UNIVERSIDADE D
COIMBRA

Gabriel Mendes Fernandes

QUALITY/DIVERSITY APPROACHES FOR STRUCTURAL OPTIMIZATION OF BRIDGES

Dissertation in the context of the Master in Informatics Engineering, specialization in Intelligent Systems, supervised by Prof. João Correia and Prof. Nuno Lourenço and presented to the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

July of 2023

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D
COIMBRA

DEPARTMENT OF INFORMATICS ENGINEERING

Gabriel Mendes Fernandes

Quality/Diversity Approaches for Structural Optimization of Bridges

Dissertation in the context of the Master in Informatics Engineering, specialization in Intelligent Systems, supervised by Prof. João Correia and Prof. Nuno Lourenço and presented to the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra.

July of 2023

FACULDADE DE
CIÊNCIAS E TECNOLOGIA
UNIVERSIDADE D
COIMBRA

DEPARTAMENTO DE ENGENHARIA INFORMÁTICA

Gabriel Mendes Fernandes

Abordagens de Qualidade/Diversidade para Otimização Estrutural de Pontes

Dissertação no âmbito do Mestrado em Engenharia Informática, especialização em Sistemas Inteligentes, orientada pelos Professor Doutor João Correia e Professor Doutor Nuno Lourenço e apresentada ao Departamento de Engenharia Informática da Faculdade de Ciências e Tecnologia da Universidade de Coimbra.

Julho de 2023

Acknowledgements

I would like to express my deepest gratitude to Professor João Correia and Professor Nuno Lourenço. Without their expertise and guidance, this work would not have been possible.

I would like to extend my sincere thanks to my friends that were with me on this journey. In particular, thank you to Ana Mateus, Duarte Dias, Miguel Rabuge, and Pedro Rodrigues. Meeting and working with you these past few years was an invaluable and enjoyable experience for which I will always be grateful.

Lastly, I would like to acknowledge my family, especially my mother, my brother and my sister, for the unconditional support that they have always provided me.

Funding

This dissertation was partially funded by the project BEIS (Bridge Engineering Information System), supported by Operational Programme for Competitiveness and Internationalisation (COMPETE 2020), under the PORTUGAL 2020 Partnership Agreement, through the European Regional Development Fund (ERDF) and by the FCT - Foundation for Science and Technology, I.P./MCTES through national funds (PIDDAC), within the scope of CISUC R&D Unit - UIDB/00326/2020 or project code UIDP/00326/2020.

Abstract

Optimizing Cable-Stayed Bridges is a topic that has been gaining special attention in recent years, as multiple bridges of this type have been built. The optimization process is often done by hand or with gradient-based techniques, which require specific domain knowledge. Genetic Algorithms have been used to tackle this problem and reduce the burden on civil engineers by requiring less domain knowledge. Reducing the cost of these structures whilst keeping the structure safe is of utmost importance, as even a slight reduction can be decisive for a company to get a contract.

The main goal of this work is to use Evolutionary Algorithms to automatically create configurations for cable-stayed footbridges, defined by several configurable parameters. Specifically, our focus is to study the performance of Quality Diversity (QD) algorithms against more traditional approaches. QD algorithms are more capable of finding several fit solutions within a single execution of the algorithm. In practice, this diversity enables the user to select the desired bridge configuration. The problem will be addressed as a black box optimization problem.

In our experiments, we employed a diverse range of algorithms, including single-objective, multi-objective, and QD algorithms. By incorporating this heterogeneous set of algorithms, we aimed to comprehensively evaluate the capabilities of QD algorithms. The results are favorable toward the QD algorithms, showcasing their ability to discover a large number of diverse solutions while maintaining the optimization performance. For example, one of the QD algorithms used discovered cheaper structurally safe solutions than the baseline 19 times out of 20. Additionally, 50% of the created individuals are significantly different from each other. The results show that the search for diversity does not necessarily lead to a degradation of the optimization performance and that it is possible to find diverse and optimized solutions.

Keywords

Structural optimization of Cable-Stayed Bridges; Evolutionary Algorithms; Quality Diversity;

Resumo

A otimização de Pontes de Tirantes é um tema que tem ganho especial atenção nos últimos anos, já que várias pontes deste tipo foram construídas. O processo de otimização muitas vezes é feito manualmente ou com técnicas baseadas no gradiente, que exigem conhecimentos específicos do domínio. Algoritmos Genéticos têm sido usados para resolver este problema e reduzir a carga sobre os engenheiros civis, exigindo menos conhecimentos específicos do domínio. Reduzir o custo destas estruturas garantindo que são seguras é de extrema importância, pois mesmo uma pequena redução pode ser decisiva para uma empresa conseguir um contrato.

O principal objetivo deste trabalho é usar Algoritmos Evolucionários para criar automaticamente configurações para pontes de tirantes pedonais, definidas por diversos parâmetros configuráveis. Especificamente, o nosso foco é estudar a performance de algoritmos de Qualidade/Diversidade (QD) em relação a abordagens mais tradicionais. Algoritmos QD são mais competentes a encontrar várias soluções adequadas durante uma única execução do algoritmo. Na prática, esta diversidade permite que o utilizador selecione a configuração de ponte que prefere. O problema vai ser abordado como um problema de otimização black box.

Nas nossas experiências, nós utilizámos uma gama diversificada de algoritmos, constituído por algoritmos de optimização para um único objectivo, multi-objectivo e de QD. Ao usar este conjunto heterogéneo de algoritmos, o nosso objetivo é avaliar de forma mais abrangente as capacidades dos algoritmos de QD. Os resultados são favoráveis aos algoritmos de QD, demonstrando a sua competência para encontrar um grande número de soluções diferentes, mantendo o desempenho de otimização. Por exemplo, um dos algoritmos QD usados descobriu soluções estruturalmente seguras mais baratas que a solução base 19 vezes em 20. Adicionalmente, 50% dos indivíduos criados são significativamente diferentes entre si. Este resultado mostra que a procura de diversidade não leva necessariamente a um declínio de desempenho de otimização e que é possível encontrar soluções diversas otimizadas.

Palavras-Chave

Otimização estrutural de Pontes de Tirantes; Algoritmos Evolucionários; Qualidade/Diversidade;

Contents

Acronyms

BC Behavior Characterization.

BD Behavior Descriptor.

CMA-ME Covariance Matrix Adaptation MAP-Elites.

CMA-ES Covariance Matrix Adaptation Evolution Strategy.

CSB Cable-Stayed Bridge.

CVT Centroidal Voronoi Tesselations.

CVT-MAP-Elites Centroidal Voronoi Tesselations MAP-Elites.

DEAP Distributed Evolutionary Algorithms in Python.

DG-MAP-Elites Deep Grids MAP-Elites.

EA Evolutionary Algorithm.

ES Evolutionary Strategy.

GA Genetic Algorithm.

KNN K-Nearest Neighbors.

MAP-Elites Multi-dimensional Archive of Phenotypic Elites.

ME-MAP-Elites Multi-Emitter MAP-Elites.

NS Novelty Search.

NSGA-II Non-dominated Sorting Genetic Algorithm II.

NSLC Novelty Search with Local Competition.

QD Quality Diversity.

WSRP workforce scheduling and routing problem.

List of Figures

List of Tables

Chapter 1

Introduction

The automation of engineering processes has been attracting interest over the years with the aim of reducing the work to be performed by the engineer while improving results. This way the engineer can choose the one that better satisfies the needs and allows him to think about other parts of the problem.

For a few decades now, in civil engineering, there has been a focus on the utilization of optimization algorithms to aid in the search for solutions to the design variables of Cable-Stayed Bridges (CSBs) ([Martins et al., 2020]). These bridges are particularly interesting, due to their ability to overcome large spans and unique aesthetics. The optimization process used to be carried out by civil engineers that had to, through a tedious iterative process, calculate every design variable to guarantee structural safety as well as an acceptable construction cost. Therefore, computational-based approaches that do not require the intervention of a user (hands-free) while the optimization process is undergoing are valuable to reduce the burden on civil engineers. Another important factor to consider is that the level of optimization accomplished by machines is greater than the one accomplished by hand. In a world where having the best proposition, either in terms of cost or aesthetics, is the deciding factor for getting a contract, the use of optimization algorithms is a must.

Nevertheless, the optimization of these structures is particularly challenging, given that the fitness landscape is deceptive. As such a small modification to the value of a design variable can lead to massive deterioration in the quality of the bridge, making a feasible solution into an unfeasible one.

The main objective of this work is to assess the performance of several Quality Diversity (QD) algorithms in the structural optimization of CSBs. Specifically, we aim to retrieve a collection of diverse high-performing solutions from a single execution of the algorithm. This often is less time-consuming than multiple runs of non-QD algorithms to accomplish such diversity, allowing the user or a potential client to choose the one that pleases him the most from the several computed individuals.

We are approaching the problem as a black box optimization problem. This means that we are going to rely on a toolbox to perform the computations to get

the cost and the structural constraints of the bridges, that will guide the search. We are also not concerned with the analyses of the dynamic characteristics of the evolved structures, as we do not possess the required knowledge to do so. This work is intended to aid experts and not replace them. We intend to facilitate the optimization part, however, the real-world validity of the solutions found has to be studied by the experts.

To fulfill the proposed work, the following steps will be conducted.

- Understand the state-of-the-art of QD algorithms;
- Research the literature on structural optimization of cable-stayed bridges;
- Search for existing open-source python frameworks that implement QD algorithms;
- Implement the algorithms resorting to the frameworks found, adding required features if they are not implemented;
- Run the experiments with the selected algorithms;
- Analyse the data gathered from the experiments, comparing the selected algorithms;

In terms of contributions, we enumerate the following: (i) application of a new type of algorithm to the structural optimization of CSBs; (ii) comparison of a heterogeneous set of algorithms; (iii) demonstrate that there are QD algorithms capable of finding diverse individuals while maintaining optimization performance.

The remainder of this document is divided into 3 chapters. Chapter 2 will address the search carried out regarding CSBs and their optimization, alongside the explanation of a few optimization algorithms and multiple use cases of QD algorithms. Chapter 3 focuses on the experimental part of this work, starting with the problem definition, followed by implementation details, experimental setup, and the results obtained from our experiments. Lastly, chapter 4 concludes the work carried out, summarizing the document and presenting a few ideas regarding future work.

Chapter 2

Background

In the present chapter, the reader will find information to get familiarized with the terminologies used in the core concepts of this work. Section 2.1 briefly introduces the concept of Cable-Stayed Bridges (CSBs), addressing the main components and some design options that can be found in structures of this type. Then, Evolutionary Algorithms (EAs) are presented in Section 2.2 in the form of Genetic Algorithms (GAs) and Covariance Matrix Adaptation Evolution Strategy (CMA-ES). Finally, section 2.3 addresses Quality Diversity (QD) algorithms, starting by talking about Novelty Search (NS) and Novelty Search with Local Competition (NSLC), with the latter being regarded as the first QD algorithm, and then we present the more commonly utilized QD algorithm, Multi-dimensional Archive of Phenotypic Elites (MAP-Elites).

2.1 Cable-Stayed Bridges

Cable-Stayed Bridges (CSBs) have been widely utilized worldwide, with their first appearance in the year 1823 in Geneva. Since then, cable-stayed bridges have grown larger, being able to cover bigger distances, which, alongside their unique aesthetics, is a deciding factor when choosing the type of bridge to be utilized. CSBs are usually composed of three spans, two side spans with a larger one in the middle.

These bridges utilize cables supported by towers (also called pylons) and anchor blocks (or anchor piers) to hold the deck. It is possible to use different cable arrangements, from the number of cables utilized to how they are connected to the deck and positioned in the towers.

There are two main cable system designs for the connection of the cables to the towers, one called the fan system and the other harp system. In the fan system, all the cables radiate from the top of the towers, resembling a fan, thus the name (Fig. 2.1). In the harp system, the cables are parallel to each other, with connections spreading throughout the entire tower (Fig. 2.3). Intermediate designs can also be found, like the semi-fan system, where the cables are not derived from one single point at the top of the tower but are slightly spread (Fig. 2.2).

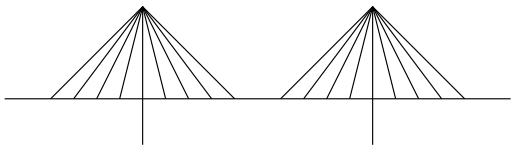


Figure 2.1: Simplistic example of a cable-stayed bridge with the fan cable system.

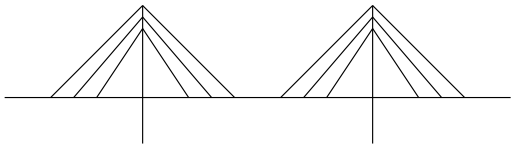


Figure 2.2: Simplistic example of a cable-stayed bridge with the semi-fan cable system.

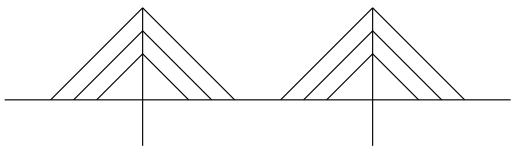


Figure 2.3: Simplistic example of a cable-stayed bridge with the harp cable system.

When longer distances have to be covered, CSBs can be designed in a multi-span configuration, which can be perceived as various smaller CSBs connected to each other.

For a more interested reader in the subject of cable-stayed bridges, the book from where the presented information was retrieved, Gimsing and Georgakis [2011], is advised.

2.2 Evolutionary Algorithms

Evolutionary Algorithms (EAs) are metaheuristic optimization algorithms based on Darwin's theory of evolution, branched into several categories such as Genetic Algorithms (GAs) and Evolutionary Strategies (ESs). These algorithms use the ideas of genetic recombination, gene mutation, and natural selection. The exploration of the search space is stochastic, meaning that the results depend on the random variables used, so it is not guaranteed that an optimum is reached. However, they are often capable of finding solutions near the optimum, while being less time-consuming than a non-stochastic method for problems of considerable dimension, when well parameterized.

2.2.1 Genetic Algorithm

According to Charles Darwin, a population of individuals evolves by the combination of mating and mutation, biased by the aptness of the individuals to the environment that surrounds them, which in practice means that the more apt individuals live longer making them more likely to procreate, passing their genes to the following generations.

The Genetic algorithm (GA) is a type of EA that aims to simulate this ideology, by evolving a population of individuals, in the search for the best individual to solve a problem, which can be seen as the population trying to adapt to the environment that they are in. The aptness (fitness) of an individual is calculated with a fitness function that needs to be specifically crafted for the problem at hand. The aptness of each individual is then taken into consideration in the selection process of the parents for the following generation. There are two common selection operators that the GA utilizes to simulate this process, tournament selection and roulette wheel selection. The first creates multiple tournaments with a predefined number (designated tournament size) of randomly sampled individuals from the population and keeps only the winner of each tournament, which is the one that has the highest fitness value. The second one uses the idea of a roulette wheel, in which the slice corresponding to each individual is proportional to his fitness value, which can be a problem for use cases that have negative values or zero as possible fitness values.

Usually, the population is randomly initialized, to promote diversity and more even coverage of the search space, however, when one knows the characteristics of the fitness landscape, there are alternative methods that may be better suited.

The representation of an individual is flexible, in the sense that it can be a binary array or a string of characters. The algorithm's designer is free to choose the representation that better suits its use case, keeping in mind that the variation operators need to support it. There are two types of variation operators, recombination and mutation, each with its specific functionality and probability of application.

Recombination operators (also designated crossover operators), as the name suggests, are responsible for emulating genetic recombination, where an offspring is a product of a mix of its parents' genes. It is considered a global search mechanism, as the resultant individuals are rarely close to either of their parents, being placed farther away from them in the search space.

On the other hand, mutation operators are local search mechanisms, charged with the task of slightly modifying an individual, enabling the exploration of the surrounding area of an individual in the search space. These operators emulate the sporadic mutations found in nature, where an individual of a species presents a characteristic not yet expressed in it. As in nature, a mutation can be beneficial or not, resulting in a gain or loss in fitness.

The algorithm's creator must consider the possibility of the variation operators generating unfeasible individuals. This can be solved in different ways, for example adding safeguards in the operators themselves so that the returned indi-

vidual is feasible, or penalizing those unfeasible individuals in terms of fitness, contributing to a lower chance of being selected to be parents in the following generation, guiding the evolution towards feasible solutions.

Algorithm 1 shows a pseudocode of a generic GA. One can see the initialization of the population, followed by the evolution loop, where the individuals' fitness is computed, the parents are selected, to whom the variation operators are applied, resulting in the offspring. Then, a new population is built, combining the parents and the offspring, often resorting to elitism. Elitism ensures that a specified number of the best individuals (elite size) found by the evolutive process are not lost by automatically allowing them to be featured in the new population, but may lead to premature convergence.

Algorithm 1 Generic GA pseudocode.

```
1: procedure GENETIC ALGORITHM(n_generations, crossover_prob, mutation_prob, pop_size)
2:     pop = initialize_population(pop_size)
3:     pop = fitness(pop)
4:     pop = sort_by_fitness(pop)
5:     gen = 1
6:     while gen < n_generations do
7:         parents = select_parents(pop)
8:         offspring = crossover(parents, crossover_prob)
9:         offspring = mutation(offspring, mutation_prob)
10:        offspring = fitness(offspring)
11:        offspring = sort_by_fitness(offspring)
12:        pop = create_new_population(pop, offspring)
13:        pop = sort_by_fitness(pop)
14:        gen += 1
15:    end while
16:    return pop[0]                    ▷ return the best solution found.
17: end procedure
```

2.2.2 Covariance Matrix Adaptation Evolution Strategy

Covariance Matrix Adaptation Evolution Strategy (CMA-ES) is a type of Evolutionary Strategy (ES), which uses a multivariate normal distribution using a covariance matrix, sampling the new individuals in the most promising direction. There are two different variants, (μ, λ) and $(\mu + \lambda)$, that differ from the way that the update of the population is handled. In the "," variant, the previous population is replaced by the new best μ individuals from the λ sampled, while in the "+" variant the best μ individuals from both the previous population and the new λ individuals are combined to build the new population.

The normal distribution is defined as $\mathcal{N}(\mathbf{m}, \sigma^2\mathbf{C})$, where $\mathbf{m} \in \mathbb{R}^n$ is a mean vector, σ is the distribution's step size and $\mathbf{C} \in \mathbb{R}^{n \times n}$ is a covariance matrix. These parameters are updated every generation. The sampling from this distribution can be regarded as the mutation operator in CMA-ES.

The algorithm can be decomposed into four repeating parts: (1) individual sampling from the normal distribution; (2) sorting the individuals in terms of fitness; (3) assembly of the new population and (4) updating the algorithm's parameters. To start the algorithm, a centroid $\mathbf{m}_0$, σ_0 and λ have to be defined. There are other parameters that can be also set, or calculated based on the dimension of the problem dimension (n) and λ, like μ.

After the sampling of the offspring and the creation of the new population, the parameters of the algorithm need to be updated, starting with the calculation of the new $\mathbf{m}$, which can be seen as the selection and recombination mechanisms. Only the best μ individuals from the population are used for the calculation of the average (selection). The recombination is accomplished by the use of weights to control the contribution of each selected individual, where one can decide to define the weights in such a way that more fit individuals contribute more than less fit ones, resulting in a weighted average. Then, the new σ and $\mathbf{C}$ are computed, considering their current values and population, either diverging to explore multiple regions with favorable fitness values or converging into a specific one.

Algorithm 2 presents a pseudo-code for the CMA-ES algorithm with the steps previously described.

Algorithm 2 CMA-ES pseudo-code.

```
1:  procedure CMA-ES(σ, m, λ)
2:      Initialize μ, C = I, pop = []                ▷ I is the identity matrix
3:      while not terminate do
4:          offspring = sample_population(m, C, σ)
5:          offspring = fitness(offspring)
6:          offspring = sort_by_fitness(offspring)   ▷ sort in descending order
7:          pop = create_pop(pop, offspring)   ▷ Depends on the CMA-ES variant
8:          previous_m = m
9:          m = update_m(pop)
10:         σ = update_sigma(σ)
11:         C = update_covariance(C)
12:     end while
13:     return pop[0]                              ▷ return the best solution found.
14: end procedure
```

An in-depth CMA-ES analysis can be found in Hansen and Ostermeier [2001] and a comparison with similar evolution strategies in Hansen [2006].

2.3 Quality Diversity

Quality Diversity (QD) (Pugh et al. [2015, 2016]) is a fairly new optimization approach, gaining attraction in the last few years. Its main goal is to find a set of high-performing, but also diverse, solutions in a single algorithm run. There are two main QD algorithms, Novelty Search with Local Competition and MAP-Elites, then branching to several variants, especially MAP-Elites, due to its sim-

ple algorithm, being an easy base to build upon. In general, an archive has to be maintained to assure the diversity of solutions, although other techniques can be used, like neural networks as in Salehi et al. [2021]. Over the course of optimization, less performing individuals will be stored in the archive, which may not be good enough as solutions to the problem, but serve as stepping stones for the finding of high-performing solutions. The fact that QD algorithms search not only for fitness but also for diversity in the behavior space helps avoid local optimum in the fitness landscape (Lehman and Stanley [2008]).

2.3.1 Novelty Search

Novelty Search (NS) (Doncieux et al. [2019]; Lehman and Stanley [2008]) was a revolutionary approach to optimization because instead of guiding the search by fitness, like traditional optimization algorithms until then, it uses a *novelty metric*, also designated *sparseness*. A population is maintained alongside an archive, used to store novel solutions while the population is evolved. Novelty here means that a solution is significantly different from the previously seen. NS utilizes a distance threshold and a neighborhood size, k, defined by the user to classify a solution as being novel or not. It uses the K-Nearest Neighbors (KNN) algorithm to then calculate the average distance of one individual to the k neighbors in the rest of the population and the archive, Eq. 2.1. If the *sparseness* value, $\rho(x)$, is small, it means that there are a lot of solutions with similar behavior of x, otherwise, it means that x is far from the others, thus, can be considered novelty, considering the threshold defined by the user.

$$\rho(x) = \frac{1}{k}\sum_{i=1}^{k} dist(B(x), B(y_i)) \tag{2.1}$$

where $B(..)$ is the mapping function from the genotypic space to the behavior space and $\{y_1, ..., y_k\}$ is the set of the k closest individuals to x in the behavior space.

Since the search is not being done at the fitness level, the algorithm avoids local optimums of the fitness landscape, because once an individual is found by the algorithm and added to the archive, similarly behaving individuals will have a low value of novelty assigned to them, meaning that the search will not pursue their direction and instead follows the direction of more novel ones. This means that one individual will have different values of novelty at different generations.

Although NS is better at avoiding local optimums at the fitness level, it still can lead to premature convergence at the archive level. For example, the convergence can appear in different ways: (1) imagine that the individuals from the archive are never removed. Since the archive has a limited size, due to performance (KNN is computationally expensive) and memory constraints, there will be a time when the archive is full and no other individual is ever added to it; (2) now, imagine that by using a mechanism, some individuals are eventually removed from the archive. Even in this scenario, which apparently would not suffer from convergence at the archive level, convergence can still exist. Since the novelty of an

individual is calculated considering its distance to its neighbors, we can have an archive constructed in such a way, that no other individual, but the one that was already there, can be added to it. This is a limit scenario, possibly mitigated by increasing the rate of removal of individuals from the archive.

However, the removal of individuals from the archive comes with its own set of issues, like which mechanism to use for the individual removal and cycling in the behavior space (Salehi et al. [2021]). For the removal of individuals from the archive, a mechanism simulating time can be used, using the generation number at which the individual was added to the archive, being removed from it if a predetermined number of generations have passed. A mechanism considering the density of individuals in the archive can also be used, removing individuals from zones with a high number of individuals (basically recomputing the novelty of the individuals in the archive, if a zone is too crowded, the novelty of those individuals will decrease), allowing others to be added, possibly from a completely different area of the search space.

The cycling in the behavior space is an emergent phenomenon from the removal of individuals from the archive. The archive is used so that an inventory of patterns (individuals) is kept, as a way of knowing what was already found, enabling the search for novelty. When a pattern is removed from the archive, if again found by the algorithm can be classified as novelty once more. This can lead to individuals being removed and added in a cycling manner, thus the name cycling in the behavior space, never reaching actual novelty again.

Even though diversity is desirable, more often one wants several quality solutions rather than simply diverse ones, and that is something that NS does not search for. For that Novelty Search with Local Competition (NSLC) (Lehman and Stanley [2011]) can be used. In addition to calculating the novelty of the individuals considering their neighborhood, it also computes a metric to assert the quality of an individual in relation to his neighbors, called local competition. Both these metrics are combined to guide the search in a multi-objective optimization algorithm like Non-dominated Sorting Genetic Algorithm II (NSGA-II) (Deb et al. [2002]), forming a Pareto-front of both the diversity and the local competition ranking.

The idea of local competition arises from what happens in nature, where the aptness of a species is not affected by others if they are not related to each other, for example as prey or predator, instead, there is competition in each of the species. The local competition emulates this intra-species competition by assigning to an individual the count of neighbors in his neighborhood that are beaten by him in terms of fitness. This metric can then be used as a way to quantify one individual's quality.

Combining the two objectives seems to be advantageous, with NSLC presenting diversity levels similar to NS and values of fitness close to the ones of fitness-based search (Lehman and Stanley [2011]).

2.3.2 MAP-Elites

Multi-dimensional Archive of Phenotypic Elites (MAP-Elites) was introduced in Mouret and Clune [2015] and is a QD algorithm that draws inspiration from NSLC. Its objective is also to find several high-quality diverse individuals, however, it accomplishes it without resorting to KNN to compute the novelty of an individual and also does not use the local competition mechanism. NSLC utilizes an archive to store the novel individuals and evolves a population, but MAP-Elites maintains only an archive. The search is not conducted in the genotypic space but in a feature space defined by the user, called Behavior Characterization (BC). A mapping function that given the genes of an individual, returns his feature values, designated Behavior Descriptor (BD), is required. To assure diversity MAP-Elites uses a grid in the feature space, dividing it into cells, each for a combination of a certain value of each feature, instead of calculating the distance between individuals, therefore the algorithm is less computationally expensive. The granularity of the grid can be defined by the user and can be changed throughout the algorithm run, enabling a finer search in the feature space by increasing the archive granularity.

Algorithm 3 shows the MAP-Elites pseudo-code. The algorithm starts by initializing the archive, X and P, for the solution and the performance respectively. Then, for a pre-determined number of iterations, generates individuals randomly, to populate the archive. After those initial iterations, until the termination criteria is reached, the sampling is done in the archive, selecting randomly one of the stored solutions, to which are applied variation operators (crossover and/or mutation). A behavior descriptor and the performance of the new solution, either randomly generated or a variation from one sampled from the archive are calculated and the new solution is then added to it if the respective cell is empty, or if the performance of the new one is higher than the current stored solution. For a more thorough review of MAP-Elites, the reader is advised to look at Mouret and Clune [2015].

MAP-Elites, due to the discretization of the archive, is not suitable for problems with high dimensional BC, because the number of bins (cells in the BC) grows exponentially. With this in mind, a variation of MAP-Elites called Centroidal Voronoi Tesselations MAP-Elites (CVT-MAP-Elites) was introduced in Vassiliades et al. [2018], where instead of discretizing every BC dimension, the BC is divided into k regions defined by a centroid inspired by the Centroidal Voronoi Tesselations (CVT). A solution is then assigned to the closest centroid in the BC. This way, the number of bins in the archive is independent of the dimension of the BC. The results presented by the authors show that the CVT-MAP-Elites in low dimensions achieves similar performance to the regular MAP-Elites, but the performance does not degrade in high dimensions.

In Fontaine et al. [2019b] the algorithm Covariance Matrix Adaptation MAP-Elites (CMA-ME) was introduced, aiming to get the optimization power of CMA-ES and the diversity of MAP-Elites. The results demonstrate that CMA-ME is an improvement over MAP-Elites, obtaining higher coverage of the behavioral space whilst reaching similar fitness values of CMA-ES, which is purely an optimizer.

Algorithm 3 MAP-Elites pseudo-code.

```
1:  procedure MAP-ELITES(number_iterations, iterations_threshold)
2:      X, P = create_archive()   ▷ store solutions in X and their performance in P
3:      iter = 0
4:      while end_criteria(iter, number_iterations) do   ▷ until the end criteria is
    reached
5:          if iter < iterations_threshold then
6:              x′ = random_solution()
7:          else
8:              x = random_selection(X)                  ▷ select from the archive
9:              x′ = apply_variation(x′)
10:         end if
11:         b′ = feature_descriptor(x′)                  ▷ compute x′ feature descriptor
12:         p′ = performance(x′)                         ▷ compute x′ performance (fitness)
13:         if P(b′) is empty or P(b′) < p′ then         ▷ Add or update b′ entry
14:             P(b′) = p′
15:             X(b′) = x′
16:         end if
17:         iter += 1
18:     end while
19:     return X, P                                      ▷ return the archive
20: end procedure
```

Three types of emitters were tested, random direction, optimizing and improvement, differing in the way that the parameters of the CMA-ES are updated, directly impacting the generated solutions. Random direction emitters induce the algorithm to search for individuals in a defined direction until it stops generating solutions that improve the archive. If that happens, the emitter is restarted with a different elite from the archive and a new direction bias. The optimizing emitter is very similar to CMA-ES with restarts, but instead of restarting from the best individual discovered, it uses one of the elites from the archive. The improvement emitter ranks the solutions by their improvement, instead of fitness. For solutions that fill an empty cell a rank equal to their fitness is assigned, while for solutions that are replacing an existing elite the difference between fitnesses is assigned, thus, moving the search towards the discovery of solutions that fill new cells or better-fit ones.

Later, in Cully [2020], Multi-Emitter MAP-Elites (ME-MAP-Elites) was introduced, stating that it is a direct extension of CMA-ME, improving its quality, diversity and data efficiency. It also utilizes emitters, but, instead of a homogeneous set of emitters, it utilizes a heterogeneous set of emitters. The authors used the three types of emitters of CMA-ME and introduced a fourth, random emitter, inspired by the selection operator of MAP-Elites. A batch of randomly selected individuals from the archive is modified by a variation operator. The emitters are selected from a pool of emitters with a bandit algorithm, considering the number of times the emitter was selected and the number of solutions generated by it that were added to the archive of elites. All the emitter types have the same number of instances and the active pool of emitters (the emitters that were selected) has

the same size. Thus, ME-MAP-Elites enables the active pool to be comprised of emitters of the same type, at which point it becomes CMA-ME. The fact that the emitter selection is adaptive, enables the algorithm to choose the ones that are more suited for the current search state. The results showed that ME-MAP-Elites either outperformed or was in pair with the various QD algorithms tested.

2.4 Related Work

Early works that studied the optimization of Cable-Stayed Bridges (CSBs) considered a fixed geometry, meaning that the number of cables and tower sizes were pre-defined. In Baldomir et al. [2010], only the optimization of the cross-section of the cables was considered. In Qin [1992], the cable stretching during construction was studied. The post-tensioning cable forces were addressed in Sung et al. [2006]. Later in Hassan [2010, 2013], similar problems were studied but a Genetic Algorithm (GA) was added into the optimization process used.

The optimization of CSBs to withstand the vibrations caused by earthquakes is of the most importance, particularly in countries that are greatly affected by this natural phenomenon, given that these are expensive structures that take a considerable time to be constructed and may greatly affect the lives of the surrounding population. Works like Ferreira and Simoes [2011]; Simões and Negrão [1999] have taken into consideration this phenomenon during the optimization process of cable-stayed bridges. In a similar fashion, strong winds can also lead to catastrophic failures in structures of this type, in Baldomir et al. [2013]; Jurado et al. [2008]; Nieto et al. [2011] wind aerodynamics are studied in the case of cable-stayed bridges. The usage of control devices to limit the vertical and horizontal vibrations of Cable-Stayed Brigdes has been addressed in Ferreira and Simões [2012, 2019a,b].

In Hassan et al. [2015], a database of optimal values for the number of cables, pylon (tower) and girder (deck) dimensions for CSBs with two or four lanes across a wide range of bridge lengths is presented. To reach the shown results, a combination of a GA to evolve bridge configurations and finite element models of each of the evolved bridges are produced to assess their fitness, aiming to discover the least expensive bridge. More recently, in Correia and Ferreira [2020]; Correia et al. [2020], GAs have been used to find the optimum configuration for controlled cable-stayed footbridges.

