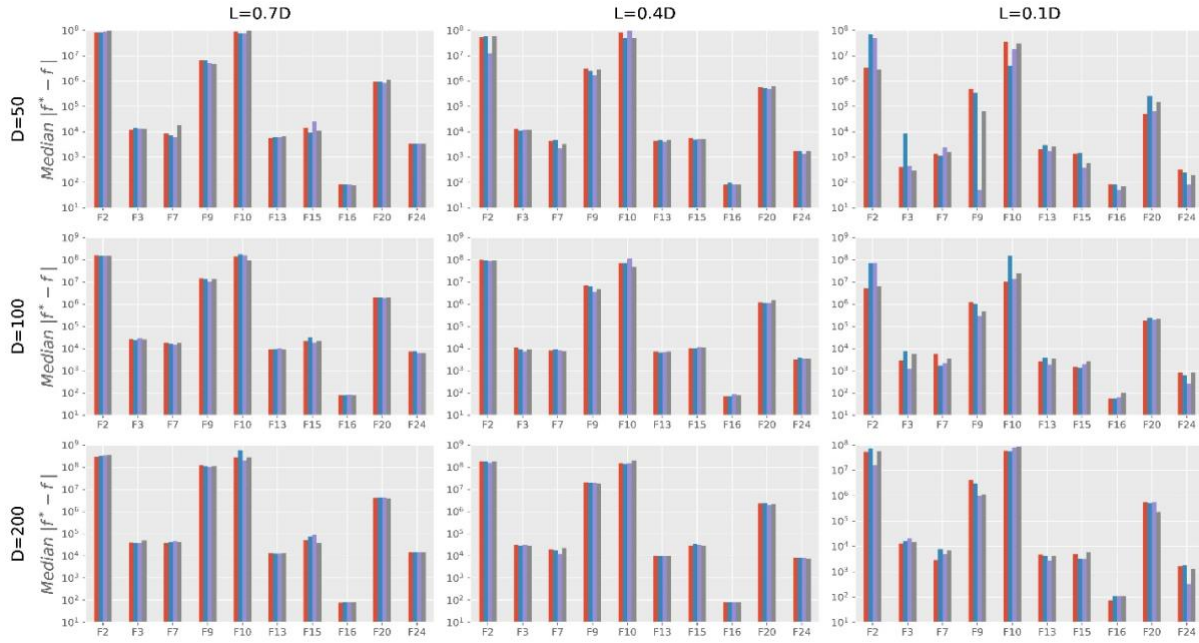


Figure 14. Median absolute difference of the globally optimal function values with the proposed optimal function values for all test cases based on Polynomial surrogates is presented. The test cases were defined on combinations of ten test functions, four dimensionality reduction techniques and three distinct values for  $D$  and  $L$  each.



Figure 15. Median absolute difference of the global optimum with the proposed optimum for all test cases based on Kriging surrogates is presented. The test cases were defined on combinations of ten test functions, four dimensionality reduction techniques and three distinct values for  $D$  and  $L$  each.



