Evaluating Exploratory Landscape Analysis for Replacing Real-World Optimization Problems with LLM-generated Proxy Functions

Foppe van Berkel (s2995387)

Supervisors:
Thomas Bäck & Elena Raponi

Bachelor Thesis– Data Science and Artificial Intelligence

Leiden Institute of Advanced Computer Science (LIACS)
liacs.leidenuniv.nl

August 7, 2026

Abstract

Black Box Optimization test suites, such as the Black Box Optimization Benchmarking suite, often contain benchmarks that are abstract and too simple, while real-world inspired suites like MECHBench are very expensive to evaluate on. One solution has been to instead evaluate on representative functions with similar landscapes. This thesis investigates to which extent algorithm performance on expensive problems can be informed by performance on LLM-generated proxy functions with similar landscape characteristics. Using the LLaMEA framework and Exploratory Landscape Analysis, we constructed mathematical functions in Python with specific ELA feature values, and compared the performance of algorithms on these functions with the original problems. We find no significant correlation between problem landscape and algorithm performance, underlining the limitations of ELA in informing optimization problem classes and algorithm selection.

Contents

1 Introduction **1**
- 1.1 Thesis overview . . . 2

2 Background **2**
- 2.1 Black Box Optimization . . . 2
- 2.2 Optimization algorithms for BBO . . . 3
  - 2.2.1 Bayesian Optimization . . . 3
  - 2.2.2 Evolutionary Strategies . . . 3
  - 2.2.3 Differential Evolution . . . 4
- 2.3 Benchmarking . . . 4
  - 2.3.1 BBOB . . . 4
  - 2.3.2 MECHBench . . . 5
- 2.4 Exploratory Landscape Analysis . . . 6
- 2.5 Expanding The Problem Space . . . 7
  - 2.5.1 Previous Methods . . . 8
  - 2.5.2 LLaMEA . . . 8

3 Method **8**
- 3.1 MECHBench Data Collection . . . 8
- 3.2 LLaMEA Loop . . . 11
- 3.3 Algorithms and Analysis . . . 11

4 Results **11**

5 Conclusions and Future Research **15**

References **21**

1 Introduction

Black Box Optimization (BBO) involves finding minima or maxima of optimization problems without an analytic form, a process that has applications in many real-world research domains [4]. Evaluating the performance of optimization algorithms is an integral part of all optimization research, especially because no one algorithm is the best across a large problem space, formally stated by the No Free Lunch (NFL) theorems [1]. The inverse implication of these theorems is that, to outperform random search, the algorithm must exploit the specific problem structure. This makes automated algorithm selection (AAS), the process of automatically choosing the most suitable algorithm for a given problem instance, one of the core components of BBO research.

One of the main steps in the selection process is to test the algorithms on benchmarks: standardized 'example' problems. One collection of such BBO benchmarks is the Black-Box Optimization Benchmarking (BBOB) suite, which is widely used in BBO research to evaluate optimization algorithms [15]. However, while these benchmarks allow for cheap evaluation of algorithms, the problems are often abstract and not representative of real-world problems. This greatly limits the practical insights and applications of such evaluations, leading to the creation of complex benchmark suites rooted in structural mechanics, such as MECHBench [34]. These complex problem properties are designed to provide well-informed algorithm benchmarking and algorithm selection. On the other hand, the same mathematical complexities make for slow computer simulations, meaning AAS that is both well-informed as well as quick remains an ongoing research goal in BBO.

As opposed to the original, expensive problem, one proposed solution is to instead use cheap, representative functions that are similar in terms of problem landscape [26, 20, 40]. The pioneering method to quantify properties of problem landscapes is known as Exploratory Landscape Analysis (ELA), and was proposed by Mersmann et al. [23]. Starting with expert-designed, high-level landscape characteristics such as global structure, multi-modality and separability, these are extended to numerical low-level features, which can be computed relatively cheaply given a number of samples of the real problem, typically obtained from a Design of Experiments (DoE) sample. Because algorithm performance on a given problem is influenced by the problem class, and ELA features are hypothesized to map to the problem class, algorithm performance is expected to correlate with the problem's ELA features.

The current study will evaluate the performance of several optimization algorithms on the three MECHBench problems. This performance is compared with that on proxy functions, which we generate and evolve to have specific landscape features with the LLM-based, evolutionary framework LLaMEA [37]. We contribute to BBO research in the following ways. First, we investigate to which extent ELA features map to problem class and as such inform algorithm performance. Second, we provide as much practical insight as possible by running the algorithms on expensive, engineering-inspired benchmarks in the form of MECHBench. Third, we further explore the possibilities of LLM-based function generation, a new and promising avenue in BBO research.

Research Question: Given the performance ranking of a number of optimization algorithms $A_1, ..., A_k$ on the original real-world problems MB_1, MB_2 and MB_3, and per original problem MB a number of proxy functions $p_{MB1}, ..., p_{MBi}$ with a particular ELA distance to MB:

When comparing MB with $p_{MB1}, ..., p_{MBi}$, to what extent is there a correlation between similarity in algorithm performance ranking and ELA distance?

1.1 Thesis overview

The thesis is structured as follows: Section 2 provides background on BBO, including algorithms and benchmarking, as well as ELA and expanding the problem space. After, Section 3 describes the set-up of the study methods and experiments. Section 4 provides an overview of the obtained results, showing whether MECHBench and proxy problems with similar landscape characteristics produce similar algorithm performance. Lastly, we discuss the results, implications and limitations in Section 5, ending with a final conclusion. All code is made publicly available on the GitHub page to allow for replicability.

2 Background

2.1 Black Box Optimization

Black Box Optimization (BBO) aims to study and design optimization algorithms whose objective values are given in the form of black boxes [4]. Generally, this means an analytic form of the problem is unknown or infeasible to get, and optimization occurs instead through laboratory experiments or computer simulations, i.e., gradient-free as opposed to gradient-based methods. Because these evaluation methods are expensive, such problems are also known as *expensive* BBO problems. A general form of a continuous BBO problem is given in equation 1:

$$min\{f(\mathbf{x}) : \mathbf{x} \in \Omega\}, \Omega \subset \mathbb{R}^n \tag{1}$$

where f is an objective function of the form

$$f : \Omega \to \mathbb{R}^m. \tag{2}$$

The dimensionality of the input space n is at least 1. If the dimensionality of the output space $m = 1$, then f is single-objective, if $m = 2$ it is bi-objective, and generally for $m \geq 2$, f is multi-objective. The current study only investigates single-objective functions.

A BBO problem can also be constrained, meaning f is optimized with respect to the variables $\mathbf{x}$ that adhere to some constraint functions h_i [22]. These can be equality constraints as in $h_i(\mathbf{x}) = c_i$, or inequality constraints as in $h_j(\mathbf{x}) \geq c_j$. The constraints on $\mathbf{x}$ can also be formulated as a constraint on the entire input space, like $\Omega \subset \mathbb{R}^n$. While these problems technically require specialized, constrained optimizers, the problem can typically be reformulated as an unconstrained function as well. For example, Rodriguez et al. [34] do this for their MECHBench problems by reconstructing them as piece-wise functions.

One could be led to think that the search for 'the best' optimization algorithm would be the primary driver of BBO research, but this is not the case: Wolpert & Macready [39] proved that the performance of many optimization algorithms is all equal when averaged over a large problem space. This finding, formally known as a collection of No Free Lunch (NFL) theorems, indicates that the choice for the best algorithm goes to the one that best exploits the specific problem structure at hand. Rice [33] formulated this process as the Algorithm Selection Problem (ASP) and proposed a pipeline meant to guide algorithm selection, where the problem space directly maps to a problem feature space, which maps to the algorithm space. Automatically identifying these features thus

becomes a main component of automated algorithm selection (AAS) and will be discussed in more detail in Section 2.4.

2.2 Optimization algorithms for BBO

Without any analytic information, derivative-free optimization algorithms must use other strategies to optimize the function. Generally, there are two different types of methods used, namely surrogate-based and bio-inspired algorithms [7]. Surrogate-based methods use sampled points to build a surrogate model that estimates the objective function, and balance between exploration of unknown regions and exploitation of the best values so far. Most of the bio-inspired algorithms, on the other hand, are based on evolutionary strategies: they evolve a population of solutions based on survival of the fittest and mutations [13, 36]. In this study, we investigate the performance of six influential optimization algorithms, three Bayesian and three bio-inspired. The current section briefly presents every algorithm, its motivation and how it works.