Since the optimization of CSBs has become a popular research theme, review works that summarize the contributions and advancements in the field have been created, for example, Martins et al. [2020]; MARTINS et al. [2022].

When Multi-dimensional Archive of Phenotypic Elites (MAP-Elites) was first introduced in Mouret and Clune [2015], it was utilized to evolve neural networks, outperforming fitness-based algorithms and Novelty Search with Local Competition (NSLC), in terms of raw performance, as well as in coverage of the behavior space. Tests with simulated soft locomotion robot morphologies are also shown, with MAP-Elites presenting by far the best coverage of the space and a similar

performance to the compared algorithms. The algorithm is also tested in the evolution of real soft arm controllers, and once again shows the best coverage of the space and the best performance in terms of fitness. The most interesting factor is that MAP-Elites is able to return a set of high-performing solutions from a single algorithm run, allowing us to understand how the fitness is spread in the feature space, beating the best solutions found by pure fitness-based approaches. In Pugh et al. [2016] MAP-Elites was used to solve robot maze navigation problems inspired by the ones presented in Lehman and Stanley [2008], showing promising results.

Since its introduction MAP-Elites has been adopted by researchers from the field of robotics. In Fontaine and Nikolaidis [2020] it was used to generate a vast number of testing scenarios for a robot being operated by a human and a shared autonomy algorithm, in which a high-performing scenario is one that leads the robot to fail in reaching the goal. This is important because knowing the scenarios where a robot fails allows us to study and evolve the robot to reduce its failure rate. Results show that the Quality Diversity (QD) approach is superior at generating scenarios compared to Monte Carlo and CMA-ES. MAP-Elites was also applied to evolve controllers for robots to recover from damage online in Allard et al. [2022]. There are also works concerned with evolving controllers for large-legged robots, which are particularly interesting due to their capacity of traversing rough terrain, for example in Howard et al. [2020]. Differing from the previous works, in Nordmoen et al. [2020], the authors use MAP-Elites to evolve both the control and the morphology of modular robots to adapt to changes in the environment. The evolved populations of the different approaches utilized were tested in two environments to understand if the diversity brought by QD algorithms is impactful in the adaptation of the robots to the new adversities. The results demonstrated that the QD approach was superior, finding individuals with higher fitness and assembling a more diverse collection of solutions.

In the field of games, MAP-Elites has been adopted in Pérez-Liébana et al. [2021] to evolve diverse competitive play styles in a strategy game framework called Tribes, by evolving arrays of weights that influence the scripts (a series of actions) that will be used by a player controlled by a modification of Monte Carlo Tree Search (MCTS). The results of the simulations are then used to compute the features that define the behavior characterization space of MAP-Elites and assert the fitness of the constructed array of weights. The results are promising, demonstrating that MAP-Elites was able to produce diverse play styles. In Warriar et al. [2019] it was used to design levels for 2D platform games. The authors of Fontaine et al. [2019a] present a new MAP-Elites variation designated MAP-Elites with Sliding Boundaries and use it to create card decks in Hearthstone. In this variation, the cells of the grid are not uniformly spaced in the behavior space, instead, they are spaced based on the underlying distribution of the behavior space, uniformly placed at some percentage marks of the distribution. In Fontaine et al. [2019b], instead of generating the card decks, the goal is to search for diverse strategies to play Hearthstone with a deck built by a human, and a variation called Covariance Matrix Adaptation MAP-Elites (CMA-ME) is presented and compared with the regular MAP-Elites and Covariance Matrix Adaptation Evolution Strategy (CMA-ES), displaying good results.

MAP-Elites was used to tackle the problem of workforce scheduling and routing problems (WSRPs) in Urquhart and Hart [2018], yielding promising results compared to a traditional EA, while, by design, enabling the user to choose the solution that pleases him the most, giving that one route might be more interesting than other due to traffic conditions or simply preference, even with lower fitness values.

Some works on constrained optimization problems, Fioravanzo and Iacca [2019], and noisy domains, Flageat and Cully [2020]; Justesen et al. [2019], have used MAP-Elites. For the latter, variations of the algorithm to cope with the particularities of the problem domain have been presented, like Deep Grids MAP-Elites (DG-MAP-Elites) and MAP-Elites with adaptive sampling and drifting elites.

When Novelty Search (NS) was introduced in Lehman and Stanley [2008], it was utilized to solve robot maze navigation problems, demonstrating promising results when compared to fitness-based approaches, due to the deceptive nature of the search space. Since then, NS has been used in several areas, such as the creation of game levels in Liapis et al. [2013a,b], automatic bug repair in Villanueva et al. [2020] and GANs co-evolution in Vinhas et al. [2016]. However, there are problems where searching just for novelty is not enough to achieve acceptable results (functional solutions), the fitness of the individuals also needs to be considered. NSLC, introduced in Lehman and Stanley [2011], considered to be the first QD algorithm, addresses this concern, considering both novelty and local fitness. In this work, the authors show the results of applying this algorithm to evolve locomoting virtual creatures and it was able to find creatures functionally diverse and not just diverse.

QD techniques have also been utilized in economics in Zhang et al. [2020], image generation in Costa et al. [2020] and configuration of urbanization layouts in Galanos et al. [2021].

2.5 Summary

Based on the literature, we selected the following algorithms: GA, CMA-ES, MAP-Elites, Multi-Emitter MAP-Elites (ME-MAP-Elites), CMA-ME, Non-dominated Sorting Genetic Algorithm II (NSGA-II) and a Hybrid algorithm from Vinhas et al. [2016].

The GA was selected because it is the algorithm chosen in various works, for example, Correia and Ferreira [2020]; Correia et al. [2020]; Hassan [2010]; Hassan et al. [2015]; Hassan [2013]. CMA-ES was chosen because it demonstrated good results in a previous experiment on this problem. The results were documented in an article for Evostar 2023 (Fernandes et al. [2023]). We chose MAP-Elites due to its versatility, demonstrating good results in various domains, for example, in Mouret and Clune [2015]; Nordmoen et al. [2020]; Pérez-Liébana et al. [2021]. ME-MAP-Elites and CMA-ME were chosen, because they are two new approaches built on top of MAP-Elites and CMA-ES, aiming to get the best of both worlds. The Hybrid algorithm from Vinhas et al. [2016] is a hybrid approach (GA with

an archive) that combines both diversity and fitness in its search. It was selected because it is a GA with mechanisms to search for diversity. Since we want to optimize the cost and the structural constraints of the bridges, we also selected NSGA-II.

Chapter 3

Experimental Work

This chapter is divided into four pieces. Section 3.1 explains the problem definition and the characteristics of the problem. Then, in section 3.2, certain implementation alternatives are covered, including the Python libraries used and some specific components that have an impact on all of the algorithms tested. The configurations of the algorithms, including the parameters and operators chosen are shown in section 3.3. Finally, the results gathered from the experiments with the selected algorithms are discussed in section 3.4.

3.1 Problem Definition

We are evolving cable-stayed footbridge configurations in an effort to develop bridges that are both economical and structurally secure. We aim to keep the cost of the structure as low as possible while keeping safety standards.

The problem studied in this work is the same as presented in Ferreira and Simões [2019], but as a black-box optimization problem, only interested in the process of evolving configurations of factors that then are used by a Matlab toolbox to create the structures. In concrete, we are evolving arrays of factors that then are used to compute the design variables presented in the work cited above. The tool expects an array of 21 float variables, the number of cables, and the bridge's total length, and returns the resulting bridge cost along with the structural constraints values. We use the returned values to compute the fitness of the array of factors, guiding the search toward better-performing bridges. The inner functions of the toolbox in use and the way the cost and the structural constraints are computed are out of the scope of this work. This toolbox allows for the creation of bridges with two towers with a variety of configurable parameters, such as the number of cables, positioning of the cables in the towers and the deck, and distance between towers, for example.

Considering the requirements of the Matlab toolbox, the individuals of the evolutionary approaches used in this study are defined as an array of 22 variables: one integer corresponding to the number of cables, and twenty-one floating point numbers for the other design variables. Their domains and the description can

be seen in Table 3.1. Between parentheses, we show the name of the variables from Ferreira and Simões [2019] that each of our design variables contributes to. Additionally, a few fixed parameters are defined in Table 3.2, like the length of the bridges.

For NSGA-II, a problem minimizing both the cost and the structural constraints value was defined. These values are used to rank the individuals and create the Pareto Fronts used in the creation of the population. However, even though it is not used for the search, we still computed the fitness of the individuals to compare their performance with the individuals from the other algorithms.

The fitness function utilized is shown in Eq. 3.1. $C(x)$ and $S(x)$ are the cost and the structural constraint value of individual x, respectively. $S(x)$ calculates all the security and structural constraints and returns the maximum value of all structural constraint variables considered in Ferreira and Simões [2019]. The cost returned is in the tens of thousands of euros.

The function is divided into three branches. The first branch produces values from 0 to 1 and the reasoning behind it is to guide the search towards individuals that have a more acceptable cost, specified by the cost reference constant, c_r. The second branch produces values from 1 to 2 and aims to guide the search from individuals with just an acceptable cost into individuals that also respect the structural constraints, which guarantee that the bridge is structurally safe. In practice, we force the cost of the individuals to be lower than the cost reference constant while rewarding the ones that have structural constraints values closer to 1. Lastly, feasible structures are the ones that are qualified to enter the third branch, given that they respect the structural constraints and have a price lower than c_r. The aim of this case is to reward the configurations that present lower cost or lower structural constraints values. The fitness values obtained from the third branch are all bigger than 3. This is a maximization problem due to how the fitness function is constructed, however, we want to minimize the cost and the value of the structural constraints.

A visual representation of the fitness function can be seen in Fig. 3.1. The function is not continuous and one can easily see the three branches of the function in action. The first branch is being used for values of $C(x)$ greater than $c_r = 150$ independently of the $S(x)$ (black region). Once we have a cost lower than the cost reference constant, we can see the second branch being used, considering the value of $S(x)$ (purple region). Finally, the third branch is utilized for cost values under c_r and structural constraints values of at most 1 (orange/yellow region). In this branch, fitness increases with decreases from $C(x)$ and $S(x)$.

In our previous work, Fernandes et al. [2023] (Appendix B, section B.1), the values of the structural constraints were forced to be close to 1. This means that we are being more lenient in the allowed amount of play that the structure can have while maintaining its safety, often leading to cheaper structures. However, in this work, we intend to assess the feasibility of the usage of Quality Diversity algorithms to gather a diversified set of fit configurations. With this in mind, we found it reasonable to drop the idea of forcing the algorithms to search in that area and instead allowed the minimization of $S(x)$, ultimately enabling a more

Table 3.1: Cable-stayed bridges design variables description and domain values.

Variable Type	**Description**	**Domain Values**
Discrete		
dv0 (N_{cables})	Number of cables	3,4,5,6,7
Geometry		
dv1 ($L_{Central}$)	Central span (tower to tower distance) of the structure	$[0.9, 1.2]$
dv2 (x_1)	Distance between the first and second cables anchorage in the lateral span of the deck	$[0.7, 1.3]$
dv3 (x_2)	Distance between the tower and the first cable in the central span	$[0.7, 1.3]$
dv4 (x_3)	Distance between the last cable anchorage and the bridge symmetry axis	$[0.7, 1.3]$
dv5 (z_1)	Height of the towers	$[0.1, 2.0]$
dv6 (z_2)	Distance where the cables are distributed in the top of the towers	$[0.1, 4.0]$
dv7 (y_2)	Distance between the top of each tower	$[0.1, 1.3]$
dv8 (y_1)	Distance between each tower at the base	$[0.1, 1.13]$
Control		
dv9 (k_y)	Transversal stiffness of the tower-deck connection	$[0.001, 1000]$
dv10 (k_z)	Vertical stiffness of the tower-deck connection	$[0.001, 1000]$
dv11 (c_y)	Transversal damping of the tower-deck connection	$[0.001, 1000]$
dv12 (k_z)	Vertical damping of the tower-deck connection	$[0.001, 1000]$
Sectional and tensioning		
dv13 (h_{slab})	Added mass of the concrete slab	$[0.1, 7.0]$
dv14 (H_{deck})	Deck section	$[0.1, 80.0]$
dv15 (B_{deck})	Deck section (triangular section)	$[0.5, 1.3]$
dv16 ($B_{2,tower}$)	Tower section (rectangular hollow section)	$[0.4, 1.5]$
dv17 ($B_{2,tower}$)	Tower section (rectangular hollow section)	$[0.1, 20.0]$
dv18 ($B_{1,tower}$)	Tower section (rectangular hollow section)	$[0.3, 20.0]$
dv19 ($H_{1,tower}$)	Tower section (rectangular hollow section)	$[0.3, 9.0]$
dv20 (*C.s* and *C.c prestress*)	Cables pre-stress	$[0.7, 3.0]$
dv21 (*C.s* and *C.c area*)	Cables cross section	$[0.5, 9.0]$

Table 3.2: Values for the fixed parameters.

Bridge Length (L_{Total})	220 meters
Bridge Width	4 meters
Tower Height below deck	10 meters

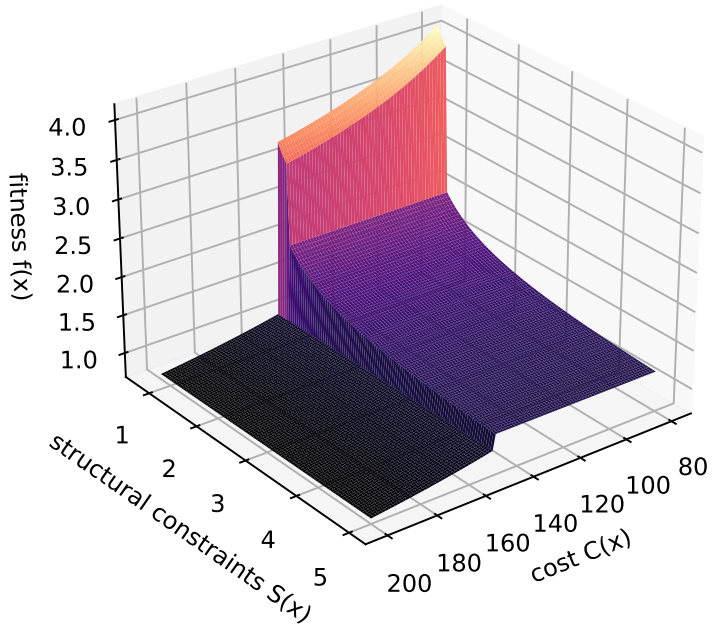


Figure 3.1: 3D representation of the fitness function.

free search. This new approach is also better in our understanding, due to the fact that it is not known for sure that the least expensive structures are in fact the ones that have structural constraints closer to 1.

$$f(x) = \begin{cases} c_r/C(x), & \text{if } C(x) > c_r \\ 1 + 1/S(x), & \text{if } C(x) < c_r \wedge S(x) > 1.0 \\ 1 + 1/S(x) + c_r/C(x), & \text{if } C(x) < c_r \wedge S(x) \leq 1.0 \end{cases} \tag{3.1}$$

To aid us in understanding our results, we are using a baseline solution obtained by the approach from Ferreira and Simões [2019], which has a cost of 91.354×10^4 € and a maximum structural constraints value of 0.9962. This baseline is crucial because it gives us an understanding of how effective a solution discovered through any of the used methodologies actually is.

The QD algorithms require a feature space (behavior characterization space) to characterize each individual according to certain behaviors that one wants to study in their solutions. The diversity is handled at the feature space level, meaning that what is important is not gene diversity but behavior diversity.

To create a feature space we need to take into account the number of features, and the discretization factor. Both choices influence memory usage and affect the analysis of the individuals stored in the archive. Imagine that we have a pool of 10 behaviors to select from and two scenarios: in A we choose all 10 behaviors, and in B choose only 4. Now we need to define the level of granularity that we want

in each feature, directly impacting the analysis that can be done by looking into the archive. If one of the behaviors is speed (m/s), we can define the granularity of speed in A as 2 (slow, fast) and in B as 4 (0-10, 10-20, 20-30, and 30-40 assuming that the bounds for speed are 0 and 40 m/s), for example. The archive of A will have $10^2 = 100$ cells and the archive of B $4^4 = 16$ (assuming that all the features have the same discretization factor). It is easily understood that the one with more cells can potentially occupy more memory. Another important point is that in scenario A we can only say "I found a fast/slow individual.", while in B we can say "I found an individual that travels at less than 10 m/s" etc. The information obtained is completely different because in A we are condensing into the same category individuals that move from 0 to 20 m/s (classified as slow), which can be extremely different. By using scenario B, only in terms of speed (because they do not have the same number of features), we can still classify individuals as slow or fast, but from scenario A we cannot say that the individual in the slow cell has a speed of 0-10 m/s (we lost information). All of this is to say that often a higher discretization factor is more informative than a small one, however, it can lead to higher usage of memory, depending on the number of features in use.

We were not able to find features to characterize the bridge configurations, so we opted to use some of the genes (design variables) as features to create the feature space. Considering all 22 genes as features was unfeasible because we would need to have a low discretization factor to keep the archive size reasonable. Since we wanted to use a considerable discretization factor, we had to select some of the genes to act as features. We conducted a study of the distribution of the individuals for each gene and selected the ones that presented values more evenly spread over the entire interval. This study was conducted with the last 100 000 individuals of each of the 60 runs from the preliminary experiments with CMA-ES and GA. From these 6 000 000 ($30 \times 100000 \times 2$) individuals, only the unique ones were considered and then filtered, based on their structural constraints value and cost (1 and 150 respectively), because we wanted to consider only feasible structures, hence the structural constraints value of at most 1 and a cost of less than 150×10^4 €. The histograms can be seen in Fig. 3.2.

Looking at Fig. 3.2, one quickly observes that various dvs tend to be located in one region of the interval. We did not consider dv0 to be a feature because it is represented by integers and would force us to choose a maximum discretization factor equal to the maximum number of possible cables for the resultant feature. Dvs 2 to 12 and 14 to 19 are concentrated in one part of the interval. This left us with dvs 1, 13, 20, and 21, which were the ones that we chose to act as features, alongside a discretization factor of 50 for all features, resulting in an archive with $50^4 = 6250000$ cells.

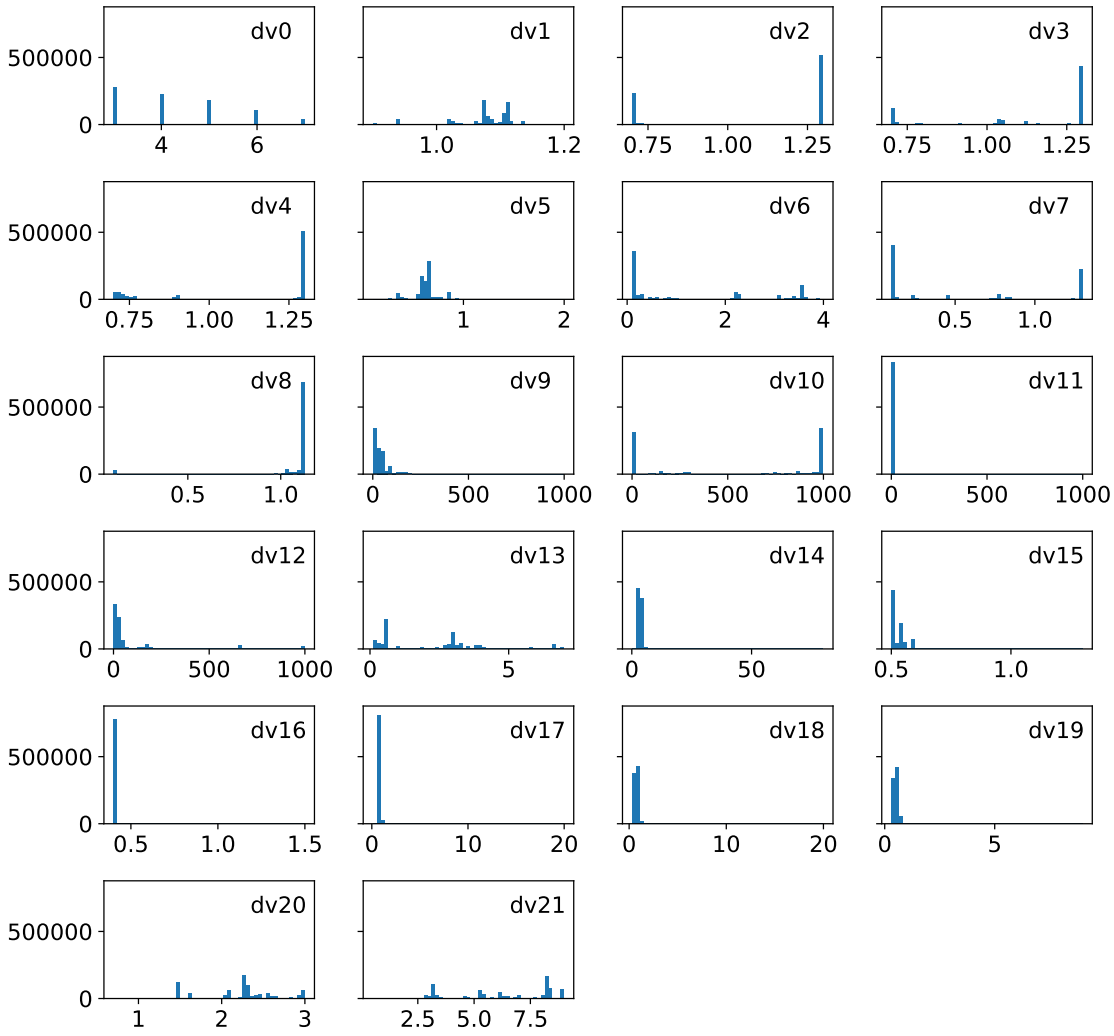


Figure 3.2: Histograms of the design variables (dv), using the unique entries of the last 100 000 individuals from every run of the preliminary experiments. 30 runs from CMA-ES and 30 runs from the GA.

3.2 Implementation Details

The main goal of this work is to evaluate QD algorithms, as such we decided to use well established Python libraries that have already implemented the algorithms that we want. We started by using QDpy (Cazenille [2018]), given that it appeared to have all the QD algorithms that we needed. Soon after some problems arose, the examples of how to use the library were few, and some of them did not work out of the box. Some modifications were made to the library code, fixing the issues that we were having, and a couple of algorithms were implemented with it. However, it seems that the library has not been updated in a long time, which led us to change the focus towards using Pyribs (Tjanaka et al. [2021]) instead.

Pyribs was used to implement MAP-Elites, ME-MAP-Elites, CMA-ME and CMA-ES. NSGA-II was implemented with Pymoo (Blank and Deb [2020]). The GA and the Hybrid were implemented with Distributed Evolutionary Algorithms in Python (DEAP) (Fortin et al. [2012]).

The individuals from Pyribs and Pymoo are defined by arrays while the ones from DEAP are objects. To enable the use of the same functions across all algorithms, we transform all individuals into DEAP individuals prior to computing fitness and logging.

Since some algorithms utilize KNN to calculate the novelty of the individuals, we decided to map all the genes from the intervals shown in Table 3.1 into intervals from 0 to 1. This mapping is utilized in every algorithm, to neutralize any advantage that this operation might give. The mapping is accomplished by using the mechanism presented in Alg. 4, with the first gene requiring special attention, because it is an integer in the original mapping, so we apply the Python round function to the mapped value.

Algorithm 4 Mapping mechanism from 0,1 to the original intervals.

```
lb                                  ▷ Array with the lower bounds for every gene.
ub                                  ▷ Array with the upper bounds for every gene.
new_x = x
for i do in range(len(x)):
    new_x[i] = (ub[i] - lb[i]) * x[i] + lb[i]
end for
new_x[0] = round(new_x[0])
```

All the individuals used to start the algorithms are randomly sampled from a uniform distribution between 0 and 1. The algorithms that utilize CMA-ES simply need a centroid, a single individual. MAP-Elites requires a set of n solutions to populate the grid archive and enable the random sampling of elites from it, and for the rest, a set of solutions with the same size as the population is created.

Due to the way that the algorithms are designed, there is a chance of getting unfeasible bridge configurations, individuals whose genes are outside of the required intervals, so, to solve it, we check all the individuals to guarantee that their

genes are within the correct bounds. If it is not the case, new genes are randomly sampled from a uniform distribution between 0 and 1 to replace the ones outside of the required bounds. Other correction mechanisms could have been used, like clipping the unfeasible values to the bounds of the allowed intervals.

3.3 Experimental Setup

Table 3.3 contains all the parameters used by all the algorithms in our experiments. The cells occupied by "-" mean that the respective algorithm does not use that parameter. The GA was executed with a population size of 10 individuals. The crossover operator used was uniform crossover, with a probability of being used of 1 (cxpb), and every gene has a probability of 0.5 (cxp_aux) of being swapped. We utilized a per gene replacement mutation operator where each gene is replaced by a random value from a uniform distribution between 0 and 1 with a probability of 0.1. The parents are chosen with tournament selection with size 3 (tourn size). The best individual from the current generation is passed untouched into the next one (elite size).

CMA-ES was parameterized with a λ of 50 and μ of 25, with an initial step size (σ) of 0.1 and an elite of size 1, meaning that the best individual from the run until then is placed within the 25 that will be used to update the parameters of the strategy. The init size of 1 is the initial centroid used to start the algorithm.

The archive of MAP-Elites is initialized with 1000 individuals (init size) and then the new individuals are created by random sampling an elite from the archive and altered by a normal distribution centered in 0 and scaled by 0.1 (σ).

The parameters of CMA-ME were the same ones of CMA-ES. Because CMA-ME utilizes the concept of emitters, which can be several instances of CMA-ES but we use just one objective emitter. Like this, CMA-ME is almost just an instance of CMA-ES, but with a grid archive, used to store the solutions that it finds and when no individuals are added to the archive in a generation, the emitter is restarted with a randomly sampled elite from the archive. This experiment was done, with the intent of seeing the impact of the archive and restart functionality added to CMA-ES.

ME-MAP-Elites is very similar to CMA-ME, with the difference being that not all the emitters have to be used in every iteration. The Upper Confidence Bound - 1 algorithm (UCB1) is utilized to sort the emitters by their potential and the ones ranked highest are selected. It all depends on the way it is parameterized because it can also behave just like CMA-ME, if the pool size is equal to the number of active emitters. In our experiment we only allow one emitter to be activated per iteration, with the emitter pool being composed by three emitters, one objective emitter, one random direction emitter and an improvement emitter, all using the same parameters as CMA-ES.

The Hybrid algorithm, consisting of a GA with a distance based archive, uses a

hybrid system, where depending on the current state of the algorithm (in terms of diversity), different metrics are used to rank the parents in the tournament. If the percentage of individuals that are above f_min is higher than t_max the tournament considers both the fitness and the novelty of the individual, using the concept of Pareto fronts to rank the individuals in the tournament, and then one of the first front is randomly selected to be the winner. On the other hand, if the percentage of individuals that have a fitness higher than f_min is less than t_min, a tournament considering only the fitness is used. Individuals that have a fitness above f_min and a novelty (dissimilarity) higher than dissim_min are added to the archive, with the novelty being computed as the average sum of the squared euclidean distances between the individual and the k nearest neighbors in the archive, and then this value is divided by the maximum possible distance, which is computed using the lower and upper bounds so that the resultant novelty value is between 0 and 1. The number of neighbors used might not be equal to k, in the case that the amount of stored solutions in the archive is less than k.

For the Hybrid, we decided to use the same parameters used in Martins et al. [2019] where they used a population of 100 individuals and an elite of 10. The crossover operator was a two-point crossover, with a probability of 0.8 (cxpb) and a probability of 0.5 (cxpb_aux) inside the operator to choose if something should be swapped. Gaussian mutation was used, with mean 0 (mut_mean) and scaled by 0.1 (mut_std), called with a probability of 0.5 (mutpb) and a probability of adding the Gaussian noise to each gene of 0.5 (mutpb_aux). The selection of parents was made by tournament selection with size 5.

The value of dissim_min was a result of a study using the best individuals of 30 runs from a previous experiment with CMA-ES. In this study, the novelty of each individual was calculated for k ranging from 1 to 29 and the minimum value for each k was saved. These values would be the minimum required for each k, so that all the individuals in the test would be considered novel and, as a consequence, added to the archive. We selected a value slightly lower than the value of k = 4, however, we observed that the number of individuals being added to the archive was low and settled on using the value for k = 3, maintaining k = 4.

The parameters used by NSGA-II are the default in Pymoo. A population o 100 was used, a probability of performing crossover (simulated binary crossover, sbx) of 0.9 and an eta of 15 were utilized, and for the mutation operator (polynomial mutation) a probability of 0.4 and an eta of 20 were used. The tournament type, "comp_by_rank_and_crowding" was chosen instead of the default. The two objectives, in this case, were the cost ($C(x)$) and the structural constraints value ($S(x)$), which are the same used to compute the fitness value by our fitness function.

Since the different algorithms use different population sizes, to provide a fair comparison, all were run for 400 000 evaluations (calls to the fitness function). During the execution of the algorithms, the created individuals are stored in csv files by iteration. These logs are used to create the plots and compute the statistics. The plots are presented in terms of evaluations, to more easily compare plots from one experience to another.

Table 3.3: Parameters used in the algorithms. ME stands for MAP-Elites and ME-ME for ME-MAP-Elites.

	GA	CMA-ES	ME	CMA-ME	ME-ME	Hybrid	NSGA-II
generations	40 000	8 000	400 000	8 000	8 000	4 000	4 000
pop size	10	-	1	-	-	100	100
init size	10	1	1000	1	1	100	100
cxpb	1	-	-	-	-	0.8	0.9
cxpb_aux	0.5	-	-	-	-	0.5	-
cx_npoints	2	-	-	-	-	2	-
cx_eta	-	-	-	-	-	-	15
mutpb	1	-	-	-	-	0.5	0.4
mutpb_aux	0.1	-	-	-	-	0.5	-
mut_mean	-	-	-	-	-	0	-
mut_std	-	-	-	-	-	0.1	-
mut_eta	-	-	-	-	-	-	20
f_min	-	-	-	-	-	2.0	-
dissim_min	-	-	-	-	-	0.0006	-
t_min	-	-	-	-	-	0.05	-
t_max	-	-	-	-	-	0.15	-
k	-	-	-	-	-	4	-
archive size	-	-	-	-	-	5000	-
tourn size	3	-	-	-	-	5	-
μ	-	25	-	25	25	-	-
λ	-	50	-	50	50	-	-
σ	-	0.1	0.1	0.1	0.1	-	-
elite size	1	1	-	-	-	10	-
c_r	150	150	150	150	150	150	150
num evals	400 000	400 000	400 000	400 000	400 000	400 000	400 000

3.4 Experimental Results

In the following pages, the results from each algorithm concerning optimization and diversity performance are presented separately. In the end, a comparison between all the algorithms is performed.

3.4.1 Genetic Algorithm Results

Fig. 3.3 shows the fitness of the best individual in blue and the average fitness of the population in orange. The lines in full are averages of 20 algorithm executions. The shading is the average $\pm$ the standard deviation, used to gain insight into the amount of variability between executions. In the figure, it can be seen that the fitness is going up, first increasing rapidly, but slowing down as the execution of the algorithm continues, but never stagnating. The population's average fitness also tends to increase throughout the algorithm's execution, even though the line is full of spikes, which indicates that the average fitness has a high variation. The average fitness values are far from the maximum, which suggests that the population has a diverse set of individuals.

Fig. 3.4 shows how the value of the structural constraints of the best individual evolves over time. The plot is limited in the y axis to allow for a better look near 1. This can give the impression that there are no values for some number of evaluations, but they do exist, they just cannot be seen in the plot. The line in full is the average of the 20 runs executed and the shaded area is the average $\pm$ standard deviation. In this figure, we can see that the structural constraints value is being optimized getting closer to 1. The plot can be divided into three separate parts: one where the structural values are very high; then the values begin to decrease approaching 1; and finally, we have values below 1. These parts can be mapped into the three branches of the fitness function. First only the cost matters so we are not optimizing the structural constraints value, allowing it to be large. Then, once the cost has reached reasonable values, the structural constraints optimization begins, lowering its value. Lastly, we reach the third

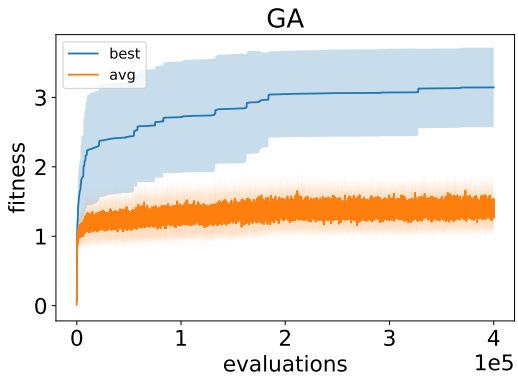


Figure 3.3: GA fitness per evaluation. The lines in full are averages of 20 seeds and the shaded is the average $\pm$ standard deviation.