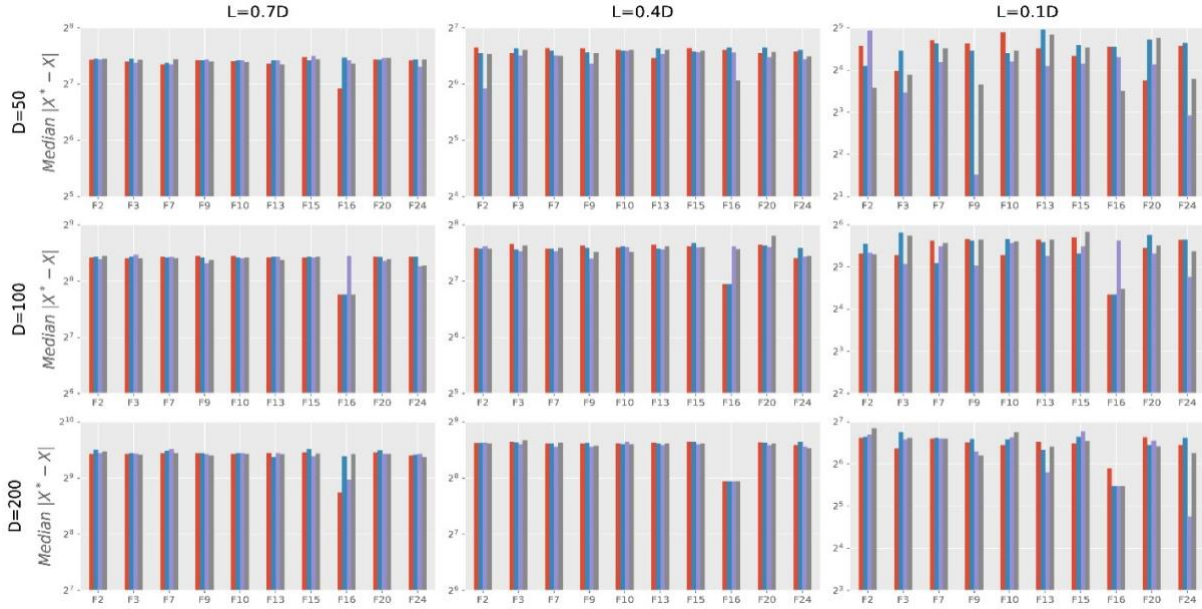


Figure 16. Median absolute difference of the global optimum with the proposed optimum for all test cases based on Polynomial surrogates is presented. The test cases were defined on combinations of ten test functions, four dimensionality reduction techniques and three distinct values for  $D$  and  $L$  each.

Next, the results concerning the criterion of the global optimality are reported. For this, the median absolute difference of the globally optimal function values with the proposed optimal function values is reported in Figure 13 and Figure 14. Both figures contain a total of nine subplots each based on three distinct values of  $D$  and  $L$ . Each subplot contains ten bar charts corresponding to the ten test functions. Furthermore, each bar chart shares the median absolute difference (lower is better) of the globally optimal function values with the proposed optimal function values. From Figure 13 which depicts these results for Kriging, it was observed that PCA, KPCA and VAEs performed similarly in most cases, whereas AEs performed poorly. The performance of the AEs especially deteriorated for  $D = 100$  and  $L = 0.1D$ . In the majority of the remaining cases, PCA performed better than the other DRTs. In the case of Polynomials in Figure 14, all four DRTs performed likewise in most cases except for  $L = 0.1D$ , where AEs performed better than the rest in 15/30 cases.

Similarly, plots depicting the median absolute difference (lower is better) of the global optima with the proposed optima for all 720 cases are provided in Figure 15 and Figure 16. In Figure 15, AEs and VAEs performed poorly, whereas PCA performed better in most cases. In Figure 16, the LDSMs performed similarly in most cases except  $L = 0.1D$ .

### 6.1.7 Conclusion

This section of the report presents some of the most important details from our research in ECOLE, which empirically evaluated and compared four of the most important DRTs for efficiently constructing the LDSMs. The DRTs discussed were PCA, KPCA, AEs and VAEs. The comparison was made on the basis of the quality assessment of the corresponding LDSMs on a diverse range of test cases. There were a total of 720 test cases based on the combinations of ten test functions, four DRTs, two modeling techniques and three distinct values for D and L each. Furthermore, the quality assessment of the LDSMs was based on two criteria: modeling accuracy and global optimality. Based on the observations so far, the following conclusions can be drawn:

- The LDSMs based on AEs had the highest modeling accuracy in **132/720** cases. In most of the remaining cases, the modeling accuracy of the LDSMs was comparable. This demonstrated the efficacy of all four DRTs and provided evidence to suggest AEs as the most competitive DRT in terms of modeling accuracy. However, future research would be necessary to validate this on more complex cases, e.g., real-world applications and optimisation under uncertainty.
- In terms of the global optimality, the LDSMs based on PCA performed better than the others in most cases for Kriging. This aspect could be verified from Figure 13 and Figure 15. For LDSMs based on Polynomials, the performance on this criterion was comparable in most cases.