2.2.1 Bayesian Optimization

BO refers to a class of surrogate-based optimization algorithms that use the principle of Bayesian inference to decide on which points to sample next [12]. In the first stage, a few initial points are sampled, ideally using a diverse, systematic sampling method such as Latin Hypercube Sampling (LHS) or Sobol [31]. These points are then used to build the surrogate model, usually a Gaussian Process (GP) which provides a mean prediction and variance. After, these predictions are used in an acquisition function that determines promising, new points to sample. There are several acquisition functions, including Expected Improvement and Upper Confidence Bound. This process continues until some termination condition is fulfilled.

While a popular and sample-efficient optimization method, BO is typically trumped by other paradigms when it comes to high-dimensional problems, which has recently led to the creation of new high-dimensional BO (HDBO) algorithms. Eriksson et al. [11] proposed an adaptation of BO called Trust Region Bayesian Optimization (TuRBO), which uses surrogate models inside a trust region that dynamically scales its boundaries based on the current best solution. A multi-armed bandit is used to allocate samples inside the trust regions, providing local optimal solutions. Papenmeier et al. [29] proposed BO with adaptively expanding subspaces (BAxUS), which optimizes over nested random subspaces that gradually increase in dimensionality, thus leveraging the efficiency of low-dimensional BO and allowing low-dimensional observations to be carried over to higher-dimensional spaces. Finally, classical BO has the limitation that an analytic form of its acquisition function is often required. Balandat et al. [5] tackle this with their algorithm BOTorch, which uses Monte-Carlo (MC) techniques to approximate acquisition function maxima.

2.2.2 Evolutionary Strategies

ES are inspired by biological evolution and fundamentally work as follows. They start with a parent population, generate new offspring that are recombinations of parents and undergo mutations, and select the surviving individuals based on a selection strategy. Here, mutations are random Gaussian noise, leading to exploration in all directions of the landscape. There are three main parameters that determine the selection strategy: parent population size μ, offspring population size λ and

elitism [2]. When using elitism, it is known as a "plus"-strategy and it means new offspring can be generated using the current population, as well as the parent population from one generation earlier. With no elitism, it is known as a "comma"-strategy, meaning new offspring can be generated by selecting from only the current generation's population of solutions.

One simple ES strategy that still has practical applications is known as (1+1)-ES. Thus, both the parent and offspring population size is 1, and the choice for the next offspring to mutate goes to the best option between the current individual and its parent [10]. Another adaptation of ES is Covariance Matrix Adaptation ES (CMA-ES), which changes how mutations are created. CMA-ES uses the evolutionary trajectory of previous parents to learn its mutation distribution by continuously adapting a covariance matrix [13].

2.2.3 Differential Evolution

As opposed to evolutionary strategies, Differential Evolution (DE) does not sample offspring from a Gaussian distribution. Instead, DE creates a mutant by calculating the difference between two random individual vectors, scaling it and adding this to a random third individual [36]. Generally, DE is popular for its simplicity; the difference vector will first be big and encourage exploration, but get smaller as the algorithm converges.

2.3 Benchmarking

One of the most important aspects of BBO research is accurate assessment and comparison of BBO algorithms. This is done by running the algorithms on pre-defined test functions, known as benchmark functions. Benchmarking algorithms in a scientifically rigorous way consists a lot of subtle, yet critical steps: selecting the objective functions and evaluation metrics, generating data from an experimental design, and interpreting and presenting results to name a few. This process has been notoriously tedious [14, 15], which is why in the past years several testing suites and benchmark frameworks have been proposed. To be clear, a testing suite refers simply to a collection of objective functions. A benchmarking framework or platform, on the other hand, encompasses multiple facets. Typically it includes a (previously created) testing suite, interface for algorithms, data post-processing and performance results that grow more expansive the more people use the framework [14].

2.3.1 BBOB

One of the most widely used testing suites is the BBO Benchmarking (BBOB) family of problems [15]. It consists of 24 noise-free, single-objective functions, all with diverse high-level properties and thus spanning multiple optimization landscapes. Due to its popularity, BBOB has been incorporated into benchmark frameworks such as COCO [14] and the Iterative Optimization Heuristic (IOH) profiler [9]. Another benefit of BBOB is that problem instances can be changed in two main ways, namely by scaling them to different dimensions and using transformations, such as rotation and translation. Principally, these scalers and transformations allow us to expand the problem space by generating diverse problem landscapes. However, as will be discussed more elaborately in section 2.4, past research has shown that BBOB is still not representative enough of all possible problem landscapes [8, 28].

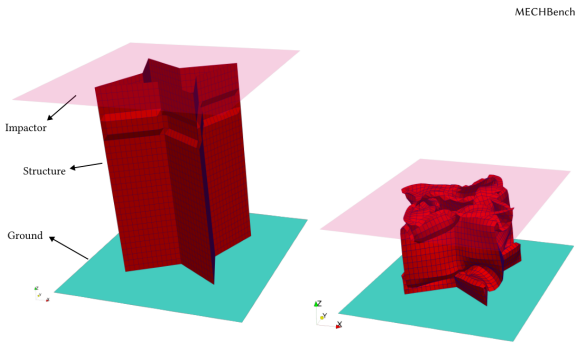

Figure 1: First MECHBench problem MB_1: star shaped crash-box [34].

Another drawback is that synthetic test functions like those in BBOB are often very distinct from real-world optimization problems. Real problems tend to exhibit more complex and irregular landscapes [6], which makes it difficult to assert positive performance of algorithms on the test functions will generalize to real-world expensive problems. This has led to the development of several benchmarking suites designed to offer real-world objective functions. In the context of structural mechanics, MOPTA08 [16] provides problems derived from vehicle crashworthiness scenarios without the high cost of online simulations. However, it has limited scalability due to being constrained to a 124-dimensional input space only. The Mazda CdMOBP suite [18] also offers real-world problems rooted in structural mechanics, but is limited to discrete design variables. Moreover, these test suites suffer slightly from complex experiment and simulation setup.

2.3.2 MECHBench

Our study uses the objective functions from MECHBench [34] as a basis for the assessment of our optimization algorithms. MECHBench offers three shape optimization problems derived from vehicle crashworthiness scenarios, capturing real-world complexities such as nonlinearity, high dimensionality and symmetries. The benchmark workflow works by assembling a Finite Element (FE) model based on a new configuration of n input variables. This FE is simulated using the open-source simulation solver OpenRadioss [3], the output of which is used to automatically parse the objective values. The following paragraphs will describe the three problems in more detail.

1) Star Shaped Crash-Box: This setup starts with a star shaped box stuck to the ground and an impactor crashing it from above (see figure 1). The objective of the structure is to absorb as much crash energy as possible, while allowing a maximum compression of 60 mm. Formally, the goal is to maximize the specific energy absorption SEA, defined as the ratio of the absorbed impact energy E_{abs} and the total mass m_s, subject to an intrusion constraint of 60 mm. To allow the problem to be solved by unconstrained optimizers, it is reformulated as an unconstrained, piecewise objective function (see equation 3).

$$\min_{\mathbf{x}} \mathrm{SEA}(\mathbf{x}) = \begin{cases} -\mathrm{SEA}(\mathbf{x}), & \text{if } \delta(\mathbf{x}) \leq 60 \text{ mm} \\ 100(\delta(\mathbf{x}) - 60), & \text{if } \delta(\mathbf{x}) > 60 \text{ mm} \end{cases} \tag{3}$$

2) Three Point Bending of Layered Beam: The second structure is a beam that consists of five ribs, which is impacted by a cilinder from above (see figure 2). The objective is to minimize

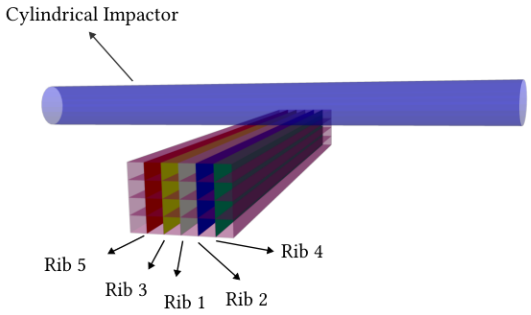

Figure 2: Second MECHBench problem MB_2: three point bending of layered beam [34].