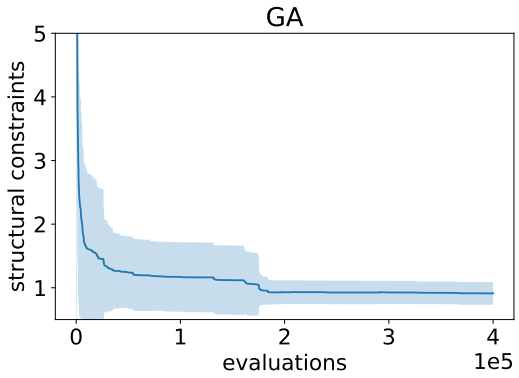


Figure 3.4: Structural constraints value achieved by the GA. The line in full is the average of 20 runs and the shaded is the average ± standard deviation of the best individual per evaluation. The maximum y value was limited.

branch, where we force the structural constraints to be at most 1.

The cost of the best individual over time is shown in Fig. 3.5. The plot is limited in the y axis to show how the algorithm behaves in the feasible price range (less than 150). This can give the impression that there are no values for some number of evaluations, but they do exist, they just cannot be seen in the plot. The line in full is the average of the 20 runs executed and the shaded area is the average ± standard deviation At the beginning of the optimization process we have really expensive structures, which is expected, given that the search starts from randomly sampled individuals. However, we quickly start to find structures with a more favorable cost. Looking at the plot, it is possible to see that the cost decreases at the beginning of the evolutionary process. However, after a few generations, it increases. This is again due to how the fitness function is defined, which starts by seeking bridges that are below a certain cost but might not satisfy the structural constraints. The cost increase takes place when the focus of the search changes, since the lowering of the structural constraints might require using different materials that impact the cost of the overall structure. However, looking at the line of the average cost, it can be seen that the lowering of the cost is the trend. When we combine the information in the cost plot and the structural constraints plot, we can observe that the structural constraints value is being maintained below 1 (after the 200 000 evaluations mark), while the cost is decreasing, resulting in bridges that are cheaper and still safe.

The boxplots in Fig. 3.6 show how the 20 best individuals are distributed in terms of fitness, cost, and structural constraints. We can see that some runs performed poorly when looking at the fitness boxplot (Fig. 3.6a), given that we have runs where the best individual could not even reach a fitness of 2, meaning that in those runs, the structural constraints value was not optimized to values below 1. The best individuals from the majority of the runs are concentrated in the fitness range of 3 to 3.5, which means that those individuals have a structural constraint value of less than 1 and a cost of less than 150k €. The boxplot for the structural constraints (Fig. 3.6b) corroborates this, showing that some runs were not able to reduce the structural constraints to values below 1. The boxplot for the cost

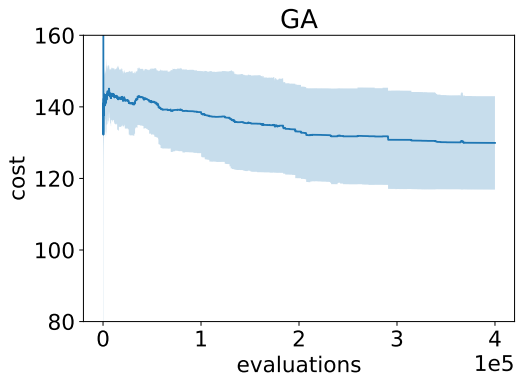
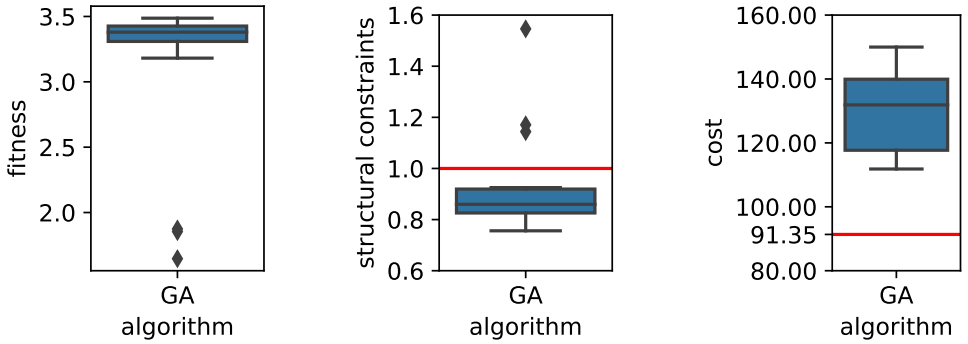


Figure 3.5: Cost achieved by the GA in the tens of thousands of euros. The line in full is the average of 20 runs and the shaded is the average ± standard deviation of the best individual per evaluation. The maximum y value was limited.



(a) Fitness. (b) Structural constraints. (c) Cost.

Figure 3.6: Boxplots created with the best individual of each of the 20 GA runs. The cost is presented in the tens of thousands. The red line in the cost boxplot is the cost of the baseline solution and in the structural constraints plot is the threshold of safety (1).

(Fig. 3.6c) is particularly helpful, informing us that all the runs were at least able to find individuals with a cost lower than $c_r = 150$. In this boxplot, we can also see that the baseline was not surpassed by any of the discovered solutions.

Taking into account these results, the GA can optimize solutions for this problem to a certain degree. The best individual found in the 20 runs has a cost of 114.147×10^4 € and a structural constraints value of 0.8527. The structural constraints value is significantly lower than the one of the baseline but the cost is larger. This individual has the highest value of fitness, but it is not the cheapest individual discovered. The cheapest safe individual has a cost of 108.352×10^4 € and a structural constraint of 0.9915. A graphical comparison of the geometry of these two bridges and the baseline is shown in Fig. 3.7. We can see that the evolved solutions have a harp cable system, while the baseline has a fan cable system.

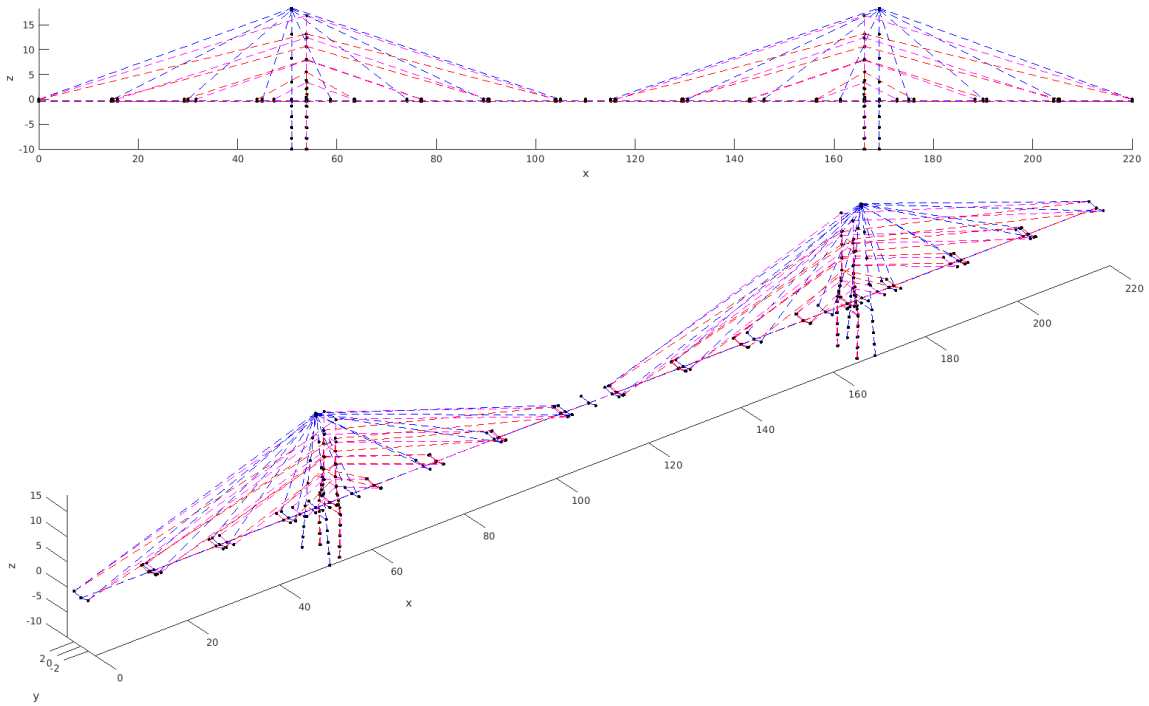
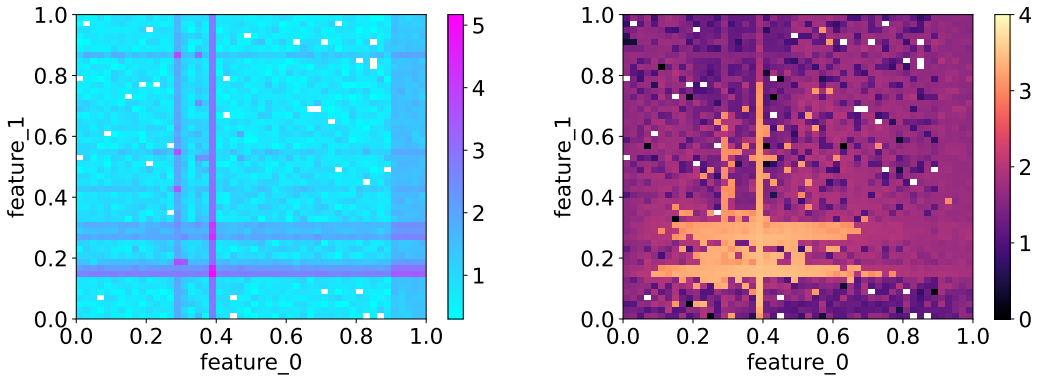


Figure 3.7: Graphical comparison of the baseline (blue), the best (pink), and the cheapest (red) from the GA.

The purpose of studying this problem with QD algorithms was trying to find a diverse set of fit solutions. However, we need a way to observe the diversity, which in algorithms that are based on MAP-Elites is accomplished by using a grid archive, then plotted as fitness heatmaps. With this in mind, we decided to build grid archives for all the approaches being tested, even if the archive does not take part in the optimization process. The archives are constructed with the help of the log files that were written by the algorithms during execution. Every run produces an archive, which means that we have 20 archives per algorithm, resulting in a large quantity of charts. Thus we only show the fitness heatmaps and activity plots for the best run of the algorithm. An example of these types of plots can be observed in Fig. 3.8, while the rest for this specific seed are presented in Appendix A, section A.1.

Since the archives have four dimensions (feature space with four features), we are not able to represent it in a figure all at once, so we make a fitness heatmap for every pair of features, resulting in 6 plots. These plots are projections of the grid archive into the pair of features being used, so one cannot infer the coverage of the grid archive as a whole because we can have an archive filled with points scattered all over and still get a projection that looks full. Even though these heatmaps do not give us an entire picture of the coverage of the archive, they still give us some insights. For instance, it is possible to see if there is an individual that has a feature that falls in a particular cell, or if all the fittest individuals have values for a certain feature, falling into the same projected cell. The features are labeled as feature_*i*, with *i* ranging from 0 to 3, which are mapped into dv1, dv13, dv20, and dv21, respectively. We also build activity plots, similar to the fitness heatmaps, but instead of fitness, we use the number of individuals found during



(a) Activity plot.

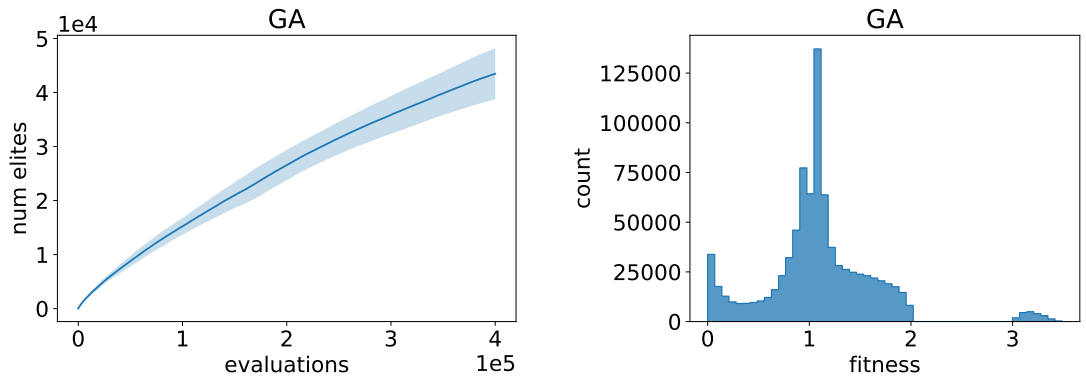
(b) Fitness heatmap.

Figure 3.8: On the right side, we present a projection of features 0 and 1 from the grid archive, showcasing the fitness results obtained by the best GA seed. On the left side, we have an activity plot that provides insights into the regions that were more explored. The activity plot employs a logarithmic scale with a base of 10.

the search that would be mapped to that cell. This gives us an idea of the areas of the space that were more visited. Since they are projections, we can have a case where the number of hits in one cell is 2 with those 2 individuals filling two different cells, due to having different values for the other two features. However, we can also have the case of those 2 individuals being mapped into the same exact archive cell. The scale of the activity plots is logarithmic of base 10, to allow us to see differences along regions that were less explored, which would otherwise be unnoticeable.

By examining Fig. 3.8b, one can see the fitness levels of individuals stored in the archive for features 0 and 1. The better individuals are mostly located in the first half of each feature interval, that is the bottom left corner. However, the space seems to be more or less evenly covered (the cells, in general, have a similar value of fitness, not too distant from the maximum value found). Additionally, a pattern can be observed among higher fitness individuals, which, for the most part, is the same pattern depicted in the activity plot of Fig. 3.8a. Although not always the case, regions with higher activity tend to exhibit better fitness values than less-explored areas.

To get a deeper understanding of the state of the archive, we present its evolution over time in terms of the number of occupied cells and how the stored individuals are distributed in terms of fitness at the end of the algorithm's execution. The plot in Fig. 3.9a shows us that the algorithm is adding new individuals to the archive over time, which means that it is continuously finding individuals that are different than the previous ones. We can see that, on average, after the 400 000 evaluations around 40 000 individuals were added to the archive (see Table 3.4). It is important to note that this number of elites accounts only for roughly 10% of the created individuals, meaning that 90% were mapped to already occupied cells. These results are not surprising since the GA is a pure optimization algorithm and our implementation has no mechanisms to search for diversity. By examining the histogram of Fig. 3.9b we observe that most of the individuals



(a) Number of elites.

(b) Fitness distribution.

Figure 3.9: Plots concerning only the individuals added to the grid archive. On the left, the number of elites in the archive is shown per evaluation. The line in full is the average of the 20 seeds and the shaded is the average $\pm$ standard deviation. On the right, we present a histogram of the distribution of fitness in the archives at the end of the executions of the algorithm. All the fitness values in the 20 archives were grouped together and then utilized to create the histogram with 50 bins.

added to the archive are not feasible (fitness values below 3). The majority of the stored individuals comply with the cost restriction (because they have fitness above 1), but do not respect the structural constraints restriction.

To complete the information displayed in Fig. 3.9, we present Table 3.4 with the data for each algorithm execution (seed). Column f stands for the number of unique feasible individuals (cost of less than 150 and structural constraints of at most 1) discovered throughout the run. Column f_arch is the number of unique feasible individuals in the archive at the end of the run. Similarly, b_b stands for the number of unique individuals that beat the baseline (cheaper but still safe) that were discovered throughout the run. Column b_b_arch is the number of unique individuals that beat the baseline in the archive at the end of the run. The number of elites is the number of occupied cells in the archive, the coverage is the number of elites divided by the total number of evaluations and the qd_score is the sum of the fitness of all the elites in the archive. Since we only have 400 000 evaluations, in the optimal case we would have a coverage value of $400000/50^4 = 0.064$ and this would mean that every created individual would land on a different cell. With this in mind, we think that a better metric is to use the percentage of individuals added as the coverage. The seeds are sorted by fitness in descending order, and at the end, we add the average of all rows.

When taking a closer look at Table 3.4 we can see that the GA was able to find, on average, more than 20 000 unique feasible individuals per run. However, only roughly 1 000 of those are significantly distinct, that is, present different behaviors to be added to the archive, the rest converge into occupied cells and lose in terms of fitness.

Table 3.4: Diversity statistics for each of the 20 GA seeds. Floating point values are rounded to 3 decimal points.

seed	num_elites	coverage	qd_score	f	f_arch	b_b	b_b_arch
2854977859	46566	0.116	57927.693	36196	2118	0	0
1040632194	40510	0.101	50107.453	34251	1486	0	0
1489817794	39818	0.100	46936.426	24334	1186	0	0
88083431	50510	0.126	57726.650	25021	1243	0	0
200205028	41372	0.103	47311.033	16604	563	0	0
3487363715	43239	0.108	52511.647	30400	1693	0	0
1592410486	48766	0.122	53775.147	21307	1131	0	0
781805057	47040	0.118	49364.718	11585	673	0	0
3262362994	40714	0.102	44868.671	36750	2459	0	0
3591128469	35058	0.088	43426.325	42409	1789	0	0
2849255314	48261	0.121	51901.839	7210	384	0	0
1406242007	34486	0.086	43036.254	40187	1449	0	0
3113515005	43635	0.109	42373.790	18634	730	0	0
2624447050	36573	0.091	36973.438	24777	1333	0	0
2631223479	43956	0.110	47889.517	23629	877	0	0
4197906329	49785	0.124	51764.948	3771	373	0	0
271943135	44125	0.110	47770.769	18299	481	0	0
3105602047	50125	0.125	47023.327	0	0	0	0
1766430675	43050	0.108	40931.500	0	0	0	0
2620038020	41999	0.105	39845.338	0	0	0	0
mean	43479.400	**0.109**	47673.324	**20768.200**	**998.400**	0	0

3.4.2 Covariance Matrix Adaptation Evolution Strategy Results

By looking at the fitness lines in Fig. 3.10 we can see that we reach high fitness values rapidly, at around 50 000 evaluations. From this point forward the evolution seems to stagnate. The line of best fitness never gets worst because the algorithm was implemented with elitism. The average fitness at first reaches values very close to the maximum and then drops off to values around 2, fluctuating throughout the algorithm's execution. This supports the idea that the best individuals are found at the beginning of the run and from there on we mostly find individuals that are significantly less fit than the best. Analyzing the data that we have, it appears that, from a pure optimization point of view, there is no reason to run the algorithm for more than 100 000 evaluations. Observing the fitness boxplot in Fig. 3.13a we can see that most of the runs reach values of fitness well above 3, but then two of the runs underperformed, pushing the average of the maximum fitness towards 3.

In Fig. 3.11 one can see that the value of the structural constraints of the best individual quickly reaches values close to 1. The line sits above 1 which, at first glance, suggests that the algorithm was not capable of minimizing this objective. However, if we look at the boxplot of Fig. 3.13b we can see that what is causing the mean to be above 1 are two runs with considerably large values of structural constraints. The other 18 runs were able to minimize this objective.

By looking at the line of the cost of the best individual in Fig. 3.12 we first see a drop in the cost of the bridges, followed by a slight increase. In this case, the increase is followed by a steep decrease until around the 70 000 evaluations, and

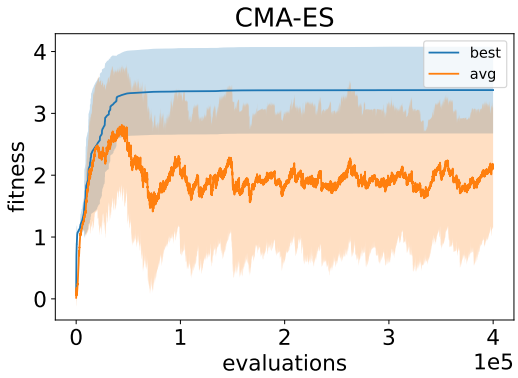


Figure 3.10: CMA-ES fitness per evaluation. The lines in full are averages of 20 seeds and the shaded is the average ± standard deviation.

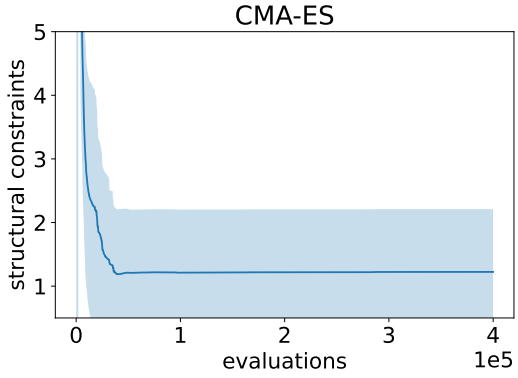


Figure 3.11: Structural constraints value achieved by CMA-ES. The line in full is the average of 20 runs and the shaded is the average ± standard deviation of the best individual per evaluation. The maximum y value was limited.

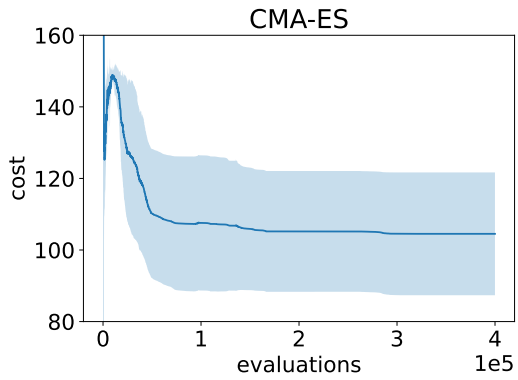
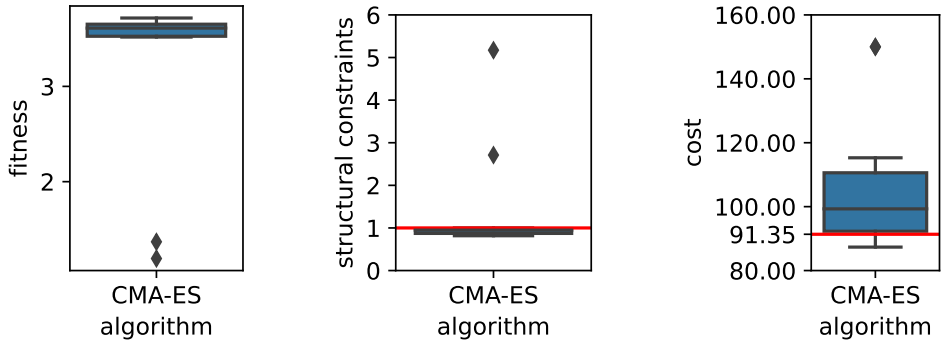


Figure 3.12: Cost achieved by CMA-ES in the tens of thousands of euros. The line in full is the average of 20 runs and the shaded is the average ± standard deviation of the best individual per evaluation. The maximum y value was limited.



(a) Fitness. (b) Structural constraints. (c) Cost.

Figure 3.13: Boxplots created with the best individual of each of the 20 CMA-ES runs. The cost is presented in the tens of thousands. The red line in the cost boxplot is the cost of the baseline solution and in the structural constraints plot is the threshold of safety (1).

from there the price decreases are very subtle, showing that past the 100 000 evaluations, we are not getting significant improvements. However, resorting to the boxplot in Fig. 3.13c, we can see that the algorithm was able to find several individuals below the 100×10^4 € mark and that at least one is cheaper than the baseline.

The best individual obtained by CMA-ES has a cost of 87.339×10^4 € and structural constraints of 1.000. This individual beats the cost of the baseline in 4.015×10^4 €, while still being safe. The graphical comparison of the geometry of both structures is presented in Fig. 3.14. We can see that the both structures are using a fan cable system, but the structure evolved is taller than the baseline.

By looking at the fitness heatmap and the activity plot of Fig. 3.15 we can see a pattern in the most searched areas and the areas of higher fitness. This run showed a particular focus on optimization and found the best individual out of all the other CMA-ES runs, but it did not find individuals for many combinations

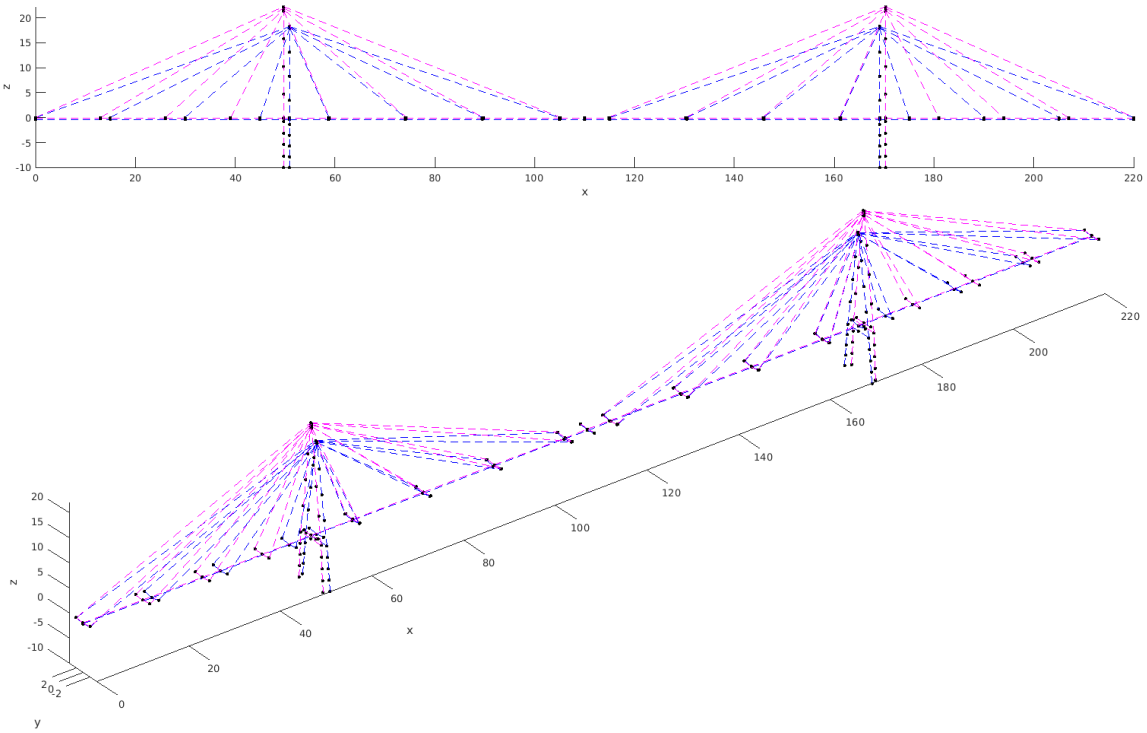
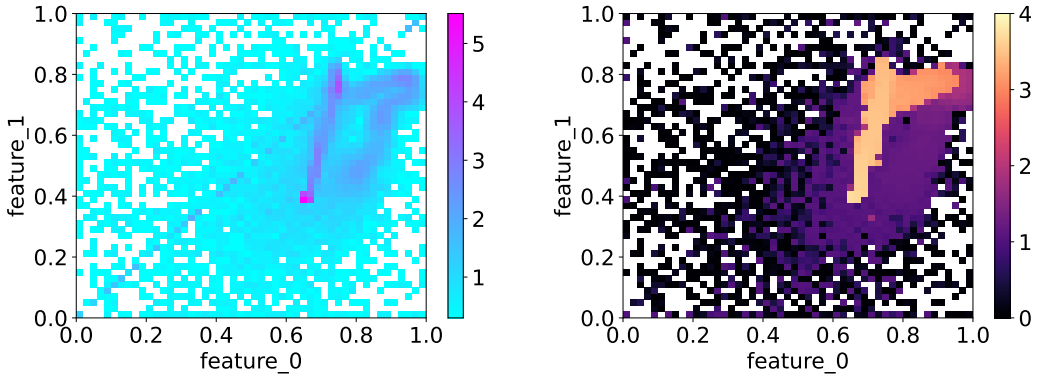


Figure 3.14: Graphical comparison of the baseline (blue) and the best from CMA-ES (pink).

of features, resulting in many empty (white) cells. The remaining plots regarding this seed are presented in Appendix A, section A.2.

Regarding the archives of the 20 runs, Fig. 3.16a shows that only an average of around 45,000 individuals (11%) are added to the archive, which accounts for 11% of the 400,000 individuals created. However, the shaded area increases throughout the execution, indicating a high level of variability in the number of elites added to the archive between runs. Most of the elites added to the archives have low fitness, easily seen by the two large peaks in the histogram of Fig. 3.16b. Once more, we are faced with a very small portion of feasible elites.

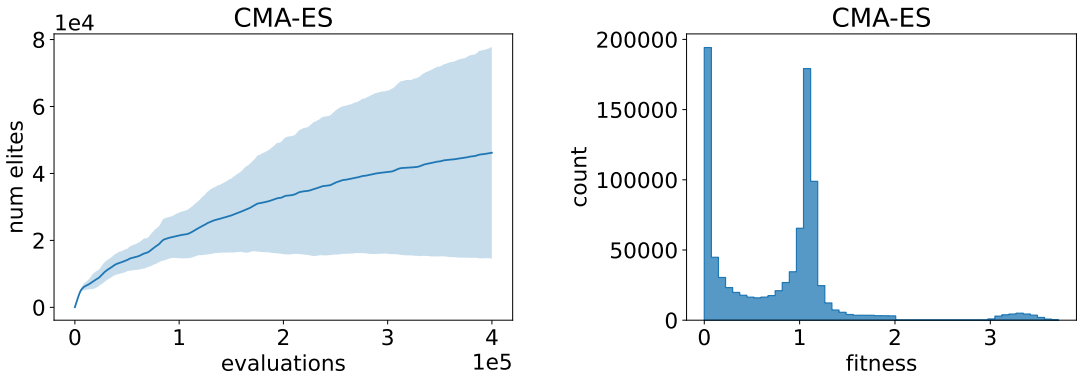
The data in Table 3.5 effectively highlights the disparity in the number of elites added to the archive across different runs (column num_elites), which accounts for the noticeable size of the shaded area in Fig 3.16a. For instance, seed 1529642067 saved 119 997 elites to the archive, while seed 3853914723 only added 8 208 elites. CMA-ES discovered on average more than 16 000 solutions that are more cost-effective than the baseline, however, this number is highly influenced by one of the runs, with 16 out of the 20 runs not being able to find at least an individual that surpasses the baseline (column b_b). We also can observe that from the individuals that beat the baseline, only a small portion ends up in the archive. This means that the majority of these individuals are mapped to the same archive cells, not being significantly different in the feature space.



(a) Activity plot.

(b) Fitness heatmap.

Figure 3.15: On the right side, we present a projection of features 0 and 1 from the grid archive, showcasing the fitness results obtained by the best CMA-ES seed. On the left side, we have an activity plot that provides insights into the regions that were more explored. The activity plot employs a logarithmic scale with a base of 10.



(a) Number of elites.

(b) Fitness distribution.

Figure 3.16: Plots concerning only the individuals added to the grid archive. On the left, the number of elites in the archive is shown per evaluation. The line in full is the average of the 20 seeds and the shaded is the average $\pm$ standard deviation. On the right, we present a histogram of the distribution of fitness in the archives at the end of the executions of the algorithm. All the fitness values in the 20 archives were grouped together and then utilized to create the histogram with 50 bins.