Although a comprehensive analysis on the performance of the LDSMs is provided, there are a few limitations to discuss. Firstly, the discussion focused on unconstrained, noiseless, single-objective optimisation problems. Therefore, the results cannot be generalized to more complex cases. Secondly, the study did not focus on the size of the training sample size which can be crucial in many cases. It is pertinent to maintain that this was infeasible to include and was left for future work. Based on these rationales, further work is necessary to validate these findings on more complex cases, e.g., multiple-objectives, optimisation under uncertainty, constraint handling and real-world applications.



## 7. Summary and Outlook

In conclusion, we find that we can design search operators as inductive biases for evolutionary algorithms by keeping statistics about the behavior of the operators. A key problem however still lies in harnessing them for high-dimensional problems and investigating their utility in application scenarios. At last, one ideally wants to adapt the operators for a given problem. Thus, it would be desirable to hybridize them with a form of self-adaption as found in many modern algorithms [20]. Our findings also indicate the usefulness of Autoencoders and Principal Component Analysis for constructing the low dimensional surrogates. In particular, Autoencoders perform excellently in terms of modeling accuracy, whilst Principal Component Analysis produce best results with respect to global optimality. Further research must be invested to validate these findings on more complex cases.



## Bibliography

- [1] T. T. H. Dinh, T. H. Chu and Q. U. Nguyen, "Transfer learning in genetic programming," in *2015 IEEE Congress on Evolutionary Computation (CEC)*, 2015.
- [2] S. Friess, P. Tiño, S. Menzel, B. Sendhoff and X. Yao, "Learning transferable variation operators in a continuous genetic algorithm," in *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2019.
- [3] S. Friess, P. Tiño, S. Menzel, B. Sendhoff and X. Yao, "Representing Experience in Continuous Evolutionary optimisation through Problem-tailored Search Operators," in *2020 IEEE Congress on Evolutionary Computation (CEC)*, 2020.
- [4] A. T. W. Min, Y.-S. Ong, A. Gupta and C.-K. Goh, "Multiproblem surrogates: Transfer evolutionary multiobjective optimization of computationally expensive problems," *IEEE Transactions on Evolutionary Computation*, vol. 23, p. 15–28, 2017.
- [5] M. Jiang, Z. Huang, L. Qiu, W. Huang and G. G. Yen, "Transfer learning-based dynamic multiobjective optimization algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 22, p. 501–514, 2017.
- [6] S. J. Louis and J. McDonnell, "Learning with case-injected genetic algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 8, p. 316–328, 2004.
- [7] B. Koçer and A. Arslan, "Genetic transfer learning," *Expert Systems with Applications*, vol. 37, p. 6997–7002, 2010.
- [8] L. Feng, Y.-S. Ong, S. Jiang and A. Gupta, "Autoencoding evolutionary search with learning across heterogeneous problems," *IEEE Transactions on Evolutionary Computation*, vol. 21, p. 760–772, 2017.
- [9] M. Jiang, Z. Wang, L. Qiu, S. Guo, X. Gao and K. C. Tan, "A fast dynamic evolutionary multiobjective algorithm via manifold transfer learning," *IEEE Transactions on Cybernetics*, 2020.
- [10] Z. Wang, H. Hong, K. Ye, M. Jiang and K. C. Tan, "Manifold Interpolation for Large-Scale Multi-Objective Optimization via Generative Adversarial Networks," *arXiv preprint arXiv:2101.02932*, 2021.
- [11] S. J. Pan, I. W. Tsang, J. T. Kwok and Q. Yang, "Domain adaptation via transfer component analysis," *IEEE Transactions on Neural Networks*, vol. 22, p. 199–210, 2010.
- [12] G. Ruan, L. L. Minku, S. Menzel, B. Sendhoff and X. Yao, "Computational Study on Effectiveness of Knowledge Transfer in Dynamic Multi-objective Optimization," in *2020 IEEE Congress on Evolutionary Computation (CEC)*, 2020.
- [13] B. Da, A. Gupta and Y. Ong, "Curbing Negative Influences Online for Seamless Transfer Evolutionary Optimization," *IEEE Transactions on Cybernetics*, vol. no. 99, pp. 1-14, 2018.
- [14] J. Zhang, W. Zhou, X. Chen, W. Yao and L. Cao, "Multisource Selective Transfer Framework in Multiobjective Optimization Problems," *IEEE Transactions on Evolutionary Computation*, vol. 24, p. 424–438, 2019.
- [15] A. T. W. Min, R. Sagarna, A. Gupta, Y.-S. Ong and C. K. Goh, "Knowledge transfer