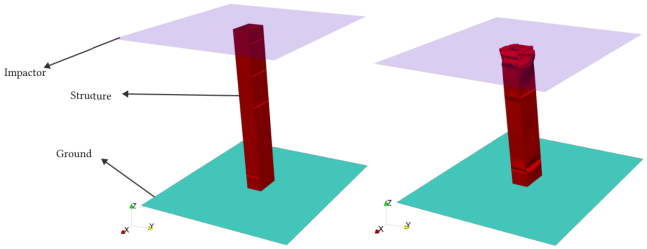

Figure 3: Third MECHBench problem MB_3: long crash tube [34].

the total mass of the structure with a maximum compression of 50 mm. Again, the constrained problem is reformulated as a piecewise function that penalizes an intrusion of more than 50 mm (see equation 4).

$$\min_{\mathbf{x}} \text{Penalized } m_s(\mathbf{x}) = \begin{cases} m_s(\mathbf{x}), & \text{if } \delta(\mathbf{x}) \leq 50 \text{ mm} \\ 4.25952 + 10(\frac{\delta}{50} - 1), & \text{if } \delta(\mathbf{x}) > 50 \text{ mm} \end{cases} \tag{4}$$

3) Long Crash Tube: The final structure is a long crash box clamped to the ground and impacted by a solid from above, which needs its shape and trigger placements optimized. Specifically, the objective is to minimize peak force to the structure relative to the average crash load, because high peak forces can cause localized stress and failures. The ratio of the maximum force F_{peak} and average force F_{mean} is named the load uniformity $LU(\mathbf{x})$. This problem is unconstrained and formulated in equation 5.

$$\min_{\mathbf{x}} \text{LU}(\mathbf{x}) = \left| \frac{F_{\text{peak}}}{F_{\text{mean}}} \right| \tag{5}$$

2.4 Exploratory Landscape Analysis

As mentioned in section 2.1, no one algorithm performs the best across all problem types. Instead, performance differs greatly depending on problem landscape properties, motivating the need for efficient landscape-aware algorithm selection. ELA is one of the most used techniques to cheaply extract high-level landscape features, based on various numerical low-level features. Originally

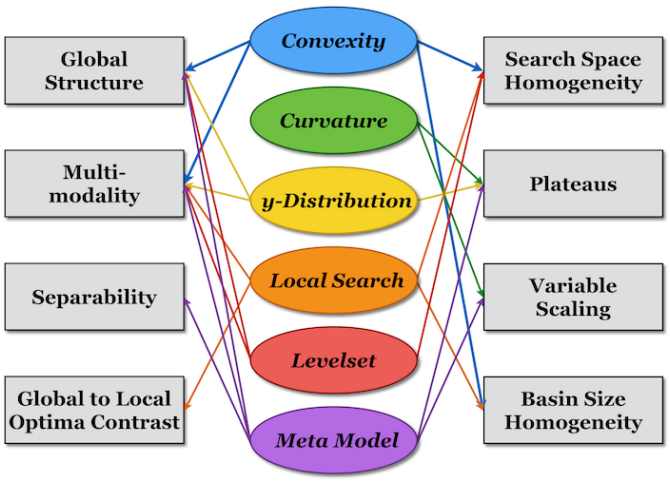

Figure 4: Taken from pflacco documentation [30] and inspired by Mersmann et al. [23]: Relationship between high-level ELA features (white) and low-level features (grey).

proposed by Mersmann et al. [24, 23], the initial ELA collection consisted of the following low-level features: convexity, y-distribution, levelset, meta-model, local search and curvature. These properties inform other high-level features such as global structure, multimodality and separability (see Figure 4). Since this pioneering work, the ELA set has expanded to include features related to Nearest Better Clustering [17]; dispersion [21]; principal component analysis (PCA), information content (IC) and linear model [25].

While investigated extensively in the context of BBOB, research into landscape characteristics of real-world expensive BBO problems has been limited. One such study is that by Long, F.X. [20], who compared BBOB ELA features with those of BMW automotive crash problems. However, their ELA feature computations suffer from small sample sizes due to the high complexity of the optimization problems. Moreover, as opposed to the MECHBench suite, each of these problems is limited to one input dimensionality.

Despite their frequent usage in BBO research, several concerns have been raised regarding ELA features as well: they can be highly correlated, insufficient to distinguish problem types, are generally less informative for high-dimensional problems and possibly too biased to the BBOB suite [20]. A similar conclusion was drawn by Vermetten et al. [38], who extended the BBOB instance space using affine combinations, analysed the instances by both computing ELA features as well as running several algorithms, and found no significant correlation between ELA features and algorithm performance ranking.

2.5 Expanding The Problem Space

Previous research has shown that many test suites, including the greatly utilized BBOB suite, are not representative of all kinds of problem landscapes. For example, Muñoz and Smith-Miles [28] quantified the landscapes of all 24 BBOB functions with eight different ELA features, and showed with principal component analysis (PCA) that the instances only cover a small region of the projected ELA space. This limitation has led to various suggested methods to automatically generate new, space-filling optimization problems. In the rest of this section, we first discuss several

other techniques to generate these functions, before presenting our method based on LLaMEA.

2.5.1 Previous Methods

As a first technique, Long F.X. [20] utilized tree-based randomly generated functions (RGFs) to expand on the BBOB problem space. These functions are constructed by randomly selecting from a set of operators and operands. Moreover, complex properties such as noise and multimodality can be increased with difficulty injection operations. A second method by Dietrich and Mersmann [8] used affine recombinations of two BBOB functions. Both of these techniques are computationally inexpensive and succeed in generating functions with new landscape combinations. However, one drawback is that the RGFs and recombinations cannot be guided towards specific ELA values, making it difficult to populate the problem space in a rigorous way.

The approach that is arguably most similar to ours, is that of Muñoz and Smith-Miles [28]. They use the bio-inspired Genetic Programming (GP) method to generate programs, representing functions, whose sampled ELA feature vector should be equal to a user-defined input. This strategy successfully generated problem instances at new locations of the projected feature space. However, the approach is still limited by the quality of the GP algorithm and its configurations.

2.5.2 LLaMEA

Our method uses the LLM-based evolutionary framework LLaMEA, first proposed by Van Stein and Bäck [37]. LLaMEA iteratively generates code with selected mutations, optimizing solutions based on pre-defined performance metrics, by prompting an LLM caught in an evolutionary loop. In each generation, the LLM is prompted multiple times to create a population of solutions, akin to other evolutionary optimization strategies.

The LLaMEA framework has been previously applied to function generation with specific landscape properties by Skvorc et al. Specifically, they trained an XGBoost model to predict high-level properties such as multimodality and seperability based on low-level ELA values, and guided the LLM towards solutions with target high-level properties [35]. Contrary to optimizing for high-level properties, though, this study directly optimizes for specific, target low-level ELA values. This way, we can generate functions with precise landscapes that closely resemble those of the original, expensive MECHBench problems.

3 Method

The pipeline for this study is visualized in Figure 5, consisting of four main stages. The next subsections will explain every step in more detail.

3.1 MECHBench Data Collection

To facilitate easier comparison of ELA values and problem landscapes, we sample points from the MECHBench problems in the domain $\mathbf{x}_i \in (-5, 5)$. For MB_1 and MB_2, we constrict our dimensionality to $n = 5$, because higher dimensionalities keep the same five initial variable definitions and only add varying thicknesses of control points for the remaining variables. For MB_3, we use $n = 15$ for a similar reason: for $n \leq 15$, the variables control the triggers of the crash tube in