Table 3.5: Diversity statistics for each of the 20 CMA-ES seeds. Floating point values are rounded to 3 decimal points.

seed	num_elites	coverage	qd_score	f	f_arch	b_b	b_b_arch
1961415599	10833	0.027	10834.563	337098	1266	311107	52
265866547	37082	0.093	27991.795	33738	1323	8753	15
921333072	98768	0.247	63501.597	59620	2040	0	0
134371839	56858	0.142	55145.321	48217	3162	3	3
3864806146	14406	0.036	16940.478	200385	2009	0	0
3458633712	64555	0.161	52679.862	42567	1664	0	0
2230458389	32014	0.080	30248.440	13123	2076	0	0
953075784	53408	0.134	46039.115	129376	1554	5947	2
4096528757	61670	0.154	36435.490	56612	799	0	0
1517680419	57691	0.144	47258.291	32074	1470	0	0
415886967	98737	0.247	63978.253	52655	1178	0	0
1529642067	**119997**	0.300	83126.045	51275	1270	0	0
3181050477	23744	0.059	22183.984	343954	1653	0	0
2091038421	39768	0.099	25872.586	49384	1369	0	0
459054007	68870	0.172	64063.742	21923	885	0	0
1649963927	9854	0.025	10747.097	378620	1484	0	0
786905057	19506	0.049	15801.130	13128	1192	0	0
3338455042	29625	0.074	23461.752	317991	1379	0	0
3625157141	18076	0.045	12049.710	0	0	0	0
3853914723	**8208**	0.021	6151.508	0	0	0	0
mean	46183.500	**0.115**	35725.538	109087	1388.650	**16290.500**	3.600

3.4.3 Multi-dimensional Archive of Phenotypic Elites Results

In Fig. Fig. 3.17, the fitness of the best and the average fitness of the population is presented. In this case, since the population has a size of 1, the average is equal to the best. By examining the figure we see that MAP-Elites presents poor results in terms of optimization. The fitness values are low, ultimately rendering the discovered individuals completely unusable. The plots regarding the cost and the structural constraints of the best show that these objectives were never close to being minimized (the average lines can never be seen in the limited plots). The boxplots of Fig. 3.20 clearly show us that the algorithm was not able to minimize the structural constraints. These results could be due to poor parameterization, bad luck with the seeds used (it is very unlikely) or the algorithm may not be suited for this type of problem.

We decided not to present a comparison with the baseline, since the algorithm was not capable of finding at least one feasible configuration.

Since there is no optimization occurring, the fitness heatmap of Fig. 3.21b is mostly covered in black with a few purple points. In terms of search activity, we can see in Fig. 3.21a that the algorithm focused its search on the cells located in the diagonal. Aside from the diagonal, the search of the space seems very uniform over the feature space. The remainder of the fitness heatmaps and activity plots concerning this seed are shown in Appendix A, section A.5.

What MAP-Elites lacks in optimization power, excels in terms of diversity, being able to consistently add around 95% of the created individuals into the archive (Table 3.6, column coverage, and Fig. 3.22a), which is very impressive. However,

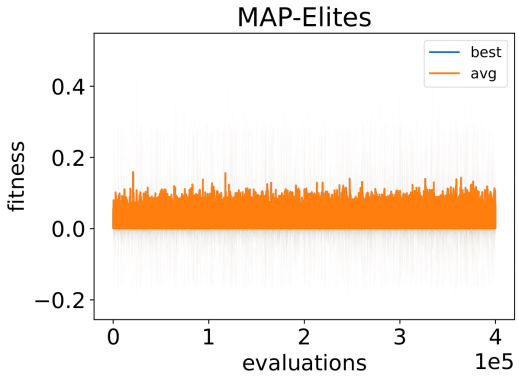
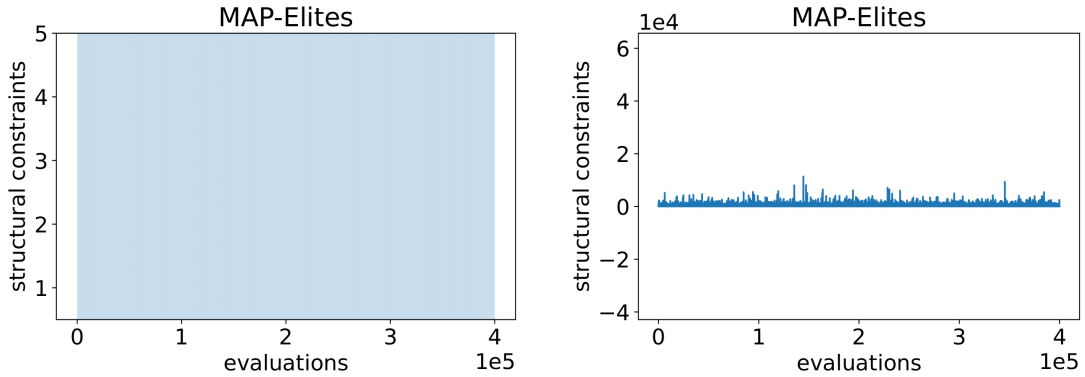


Figure 3.17: MAP-Elites fitness per evaluation. The lines in full are averages of 20 seeds and the shaded is the average ± standard deviation. Since the population size is 1, the best is the same as the average, so the lines are overlapped.

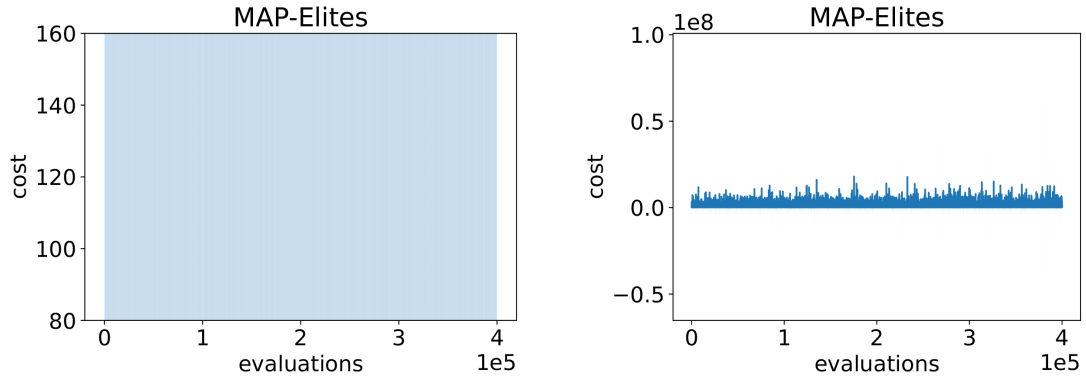


(a) Structural constraints limited in the y axis.

(b) Structural constraints not limited in the y axis.

Figure 3.18: Structural constraints value achieved by MAP-Elites. The line in full is the average of 20 runs and the shaded is the average ± standard deviation of the best individual per evaluation. The same data was used for both plots, however, on the left, the maximum y value was limited.

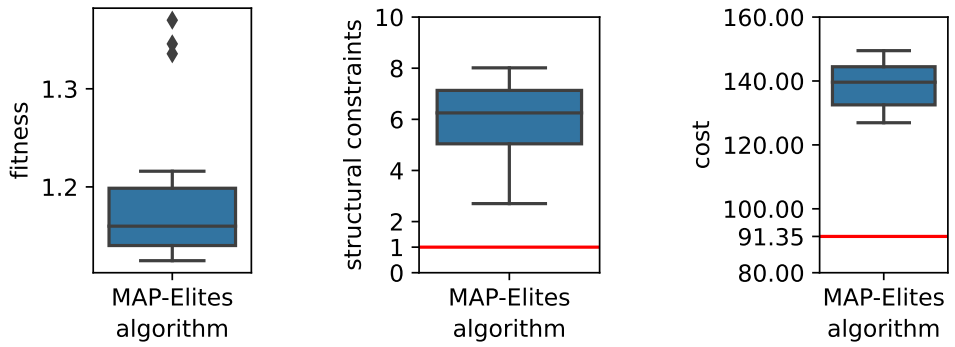
this diversity came at a cost of performance, given that almost all individuals have a fitness lower than 0.25, as can be seen in the histogram of Fig. 3.22b.



(a) Cost limited in the y axis.

(b) Cost not limited in the y axis.

Figure 3.19: Cost achieved by MAP-Elites in the tens of thousands of euros. The line in full is the average of 20 runs and the shaded is the average ± standard deviation of the best individual per evaluation. The same data was used for both the right and left plots, however, on the left, the maximum y value was limited.

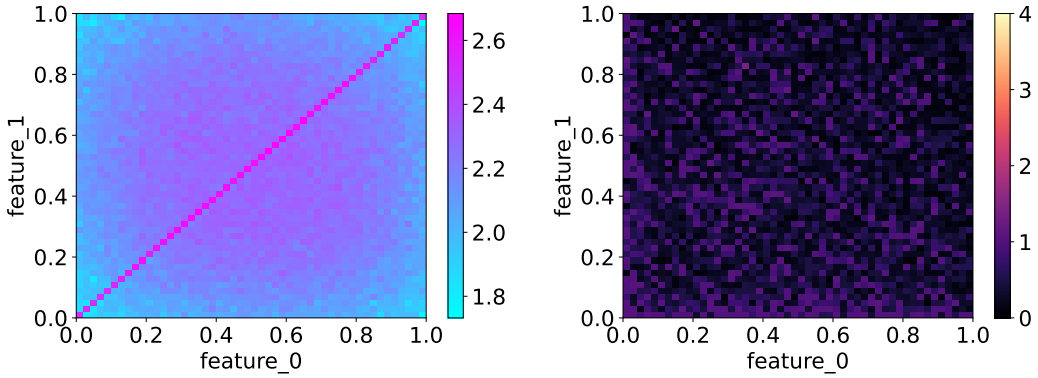


(a) Fitness.

(b) Structural constraints.

(c) Cost.

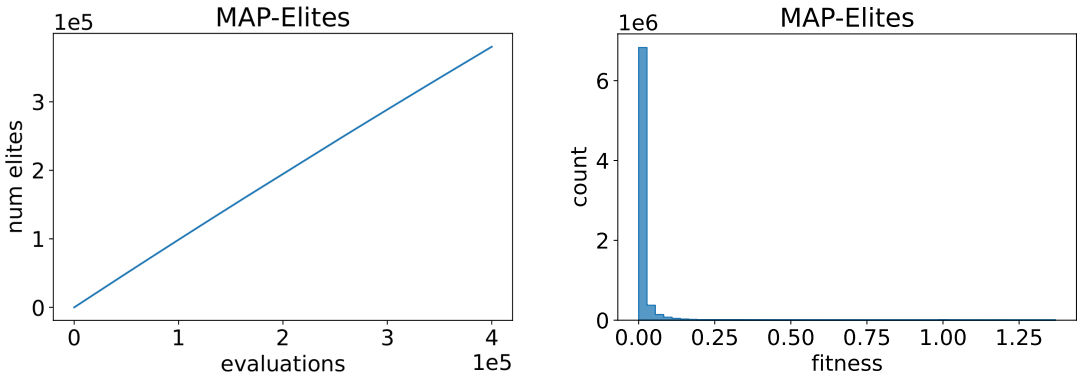
Figure 3.20: Boxplots created with the best individual of each of the 20 MAP-Elites runs. The cost is presented in the tens of thousands. The red line in the cost boxplot is the cost of the baseline solution and in the structural constraints plot is the threshold of safety (1).



(a) Activity plot.

(b) Fitness heatmap.

Figure 3.21: On the right side, we present a projection of features 0 and 1 from the grid archive, showcasing the fitness results obtained by the best MAP-Elites seed. On the left side, we have an activity plot that provides insights into the regions that were more explored. The activity plot employs a logarithmic scale with a base of 10.



(a) Number of elites.

(b) Fitness distribution.

Figure 3.22: Plots concerning only the individuals added to the grid archive. On the left, the number of elites in the archive is shown per evaluation. The line in full is the average of the 20 seeds and the shaded is the average $\pm$ standard deviation. On the right, we present a histogram of the distribution of fitness in the archives at the end of the executions of the algorithm. All the fitness values in the 20 archives were grouped together and then utilized to create the histogram with 50 bins.

Table 3.6: Diversity statistics for each of the 20 MAP-Elites seeds. Floating point values are rounded to 3 decimal points.

seed	num_elites	coverage	qd_score	f	f_arch	b_b	b_b_arch
977503172	380647	0.952	5162.759	0	0	0	0
3211678240	381202	0.953	5661.815	0	0	0	0
3928983013	380388	0.951	5609.315	0	0	0	0
857742291	380558	0.951	5185.294	0	0	0	0
2714227225	379905	0.950	5969.032	0	0	0	0
3790900665	379966	0.950	5877.189	0	0	0	0
1100567897	380135	0.950	5535.930	0	0	0	0
2723373142	379786	0.949	5622.072	0	0	0	0
3867363402	380383	0.951	5735.830	0	0	0	0
1290171389	381468	0.954	5605.492	0	0	0	0
2309899029	380164	0.950	5391.221	0	0	0	0
1354215289	380345	0.951	4998.696	0	0	0	0
3852609653	380549	0.951	5398.884	0	0	0	0
3885547211	380820	0.952	5490.403	0	0	0	0
2814991318	380405	0.951	5589.959	0	0	0	0
816692951	380195	0.950	5186.647	0	0	0	0
4240765728	380364	0.951	5478.891	0	0	0	0
2160711787	380007	0.950	5309.213	0	0	0	0
1860587127	380682	0.952	4764.029	0	0	0	0
3602535396	379182	0.948	5816.020	0	0	0	0
mean	380357.550	0.951	5469.435	0	0	0	0

3.4.4 Covariance Matrix Adaptation Multi-dimensional Archive of Phenotypic Elites Results

Fig. 3.23 displays the fitness evolution plots for the CMA-ME. Looking at the result one can see that during the optimization process, the fitness varies. This is due to the fact that the algorithm does not use elitism. Under the hood CMA-ME is using instances of CMA-ES (emitters), in our case just one. The differentiating factor is that when one emitter fails to produce at least one individual that ends up being added to the archive in the current generation, it is restarted. This restart defaults all parameters to the initially defined ones apart from the centroid. A randomly sampled elite from the grid archive is used as the new strategy centroid. With this in mind, the drops in fitness that are observed on both plots of Fig. 3.23 are due to the restart of the emitter. An interesting aspect to notice is that the average fitness values are much closer to the maximum value than in the GA and CMA-ES. Since the values in these plots are averages, we also show the fitness of the best individual found by each run in a boxplot in Fig. 3.26a. By looking at it, it is easy to realize that most of the runs were able to find individuals with high fitness, given that all are above 3.5.

Since the fitness is computed considering both the cost and the structural constraints values, it is only natural that the lines concerning the two also exhibit high variability. By looking at Fig. 3.24 we see that the average line of the structural constraints of the best never reaches values close to 1, which may bear an indication that the algorithm failed in optimizing this objective. This is due to the variability presented by the algorithm between runs and the absence of elitism. Since the desired values for this objective are small, it may take only one run at a

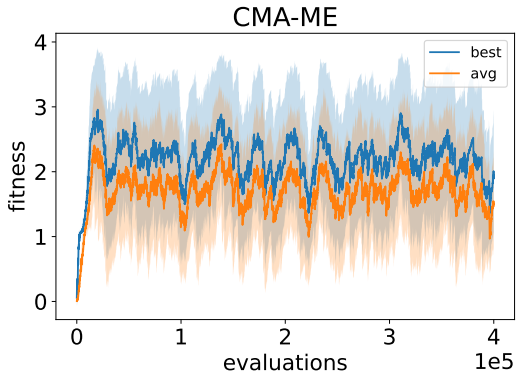


Figure 3.23: CMA-ME fitness per evaluation. The lines in full are averages of 20 seeds and the shaded is the average ± standard deviation.

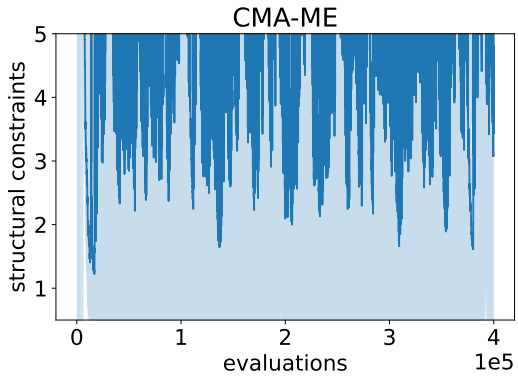


Figure 3.24: Structural constraints value achieved by CMA-ME. The line in full is the average of 20 runs and the shaded is the average ± standard deviation of the best individual per evaluation. The maximum y value was limited.

particular time to produce a best with poor performance to greatly influence this average. By looking at the boxplot in Fig. 3.26b we can see that the algorithm was capable of minimizing this objective. In fact, the best individual from each of the 20 algorithm executions have all structural constraints of at most 1.

Similarly to the structural constraints, the average cost of the best individuals presented in Fig. 3.25 shows high variability throughout the execution. There is also a high variability between runs, perceived by the large standard deviation values. Looking at the boxplot of Fig. 3.26c we can see that only 1 out of the 20 seeds failed to beat the baseline, and even then it was only by 0.605×10^4 €.

The individual with the highest fitness found by CMA-ME costs 87.652×10^4 € and 0.9914 of structural constraints, which effectively beats the baseline solution by 3.702×10^4 €. However, this is not the cheapest feasible structure that this algorithm reached. That one has a price of 87.390×10^4 € and a value of structural constraints of 0.9994, which is a saving of 3.964×10^4 € compared to the baseline. The geometry of both structures is compared with the geometry of the baseline solution in Fig. 3.27. We can see that the individuals evolved are similar. They are taller and have the towers placed further away from the center of the bridge.

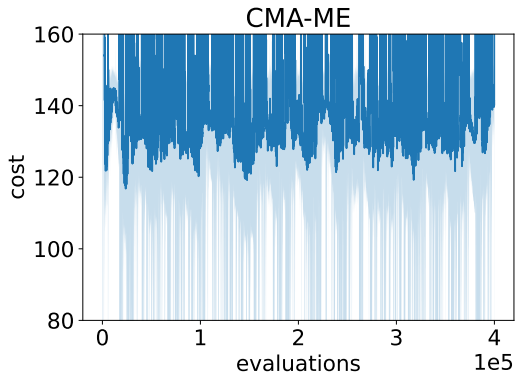
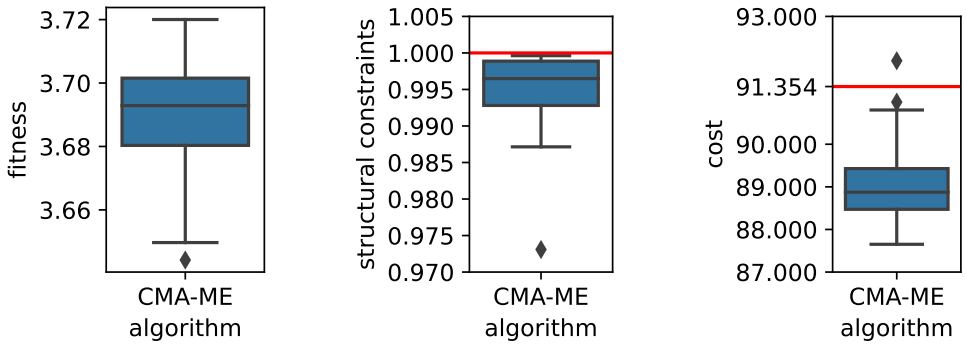


Figure 3.25: Cost achieved by CMA-ME in the tens of thousands of euros. The line in full is the average of 20 runs and the shaded is the average ± standard deviation of the best individual per evaluation. The maximum y value was limited.



(a) Fitness. (b) Structural constraints. (c) Cost.

Figure 3.26: Boxplots created with the metrics of the best individual of each of the 20 CMA-ME runs. The cost is presented in the tens of thousands. The red line in the cost boxplot is the cost of the baseline solution and in the structural constraints plot is the threshold of safety (1).

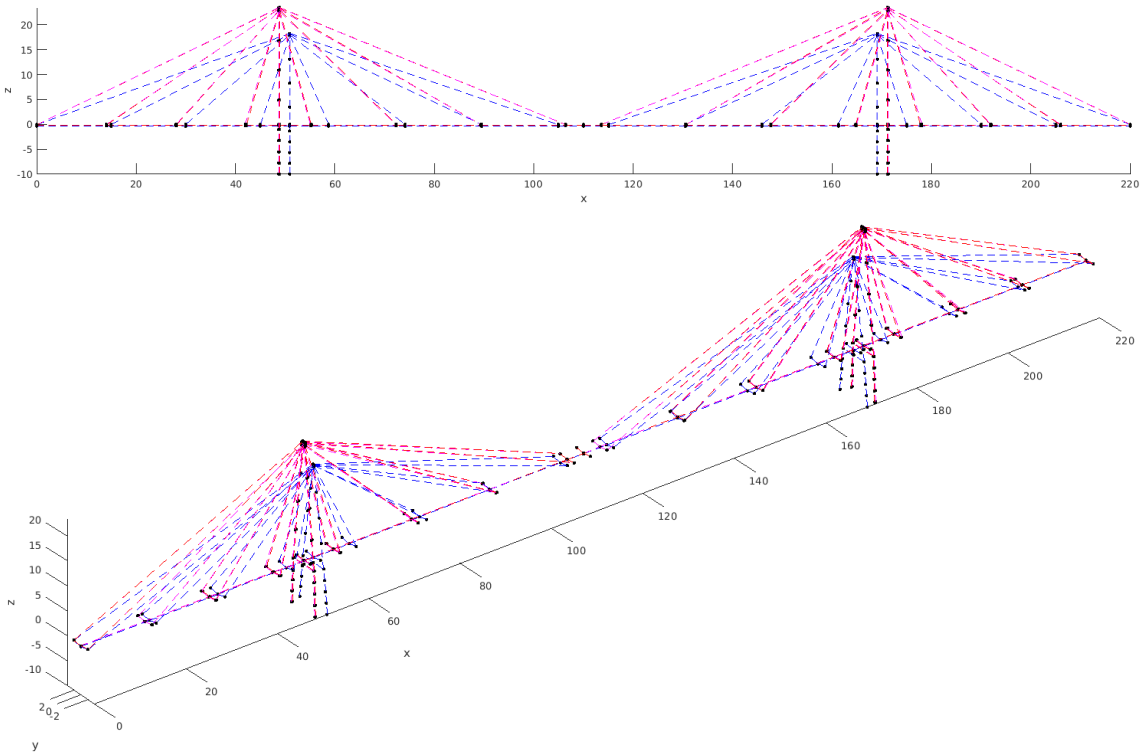
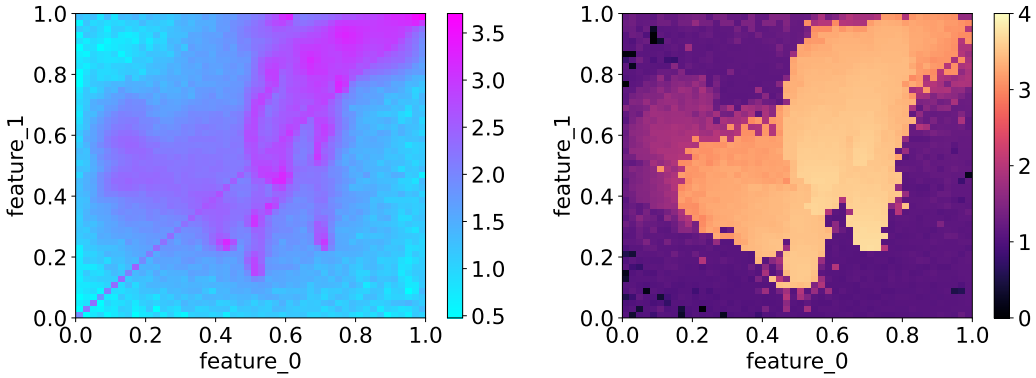


Figure 3.27: Graphical comparison of the baseline (blue), the best (pink), and the cheapest (red) from CMA-ME.

The fitness heatmap of Fig. 3.28b shows that the algorithm was able to find many feasible individuals (fitness higher than 3) with various combinations of feature_0 and feature_1. The coverage of these individuals, although not uniform, is spread among almost all of the possible values of each feature, which suggests that there could be feasible individuals with every possible value of the interval. Once more, the activity plot (Fig. 3.28a) shows a similar pattern of search activity to the one of the fitness, making it more and more reasonable to think that if the algorithm had spent more time searching a region of lower fitness it would end up discovering better individuals. The remainder of the fitness heatmaps and activity plots concerning this seed are shown in Appendix A, section A.3.

Looking at the line of the number of elites in the archive from Fig. 3.29a we observe that 50% of the generated individuals consistently enter the archive, maintaining a steady population growth rate (represented by an almost straight-line). Notably, the algorithm demonstrates a high level of consistency across runs, as indicated by the narrow shaded area. This consistency implies that the values obtained from different seeds are highly similar at each time point.

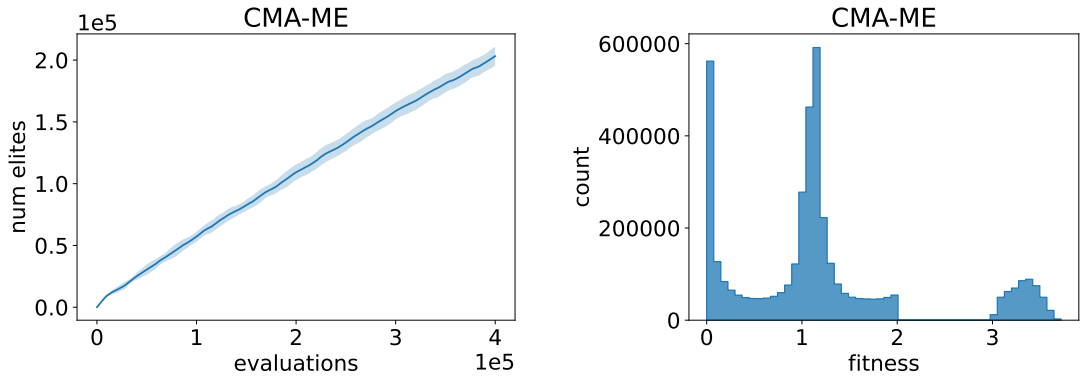
This result presents a significant advantage compared to pure optimization algorithms, showcasing the capability of this QD algorithm to discover a more diverse range of configurations. Examining the fitness distribution in Fig. 3.29b, it becomes apparent that this improvement in diversity does not come at the expense of fitness performance. In fact, the proportion of feasible individuals in the archives has increased compared to the GA and CMA-ES. However, the majority of elites in the archives are unfeasible.



(a) Activity plot.

(b) Fitness heatmap.

Figure 3.28: On the right side, we present a projection of features 0 and 1 from the grid archive, showcasing the fitness results obtained by the best CMA-ME seed. On the left side, we have an activity plot that provides insights into the regions that were more explored. The activity plot employs a logarithmic scale with a base of 10.



(a) Number of elites.

(b) Fitness distribution.

Figure 3.29: Plots concerning only the individuals added to the grid archive. On the left, the number of elites in the archive is shown per evaluation. The line in full is the average of the 20 seeds and the shaded is the average $\pm$ standard deviation. On the right, we present a histogram of the distribution of fitness in the archives at the end of the executions of the algorithm. All the fitness values in the 20 archives were grouped together and then utilized to create the histogram with 50 bins.

Table 3.7: Diversity statistics for each of the 20 CMA-ME seeds. Floating point values are rounded to 3 decimal points.

seed	num_elites	coverage	qd_score	f	f_arch	b_b	b_b_arch
1501731213	204308	0.511	245683.084	109832	26424	1420	82
665875114	184764	0.462	231673.454	121105	26443	3333	167
143348516	205277	0.513	233916.111	101806	22843	3669	127
2212182297	218030	0.545	254073.207	95190	24102	2995	139
1872887387	204251	0.511	261241.410	124602	29779	2816	148
4202634575	191506	0.479	240150.265	132265	30242	3301	167
223019384	198062	0.495	239225.156	112205	27091	2933	106
3160294597	196483	0.491	226626.981	116538	24092	2747	139
387715579	201324	0.503	241148.173	109282	25568	1577	55
2477021820	199682	0.499	240990.551	117695	23921	2561	126
1740708445	197157	0.493	244925.523	131041	30041	1731	122
3671252546	205756	0.514	237485.411	104004	25087	1528	117
2771474702	207411	0.519	244161.625	123205	27281	2702	99
1939868274	207622	0.519	249587.755	97215	25813	941	33
3049757501	209264	0.523	242647.721	122071	25970	1983	94
1404091650	202843	0.507	239610.895	119625	25217	85	13
2579782080	212033	0.530	249293.118	115593	26799	876	59
3356617878	214647	0.537	258046.850	110824	25819	271	32
3024264489	203675	0.509	230191.350	111099	24024	6	6
3639591557	199059	0.498	220571.290	101896	19993	0	0
mean	203157.700	**0.508**	241562.496	113854.650	25827.450	1873.750	**91.550**

Table 3.7 confirms the consistent performance of CMA-ME, with metrics showing remarkable similarity across runs. It also highlights the algorithm's effectiveness, as it outperformed the baseline in 19 times out of 20 (only one 0 in column b_b). Another interesting aspect is that of all the individuals that beat the baseline, on average, more than 91 were significantly different from each other at the feature space level, resulting in their addition to the archive.

3.4.5 Multi-Emitter Multi-dimensional Archive of Phenotypic Elites Results

Looking at the fitness lines of the best and the average of the population in Fig. 3.30 we can see a rapid increase in fitness, followed by a considerable decrease with good individuals found before the 100 000 evaluations mark. This algorithm was not implemented with elitism, so the fluctuation in the fitness values is completely expected. Considering this, the decrease is normal, because it takes only a few examples with poor performance in one generation to greatly impact the average line of the best. The plots of figures 3.31 and 3.32, regarding the structural constraints and the cost respectively, also feature the high variability of the fitness, since the fitness is a result of the combination of these two objectives.

From a pure optimization perspective, in addition to the performance over time, one is interested in the best individuals found by the algorithm, and for that, we present a boxplot in Fig. 3.33a. This plot shows how the best individuals from the 20 runs are distributed in terms of fitness. We can see that the algorithm consistently discovered individuals with fitness higher than 3.5. When we look

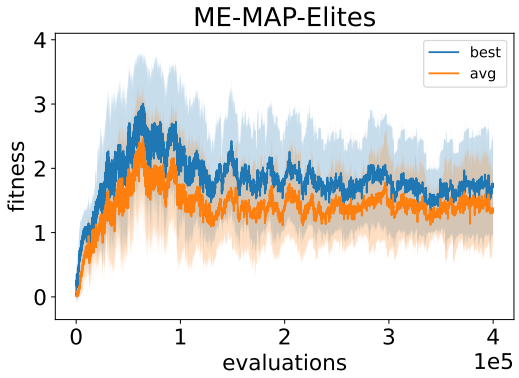


Figure 3.30: ME-MAP-Elites fitness per evaluation. The lines in full are averages of 20 seeds and the shaded is the average ± standard deviation.

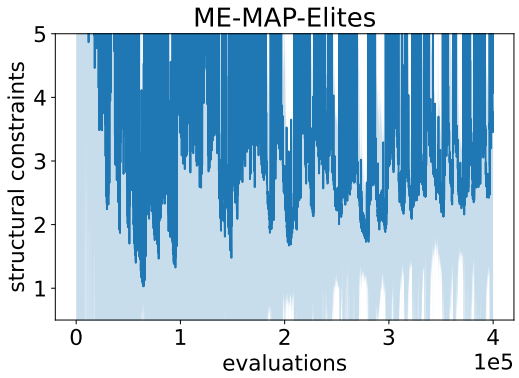


Figure 3.31: Structural constraints value achieved by ME-MAP-Elites. The line in full is the average of 20 runs and the shaded is the average ± standard deviation of the best individual per evaluation. The maximum y value was limited.

at the structural constraints boxplot of Fig. 3.33b, we observe that the algorithm was capable of finding bridges that are safe in every single run, with some having this objective around 0.9. Concerning the cost, by analyzing the boxplot of Fig. 3.33c we observe that most of the best individuals cost less than 100×10^4 €.

The best individual found by ME-MAP-Elites costs 92.422×10^4 €, with a structural constraints value of 0.9876. However, this was not the cheapest feasible configuration discovered. That has a cost of 91.908×10^4 € and a structural constraints value of 0.9983, but it is not enough to beat the baseline. A graphical comparison of the baseline against these two structures is presented in Fig. 3.34. The structures discovered by ME-MAP-Elites are taller than the baseline and the towers are further away from the center of the bridge.