- through machine learning in aircraft design," *IEEE Computational Intelligence Magazine*, vol. 12, p. 48–60, 2017.
- [16] Y. Jin, H. Wang, T. Chugh, D. Guo and K. Miettinen, "Data-driven evolutionary optimization: An overview and case studies," *IEEE Transactions on Evolutionary Computation*, vol. 23, p. 442–458, 2018.
- [17] A. W. Tan, R. Sagarna, A. Gupta, R. Chandra and Y. S. Ong, "Coping with data scarcity in aircraft engine design," in *18th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, 2017.
- [18] K. Swersky, J. Snoek and R. P. Adams, "Multi-task bayesian optimization," in *Advances in neural information processing systems*, 2013.
- [19] P. Larrañaga and J. A. Lozano, Estimation of distribution algorithms: A new tool for evolutionary computation, vol. 2, Springer Science & Business Media, 2001.
- [20] N. Hansen, "The CMA evolution strategy: a comparing review," in *Towards a New Evolutionary Computation*, J. A. Lozano, P. Larrañaga, I. Inza and E. Bengoetxea, Eds., Springer, 2006, p. 75–102.
- [21] T. M. Mitchell, The need for biases in learning generalizations, Department of Computer Science, Laboratory for Computer Science Research ..., 1980.
- [22] J. B. Losos, The Princeton guide to evolution, Princeton University Press, 2017.
- [23] T. Bäck, G. Rudolph and H.-P. Schwefel, "Evolutionary programming and evolution strategies: Similarities and differences," in *In Proceedings of the Second Annual Conference on Evolutionary Programming*, 1993.
- [24] L. Devroye, Non-Uniform Random Variate Generation, 1st edition ed., Springer-Verlag New York, 1986.
- [25] N.-B. Heidenreich, A. Schindler and S. Sperlich, "Bandwidth selection for kernel density estimation: a review of fully automatic selectors," *AStA Advances in Statistical Analysis*, vol. 97, p. 403–433, 2013.
- [26] D. Simon, Evolutionary optimization algorithms, John Wiley & Sons, 2013.
- [27] G. Celeux, S. Frühwirth-Schnatter and C. P. Robert, "Model selection for mixture models—perspectives and strategies," *Handbook of mixture analysis*, p. 121–160, 2018.
- [28] S. Ullah, D. A. Nguyen, H. Wang, S. Menzel, B. Sendhoff and T. Bäck, "Exploring Dimensionality Reduction Techniques for Efficient Surrogate-Assisted Optimization," in *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2020.
- [29] N. Hansen, A. Auger, R. Ros, O. Mersmann, T. Tušar and D. Brockhoff, "COCO: A platform for comparing continuous optimizers in a black-box setting," *Optimization Methods and Software*, p. 1–31, 2020.
- [30] S. Ullah, H. Wang, S. Menzel, B. Sendhoff and T. Back, "An empirical comparison of meta-modeling techniques for robust design optimization," in *2019 IEEE Symposium Series on Computational Intelligence (SSCI)*, 2019.
- [31] T. Bäck, Evolutionary algorithms in theory and practice: evolution strategies, evolutionary programming, genetic algorithms, Oxford university press, 1996.