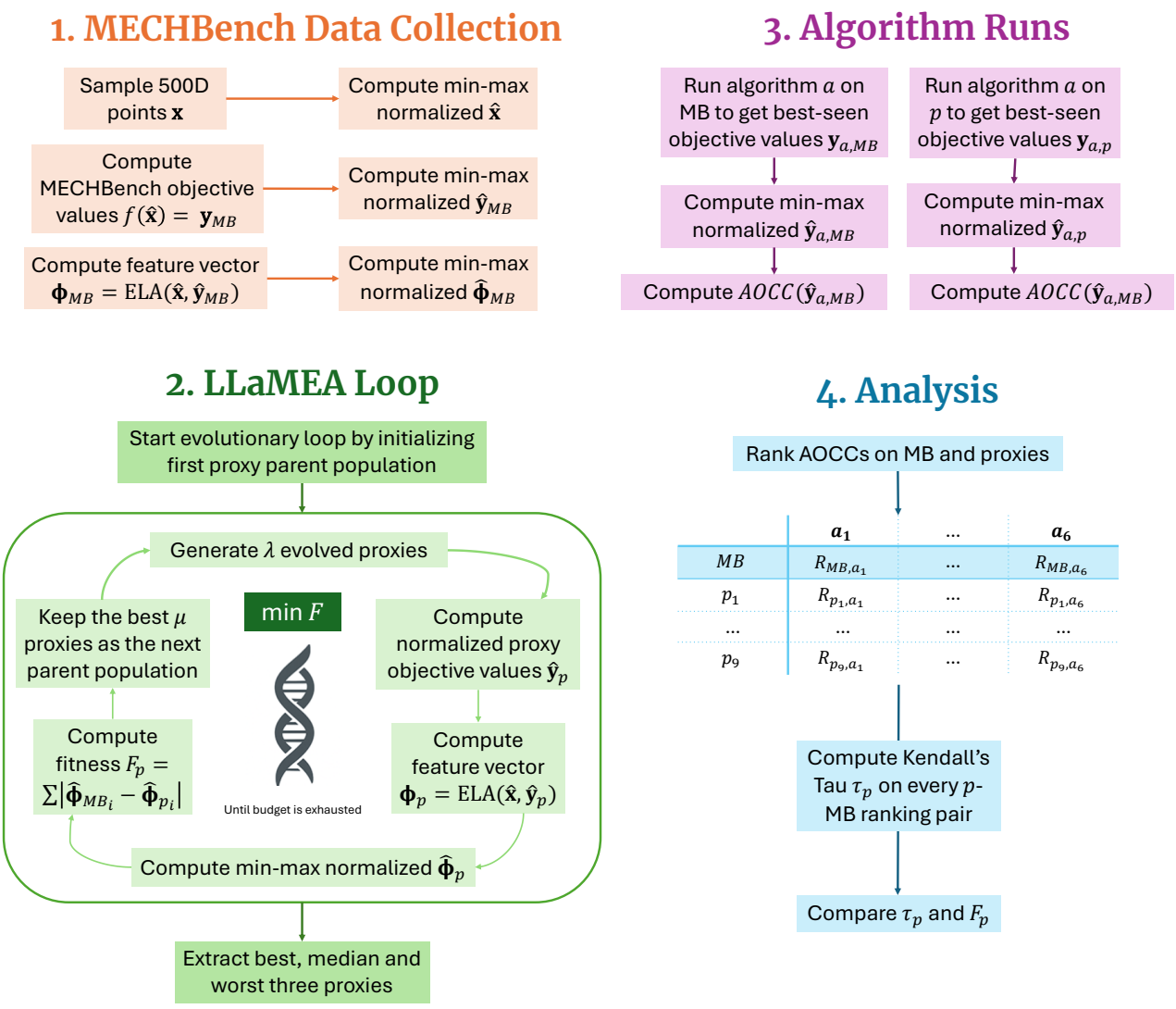

Figure 5: Thesis pipeline, repeated for every original problem MB1, MB2 and MB3.

a mirrored way, while for $15 < n \leq 30$, the triggers are controlled independently. For problem sampling, previous research has often stuck with sample sizes of $250D$ [8, 35]. This choice is mostly based on Renau et al. [32], who tested the classification accuracy of different ELA sets on BBOB functions given various sample sizes. However, previous research by Muñoz and Kirley [27] has shown that various features are particularly volatile at low sample sizes, which gets even worse as a function's ruggedness increases. Considering the mathematical complexity of the MECHBench problems, we increase the sample size to 500D.

We perform min-max normalization on the points and objective values, so that all values are between 0 and 1. This is necessary for a fair comparison of ELA features between the original problem and proxy function: while both use points in the range $(-5, 5)$, the objective values might be at a totally different scale. This can heavily shift some ELA feature values, even if the two problem landscapes are relatively similar.

With the 500D dataset, we can compute the ELA feature values using the Python library pflacco [30]. We make use of 11 different features based on Renau et al. [32], as their set was shown to be sufficient to correctly classify the BBOB functions. There are two exceptions: we firstly exclude the PCA feature, because it is not very descriptive of the problem landscape, shown by how it does not use the objective function values in its computation; and secondly, we add two level features. Granted, we cannot assume the MECHBench problem landscapes should be discernible with the same feature set as for the investigation into BBOB, but the feature set covers every feature class and is diverse nonetheless. This leads us to the following feature collection:

- **disp.ratio_mean_02**: The ratio of the pairwise distances between the 2% most optimal points and all other points.
- **ela_distr.skewness**: The skewness of the y-distribution.
- **ela_meta.lin_simple.adj_r2**: The adjusted correlation coefficient R^2 of a linear model fitted to the data.
- **ela_meta.lin_simple.intercept**: The linear model's intercept coefficient.
- **ela_meta.lin_simple.coef.max**: The linear model's largest coefficient, excluding the intercept.
- **ela_meta.quad_simple.adj_r2**: The R^2 value of a quadratic model fitted to the data.
- **ic.eps_ratio**: The half partial information sensitivity.
- **ic.eps_s**: The settling sensitivity.
- **nbc.nb_fitness.cor**: The correlation between the objective values and their indegree in the nearest-better point graph.
- **ela_level.mmce_qda_25**: The dataset is split into two classes: the points with the lowest 25% objective values and the remaining 75%. A quadratic discriminant analysis (QDA) classifier is trained on X to predict for each point whether it belongs to class 1 or 2. The resulting feature value is the mean misclassification error (mmce) between QDA and the actual split.
- **ela_level.lda_qda_25**: This is the ratio of the mmce from a linear discriminant analysis (LDA) and that of a QDA.

Finally, we also perform min-max normalization on the ELA feature values, using a pre-defined selection of feature statistics calculated from samples of all MECHBench and BBOB problems. With these ranges, we define an ELA scale that is based on all types of landscapes spanning these benchmark suites, ensuring a valid comparison of any ELA values calculated during the LLaMEA loop.

3.2 LLaMEA Loop

We make use of the framework by Skvorc et al. [35], who used LLaMEA to generate proxy functions with specific high-level landscape properties. We extend the LLaMEA class with our own evaluation function using a new fitness metric, namely the Manhattan distance between the normalized MECHBench ELA values and the normalized proxy ELA values. We send the LLM the exact same task prompt in all three problem settings. Thus, the initial parent proxies are unbiased to the original problem landscape and should, on average, be the same across problems. To test LLaMEA against different ES selection strategies, we run three configurations per problem, namely (1+1), (2+4) and (4+8). Because of the stochasticity of LLM output and by extension the LLaMEA framework, we do five runs per configuration to generate multiple population evolution trajectories. Each run uses a budget of 100, where budget refers to the total number of individuals generated. This means the (1+1)-strategy runs for 99 generations, (2+4) for 24 and (4+8) for 12.

3.3 Algorithms and Analysis

We make use of six different, popular optimization algorithms. They are **TuRBO**, **BAxUS**, **BOTorch**, **1+1-ES**, **CMA-ES** and **DE**. With this selection of three Bayesian and three bio-inspired algorithms, we cover many of the most commonly used optimizers of varying strategies. For all three problems, we use a selection of nine proxy functions coming out the LLaMEA loops. It consists of the best, median and worst three proxies in terms of fitness, spanning all five runs across each of the three configurations. This selection ensures we cover the three main areas in the space of landscape preservation. For each problem type, we ran the algorithms with a budget of $30D$ on the MECHBench problem and the corresponding nine proxies.

Using the evaluation runs, we compute for each algorithm-function pairing the area over the convergence curve $AOCC(\mathbf{y}_{a,f})$ based on the definition from [19] and repeated in Equation 6:

$$AOCC(\mathbf{y}_{a,f}) = \frac{1}{B}\sum_{i=1}^{B}\left(1 - \frac{\min(\max(y_i, lb), ub) - lb}{ub - lb}\right), \tag{6}$$

where $y_i = \log(f(x_i) - f^*$ is the log-precision after the i-th evaluation and f^* is the global optimum of f. For all ten functions within a problem type, we then rank algorithm performance in terms of AOCC and turn this into an ordinal ranking. We finally compute Kendall's $\tau(MB, p_i)$ of the MECHBench ranking and each of the nine proxy rankings p_i as a measure of algorithm performance preservation.

4 Results

For all three MECHBench problems, the LLaMEA loop generated around 140 functions until the budget expired. Figure 6 shows for every individual the standard fitness and the best fitness found so far, averaged over all five runs, as well as the standard error represented by the shaded area around the lines. Multiple observations can be made from the graphs. First, for almost all loops, the largest fitness improvement can be found in the first twenty individuals or so, after which the curves get relatively stable. One explanation is that, after the large initial improvements, the LLM continues to add refinements, but they improve the fitness score much less frequently. Another

possibility is that the LLM gets stuck in something of a local optimum, meaning the LLM makes much fewer changes to the code. In any case, the fitness scores converge to a relatively stable point quite quickly.