By looking at the fitness heatmap in Fig. 3.35b, we can see that the feasible structures discovered by the algorithm are all located in the top right corner of the fitness heatmap. The feasible region (fitness higher than 3) is significantly large, indicating that the algorithm was capable of finding several distinct feasible solutions. The activity plot of Fig. 3.35a shows us that the algorithm spent more time searching in the second half of the intervals of feature_0 and feature_1. Once

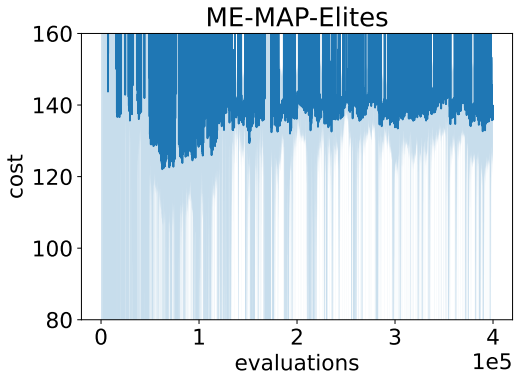
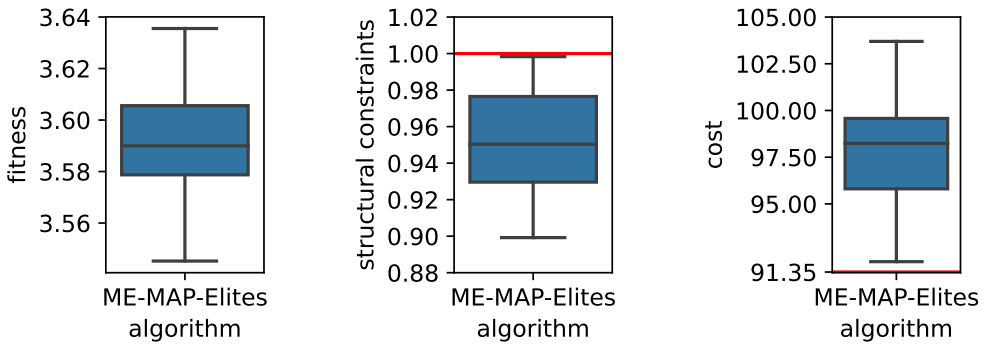


Figure 3.32: Cost achieved by ME-MAP-Elites in the tens of thousands of euros. The line in full is the average of 20 runs and the shaded is the average ± standard deviation of the best individual per evaluation. The maximum y value was limited.



(a) Fitness. (b) Structural constraints. (c) Cost.

Figure 3.33: Boxplots created with the best individual of each of the 20 ME-MAP-Elites runs. The cost is presented in the tens of thousands. The red line in the cost boxplot is the cost of the baseline solution and in the structural constraints plot is the threshold of safety (1).

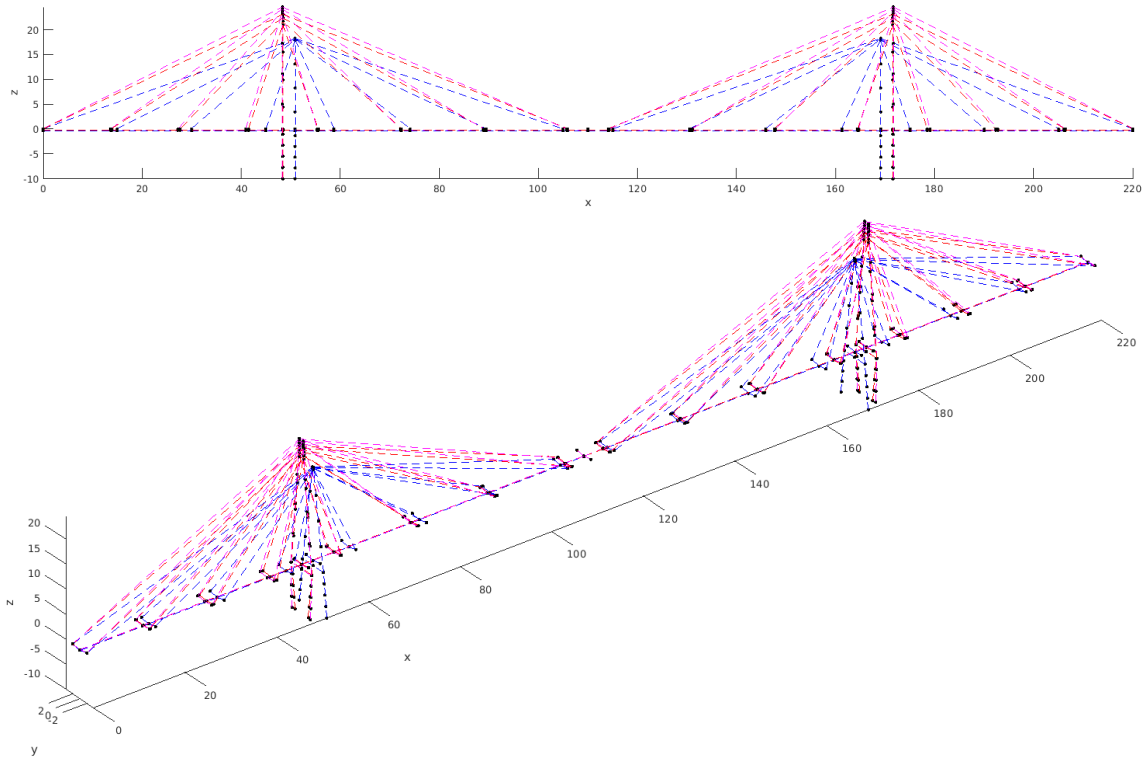
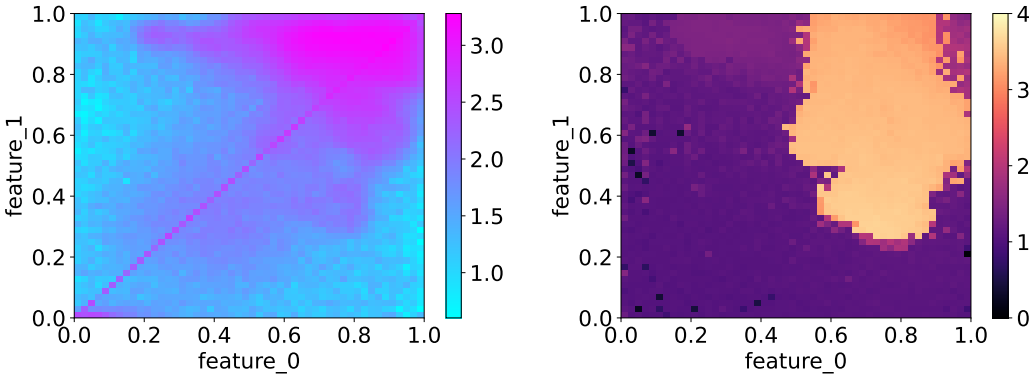


Figure 3.34: Graphical comparison of the baseline (blue), the best (pink), and the cheapest (red) from the ME-MAP-Elites.

more, we can see a pattern in the regions of more activity and higher fitness. The remainder of the fitness heatmaps and activity plots concerning this seed are shown in Appendix A, section A.4.

In terms of elites added to the archive (Fig. 3.36a), ME-MAP-Elites consistently added close to 300 000 individuals to the archive, at a very constant rate throughout the algorithm's execution. This means that 75% of the created individuals in each run are added to the archive, telling us that the algorithm is searching in different regions of the feature space most often than not. Resorting to the fitness histogram of Fig. 3.36b, we get the confirmation that most of the individuals in the archive are not feasible, with a large number of configurations having a little more than 1 of fitness. We also have a considerable amount of individuals in the 3 plus fitness range, which tells us that the algorithm was capable of finding various feasible individuals.

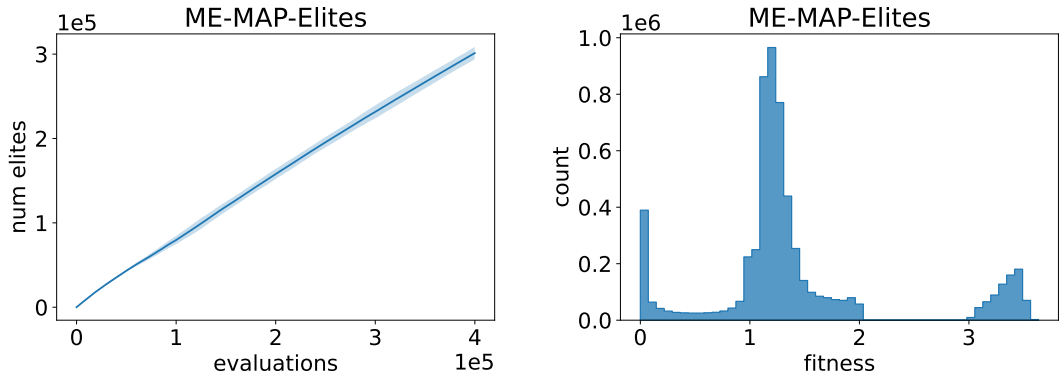
Table 3.8 corroborates what was previously said about the performance of the algorithm in terms of diversity, given that the average number of elites in the archive at the end of the execution of the algorithm is higher than 300 000. We can see that run 2900709671 was the one that had the fittest archive because it achieved the higher qd_score of all the seeds, even though it is one of the runs with the least elites in the archive. This is due to the fact that from the 290904 elites in the archive, 86854 were feasible, pushing the qd_score up. Another interesting point is that, on average, almost 70% of the feasible solutions created were significantly different in behavior.



(a) Activity plot.

(b) Fitness heatmap.

Figure 3.35: On the right side, we present a projection of features 0 and 1 from the grid archive, showcasing the fitness results obtained by the best ME-MAP-Elites seed. On the left side, we have an activity plot that provides insights into the regions that were more explored. The activity plot employs a logarithmic scale with a base of 10.



(a) Number of elites.

(b) Fitness distribution.

Figure 3.36: Plots concerning only the individuals added to the grid archive. On the left, the number of elites in the archive is shown per evaluation. The line in full is the average of the 20 seeds and the shaded is the average $\pm$ standard deviation. On the right, we present a histogram of the distribution of fitness in the archives at the end of the executions of the algorithm. All the fitness values in the 20 archives were grouped together and then utilized to create the histogram with 50 bins.

Table 3.8: Diversity statistics for each of the 20 ME-MAP-Elites seeds. Floating point values are rounded to 3 decimal points.

seed	num_elites	coverage	qd_score	f	f_arch	b_b	b_b_arch
4234418804	306854	0.767	382189.719	30925	24054	0	0
1348952993	306459	0.766	411657.422	43037	29506	0	0
2900709671	**290904**	0.727	**525237.801**	128612	**86854**	0	0
1360480637	300557	0.751	388336.003	27412	20957	0	0
2829501598	294816	0.737	446029.423	74610	50777	0	0
1511567270	292839	0.732	490383.830	111340	74076	0	0
3187816561	302894	0.757	424243.415	52458	35882	0	0
3571708301	304598	0.761	452983.731	70172	49343	0	0
4287285846	307569	0.769	394860.832	34357	26828	0	0
1847013518	302954	0.757	366948.821	16353	11705	0	0
204662516	301165	0.753	384624.533	24184	17441	0	0
477761632	305921	0.765	375931.615	12413	9651	0	0
4077363031	305964	0.765	403111.899	42957	31857	0	0
3071960880	312273	0.781	419452.484	43187	32468	0	0
439834480	282594	0.706	520554.371	149426	95048	0	0
3998953568	288258	0.721	471174.065	89850	60099	0	0
1360000300	306826	0.767	369089.778	17724	14185	0	0
2201719119	304069	0.760	415315.466	39169	29627	0	0
4095548407	302326	0.756	415991.069	46892	30552	0	0
618737211	305208	0.763	388083.716	25127	17946	0	0
mean	**301252.400**	**0.753**	422310	**54010.250**	**37442.800**	0	0

3.4.6 Hybrid Results

Looking at Fig. 3.37 one can see that the fitness of the best tends to increase during the algorithm's execution, and since the Hybrid uses elitism, it never decreases. The average fitness, although slowly, also increases throughout the run. This means that the population is getting filled by individuals with better fitness.

Fig. 3.38 shows us that on average, the algorithm is capable of finding structurally safe bridges within 150 000 evaluations, given that after that the line stays very close to 1 with low variability. By looking at the boxplot presented in Fig. 3.40b and column f of Table 3.9 one can see that in 19 of the runs, the algorithm was

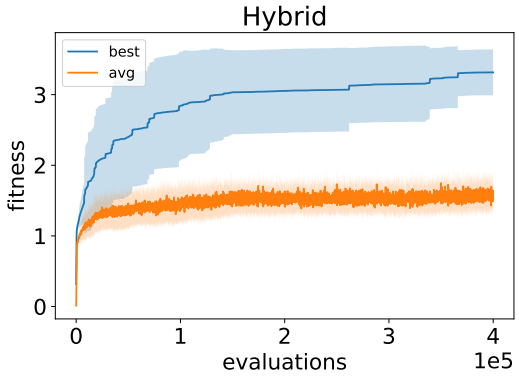


Figure 3.37: Hybrid fitness per evaluation. The lines in full are averages of 20 seeds and the shaded is the average ± standard deviation.

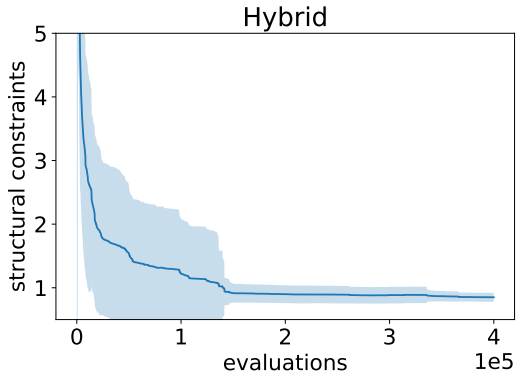


Figure 3.38: Structural constraints value achieved by the Hybrid. The line in full is the average of 20 runs and the shaded is the average ± standard deviation of the best individual per evaluation. The maximum y value was limited.

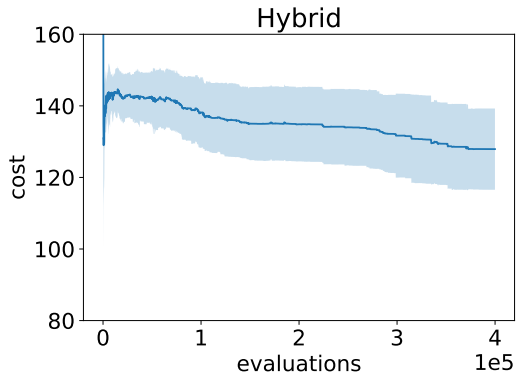
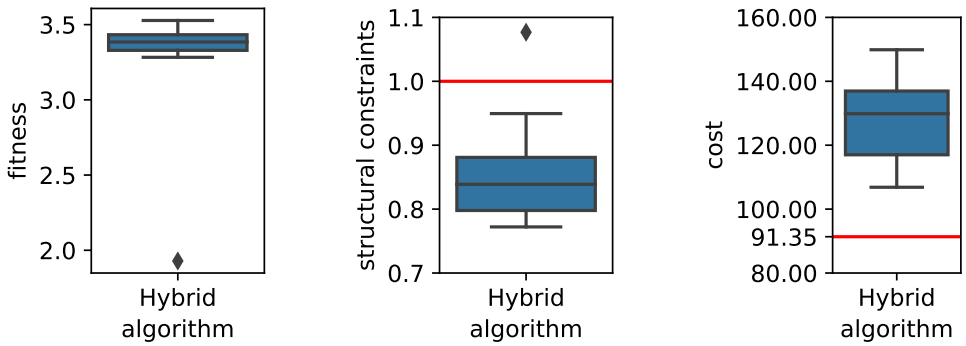


Figure 3.39: Cost achieved by the Hybrid in the tens of thousands of euros. The line in full is the average of 20 runs and the shaded is the average ± standard deviation of the best individual per evaluation. The maximum y value was limited.

able to minimize this objective.

By analyzing the plot in Fig. 3.39 we can see that the cost of the best individual quickly reaches acceptable values (below $c_r = 150$). Similarly to what happens in the GA, we witness a drop in the price, followed by a slight increase, and then from there on it decreases until the end of the execution of the algorithm. The boxplot concerning the cost, Fig. 3.40c, shows us that the baseline was not surpassed and that there is relatively high variability in the prices of the best individuals.

The individual with the best fitness found by the Hybrid costs 111.021×10^4 € and has a structural constraints value of 0.8500. This is not the cheapest safe configuration that the algorithm reached, that one costs 105.288×10^4 € and has a structural constraints value of 0.981. The graphical comparison of the geometry of these bridges against the baseline is shown in Fig. 3.41. The evolved solutions are using a harp cable system while the baseline uses a fan cable system. The towers of the evolved structures are closer to the center of the bridge than the ones on the baseline.



(a) Fitness. (b) Structural constraints. (c) Cost.

Figure 3.40: Boxplots created with the best individual of each of the 20 Hybrid runs. The cost is presented in the tens of thousands. The red line in the cost boxplot is the cost of the baseline solution and in the structural constraints plot is the threshold of safety (1).

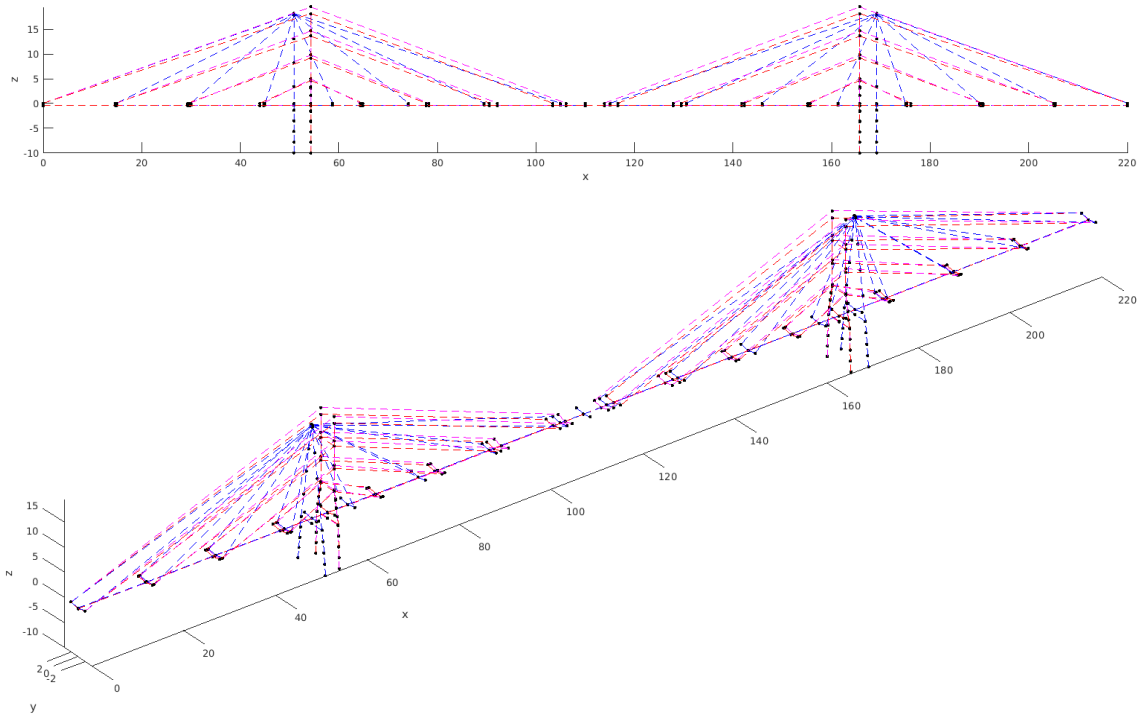
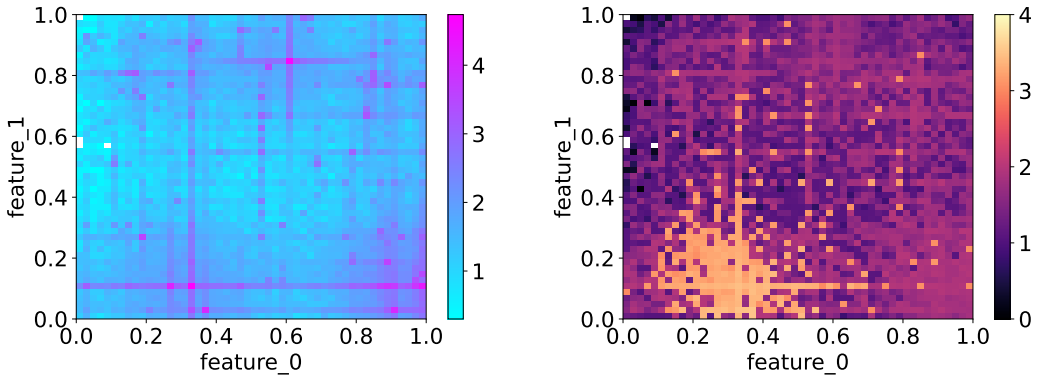


Figure 3.41: Graphical comparison of the baseline (blue), the best (pink), and the cheapest (red) from the Hybrid.



(a) Activity plot.

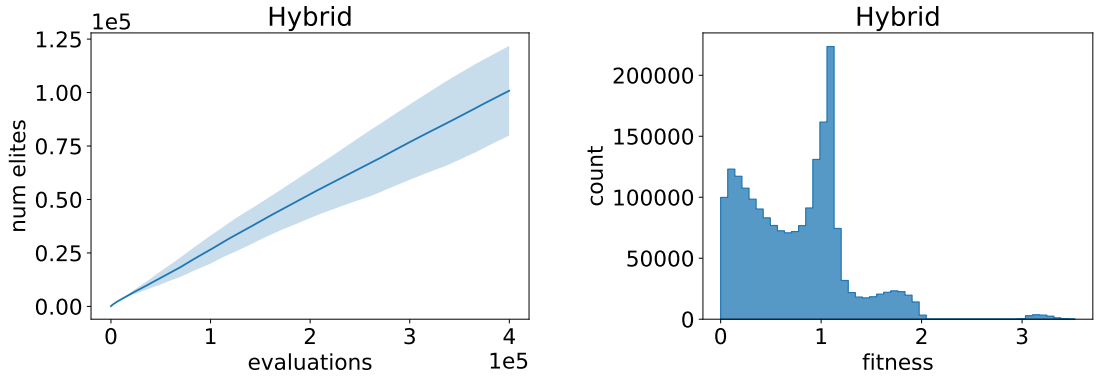
(b) Fitness heatmap.

Figure 3.42: On the right side, we present a projection of features 0 and 1 from the grid archive, showcasing the fitness results obtained by the best Hybrid seed. On the left side, we have an activity plot that provides insights into the regions that were more explored. The activity plot employs a logarithmic scale with a base of 10.

This time we cannot see a clear correspondence between more searched areas and higher values of fitness. The spots of high activity in the activity plot of Fig. 3.42a are more sparse throughout the feature space. The individuals of higher fitness are mostly located in the bottom left corner of the fitness heatmap (Fig. 3.42b), which is not commonly the case. The majority of the fitness heatmaps show higher levels of fitness at the top right corner of this projection of the feature space. The space seems to be illuminated in a more uniform manner than in the ones shown by ME-MAP-Elites and CMA-ME, closer to what happened with the GA. The remainder of the fitness heatmaps and activity plots concerning this seed are shown in Appendix A, section A.6.

Looking at the plot concerning the number of elites added to the archive over time in Fig. 3.43a we can see that on average the Hybrid is capable of finding 100 000 distinct individuals in the feature space. Nonetheless, we can see that there is a noticeable difference in the number of elites in the archives from different runs, perceived by the size of the shaded area. By looking at the fitness histogram in Fig. 3.43b we gain some insight into the fitness level of the individuals in the archives at the end of the execution of the algorithm. We can see that the large majority of the individuals are unfeasible and that the feasible portion is small.

Table 3.9 helps us see some aspects that were already stated. The average number of unique feasible solutions discovered by the Hybrid (column f) is slightly over 3 000, which is low considering the numbers registered by ME-MAP-Elites. The percentage of elites added to the archive (column coverage) is noteworthy, with on average 25% of the created individuals being added to the archive, beating CMA-ES and the GA, for example. Only 1 out of the 20 runs was not able to find a feasible individual (seed 3093707613).



(a) Number of elites.

(b) Fitness distribution.

Figure 3.43: Plots concerning only the individuals added to the grid archive. On the left, the number of elites in the archive is shown per evaluation. The line in full is the average of the 20 seeds and the shaded is the average $\pm$ standard deviation. On the right, we present a histogram of the distribution of fitness in the archives at the end of the executions of the algorithm. All the fitness values in the 20 archives were grouped together and then utilized to create the histogram with 50 bins.

Table 3.9: Diversity statistics for each of the 20 Hybrid seeds. Floating point values are rounded to 3 decimal points.

seed	num_elites	coverage	qd_score	f	f_arch	b_b	b_b_arch
3312140771	100193	0.250	75516.964	3994	873	0	0
3904300547	58530	0.146	45285.664	604	149	0	0
3632522327	108938	0.272	81169.749	3668	596	0	0
1615975349	111617	0.279	84361.641	4567	888	0	0
339781825	112224	0.281	88575.836	6013	1300	0	0
2167276161	107932	0.270	80298.632	3187	749	0	0
2843920188	120148	0.300	98236.185	5492	1297	0	0
4292015115	112594	0.281	87278.825	4551	1160	0	0
4283489550	99145	0.248	74219.072	3491	838	0	0
4192130457	94017	0.235	70998.009	3335	576	0	0
2296412008	108845	0.272	86096.757	3131	793	0	0
3751734061	80710	0.202	59769.505	1677	353	0	0
2436285545	108710	0.272	76271.915	3413	544	0	0
380819443	118614	0.297	85976.895	4033	983	0	0
2361473753	121551	0.304	90869.093	3990	863	0	0
2429895707	111386	0.278	82723.653	3317	680	0	0
3350892851	114776	0.287	88929.175	4261	821	0	0
4273106102	59943	0.150	45372.794	470	109	0	0
1225867867	117052	0.293	88984.052	3532	679	0	0
3093707613	50151	0.125	37372.135	**0**	0	0	0
mean	100853.800	**0.252**	76415.328	**3336.300**	712.550	0	0

3.4.7 Non-dominated Sorting Genetic Algorithm II Results

By analyzing the fitness plot of Fig. 3.44, it can be seen that the fitness of the best increases rapidly at the beginning of the optimization, then decreases until the 100 000 evaluations mark, stabilizing a little over 2 of fitness. Close to the 400 000 evaluations mark, we can see a slight increase in fitness. The average fitness of the population is slowly increasing for the most part of the execution of the algorithm but remains at values considerably lower than the best, which means that the population is mostly filled with individuals with low fitness.

The structural constraints plot of Fig. 3.45 presents some fluctuations and shows that the performance of the algorithm is highly irregular, due to the large shaded area and the constant spikes in the average line. By observing the plot concerning the cost of the best in Fig. 3.46 we can see that at the start it rapidly decreases, reaching values around 125×10^4€ at the 50 000 evaluations mark. From there on, the cost stabilizes.

By examining the boxplots of Fig. 3.47 we see that NSGA-II was able to find feasible bridge configurations in between runs. This is supported by the values shown in column f of Table 3.10, concerning the number of unique feasible individuals that were discovered by each seed. Only 4 seeds failed to find feasible structures, and 2 others found less than 5 during the entire run.

The baseline was not surpassed by any of the structures evolved by NSGA-II. The best individual has a price of 102.660×10^4 € and a structural constraints value of 0.8963, rendering it 11.306×10^4 € more expensive. The cheapest solution discovered costs 98.114×10^4 € and has a structural constraints value of 0.9972, resulting in an additional cost of 6.76×10^4 €. The graphical comparison of the geometry of these three structures is depicted in Fig. 3.48. We can see that the NSGA-II structures are taller than the baseline and have a different cable system. The baseline uses a fan cable system, while the evolved bridges use a semi-fan cable system.

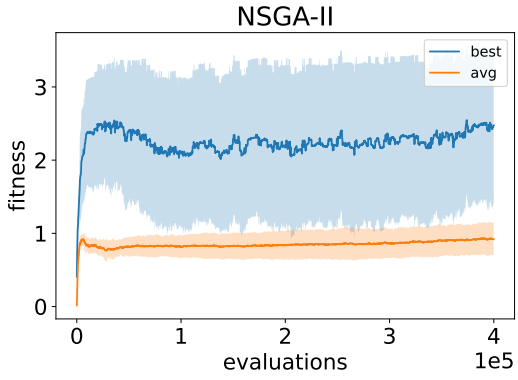


Figure 3.44: NSGA-II fitness per evaluation. The lines in full are averages of 20 seeds and the shaded is the average ± standard deviation.

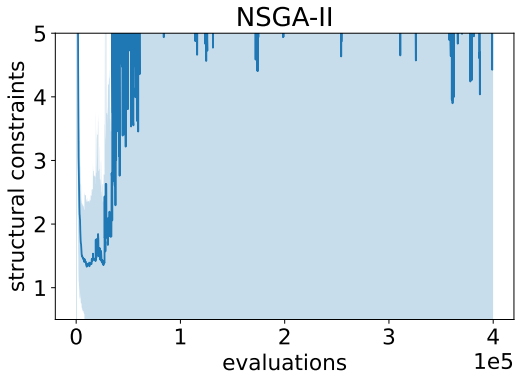


Figure 3.45: Structural constraints value achieved by NSGA-II. The line in full is the average of 20 runs and the shaded is the average ± standard deviation of the best individual per evaluation. The maximum y value was limited.

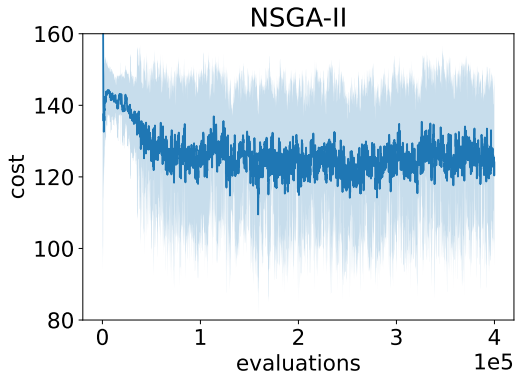
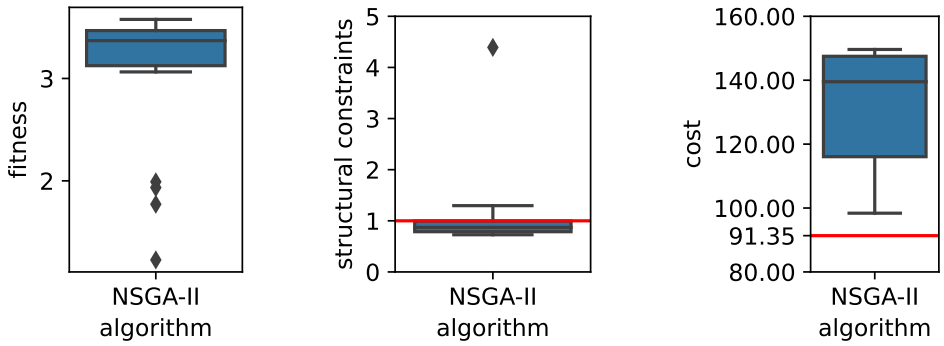


Figure 3.46: Cost achieved by NSGA-II in the tens of thousands of euros. The line in full is the average of 20 runs and the shaded is the average ± standard deviation of the best individual per evaluation. The maximum y value was limited.



(a) Fitness. (b) Structural constraints. (c) Cost.

Figure 3.47: Boxplots created with the best individual of each of the 20 NSGA-II runs. The cost is presented in the tens of thousands. The red line in the cost boxplot is the cost of the baseline solution and in the structural constraints plot is the threshold of safety (1).