As a second observation, the standard error fitness tends to be slightly higher in the $1+1$ settings, which hints at a higher stability across runs in the $2+4$ and $4+8$ settings. One might find this counter-intuitive, expecting more diversity in the configuration with eight different individuals every generation as opposed to only one. One explanation might be that the LLM converges to a similar code base more quickly with a larger population. If every run contains eight offspring, there is a higher chance for the same code base to be generated across runs than with only one offspring. In other words, larger populations might be more likely to nudge LLaMEA into similar proxy trajectories across runs.

Finally, there are notable differences in the fitness progression between the three problems. MB_1 starts with an average fitness of around $3.2-3.5$ and decreases to around $1.8-2.0$. In contrast, MB_2 has worse starting fitness, especially in the $2+4$ and $4+8$ configurations, but manages to decrease to a similar score as MB_1. Finally, MB_3 both starts and ends with a notably lower fitness score than the other two. Because the initialization prompt is the same across problems, the difference in starting fitness should on average only be due to the original problem landscape. In other words, these results imply that the MB_3 landscape is more average in terms of ELA features, based on the code an LLM generates without yet any feedback.

A better understanding of these fitness scores can be derived from Figure 7 which shows per setting the average evolution of fitness scores for every feature, i.e. the pairwise distance between it and the MECHBench feature value. Based on these differences in fitness, there are clearly features that are more difficult than others for the LLM to get correct. For example, `ela_meta.lin_simple.coef.max` - the largest absolute coefficient value of a linear model fit to the data - is the worst-performing feature in all MB_1 runs, as well as some MB_2 and MB_3 runs. In MB_2, the feature most different from the original landscape is `ela_level.lda_qda_25`. These plots also reinforce why MB_3's best fitness is so much lower than that of the other problems, seen by how almost all features decrease to nearly 0.0 very quickly. Finally, it shows that `ela_meta.lin_simple.coef.max` is the reason for MB_2's worse initialization in the $2+4$ and $4+8$ configurations, being particularly far away from the original landscape.

Figure 8 plots per feature the normalized feature value of every proxy function from the initial parent populations, across all runs, configurations and problems. This gives a look at the diversity in unbiased, initialized proxy landscapes. We can see how some features are stable and have very similar values every initialization, such as `ela_distr.skewness`, `ela_level.lda_qda_25` and the IC features. Most other features have a few value clusters, hinting at two or three distinct landscape implementations chosen by the LLM. Also plotted are the MECHBench feature values, showing in which properties the original problem landscapes overlap or not. Finally, the plots provide some insight into earlier conclusions on fitness progression. For example, while most `ela_meta.lin_simple.coef.max` values are initialized around 0, there is also a smaller cluster completely at the top, which would explain the bad feature initialization for MB_2 for the $(2+4)$ and $(4+8)$ algorithms we saw in Figure 7. Additionally, we see how the MB_1 and MB_3 values of `ela_level.lda_qda_25` are quite close to the initialized proxies, while MB_2 is distinctly further away, possibly motivating why that feature progresses the worst for MB_2.

As a final look at the LLaMEA results, Figure 9 shows a PCA projection of the original MECHBench ELA vector and that of every generated proxy function, reduced from the 11-

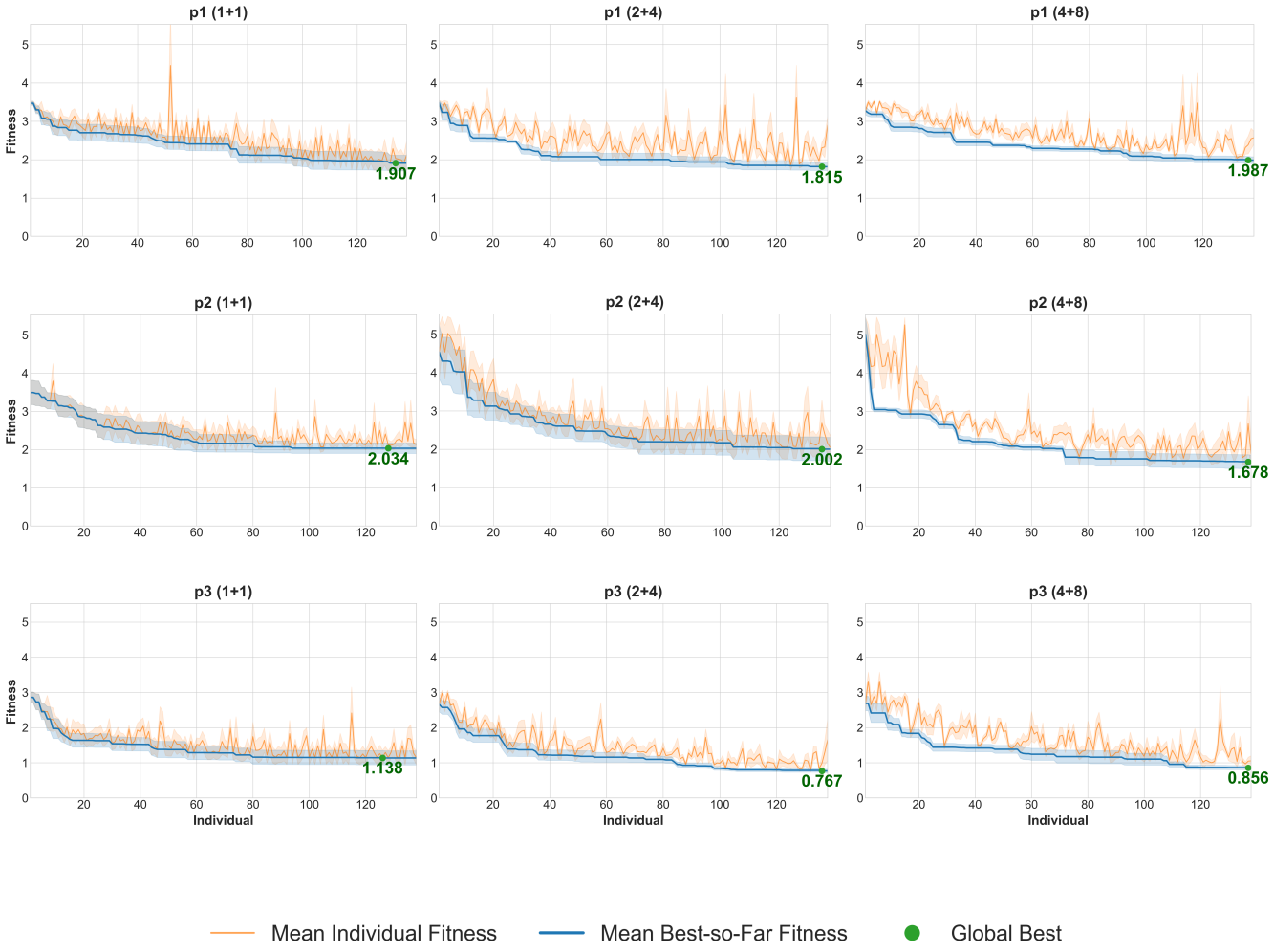

Figure 6: LLaMEA fitness evolution for each problem and ES configuration, across generations averaged over five runs. Shown are the standard and best-so-far fitness of each individual, as well as the global best.

dimensional ELA space to a 2D plot. All nine configurations share the same PCA model, with two principal components that in total explain around 72% of the variance in ELA vectors. As expected, all three configurations within each problem show a similar projection, as well as a similar trajectory in terms of landscape from the initial to the later generations. In MB_1, many individuals start in the (top) left and scatter slightly more to the right. In MB_2 and MB_3, they tend to move from the top left to the bottom left, with MB_2 showing a peculiar curve of individuals that is significantly more to the right. These similarities in trajectory show that many individuals within a problem follow a similar landscape progression, guided by only one fitness metric.

The final results of the algorithm evaluations are shown in Figure 10, which plot the best, median and worst three proxies in terms of fitness and Kendall's τ, i.e. its similarity in terms of algorithm performance ranking to the ranking on the MECHBench problem. If there is a correlation between landscape similarity and similarity in algorithm performance, we would expect a downward

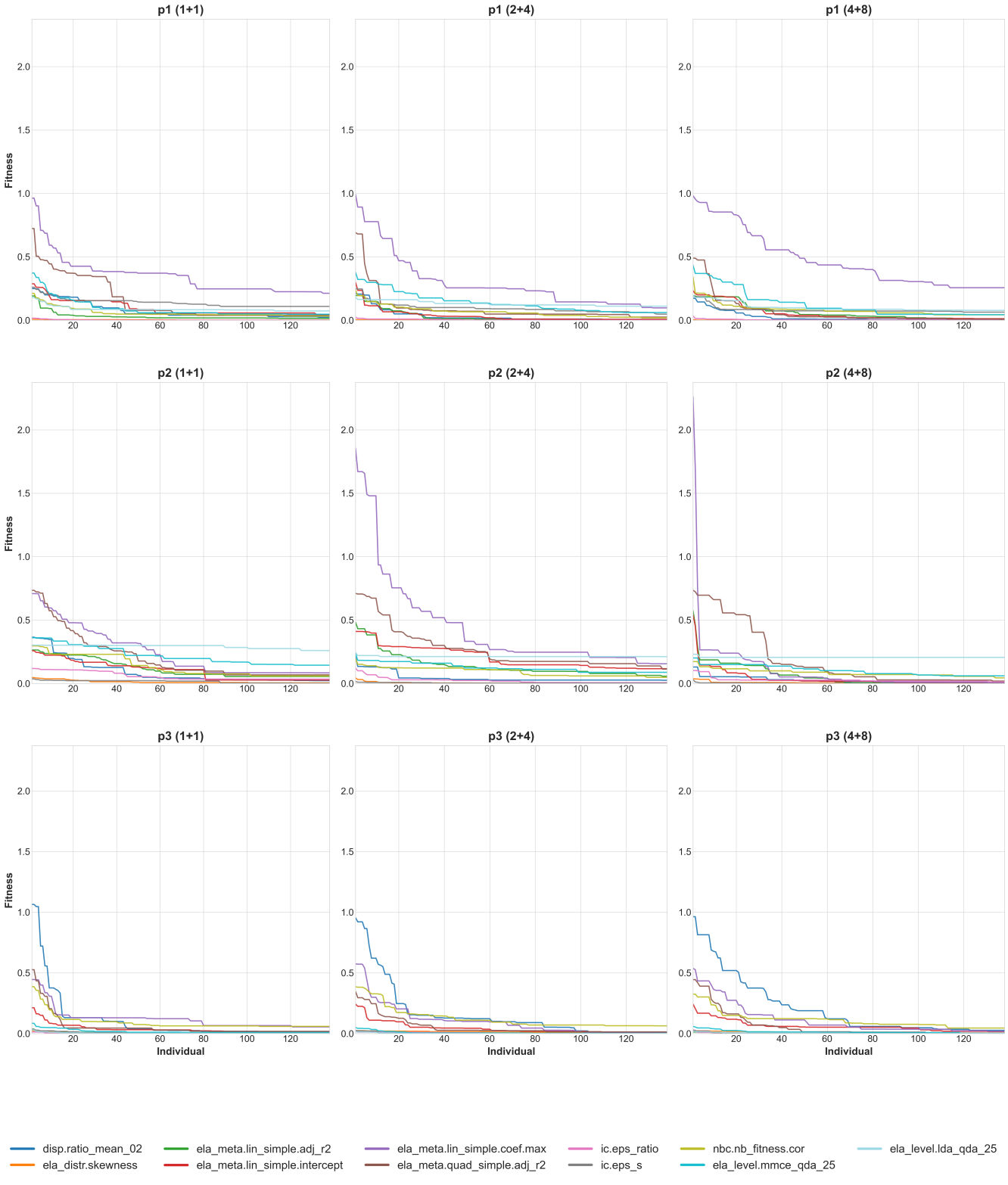

Figure 7: LLaMEA fitness evolution for each feature per problem and ES configuration, across generations averaged over five runs. Shown are the best-so-far fitnesses of each individual per feature.

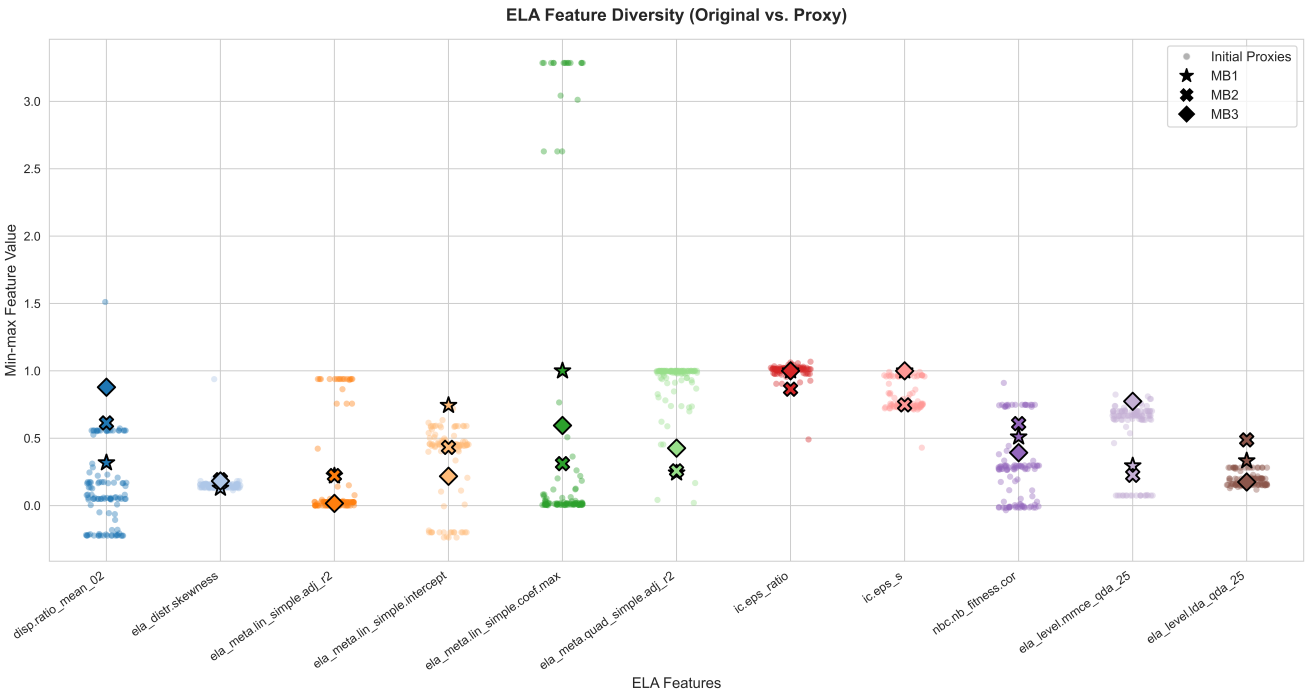

Figure 8: ELA feature values of all initial, generation 0 proxy functions, as well as the original three MECHBench problems. This shows the diversity and clustering of proxy landscape properties as generated by the unbiased, initial task prompt.

trend: as fitness improves (decreases to 0), Kendall's τ improves (increases to 1). This is not the case in MB_1, mostly due to the proxy with the worst fitness score having the most similar algorithm ranking of all proxies. In MB_2, there is a slight downward trend, with the best three proxies indeed showing the highest similarity in algorithm ranking. Interestingly, though, the worst three proxies perform better than the median three, showing that the correlation between landscape preservation and preservation of algorithm performance is relatively weak. MB_3 shows a very minimal upward trend, but the correlation is much weaker than in MB_1.

5 Conclusions and Future Research

Following Rice's [33] framework, ELA features have been prominently used as the feature space bridge between the problem and algorithm space, with varying levels of success. With these new results, we aim to answer the research question introduced at the start of the paper: When comparing the original optimization problem with its proxy functions, to what extent is there a correlation between similarity in algorithm performance ranking and ELA distance? The current study follows the footsteps of e.g. [38] in that we could not find a correlation between numerical ELA features and algorithm performance preservation. This finding indicates that ELA is insufficient to describe the problem landscape in a way that can accurately guide algorithm selection. By using a real-world inspired benchmarking suite in the form of MECHBench, we additionally underline this limitation of ELA in the context of performing algorithm selection on real-world optimization problems.