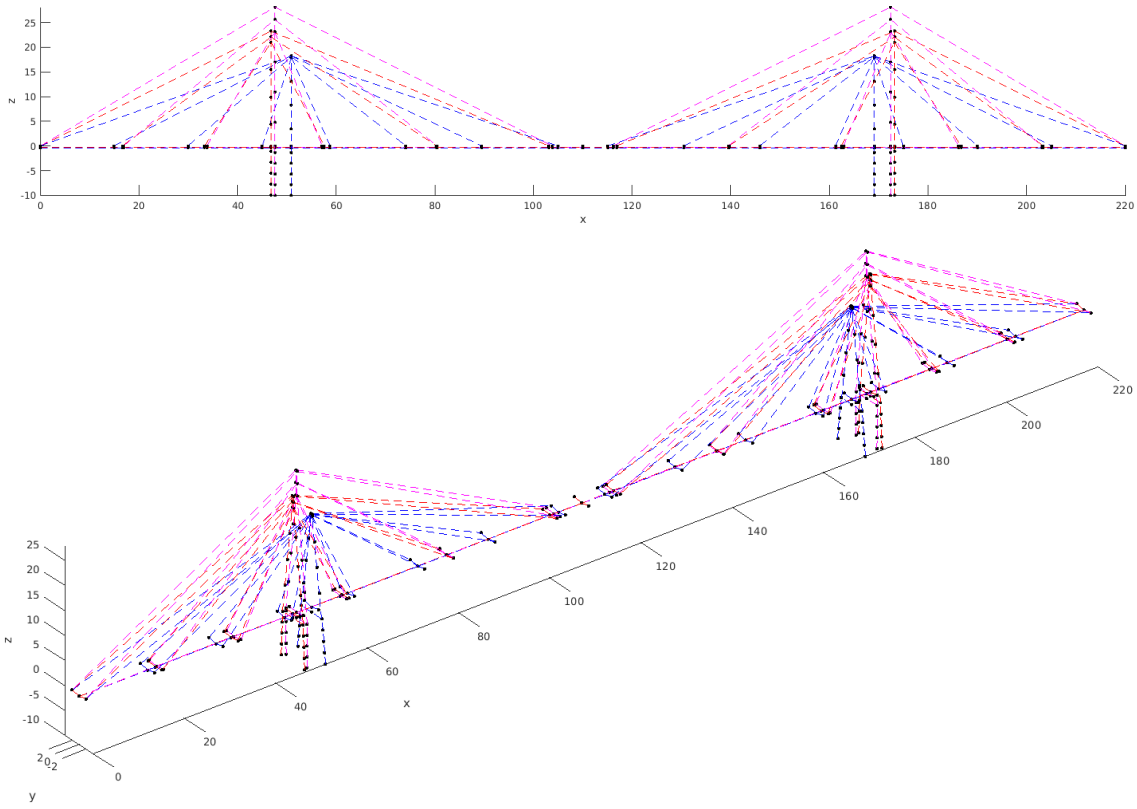
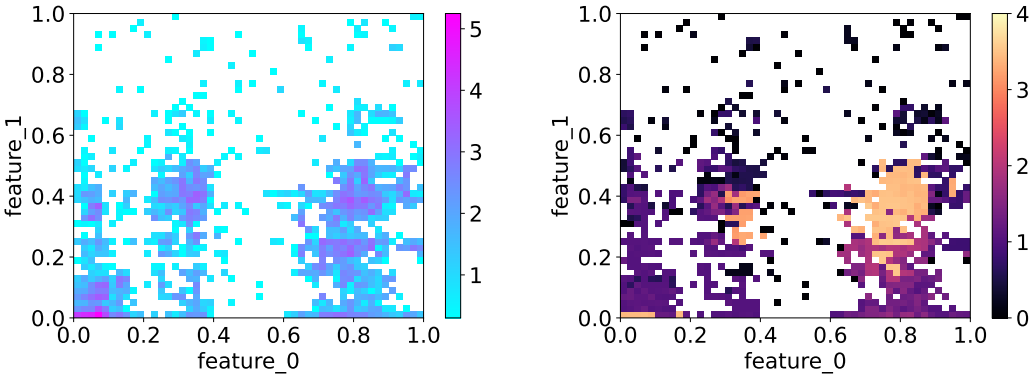


Figure 3.48: Graphical comparison of the baseline (blue), the best (pink), and the cheapest (red) from the NSGA-II.

The activity plot (Fig. 3.49a) tells us that the algorithm spent most of the time in the same regions, which resulted in a poor exploration of the feature space, seen by a large amount of empty (white) cells. Despite the low exploration, the algorithm was still able to discover several distinct feasible individuals, seen by the orange/yellow portion of the filled cells in the fitness heatmap of Fig. 3.49b. The remainder of the fitness heatmaps and activity plots concerning this seed are shown in Appendix A, section A.7.

The plot concerning the number of elites in the archive over time, Fig. 3.50a, is particularly interesting because it differs the most from the similar plots for the other approaches in terms of shape. The line is shaped in a logarithmic style, instead of the straight line style, indicating that the algorithm was reaching its peak in terms of diversity. We can also see that there is some variability between runs, by the increasing of the shaded area towards the end of the evolution. Aside from the shape differences, we also see that the average number of elites in the archive is very low, in concrete, only around 3 000 individuals out of the 400 000 created were saved in the archive. By analyzing the fitness distribution of Fig. 3.50b we notice that from the few individuals stored in the archive, most are unfeasible solutions (fitness below 3).

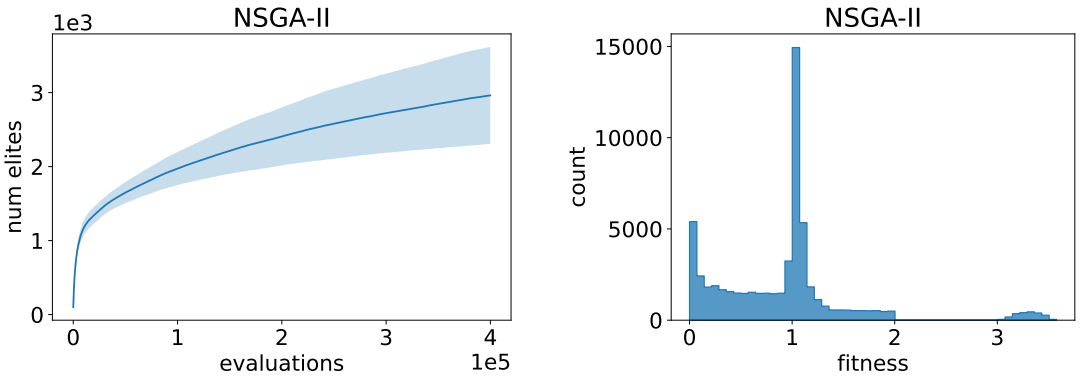
By examining the information in Table 3.10 we can see that the algorithm had difficulty finding feasible solutions and the ones that it managed to find are not competitive with the baseline. The individuals added to the archive account for less than 1% of the created individuals.



(a) Activity plot.

(b) Fitness heatmap.

Figure 3.49: On the right side, we present a projection of features 0 and 1 from the grid archive, showcasing the fitness results obtained by the best NSGA-II seed. On the left side, we have an activity plot that provides insights into the regions that were more explored. The activity plot employs a logarithmic scale with a base of 10.



(a) Number of elites.

(b) Fitness distribution.

Figure 3.50: Plots concerning only the individuals added to the grid archive. On the left, the number of elites in the archive is shown per evaluation. The line in full is the average of the 20 seeds and the shaded is the average $\pm$ standard deviation. On the right, we present a histogram of the distribution of fitness in the archives at the end of the executions of the algorithm. All the fitness values in the 20 archives were grouped together and then utilized to create the histogram with 50 bins.

Table 3.10: Diversity statistics for each of the 20 NSGA-II seeds. Floating point values are rounded to 3 decimal points.

seed	num_elites	coverage	qd_score	f	f_arch	b_b	b_b_arch
1498273391	3796	0.009	4581.014	1455	353	0	0
566925520	3587	0.009	4050.353	1035	287	0	0
412576734	3827	0.010	3889.529	838	242	0	0
1430896779	2534	0.006	2624.701	1714	199	0	0
841651272	2217	0.006	2135.847	1349	111	0	0
1487410694	3385	0.008	3175.047	490	139	0	0
2866475345	2448	0.006	2463.373	758	142	0	0
3821237647	3217	0.008	3050.584	450	164	0	0
1953163308	4116	0.010	4475.543	689	335	0	0
426689665	2365	0.006	1775.064	105	30	0	0
1715958619	2894	0.007	2332.767	78	36	0	0
3407815407	2916	0.007	2340.034	101	45	0	0
188789644	3744	0.009	3466.366	112	55	0	0
2578609284	1930	0.005	1300.834	30	18	0	0
3071952282	2502	0.006	1713.484	2	2	0	0
2639351381	3644	0.009	2956.873	1	1	0	0
745460049	3108	0.008	2369.038	0	0	0	0
538106981	2724	0.007	1912.609	0	0	0	0
2943103388	2311	0.006	1532.266	0	0	0	0
1314091562	1986	0.005	1350.948	0	0	0	0
mean	**2962.550**	**0.007**	2674.814	460.350	107.950	0	0

3.4.8 Results Combined

Now that we already analyzed the performance of the algorithms separately, we are going to see how they compare with each other. For that, we will analyze how the algorithms performed in terms of the best individuals found. When we examine the boxplot of Fig. 3.51 we can divide the algorithms into three groups. The first group is solely comprised by the MAP-Elites variant because it exhibits the worst level of optimization. Since we are using the best individuals from each run, we can see that MAP-Elites was not able to find even one feasible solution, that is, an individual with a cost below $c_r = 150$ and a structural constraints value of at most 1.

The second group includes the Hybrid, the GA, and the NSGA-II. These algorithms show a decent amount of optimization in most of the runs executed but do not reach the level of the remaining algorithms.

The third group contains the remaining three algorithms, CMA-ME, ME-MAP-Elites, and CMA-ES. All use instances of CMA-ES under the hood, indicating that the optimization capabilities of CMA-ES is the deciding factor. Within the high-performant algorithms, we can still see that CMA-ME basically ties up with the best individuals discovered by CMA-ES but is more consistent, with all the individuals (the best) scoring higher than 3.5 fitness. ME-MAP-Elites is as consistent as CMA-ME, but falls a little short in raw performance.

The structural constraints values of the best individuals from each algorithm are presented in the boxplots of Fig. 3.52. By looking at the boxplot of Fig. 3.52a we see that the worst algorithm is MAP-Elites, not being able to minimize this

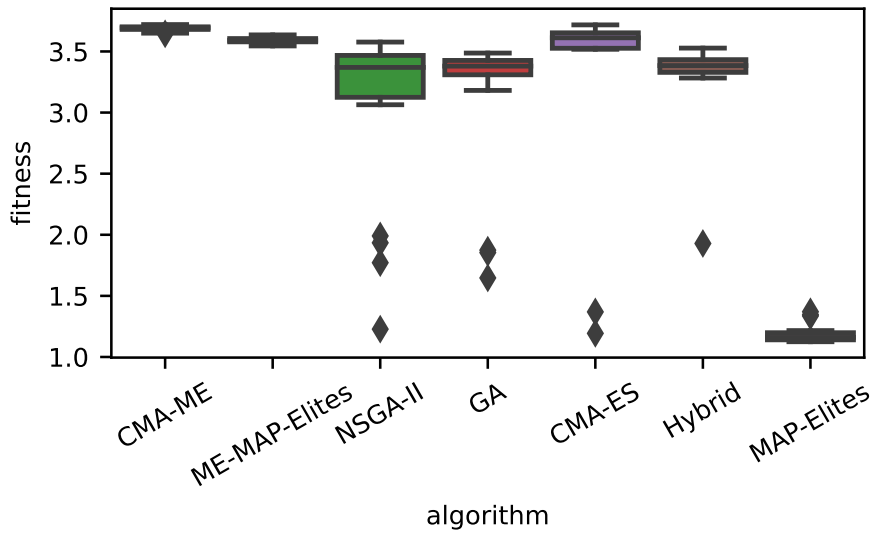
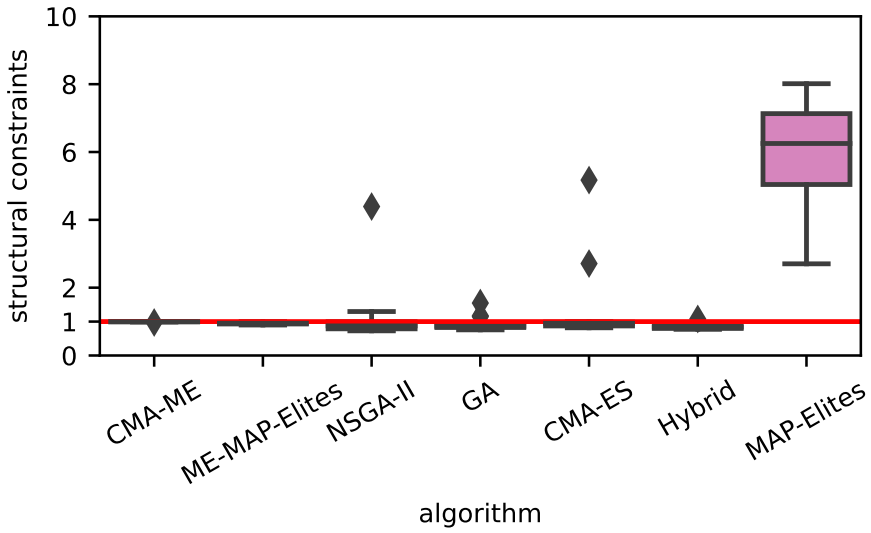


Figure 3.51: Boxplots created with the fitness of the best individuals found in each of the 20 runs executed by the algorithms.

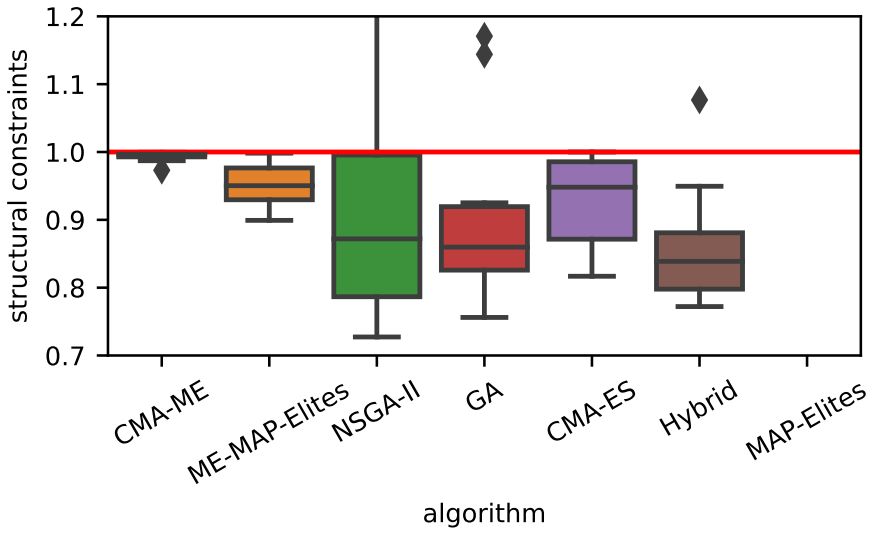
objective. The remaining algorithms were all capable of minimizing it, however, we can see that NSGA-II, GA, CMA-ES, and the Hybrid failed to optimize it in some runs. In the close up of this boxplot, Fig. 3.52b, we can see that the outlier in CMA-ME is from a run that was also able to minimize this objective. Similarly to CMA-ME, ME-MAP-Elites optimized the structural constraints value in all 20 executions of the algorithm but presents higher variability.

If we had only the cost boxplot from Fig. 3.53, we would assume that MAP-Elites was a somewhat competent option, but we showed that this is not the case, we also need to consider the structural constraints values. This only means that MAP-Elites was able to find some individuals priced under c_r, however, they do not meet the structural constraints in order for the bridge to be safe. Besides the case of MAP-Elites we can once again separate the algorithms into groups, but this time in terms of cost. We can see that NSGA-II, the GA, and the Hybrid are able to discover reasonably priced structures, but they fail to beat the cost of the baseline solution. The approaches utilizing instances of CMA-ES demonstrate the best results in terms of cost. CMA-ES by itself has some runs in which it surpassed the baseline, but it is not consistent in doing it. ME-MAP-Elites, although more consistent, failed at finding cheaper bridges than the baseline. CMA-ME on the other hand, only failed to beat the baseline in one instance.

Table 3.11 presents the cost and structural constraints values of the best individuals and the least expensive individuals from all the algorithms that found feasible solutions. Then the differences between these values and the counterparts from the baseline are added to demonstrate how they compare. Since MAP-Elites failed to find feasible solutions, its row is empty. For CMA-ES the best individual is also the cheapest, so only one row was added. Additionally, the column beat_baseline was included to inform the reader how the algorithms performed between runs in terms of beating the baseline.



(a) Structural constraints not limited in the y axis.



(b) Structural constraints limited in the y axis.

Figure 3.52: Boxplots created with the structural constraints of the best individuals found in each of the 20 runs executed by the algorithms. The red line is the threshold of safety (1). A close up was added to shown the behavior of the algorithms around 1.

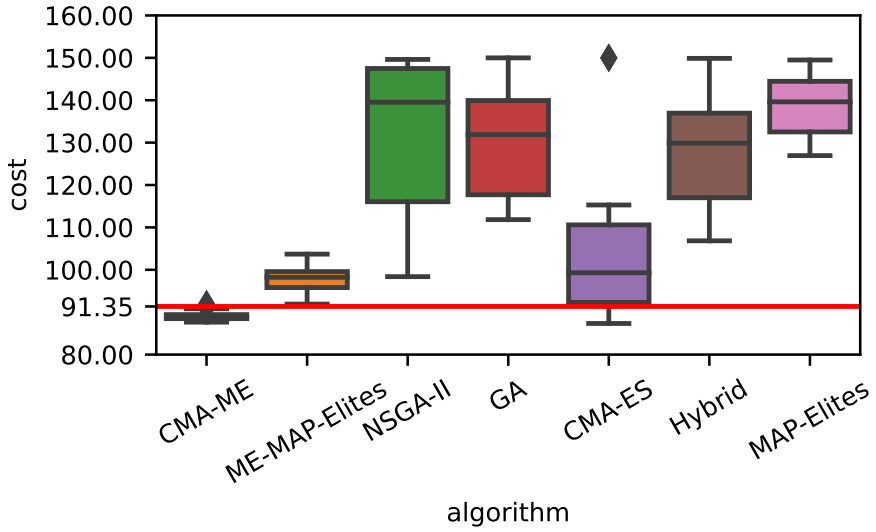


Figure 3.53: Boxplots created with the cost of the best individuals found in each of the 20 runs executed by the algorithms. The cost is presented in tens of thousands. The red line is the cost of the baseline solution.

We see again that only CMA-ES and CMA-ME were capable of beating the baseline solution. The configurations in question present a saving of around 4×10^4 €, which is a considerable amount. The best individuals from the GA and the Hybrid are considerably safer than the baseline. Still, they are much more expensive than it, redeeming their selection almost out of option in most cases, due to the preference for the least expensive options. Graphical comparisons of the structures featured in the table with the baseline are shown in the subsection of the respective algorithm.

Even though CMA-ES was able to surpass the baseline, Table 3.12 shows us that on average the algorithm is relatively far from the baseline cost, losing to ME-MAP-Elites, an algorithm that failed to beat the baseline in the 20 runs of our experiment. This is supported by the variability demonstrated by CMA-ES in the boxplot of Fig. 3.53. CMA-ME, as previously shown in the cost boxplot, easily surpassed the baseline, with the average cost of the best individuals being lower than the one of the baseline. The Hybrid and the GA are very similar, however, the Hybrid found on average the safest structures while still beating the average cost of the GA.

An important part of this work is the study of the diversity presented by each of the approaches tested. By looking at Fig. 3.54 and Table 3.13 we can see that the worst algorithm in terms of diversity is NSGA-II, storing on average in the archive less than 1% of the 400 000 individuals created per run. The number of elites in the grid archive by evaluation presented by the GA and CMA-ES are very similar on average, however, the latter has a significant amount of variability between runs, perceptible by the size of the respective shaded area. Then we have the Hybrid, which consistently added around 25% of the total individuals into the archive. This makes it the worst performing QD algorithm from the ones tested. From the 400 000 individuals created, CMA-ME and ME-MAP-Elites store 50%

Table 3.11: Cost and structural constraints value of the cheapest and best individuals found by each algorithm. The rows in bold represent the cheapest solutions. Floating point values are rounded to 3 decimal points. $S(x)$ is the structural constraints value and $C(x)$ is the cost, which is presented in the tens of thousands. The beat_baseline column shows how many of the runs had a solution that surpassed the baseline (cheaper but still safe).

algorithm	$C(x)$	diff $C(x)$	$S(x)$	diff $S(x)$	beat_baseline
GA	114.147	22.793	0.8527	-0.1435	0/20
	108.352	**16.998**	**0.9915**	**-0.0047**	
CMA-ES	**87.339**	**-4.015**	**1.000**	**0.0038**	4/20
CMA-ME	87.652	-3.702	0.9914	-0.0048	19/20
	87.39	**-3.964**	**0.9994**	**0.0032**	
ME-ME	92.422	1.068	0.9876	-0.0086	0/20
	91.908	**0.554**	**0.9983**	**0.0021**	
Hybrid	111.021	19.667	0.85	-0.1462	0/20
	105.288	**13.934**	**0.981**	**-0.0152**	
NSGA-II	102.66	11.306	0.8963	-0.0999	0/20
	98.114	**6.760**	**0.9972**	**0.0010**	
ME	-	-	-	-	0/20

Table 3.12: Averages of the fitness, cost ($C(x)$) and structural constraints value ($S(x)$) for each algorithm, considering the best individual from each run. Floating point values are rounded to 3 decimal points.

algorithm	fitness	$C(x)$	$S(x)$
GA	3.144	129.930	0.912
CMA-ES	3.376	104.519	1.224
ME	1.188	139.006	5.908
CMA-ME	**3.688**	**89.150**	0.995
ME-ME	3.591	97.657	0.950
Hybrid	3.315	127.911	**0.851**
NSGA-II	3.061	131.487	1.066

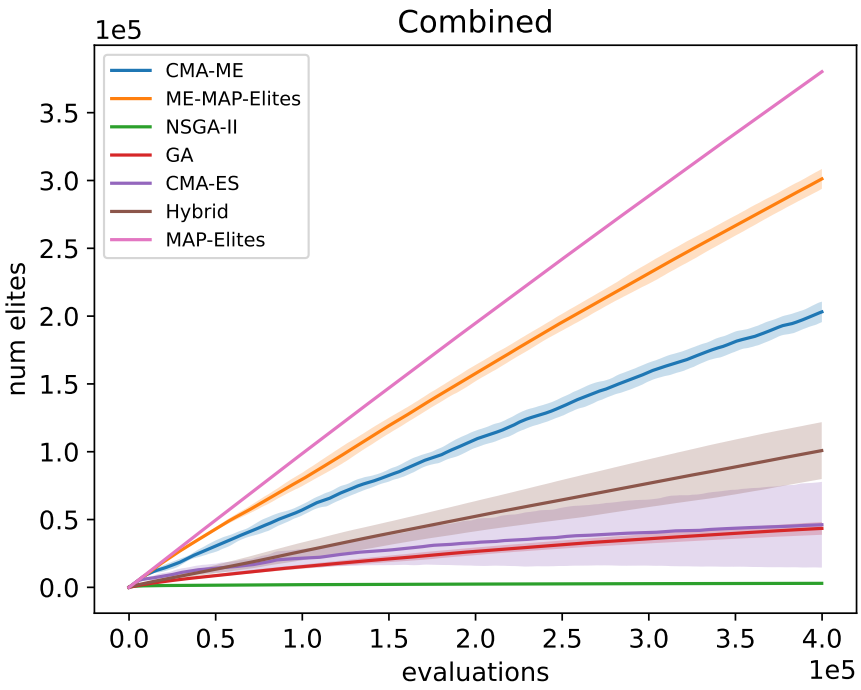
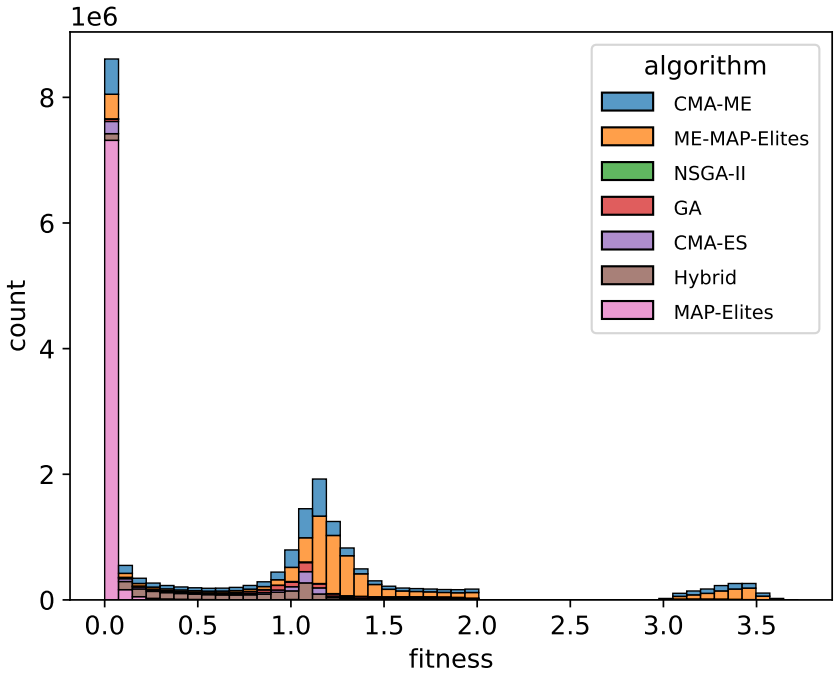


Figure 3.54: The number of elites in the archive during the execution for each algorithm. The line in full is the average of the 20 seeds and the shaded is the average $\pm$ standard deviation.

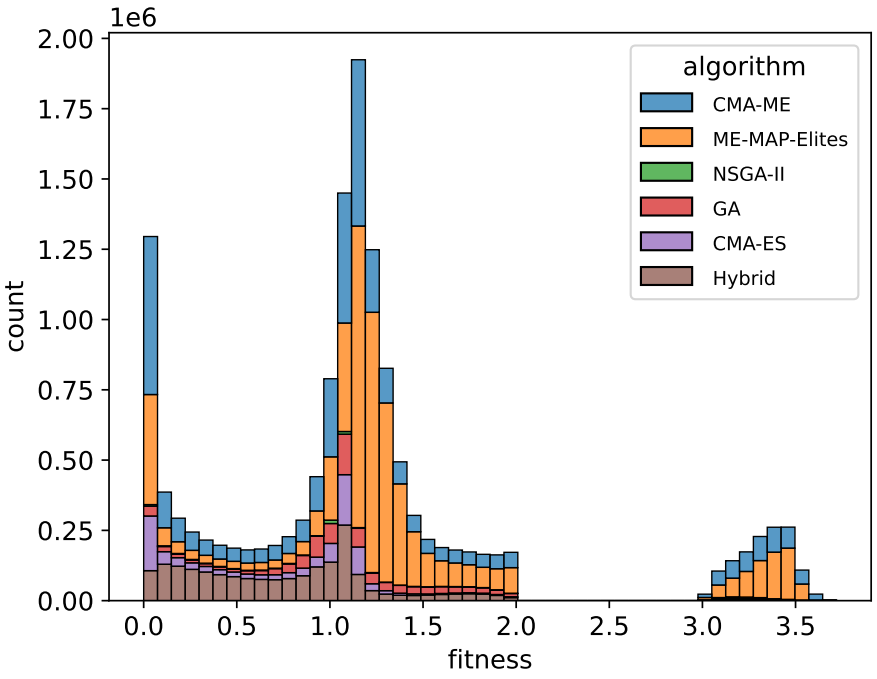
and 75% in the archive in a very consistent manner. However, the best algorithm is MAP-Elites, with an impressive 95% of the created individuals being added to the archive, which means that almost all of the individuals were significantly different from each other.

Another important aspect is understanding the fitness distribution of the individuals stored in the archives. For that we created fitness histograms for all the algorithms, using the aggregation of all the fitness values from the individuals stored in the 20 different archives of each algorithm. These histograms are shown in Fig. 3.55. In Fig. 3.55a we show the histograms from all the algorithms. It can be seen that the archives from MAP-Elites were filled by mostly individuals with fitness close to 0. We then, in Fig. 3.55b, show the same plot but without the data from MAP-Elites, to better demonstrate the results from the other algorithms. Aside from MAP-Elites, all the algorithms present a very similar behavior, summarized by two large peaks near 0 and 1 fitness, and then a small peak after 3 of fitness, which accounts for the feasible portion of the stored individuals. We can see that the algorithms that added more feasible solutions into the archives were CMA-ME and ME-MAP-Elites. The portions of feasible solutions of the remaining options are considerably smaller.

In Table 3.13 we present the average diversity statistics for each of the algorithms. The average number of unique feasible configurations for CMA-ES is a large number, that does not translate into a high qd_score. This is due to what happened in one of the runs, where more than 300 000 individuals found were feasible and considered different in the search space (meaning that the array of genes is different), but are all mapped into a few cells in the feature space, easily per-



(a) Fitness distribution.



(b) Fitness distribution.

Figure 3.55: Plots concerning the individuals in the archive at the end of the run. The data from every run of each algorithm was grouped and then used to create a histogram. The resulting histograms are then stacked on top of each other. Figure (a) shows the data from the 7 algorithms, while figure (b) omits the results from MAP-Elites, to better show the results from the remaining algorithms.

Table 3.13: Averages of the diversity statistics for all the algorithms. Floating point values are rounded to 3 decimal points.

algorithm	coverage	qd_score	f	f_arch	b_b	b_b_arch
GA	0.109	47673.324	20768.200	998.400	0.000	0.000
CMA-ES	0.115	35725.538	109087.000	1388.650	**16290.500**	3.600
ME	**0.951**	5469.435	0.000	0.000	0.000	0.000
CMA-ME	0.508	241562.496	**113854.650**	25827.450	1873.750	**91.550**
ME-ME	0.753	**422310.000**	54010.250	**37442.800**	0.000	0.000
Hybrid	0.252	76415.328	3336.300	712.550	0.000	0.000
NSGA-II	0.007	2674.814	460.350	107.950	0.000	0.000

ceived by the low number of feasible individuals in the archive. The qd_scores of CMA-ME and ME-MAP-Elites are, as expected by observing the fitness histograms, the highest. The latter is able to achieve a higher qd_score than the first, even though it found less feasible solutions. This tells us that ME-MAP-Elites found more diverse individuals in the feature space and not only in the search space, leading to a higher number of high-fit individuals saved in the archive. CMA-ME stands out because on average is able to find more than 90 significantly different solutions in the feature space that are structurally safe and cheaper than the baseline while having a coverage of 50%.

To check if the results from the algorithms are significantly different we performed a statistical analysis in terms of fitness, cost ($C(x)$), structural constraints ($S(x)$), coverage, f_arch (number of feasible individuals in the archive at the end of the execution), and b_b_arch (number of individuals that beat the baseline in the archive at the end of the run). The first three will help us understand if the algorithms differ in terms of raw optimization, and the others are used to gain insight into the magnitude of the differences observed in terms of diversity.

First, we verified if the assumptions for the employment of parametric tests were met by our data, which did not hold. With this in mind, we used the Kruskal-Wallis test for the multiple comparisons. This test's results told us that there are significant differences for all the cases, so we followed it with a post-hoc analysis with the Mann-Whitney test and Bonferroni correction. A significance level of 0.05 was used for every statistical test, however, it was corrected for the post-hoc analysis.

The effect sizes of the pairwise one-tailed Mann-Whitney tests are presented in separate tables, each one concerning one of the aspects of the study mentioned previously. The null hypothesis, H0, is that one is no better than the other, and the alternative hypothesis, H1, is that one is better than the other. In our case, the test is used between the algorithm in the row against the algorithm in the column, which means that if there are significant differences, the algorithm of the row is statistically better than the other. The effect sizes are only calculated for the pairs in which there are significant differences, that is, the statistical test rejected the null hypothesis, and so, the ones in which this is not verified are represented as blank cells in the tables. The effect sizes were computed with the formulas presented in equations 3.2 and 3.3. The values of the effect sizes are presented in the following notation (the negative effect sizes follow the same rules, but with the negative values and use the symbol "-" instead):

Table 3.14: Mann-Whitney effect sizes for the fitness.

	GA	CMA-ES	ME	CMA-ME	ME-ME	Hybrid	NSGA-II
GA			+++				
CMA-ES	+++		+++			+++	+++
ME							
CMA-ME	+++	+++	+++		+++	+++	+++
ME-ME	+++		+++			+++	+++
Hybrid			+++				
NSGA-II			+++				

Table 3.15: Mann-Whitney effect sizes for the cost.