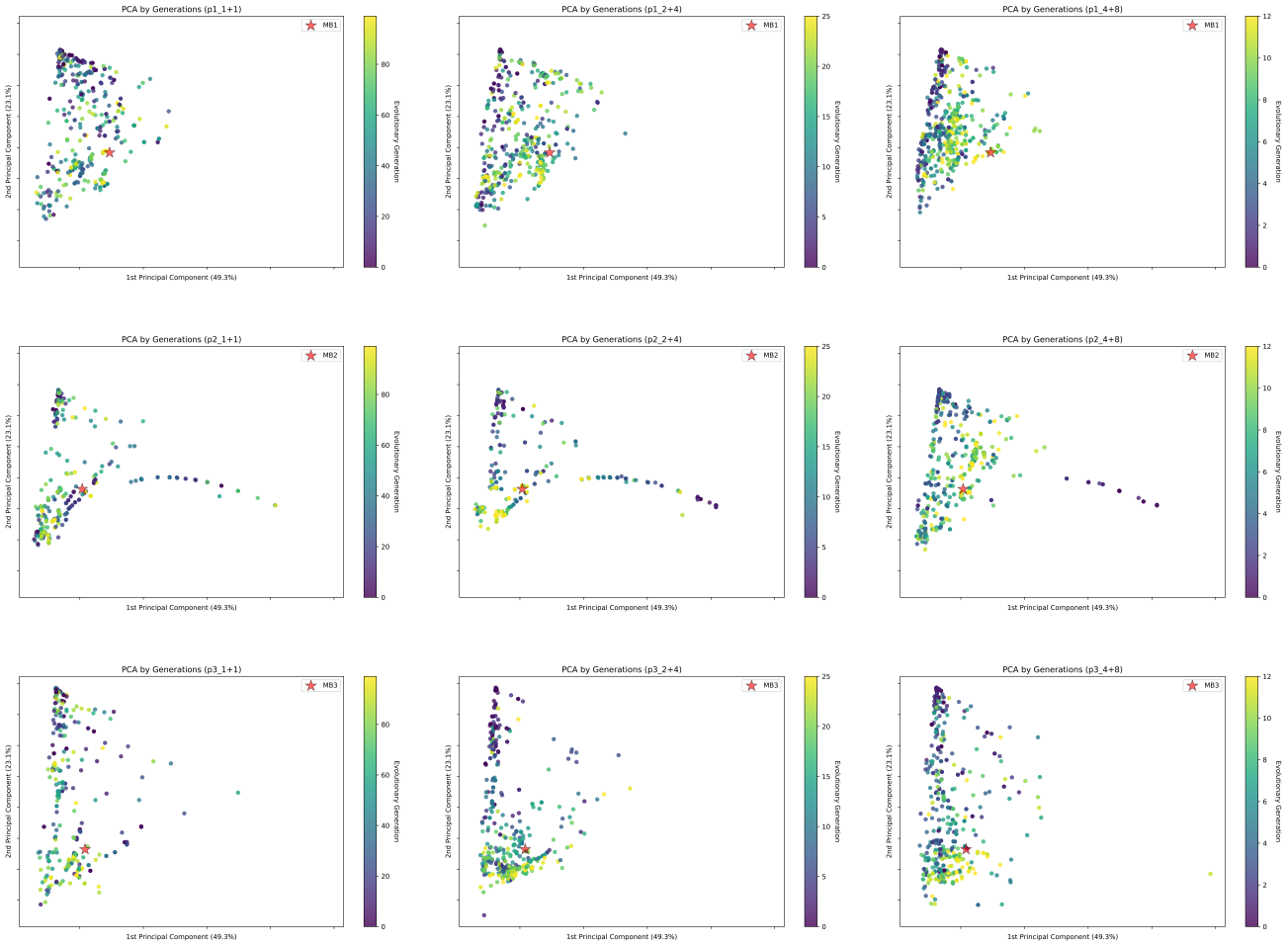

Figure 9: Projection of all five runs' proxy ELA vectors, as well as the original MECHBench vector, per problem and ES configuration. All 9 configurations share the same, global PCA model and thus the same principal components.

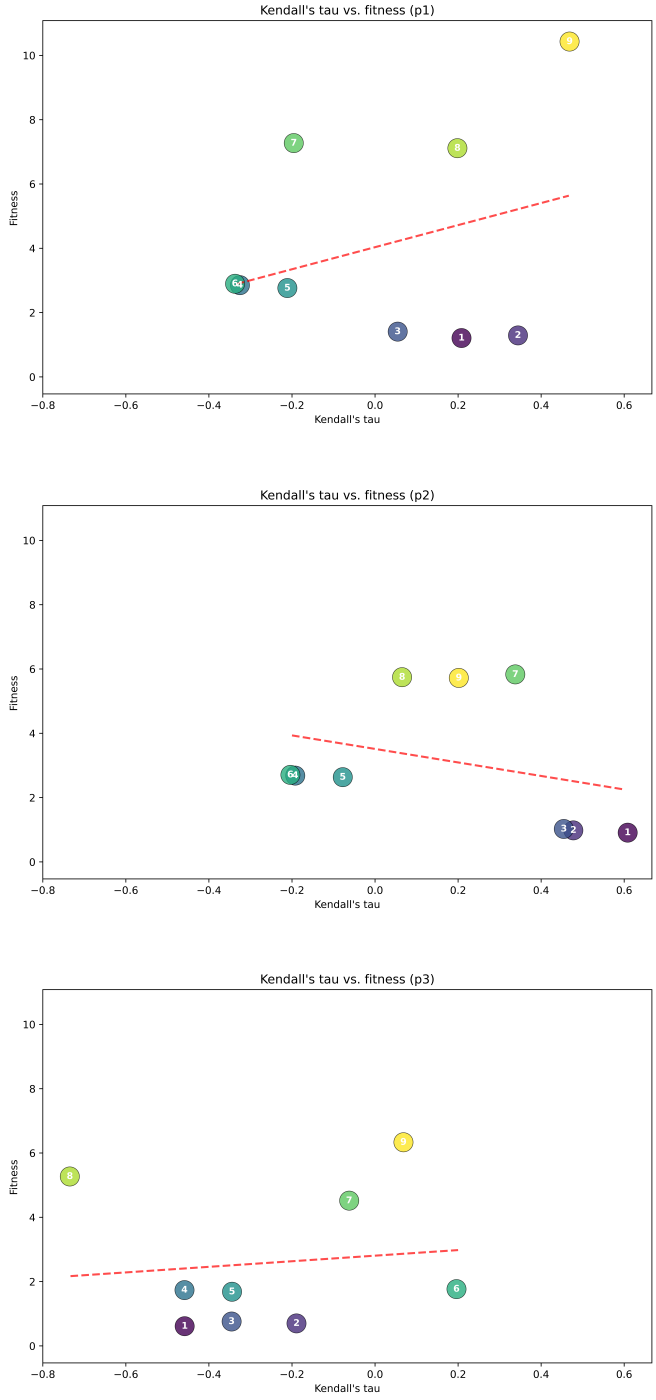

Figure 10: Scatterplot showing the Kendall's τ (x-axis) and fitness score (y-axis) of the best (labelled 1-3), median (4-6) and worst (7-9) proxies for each of the three problem settings. Kendall's τ represents preservation of algorithm performance ranking; fitness represents preservation of ELA landscape.

On the other hand, we further explore the possibilities of LLM-based function generation in BBO research. We provide several insights into the problem landscapes generated by an LLM when analysed using ELA, such as that some feature values are particularly difficult to match, features of initial landscapes tend to cluster around a few specific values, and proxy landscapes evolve in similar directions across runs and ES configurations. These findings show the opportunities and limitations of LLaMEA in function generation, when it comes to both broadly evolving problem landscapes, as well as the ease or difficulty in matching particular ELA feature values.

Still, the current study has several limitations that should be taken into account when considering the conclusions and conducting future studies. Perhaps most notably, our sample size of nine proxy functions per original problem is quite small, which could lead to one outlier proxy being able to completely shift the correlation between landscape and algorithm performance preservation. Another limitation lies in our measure of landscape similarity, in the form of the Manhattan distance between ELA feature vectors. Two proxy functions could be equally distant from the original landscape, and thus have equal fitness score, but still have very different landscapes, especially because the best fitness scores only got as low as around $1.0 - 2.0$. Similarly, this metric does not consider how an algorithm might be influenced greatly by one or multiple particular features.

References

[1] *Black Box Optimization, Machine Learning, and No-Free Lunch Theorems*. Springer International Publishing, 2021.

[2] Zaid Alrashdi and Mohammad Sayyafzadeh. Evolution strategy algorithm in well placement, trajectory, control and joint optimisation. *Journal of Petroleum Science and Engineering*, 177:1042–1058, 2019.

[3] Altair and Contributors. Openradioss, 2025. GitHub commit 6c16e4b.

[4] Charles Audet and Warren Hare. *Derivative-Free and Blackbox Optimization*. Springer International Publishing, 2017.

[5] Maximilian Balandat, Brian Karrer, Daniel Jiang, Samuel Daulton, Ben Letham, Andrew G Wilson, and Eytan Bakshy. BOTorch: A framework for efficient monte-carlo bayesian optimization. *Advances in neural information processing systems*, 33:21524–21538, 2020.

[6] Laurens Bliek, Arthur Guijt, Rickard Karlsson, Sicco Verwer, and Mathijs de Weerdt. Benchmarking surrogate-based optimisation algorithms on expensive black-box functions. *Applied Soft Computing*, 147:110744, November 2023.

[7] Thomas H. W. Bäck, Anna V. Kononova, Bas van Stein, Hao Wang, Kirill A. Antonov, Roman T. Kalkreuth, Jacob de Nobel, Diederick Vermetten, Roy de Winter, and Furong Ye. Evolutionary algorithms for parameter optimization—thirty years later. *Evolutionary Computation*, 31(2):81–122, June 2023.

[8] Konstantin Dietrich and Olaf Mersmann. Increasing the diversity of benchmark function sets through affine recombination. In *International Conference on Parallel Problem Solving from Nature (PPSN)*, pages 590–602. Springer, 2022.

[9] Carola Doerr, Hao Wang, Furong Ye, Sander Van Rijn, and Thomas Bäck. IOHprofiler: A benchmarking and profiling tool for iterative optimization heuristics. *arXiv preprint arXiv:1810.05281*, 2018.

[10] Stefan Droste, Thomas Jansen, and Ingo Wegener. On the analysis of the (1+1)-evolutionary algorithm. *Theoretical Computer Science*, 276(1-2):51–81, 2002.

[11] David Eriksson, Michael Pearce, Jacob R Gardner, Ryan Turner, and Matthias Poloczek. *Scalable global optimization via local Bayesian optimization*. Curran Associates Inc., Red Hook, NY, USA, 2019.

[12] Roman Garnett. *Bayesian Optimization*. Cambridge University Press, January 2023.

[13] N. Hansen and A. Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: the covariance matrix adaptation. In *Proceedings of IEEE International Conference on Evolutionary Computation*, pages 312–317, 1996.

[14] Nikolaus Hansen, Anne Auger, Raymond Ros, Olaf Mersmann, Tea Tušar, and Dimo Brockhoff. Coco: a platform for comparing continuous optimizers in a black-box setting. *Optimization Methods and Software*, 36(1):114–144, August 2020.

[15] Nikolaus Hansen, Steffen Finck, Raymond Ros, and Anne Auger. Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions. Research Report RR-6829, INRIA, 2009.

[16] Donald R Jones. Large-scale multi-disciplinary mass optimization in the auto industry. In *MOPTA 2008 Conference (20 August 2008)*, volume 64, 2008.

[17] Pascal Kerschke, Mike Preuss, Simon Wessing, and Heike Trautmann. Detecting funnel structures by means of exploratory landscape analysis. In *Proceedings of the 2015 Annual Conference on Genetic and Evolutionary Computation*, GECCO '15, page 265–272. ACM, 2015.

[18] Takehisa Kohira, Hiromasa Kemmotsu, Oyama Akira, and Tomoaki Tatsukawa. Proposal of benchmark problem based on real-world car structure design optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*, GECCO '18, page 183–184. ACM, 2018.

[19] Wenhu Li, Niki van Stein, Thomas Bäck, and Elena Raponi. LLaMEA-BO: A large language model evolutionary algorithm for automatically generating bayesian optimization algorithms. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 909–918, 2026.

[20] Fu Xing Long. *Optimizing solvers for real-world expensive black-box optimization with applications in vehicle design*. PhD thesis, Leiden University, 2025.

[21] Monte Lunacek and Darrell Whitley. The dispersion metric and the cma evolution strategy. In *Proceedings of the 8th annual conference on Genetic and evolutionary computation*, GECCO06, page 477–484. ACM, 2006.

[22] Joaquim R. R. A. Martins and Andrew Ning. *Engineering Design Optimization*. Cambridge University Press, November 2021.

[23] Olaf Mersmann, Bernd Bischl, Heike Trautmann, Mike Preuss, Claus Weihs, and Günter Rudolph. Exploratory landscape analysis. In *Proceedings of the 13th Annual Conference on Genetic and Evolutionary Computation (GECCO)*, pages 829–836, 2011.

[24] Olaf Mersmann, Mike Preuss, and Heike Trautmann. Benchmarking evolutionary algorithms: Towards exploratory landscape analysis. In *International conference on Parallel Problem Solving from Nature (PPSN)*, pages 73–82. Springer, 2010.

[25] Mario A. Munoz, Michael Kirley, and Saman K. Halgamuge. Exploratory landscape analysis of continuous space optimization problems using information content. *IEEE Transactions on Evolutionary Computation*, 19(1):74–87, February 2015.

[26] Mario A Muñoz, Yuan Sun, Michael Kirley, and Saman K Halgamuge. Algorithm selection for black-box continuous optimization problems: A survey on methods and challenges. *Information Sciences*, 317:224–245, 2015.

[27] Mario Andrés Muñoz and Michael Kirley. Sampling effects on algorithm selection for continuous black-box optimization. *Algorithms*, 14(1):19, 2021.

[28] Mario A. Muñoz and Kate Smith-Miles. Generating new space-filling test instances for continuous black-box optimization. *Evolutionary Computation*, 28(3):379–404, 2020.

[29] Leonard Papenmeier, Luigi Nardi, and Matthias Poloczek. Increasing the scope as you learn: Adaptive Bayesian optimization in nested subspaces. *Advances in Neural Information Processing Systems*, 35:11586–11601, 2022.

[30] Raphael Patrick Prager and Heike Trautmann. Pflacco: Feature-based landscape analysis of continuous and constrained optimization problems in python. *Evolutionary Computation*, 32(3):211–216, 2024.

[31] Marissa Renardy, Louis R. Joslyn, Jess A. Millar, and Denise E. Kirschner. To Sobol or not to Sobol? The effects of sampling schemes in systems biology applications. *Mathematical Biosciences*, 337:108593, 2021.

[32] Quentin Renau, Johann Dréo, Carola Doerr, and Benjamin Doerr. Towards explainable exploratory landscape analysis: extreme feature selection for classifying bbob functions. In *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*, pages 17–33. Springer, 2021.

[33] John R Rice. The algorithm selection problem. In *Advances in Computers*, volume 15, pages 65–118. Elsevier, 1976.

[34] Iván Olarte Rodríguez, Maria Laura Santoni, Fabian Duddeck, Carola Doerr, Thomas Bäck, and Elena Raponi. Mechbench: A set of black-box optimization benchmarks originated from structural mechanics. *arXiv preprint arXiv:2511.10821*, 2025.

[35] Urban Skvorc, Niki van Stein, Moritz Seiler, Britta Grimme, Thomas Bäck, and Heike Trautmann. Llm driven design of continuous optimization problems with controllable high-level properties. In *International Conference on the Applications of Evolutionary Computation (Part of EvoStar)*, pages 183–199. Springer, 2026.

[36] Rainer Storn and Kenneth Price. Differential evolution – a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4):341–359, December 1997.

[37] Niki Van Stein and Thomas Bäck. LLaMEA: A large language model evolutionary algorithm for automatically generating metaheuristics. *IEEE Transactions on Evolutionary Computation*, 29(2):331–345, 2024.

[38] Diederick Vermetten, Furong Ye, Thomas Bäck, and Carola Doerr. Ma-bbob: Many-affine combinations of BBOB functions for evaluating AutoML approaches in noiseless numerical black-box optimization contexts. In *International Conference on Automated Machine Learning*, pages 7–1. PMLR, 2023.

[39] David H Wolpert and William G Macready. No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation*, 1(1):67–82, 2002.

[40] Haoran Yin, Shuaiqun Pan, Zhao Wei, Jian Cheng Wong, Yew-Soon Ong, Anna Kononova, Thomas Bäck, and Niki Stein. Landscape-aware automated algorithm design: An efficient framework for real-world optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO)*, pages 1239–1248, 2026.