	GA	CMA-ES	ME	CMA-ME	ME-ME	Hybrid	NSGA-II
GA							
CMA-ES	+++		+++			+++	+++
ME							
CMA-ME	+++	+++	+++		+++	+++	+++
ME-ME	+++		+++			+++	+++
Hybrid							
NSGA-II							

- "+" - small effect size ($<= 0.3$)
- "++" - medium effect size (> 0.3 & $<= 0.5$)
- "+++" - large effect size ($>= 0.5$)

$$Z = \frac{U - \frac{n1*n2}{2}}{\sqrt{\frac{n1*n2*(n1+n2+1)}{12}}} \tag{3.2}$$

$$r = Z/\sqrt{n} \tag{3.3}$$

where U is the test statistic, r the effect size, $n1$ and $n2$ are the number of observations of from each sample, and n is the sum of all observations, that is, $n = n1 + n2$.

Before, in the analysis of the fitness boxplots, we mentioned that the algorithms could be separated into three groups. By examining Table 3.14 we can see that those three groups hold up for the most part. The GA, the Hybrid, and the NSGA-II do not present significant differences between each other but do with the rest. MAP-Elites significantly differs from all others, cementing itself as the worst algorithm in terms of optimization. Then in the top performer group, we see that ME-MAP-Elites and CMA-ES are not significantly different. CMA-ME is significantly different from every other algorithm. All the differences presented have a large effect size.

In terms of cost, we observe that CMA-ES and ME-MAP-Elites are significantly better than the GA, the Hybrid and the NSGA-II, however, they are not significantly different from each other. CMA-ME is better than all other algorithms.

The statistical results for the structural constraints shown in Table 3.16 are somewhat different. MAP-Elites was not capable of minimizing this objective while

Table 3.16: Mann-Whitney effect sizes for the structural constraints.

	GA	CMA-ES	ME	CMA-ME	ME-ME	Hybrid	NSGA-II
GA			+++	+++	+++		
CMA-ES			+++				
ME							
CMA-ME			+++				
ME-ME			+++	+++			
Hybrid		+++	+++	+++	+++		
NSGA-II			+++				

Table 3.17: Mann-Whitney effect sizes for the coverage, which is the ratio between the number of elites and the number of individuals created in total (400 000).

	GA	CMA-ES	ME	CMA-ME	ME-ME	Hybrid	NSGA-II
GA							+++
CMA-ES							+++
ME	+++	+++		+++	+++	+++	+++
CMA-ME	+++	+++				+++	+++
ME-ME	+++	+++		+++		+++	+++
Hybrid	+++	+++					+++
NSGA-II							

the remaining could, and the statistical tests confirm that all the algorithms are better than it. For the most part, the algorithms do not present significant differences between each other. The Hybrid is better than CMA-ES, CMA-ME and ME-MAP-Elites. The GA is significantly better than CMA-ME and ME-MAP-Elites. These results also show that ME-MAP-Elites is better than CMA-ME at optimizing this objective.

The statistical tests from Table 3.17 confirm what we could already observe in Fig. 3.54. MAP-Elites is significantly better than the remaining algorithms. Then comes ME-MAP-Elites, CMA-ME, the Hybrid, and CMA-ES and the GA. NSGA-II is the worst algorithm in this matter.

In the case of the number of feasible solutions in the archive at the end of the execution of the algorithm, Table 3.18 tells us that MAP-Elites is worst than every other algorithm, as expected, given that it did not find a single feasible solution. The second worst algorithm is NSGA-II, only beating MAP-Elites. ME-MAP-Elites and CMA-ME are better than the remaining but not better than each other. CMA-ES, besides the two worst algorithms in this matter, also beats the Hybrid.

Table 3.18: Mann-Whitney effect sizes for f_arch, which is the number of unique feasible individuals in the archive at the end of the run.

	GA	CMA-ES	ME	CMA-ME	ME-ME	Hybrid	NSGA-II
GA			+++				+++
CMA-ES			+++			+++	+++
ME							
CMA-ME	+++	+++	+++			+++	+++
ME-ME	+++	+++	+++			+++	+++
Hybrid			+++				+++
NSGA-II			+++				

Table 3.19: Mann-Whitney effect sizes for b_b_arch, which is the number of unique individuals that beat the baseline in the archive at the end of the run.

	GA	CMA-ES	ME	CMA-ME	ME-ME	Hybrid	NSGA-II
GA							
CMA-ES							
ME							
CMA-ME	+++	+++	+++		+++	+++	+++
ME-ME							
Hybrid							
NSGA-II							

Since only CMA-ES and CMA-ME were able to beat the baseline, we are only interested in the tests concerning these two algorithms. By looking at Table 3.19 we can see that, even though CMA-ES was able to beat the baseline it was not enough for it to be considered significantly different that the others. CMA-ME however, differed significantly from the rest.

To conclude, CMA-ES was able to find the least expensive bridge but lacks consistency. On the other hand, MAP-Elites presented the best results in terms of diversity but at the expense of the optimization performance. The GA, NSGA-II, and the Hybrid demonstrated similar mediocre results in terms of optimization and diversity. ME-MAP-Elites and CMA-ME are the best all-rounders but the first is better at finding diversity and the second at optimizing the structures. Overall, CMA-ME seems to be the best option, consistently succeeding at beating the baseline while maintaining very acceptable levels of diversity. These two QD algorithms show that one can search for diversity without compromising the optimization performance.

Chapter 4

Conclusion

In this work, we addressed the structural optimization of Cable-Stayed Bridges (CSBs), more specifically cable-stayed footbridges, with Quality Diversity (QD) algorithms. The optimization of these types of structures is challenging, due to the fact that the design variables are not independent of each other, meaning that a modification of one of them can cause the value of another to become unfit, which may result in an unfeasible individual. With the utilization of QD techniques, we aim to create a set of diverse bridge configurations, thus possibly reducing the time that it takes to discover novel solutions when compared with non-QD approaches. This would more rapidly allow the user to choose the configuration that better suits his preferences.

One of the goals was to research what has been done in the fields of Quality Diversity and structural optimization of Cable-Stayed Bridges. We concluded that QD algorithms have been successfully utilized in a variety of tasks and that the optimization of CSBs is mainly done through gradient-based approaches and Genetic Algorithms (GAs). This indicates that experimenting with this type of algorithm in this problem is reasonable.

Based on our research, we selected the following QD algorithms: Multi-dimensional Archive of Phenotypic Elites (MAP-Elites), Covariance Matrix Adaptation MAP-Elites (CMA-ME), Multi-Emitter MAP-Elites (ME-MAP-Elites) and a Hybrid algorithm from [Vinhas et al., 2016]. The GA and Covariance Matrix Adaptation Evolution Strategy (CMA-ES) were used as control samples, to help us understand the significance of the results gathered with the QD algorithms. We added Non-dominated Sorting Genetic Algorithm II (NSGA-II) because it is a multi-objective algorithm and since we are minimizing two objectives, we wanted to see how it fares against the other algorithms.

Concerning optimization performance alone, CMA-ES was able to find the least expensive feasible structure, closely followed by CMA-ME, however, it lacked consistency in finding individuals that beat the baseline in between runs. MAP-Elites was the best algorithm in terms of diversity, on average adding 95% of the created individuals to the grid archive, but the performance of those individuals is very poor. It ended up not being able to find even one feasible individual in the 20 runs combined. The GA and CMA-ES demonstrated similar results in terms of

diversity, adding around 10% of the created individuals into the archive. When compared to the GA, CMA-ES achieved better results, as it surpassed the baseline in 4 different runs, while the GA could not. The NSGA-II and the Hybrid demonstrated similar optimization results to the GA, but the first was the worst algorithm in terms of diversity, adding less than 1% of the created individuals to the archive, whilst the Hybrid added on average 25%. From the algorithms evaluated, two stood out, namely ME-MAP-Elites and CMA-ME, showing consistently good results both in diversity and optimization. While ME-MAP-Elites presented more diversity (75% against 50%), CMA-ME presented higher levels of optimization, surpassing the baseline in 19 instances, while ME-MAP-Elites failed to do so. With this in mind, we conclude that CMA-ME was the best algorithm, finding better individuals than the baseline while showing a good level of diversity.

In future work, one may want to extend this study by using a different feature space for the QD algorithms. It would also be interesting to develop a parametric study. For instance, increasing the number of emitters used by CMA-ME and ME-MAP-Elites, and verifying how this change impacts their performances. Also, using a different configuration for the bridges would be helpful to assess how this approach generalizes. This could be achieved through the optimization of structures of different lengths or with more options for the number of cables.

References

Maxime Allard, Simón C. Smith, Konstantinos I. Chatzilygeroudis, and Antoine Cully. Hierarchical quality-diversity for online damage recovery. *CoRR*, abs/2204.05726, 2022. doi: 10.48550/arXiv.2204.05726. URL https://doi.org/10.48550/arXiv.2204.05726.

A. Baldomir, S. Hernandez, F. Nieto, and J.A. Jurado. Cable optimization of a long span cable stayed bridge in La Coruña (Spain). *Advances in Engineering Software*, 41(7-8):931–938, jul 2010. ISSN 0965-9978. doi: 10.1016/J.ADVENGSOFT.2010.05.001.

A. Baldomir, I. Kusano, S. Hernandez, and J.A. Jurado. A reliability study for the messina bridge with respect to flutter phenomena considering uncertainties in experimental and numerical data. *Computers & Structures*, 128:91–100, nov 2013. ISSN 0045-7949. doi: 10.1016/J.COMPSTRUC.2013.07.004.

J. Blank and K. Deb. pymoo: Multi-objective optimization in python. *IEEE Access*, 8:89497–89509, 2020.

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

João Correia and Fernando Ferreira. Designing cable-stayed bridges with genetic algorithms. In Pedro A. Castillo, Juan Luis Jiménez Laredo, and Francisco Fernández de Vega, editors, *Applications of Evolutionary Computation - 23rd European Conference, EvoApplications 2020, Held as Part of EvoStar 2020, Seville, Spain, April 15-17, 2020, Proceedings*, volume 12104 of *Lecture Notes in Computer Science*, pages 228–243. Springer, 2020. doi: 10.1007/978-3-030-43722-0_15. URL https://doi.org/10.1007/978-3-030-43722-0_15.

João Correia, Fernando Ferreira, and Catarina Maçãs. Cable-stayed bridge optimization solution space exploration. In Carlos Artemio Coello Coello, editor, *GECCO '20: Genetic and Evolutionary Computation Conference, Companion Volume, Cancún, Mexico, July 8-12, 2020*, pages 261–262. ACM, 2020. doi: 10.1145/3377929.3390033. URL https://doi.org/10.1145/3377929.3390033.

Victor Costa, Nuno Lourenço, João Correia, and Penousal Machado. Exploring the evolution of gans through quality diversity. *CoRR*, abs/2007.06251, 2020. URL https://arxiv.org/abs/2007.06251.

Antoine Cully. Multi-emitter map-elites: Improving quality, diversity and convergence speed with heterogeneous sets of emitters. *CoRR*, abs/2007.05352, 2020. URL https://arxiv.org/abs/2007.05352.

Kalyanmoy Deb, Samir Agrawal, Amrit Pratap, and T. Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Trans. Evol. Comput.*, 6 (2):182–197, 2002. doi: 10.1109/4235.996017. URL https://doi.org/10.1109/4235.996017.

Stéphane Doncieux, Alban Laflaquière, and Alexandre Coninx. Novelty search: a theoretical perspective. In Anne Auger and Thomas Stützle, editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2019, Prague, Czech Republic, July 13-17, 2019*, pages 99–106. ACM, 2019. doi: 10.1145/3321707.3321752. URL https://doi.org/10.1145/3321707.3321752.

Gabriel Fernandes, Nuno Lourenço, and João Correia. Reducing the price of stable cable stayed bridges with cma-es, 2023.

Fernando Ferreira and Luis Simoes. Optimum design of a controlled cable stayed bridge subject to earthquakes. *Structural and Multidisciplinary Optimization*, 44 (4):517–528, oct 2011. ISSN 1615-147X. doi: 10.1007/s00158-011-0628-9.

Fernando Ferreira and Luís Simões. Optimum cost design of controlled cable stayed footbridges. *Computers & Structures*, 106-107:135–143, sep 2012. ISSN 0045-7949. doi: 10.1016/J.COMPSTRUC.2012.04.013.

Fernando Ferreira and Luís Simões. Optimum design of a cable-stayed steel footbridge with three dimensional modelling and control devices. *Engineering Structures*, 180:510–523, feb 2019a. ISSN 01410296. doi: 10.1016/j.engstruct.2018.11.038. URL https://linkinghub.elsevier.com/retrieve/pii/S0141029618314275.

Fernando Ferreira and Luís Simões. Optimum Design of a Controlled Cable-Stayed Footbridge Subject to a Running Event Using Semiactive and Passive Mass Dampers. *Journal of Performance of Constructed Facilities*, 33(3):04019025, jun 2019b. ISSN 0887-3828. doi: 10.1061/(ASCE)CF.1943-5509.0001285.

Fernando Ferreira and Luis Simões. Least cost design of curved cable-stayed footbridges with control devices. 19:68–83, 06 2019. doi: 10.1016/j.istruc.2018.12.004.

Stefano Fioravanzo and Giovanni Iacca. Evaluating map-elites on constrained optimization problems. *CoRR*, abs/1902.00703, 2019. URL http://arxiv.org/abs/1902.00703.

Manon Flageat and Antoine Cully. Fast and stable map-elites in noisy domains using deep grids. *CoRR*, abs/2006.14253, 2020. URL https://arxiv.org/abs/2006.14253.

Matthew C. Fontaine and Stefanos Nikolaidis. A quality diversity approach to automatically generating human-robot interaction scenarios in shared autonomy. *CoRR*, abs/2012.04283, 2020. URL https://arxiv.org/abs/2012.04283.

Matthew C. Fontaine, Scott Lee, Lisa B. Soros, Fernando de Mesentier Silva, Julian Togelius, and Amy K. Hoover. Mapping hearthstone deck spaces through map-elites with sliding boundaries. *CoRR*, abs/1904.10656, 2019a. URL http://arxiv.org/abs/1904.10656.

Matthew C. Fontaine, Julian Togelius, Stefanos Nikolaidis, and Amy K. Hoover. Covariance matrix adaptation for the rapid illumination of behavior space. *CoRR*, abs/1912.02400, 2019b. URL http://arxiv.org/abs/1912.02400.

Félix-Antoine Fortin, François-Michel De Rainville, Marc-André Gardner, Marc Parizeau, and Christian Gagné. DEAP: Evolutionary algorithms made easy. *Journal of Machine Learning Research*, 13:2171–2175, jul 2012.

Theodoros Galanos, Antonios Liapis, Georgios N. Yannakakis, and Reinhard Koenig. Arch-elites: Quality-diversity for urban design. *CoRR*, abs/2104.08774, 2021. URL https://arxiv.org/abs/2104.08774.

N.J. Gimsing and C.T. Georgakis. *Cable Supported Bridges: Concept and Design*. Wiley, 2011. ISBN 9781119951872. URL https://books.google.pt/books?id=5znYUUUGPZIC.

Nikolaus Hansen. The CMA evolution strategy: A comparing review. In José Antonio Lozano, Pedro Larrañaga, Iñaki Inza, and Endika Bengoetxea, editors, *Towards a New Evolutionary Computation - Advances in the Estimation of Distribution Algorithms*, volume 192 of *Studies in Fuzziness and Soft Computing*, pages 75–102. Springer, 2006. doi: 10.1007/3-540-32494-1_4. URL https://doi.org/10.1007/3-540-32494-1_4.

Nikolaus Hansen and Andreas Ostermeier. Completely derandomized self-adaptation in evolution strategies. *Evol. Comput.*, 9(2):159–195, 2001. doi: 10.1162/106365601750190398. URL https://doi.org/10.1162/106365601750190398.

Mahmoud M. Hassan. *Optimum design of cable-stayed bridges*. Phd, Western Ontario University, 2010.

Mahmoud M. Hassan, Ashraf A. El Damatty, and Ashraf O. Nassef. Database for the optimum design of semi-fan composite cable-stayed bridges based on genetic algorithms. *Structure and Infrastructure Engineering*, 11(8):1054–1068, aug 2015. ISSN 1573-2479. doi: 10.1080/15732479.2014.931976.

M.M. Hassan. Optimization of stay cables in cable-stayed bridges using finite element, genetic algorithm, and B-spline combined technique. *Engineering Structures*, 49:643–654, apr 2013. ISSN 0141-0296. doi: 10.1016/J.ENGSTRUCT.2012.11.036.

David Howard, Thomas Lowe, and Wade Geles. Diversity-based design assist for large legged robots. *CoRR*, abs/2004.08057, 2020. URL https://arxiv.org/abs/2004.08057.

José Á. Jurado, Félix Nieto, Santiago Hernández, and Alejandro Mosquera. Efficient cable arrangement in cable stayed bridges based on sensitivity analysis of aeroelastic behaviour. *Advances in Engineering Software*, 39(9):757–763, sep 2008. ISSN 0965-9978. doi: 10.1016/J.ADVENGSOFT.2007.10.004.

Niels Justesen, Sebastian Risi, and Jean-Baptiste Mouret. Map-elites for noisy domains by adaptive sampling. In Manuel López-Ibáñez, Anne Auger, and Thomas Stützle, editors, *Proceedings of the Genetic and Evolutionary Computation Conference Companion, GECCO 2019, Prague, Czech Republic, July 13-17, 2019*, pages 121–122. ACM, 2019. doi: 10.1145/3319619.3321904. URL https://doi.org/10.1145/3319619.3321904.

Joel Lehman and Kenneth O. Stanley. Exploiting open-endedness to solve problems through the search for novelty. In Seth Bullock, Jason Noble, Richard A. Watson, and Mark A. Bedau, editors, *Proceedings of the Eleventh International Conference on the Synthesis and Simulation of Living Systems, ALIFE 2008, Winchester, United Kingdom, August 5-8, 2008*, pages 329–336. MIT Press, 2008. URL http://mitpress.mit.edu/sites/default/files/titles/alife/0262287196chap43.pdf.

Joel Lehman and Kenneth O. Stanley. Evolving a diversity of virtual creatures through novelty search and local competition. In Natalio Krasnogor and Pier Luca Lanzi, editors, *13th Annual Genetic and Evolutionary Computation Conference, GECCO 2011, Proceedings, Dublin, Ireland, July 12-16, 2011*, pages 211–218. ACM, 2011. doi: 10.1145/2001576.2001606. URL https://doi.org/10.1145/2001576.2001606.

Antonios Liapis, Georgios N. Yannakakis, and Julian Togelius. Sentient world: Human-based procedural cartography - an experiment in interactive sketching and iterative refining. In Penousal Machado, James McDermott, and Adrián Carballal, editors, *Evolutionary and Biologically Inspired Music, Sound, Art and Design - Second International Conference, EvoMUSART 2013, Vienna, Austria, April 3-5, 2013. Proceedings*, volume 7834 of *Lecture Notes in Computer Science*, pages 180–191. Springer, 2013a. doi: 10.1007/978-3-642-36955-1_16. URL https://doi.org/10.1007/978-3-642-36955-1_16.

Antonios Liapis, Georgios N. Yannakakis, and Julian Togelius. Enhancements to constrained novelty search: two-population novelty search for generating game content. In Christian Blum and Enrique Alba, editors, *Genetic and Evolutionary Computation Conference, GECCO '13, Amsterdam, The Netherlands, July 6-10, 2013*, pages 343–350. ACM, 2013b. doi: 10.1145/2463372.2463416. URL https://doi.org/10.1145/2463372.2463416.

Alberto M. B. Martins, Luís M. C. Simões, and João H. J. O. Negrão. Optimization of cable-stayed bridges: A literature survey. *Adv. Eng. Softw.*, 149:102829, 2020. doi: 10.1016/j.advengsoft.2020.102829. URL https://doi.org/10.1016/j.advengsoft.2020.102829.

ALBERTO MB MARTINS, LUÍS MC SIMÕES, JOÃO HJO NEGRÃO, and FERNANDO LS FERREIRA. 30 years'experience on the optimization of cable-stayed bridges. *WIT Transactions on The Built Environment*, 209:57–70, 2022.

Tiago Martins, João Correia, Ernesto Costa, and Penousal Machado. Evolving stencils for typefaces: Combining machine learning, user's preferences and novelty. *Complex.*, 2019:3509263:1–3509263:16, 2019. doi: 10.1155/2019/3509263. URL https://doi.org/10.1155/2019/3509263.

Jean-Baptiste Mouret and Jeff Clune. Illuminating search spaces by mapping elites. *CoRR*, abs/1504.04909, 2015. URL http://arxiv.org/abs/1504.04909.

Félix Nieto, Santiago Hernández, José Á. Jurado, and Alejandro Mosquera. Analytical approach to sensitivity analysis of flutter speed in bridges considering variable deck mass. *Advances in Engineering Software*, 42(4):117–129, apr 2011. ISSN 0965-9978. doi: 10.1016/J.ADVENGSOFT.2010.12.003.

Jørgen Nordmoen, Frank Veenstra, Kai Olav Ellefsen, and Kyrre Glette. Map-elites enables powerful stepping stones and diversity for modular robotics. *CoRR*, abs/2012.04375, 2020. URL https://arxiv.org/abs/2012.04375.

Diego Pérez-Liébana, Cristina Guerrero-Romero, Alexander Dockhorn, Dominik Jeurissen, and Linjie Xu. Generating diverse and competitive play-styles for strategy games. *CoRR*, abs/2104.08641, 2021. URL https://arxiv.org/abs/2104.08641.

Justin K. Pugh, Lisa B. Soros, Paul A. Szerlip, and Kenneth O. Stanley. Confronting the challenge of quality diversity. In Sara Silva and Anna Isabel Esparcia-Alcázar, editors, *Proceedings of the Genetic and Evolutionary Computation Conference, GECCO 2015, Madrid, Spain, July 11-15, 2015*, pages 967–974. ACM, 2015. doi: 10.1145/2739480.2754664. URL https://doi.org/10.1145/2739480.2754664.

Justin K. Pugh, Lisa B. Soros, and Kenneth O. Stanley. Quality diversity: A new frontier for evolutionary computation. *Frontiers Robotics AI*, 3:40, 2016. doi: 10.3389/frobt.2016.00040. URL https://doi.org/10.3389/frobt.2016.00040.

Chen Qin. Optimization of cable-stretching planning in the construction of cable-stayed bridges. *Engineering Optimization*, 19(1):1–20, feb 1992. ISSN 0305-215X. doi: 10.1080/03052159208941217.

Achkan Salehi, Alexandre Coninx, and Stéphane Doncieux. BR-NS: an archiveless approach to novelty search. In Francisco Chicano and Krzysztof Krawiec, editors, *GECCO '21: Genetic and Evolutionary Computation Conference, Lille, France, July 10-14, 2021*, pages 172–179. ACM, 2021. doi: 10.1145/3449639.3459303. URL https://doi.org/10.1145/3449639.3459303.

Luís Miguel C. Simões and João Henrique J. O. Negrão. Optimization of cable-stayed bridges subjected to earthquakes with non-linear behaviour. *Engineering Optimization*, 31(4):457–478, mar 1999. ISSN 0305-215X. doi: 10.1080/03052159908941382.

Yu-Chi Sung, Dyi-Wei Chang, and Eng-Huat Teo. Optimum post-tensioning cable forces of Mau-Lo Hsi cable-stayed bridge. *Engineering Structures*, 28(10):1407–1417, aug 2006. ISSN 0141-0296. doi: 10.1016/J.ENGSTRUCT.2006.01.009.

Bryon Tjanaka, Matthew C. Fontaine, Yulun Zhang, Sam Sommerer, Nathan Dennler, and Stefanos Nikolaidis. pyribs: A bare-bones python library for quality diversity optimization. https://github.com/icaros-usc/pyribs, 2021.

Neil Urquhart and Emma Hart. Optimisation and illumination of a real-world workforce scheduling and routing application via map-elites. *CoRR*, abs/1805.11555, 2018. URL http://arxiv.org/abs/1805.11555.

Vassilis Vassiliades, Konstantinos I. Chatzilygeroudis, and Jean-Baptiste Mouret. Using centroidal voronoi tessellations to scale up the multidimensional archive of phenotypic elites algorithm. *IEEE Trans. Evol. Comput.*, 22(4):623–630, 2018. doi: 10.1109/TEVC.2017.2735550. URL https://doi.org/10.1109/TEVC.2017.2735550.

Omar M. Villanueva, Leonardo Trujillo, and Daniel E. Hernández. Novelty search for automatic bug repair. In Carlos Artemio Coello Coello, editor, *GECCO '20: Genetic and Evolutionary Computation Conference, Cancún Mexico, July 8-12, 2020*, pages 1021–1028. ACM, 2020. doi: 10.1145/3377930.3389845. URL https://doi.org/10.1145/3377930.3389845.

Adriano Vinhas, Filipe Assunção, João Correia, Anikó Ekárt, and Penousal Machado. Fitness and novelty in evolutionary art. In Colin G. Johnson, Vic Ciesielski, João Correia, and Penousal Machado, editors, *Evolutionary and Biologically Inspired Music, Sound, Art and Design - 5th International Conference, EvoMUSART 2016, Porto, Portugal, March 30 - April 1, 2016, Proceedings*, volume 9596 of *Lecture Notes in Computer Science*, pages 225–240. Springer, 2016. doi: 10.1007/978-3-319-31008-4_16. URL https://doi.org/10.1007/978-3-319-31008-4_16.

Vivek R. Warriar, Carmen Ugarte, John R. Woodward, and Laurissa Tokarchuk. Playmapper: Illuminating design spaces of platform games. In *IEEE Conference on Games, CoG 2019, London, United Kingdom, August 20-23, 2019*, pages 1–4. IEEE, 2019. doi: 10.1109/CIG.2019.8848118. URL https://doi.org/10.1109/CIG.2019.8848118.

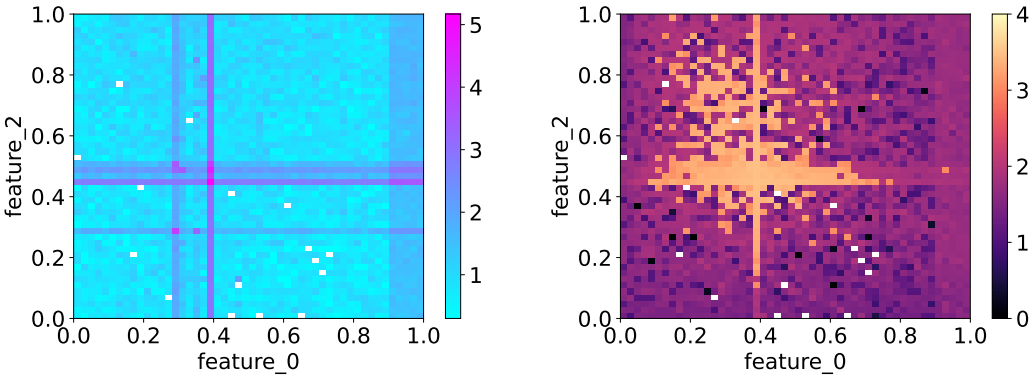
Tianping Zhang, Yuanqi Li, Yifei Jin, and Jian Li. Autoalpha: an efficient hierarchical evolutionary algorithm for mining alpha factors in quantitative investment. *arXiv preprint arXiv:2002.08245*, 2020.

Appendices

Appendix A

Experimental Results

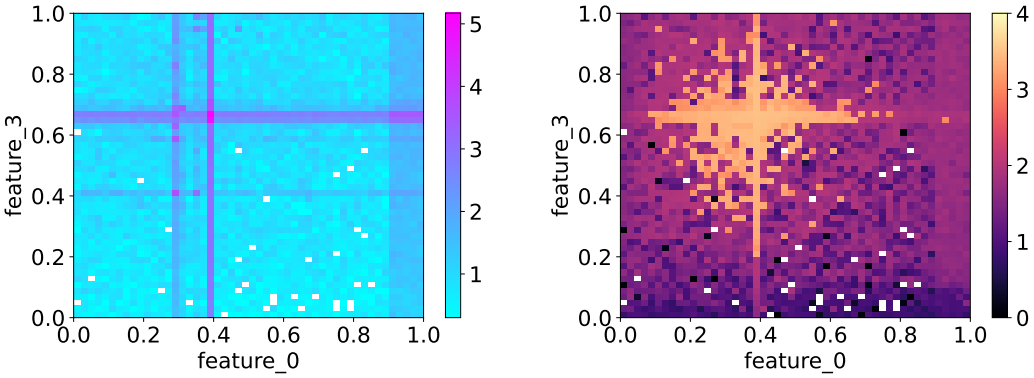
A.1 Genetic Algorithm



(a) Activity plot.

(b) Fitness heatmap.

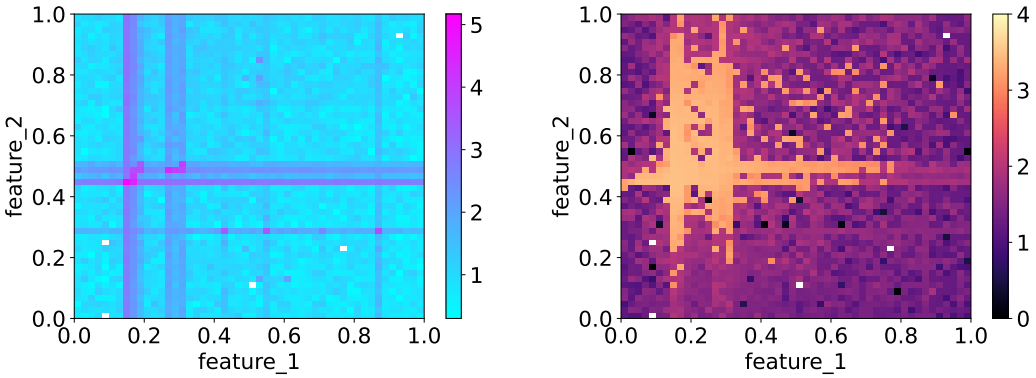
Figure A.1: On the right we have a projection of features 0 and 2 of the grid archive for the best GA seed. On the left is the activity plot, to help us understand the areas that were more searched. The activity plot uses a logarithmic scale of base 10.



(a) Activity plot.

(b) Fitness heatmap.

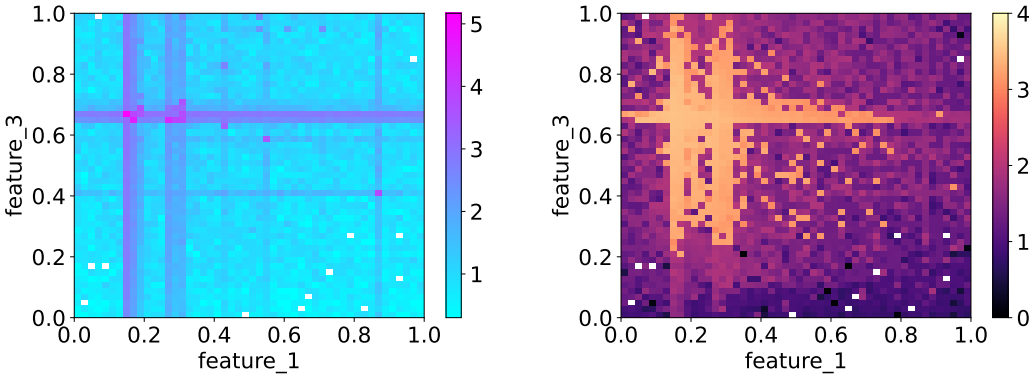
Figure A.2: On the right we have a projection of features 0 and 3 of the grid archive for the best GA seed. On the left is the activity plot, to help us understand the areas that were more searched. The activity plot uses a logarithmic scale of base 10.



(a) Activity plot.

(b) Fitness heatmap.

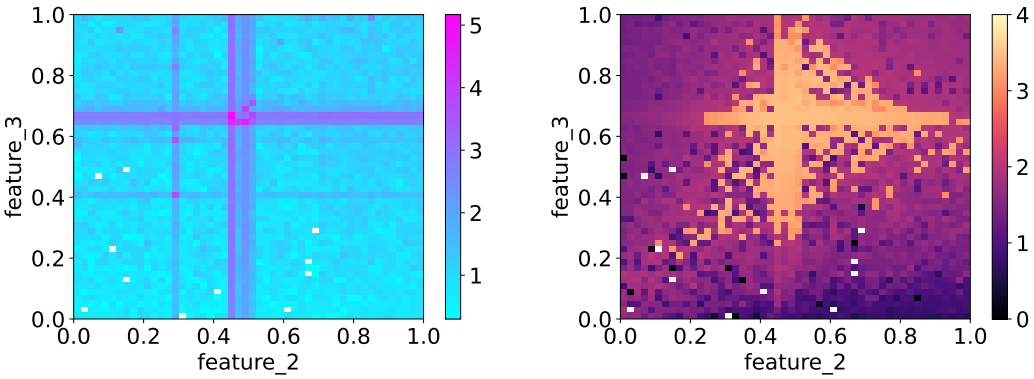
Figure A.3: On the right we have a projection of features 1 and 2 of the grid archive for the best GA seed. On the left is the activity plot, to help us understand the areas that were more searched. The activity plot uses a logarithmic scale of base 10.



(a) Activity plot.

(b) Fitness heatmap.

Figure A.4: On the right we have a projection of features 1 and 3 of the grid archive for the best GA seed. On the left is the activity plot, to help us understand the areas that were more searched. The activity plot uses a logarithmic scale of base 10.

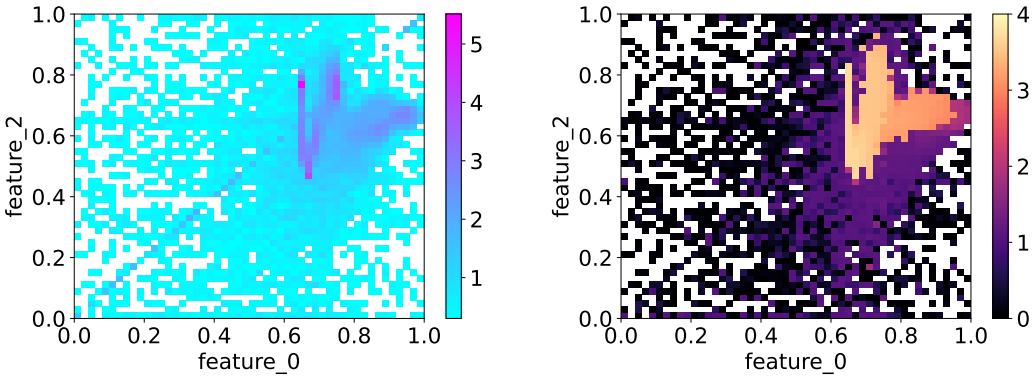


(a) Activity plot.

(b) Fitness heatmap.

Figure A.5: On the right we have a projection of features 2 and 3 of the grid archive for the best GA seed. On the left is the activity plot, to help us understand the areas that were more searched. The activity plot uses a logarithmic scale of base 10.

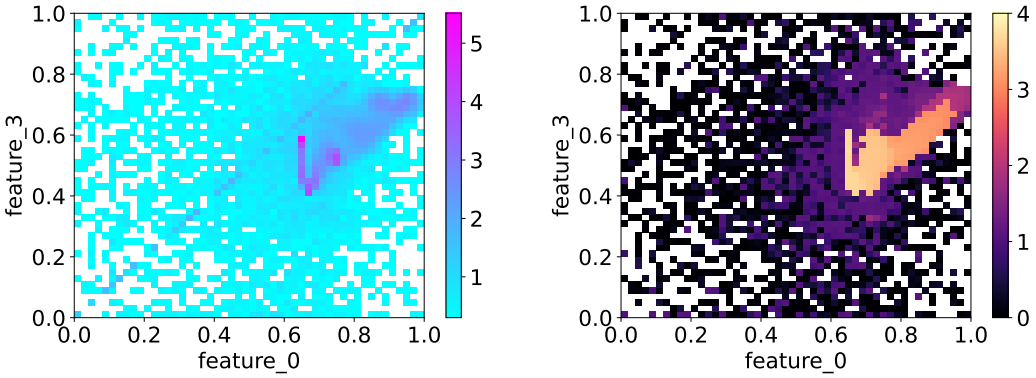
A.2 Covariance Matrix Adaptation Evolution Strategy



(a) Activity plot.

(b) Fitness heatmap.

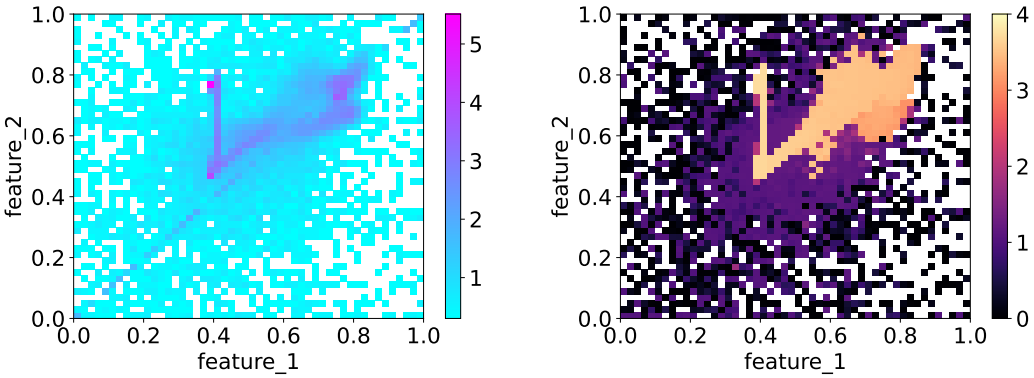
Figure A.6: On the right we have a projection of features 0 and 2 of the grid archive for the best CMA-ES seed. On the left is the activity plot, to help us understand the areas that were more searched. The activity plot uses a logarithmic scale of base 10.



(a) Activity plot.

(b) Fitness heatmap.

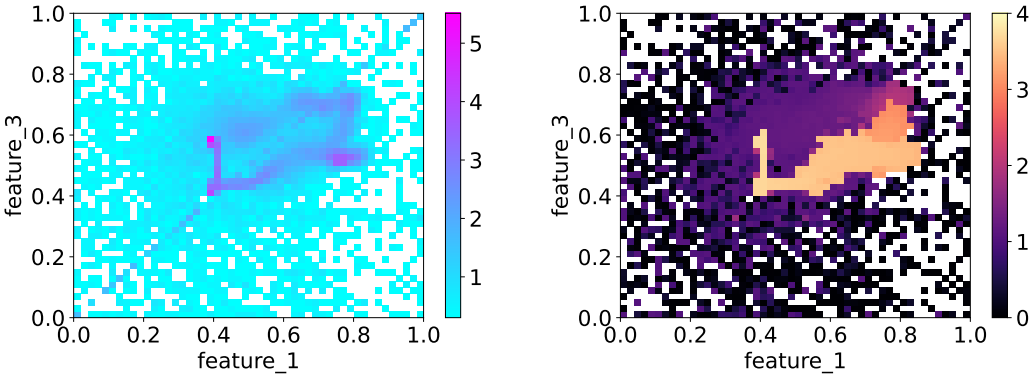
Figure A.7: On the right we have a projection of features 0 and 3 of the grid archive for the best CMA-ES seed. On the left is the activity plot, to help us understand the areas that were more searched. The activity plot uses a logarithmic scale of base 10.



(a) Activity plot.

(b) Fitness heatmap.

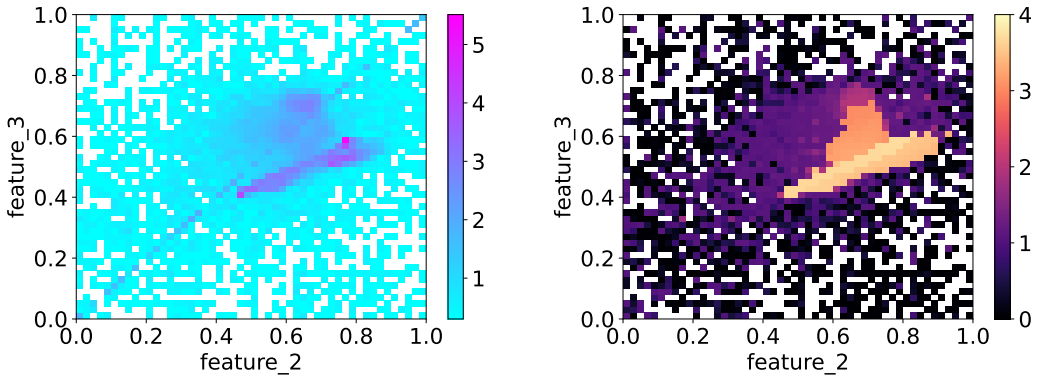
Figure A.8: On the right we have a projection of features 1 and 2 of the grid archive for the best CMA-ES seed. On the left is the activity plot, to help us understand the areas that were more searched. The activity plot uses a logarithmic scale of base 10.



(a) Activity plot.

(b) Fitness heatmap.

Figure A.9: On the right we have a projection of features 1 and 3 of the grid archive for the best CMA-ES seed. On the left is the activity plot, to help us understand the areas that were more searched. The activity plot uses a logarithmic scale of base 10.

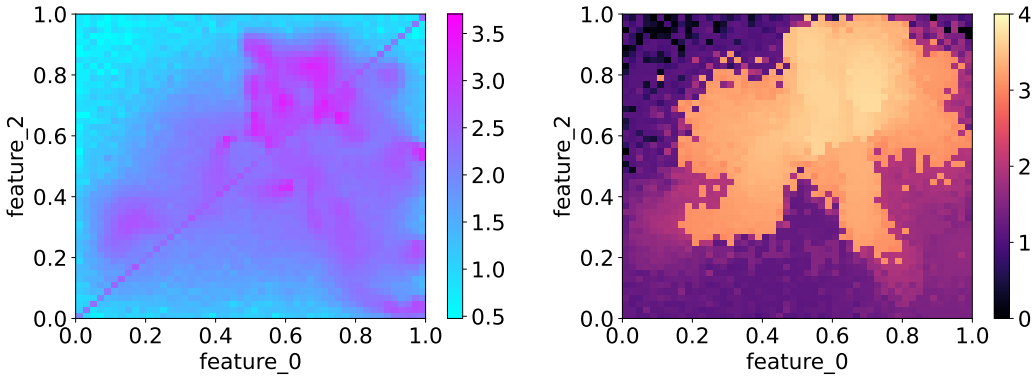


(a) Activity plot.

(b) Fitness heatmap.

Figure A.10: On the right we have a projection of features 2 and 3 of the grid archive for the best CMA-ES seed. On the left is the activity plot, to help us understand the areas that were more searched. The activity plot uses a logarithmic scale of base 10.

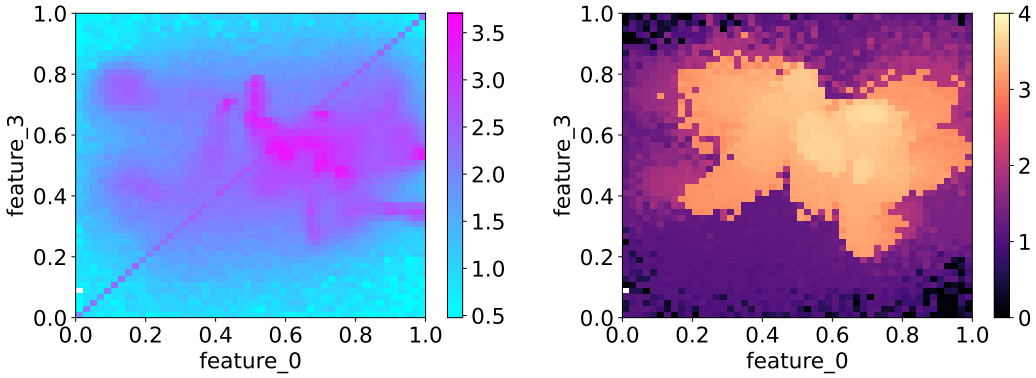
A.3 Covariance Matrix Adaptation Multi-dimensional Archive of Phenotypic Elites



(a) Activity plot.

(b) Fitness heatmap.

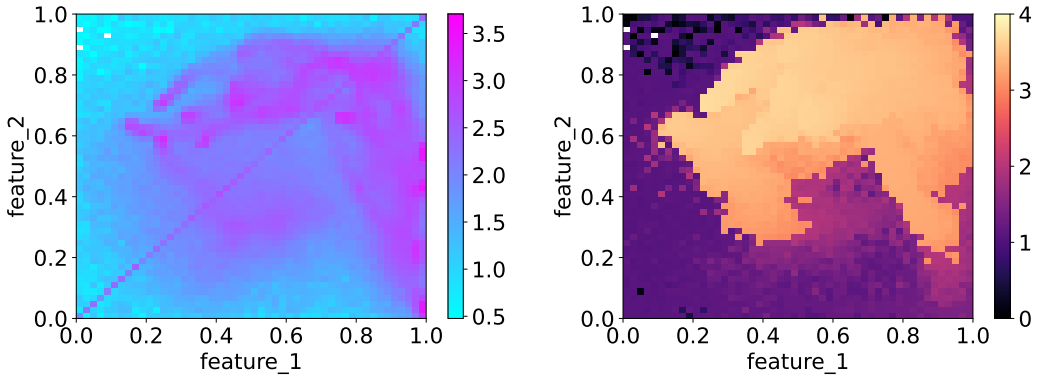
Figure A.11: On the right we have a projection of features 0 and 2 of the grid archive for the best CMA-ME seed. On the left is the activity plot, to help us understand the areas that were more searched. The activity plot uses a logarithmic scale of base 10.



(a) Activity plot.

(b) Fitness heatmap.

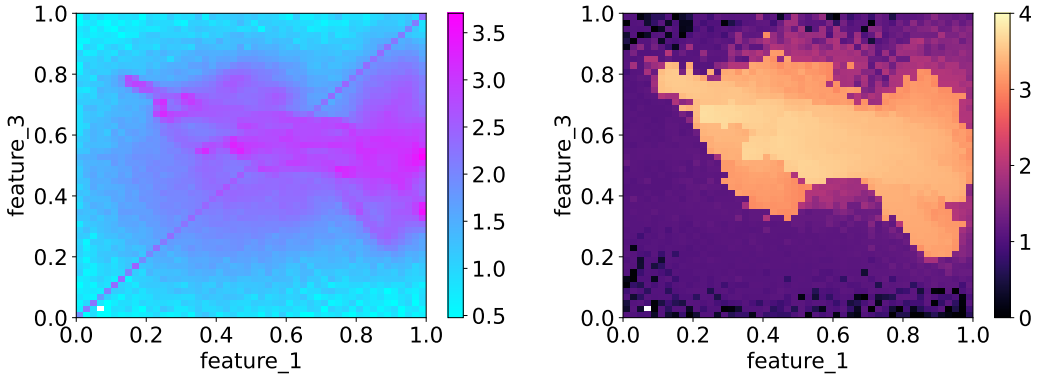
Figure A.12: On the right we have a projection of features 0 and 3 of the grid archive for the best CMA-ME seed. On the left is the activity plot, to help us understand the areas that were more searched. The activity plot uses a logarithmic scale of base 10.



(a) Activity plot.

(b) Fitness heatmap.

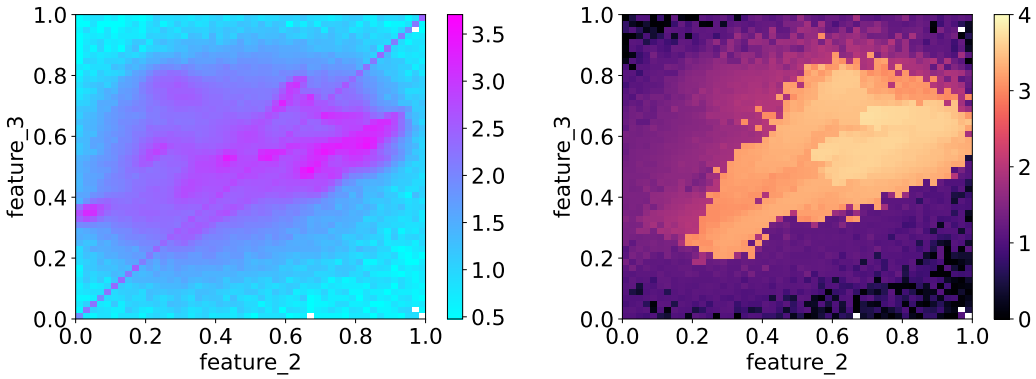
Figure A.13: On the right we have a projection of features 1 and 2 of the grid archive for the best CMA-ME seed. On the left is the activity plot, to help us understand the areas that were more searched. The activity plot uses a logarithmic scale of base 10.



(a) Activity plot.

(b) Fitness heatmap.

Figure A.14: On the right we have a projection of features 1 and 3 of the grid archive for the best CMA-ME seed. On the left is the activity plot, to help us understand the areas that were more searched. The activity plot uses a logarithmic scale of base 10.

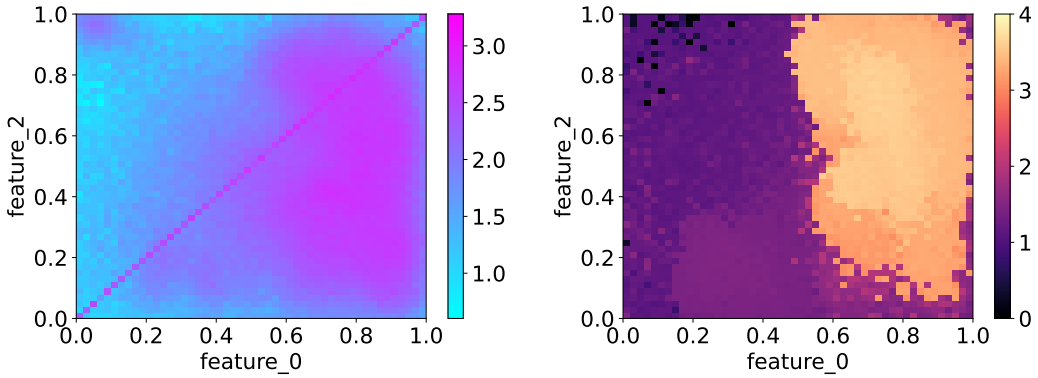


(a) Activity plot.

(b) Fitness heatmap.

Figure A.15: On the right we have a projection of features 2 and 3 of the grid archive for the best CMA-ME seed. On the left is the activity plot, to help us understand the areas that were more searched. The activity plot uses a logarithmic scale of base 10.

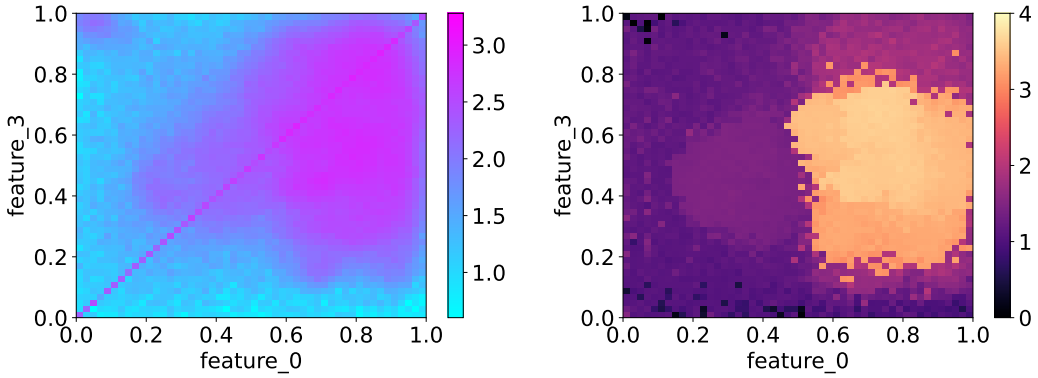
A.4 Multi-Emitter Multi-dimensional Archive of Phenotypic Elites



(a) Activity plot.

(b) Fitness heatmap.

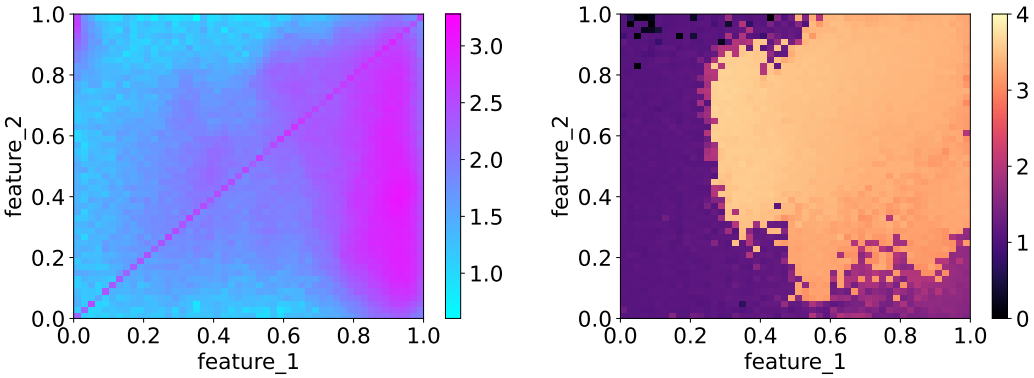
Figure A.16: On the right we have a projection of features 0 and 2 of the grid archive for the best ME-MAP-Elites seed. On the left is the activity plot, to help us understand the areas that were more searched. The activity plot uses a logarithmic scale of base 10.



(a) Activity plot.

(b) Fitness heatmap.

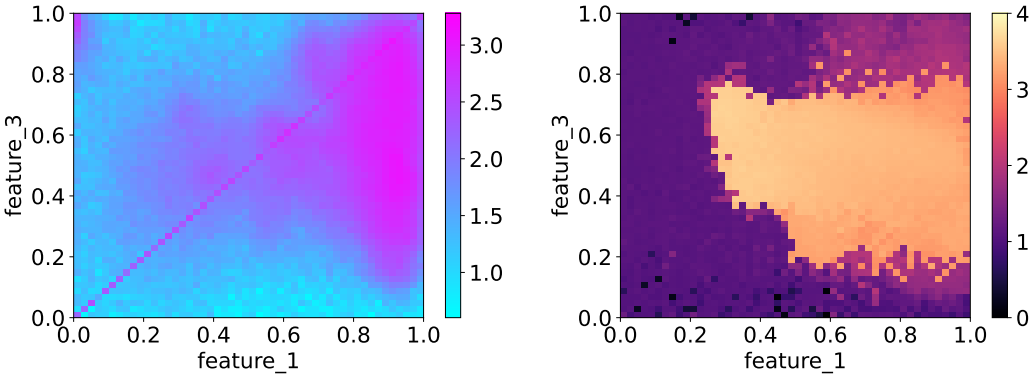
Figure A.17: On the right we have a projection of features 0 and 3 of the grid archive for the best ME-MAP-Elites seed. On the left is the activity plot, to help us understand the areas that were more searched. The activity plot uses a logarithmic scale of base 10.



(a) Activity plot.

(b) Fitness heatmap.

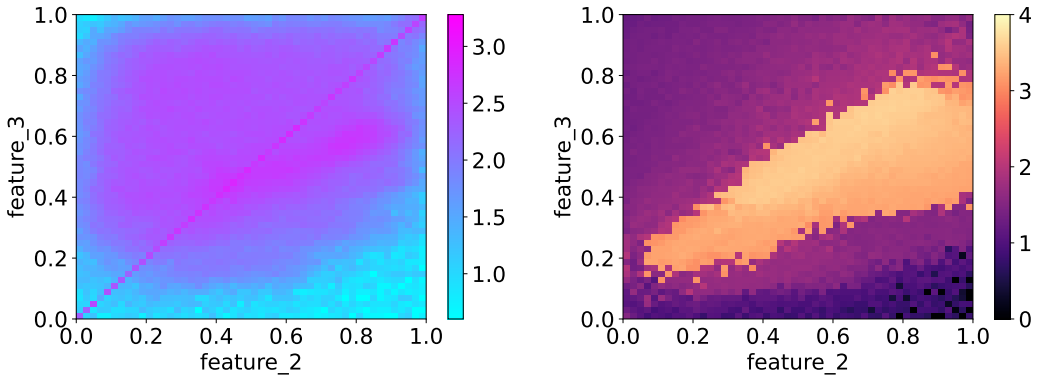
Figure A.18: On the right we have a projection of features 1 and 2 of the grid archive for the best ME-MAP-Elites seed. On the left is the activity plot, to help us understand the areas that were more searched. The activity plot uses a logarithmic scale of base 10.



(a) Activity plot.

(b) Fitness heatmap.

Figure A.19: On the right we have a projection of features 1 and 3 of the grid archive for the best ME-MAP-Elites seed. On the left is the activity plot, to help us understand the areas that were more searched. The activity plot uses a logarithmic scale of base 10.

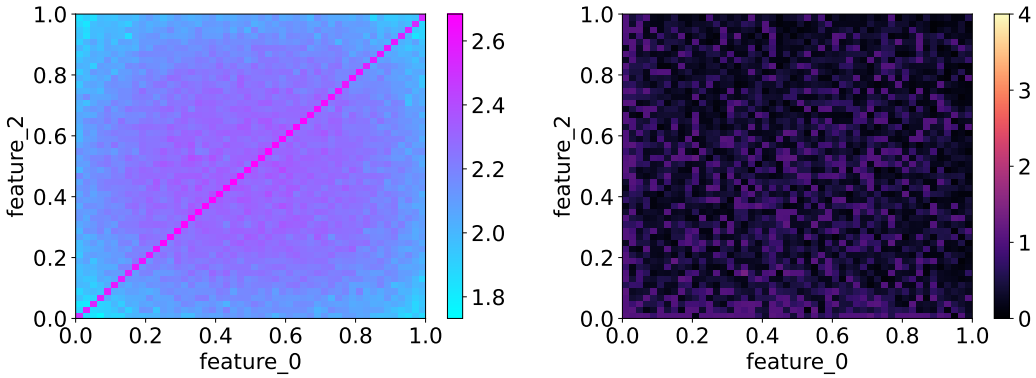


(a) Activity plot.

(b) Fitness heatmap.

Figure A.20: On the right we have a projection of features 2 and 3 of the grid archive for the best ME-MAP-Elites seed. On the left is the activity plot, to help us understand the areas that were more searched. The activity plot uses a logarithmic scale of base 10.

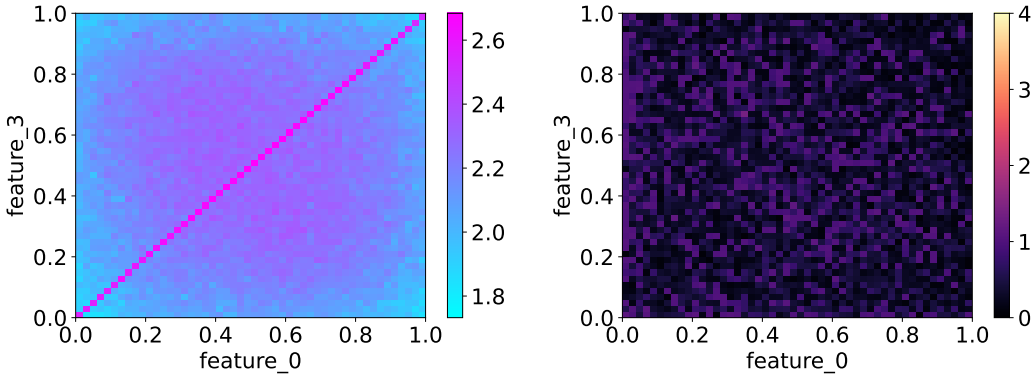
A.5 Multi-dimensional Archive of Phenotypic Elites



(a) Activity plot.

(b) Fitness heatmap.

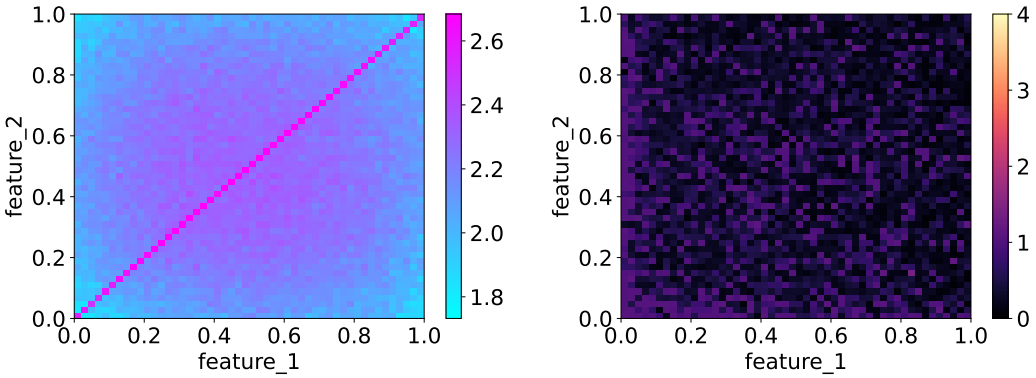
Figure A.21: On the right we have a projection of features 0 and 2 of the grid archive for the best MAP-Elites seed. On the left is the activity plot, to help us understand the areas that were more searched. The activity plot uses a logarithmic scale of base 10.



(a) Activity plot.

(b) Fitness heatmap.

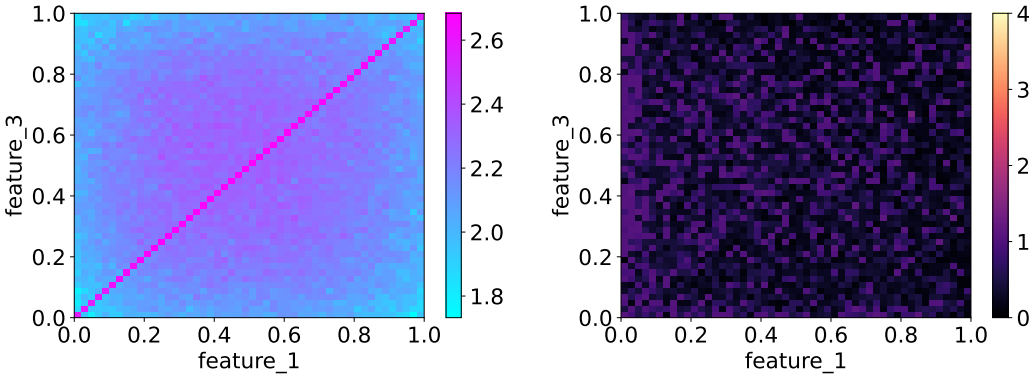
Figure A.22: On the right we have a projection of features 0 and 3 of the grid archive for the best MAP-Elites seed. On the left is the activity plot, to help us understand the areas that were more searched. The activity plot uses a logarithmic scale of base 10.



(a) Activity plot.

(b) Fitness heatmap.

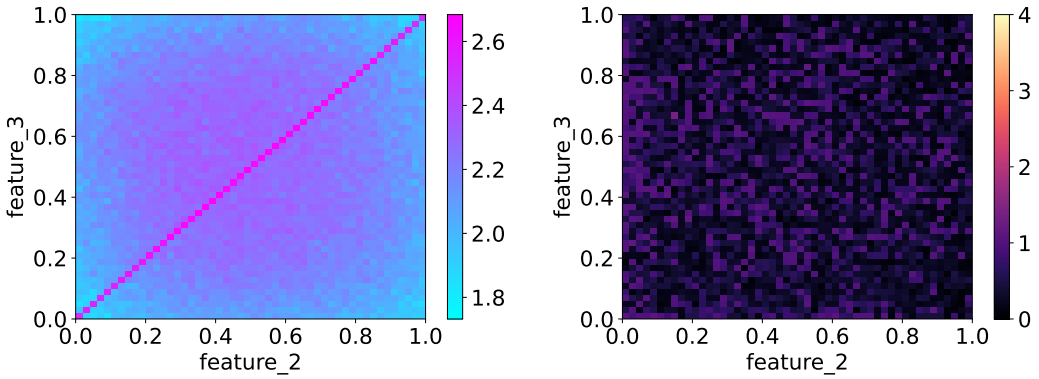
Figure A.23: On the right we have a projection of features 1 and 2 of the grid archive for the best MAP-Elites seed. On the left is the activity plot, to help us understand the areas that were more searched. The activity plot uses a logarithmic scale of base 10.



(a) Activity plot.

(b) Fitness heatmap.

Figure A.24: On the right we have a projection of features 1 and 3 of the grid archive for the best MAP-Elites seed. On the left is the activity plot, to help us understand the areas that were more searched. The activity plot uses a logarithmic scale of base 10.

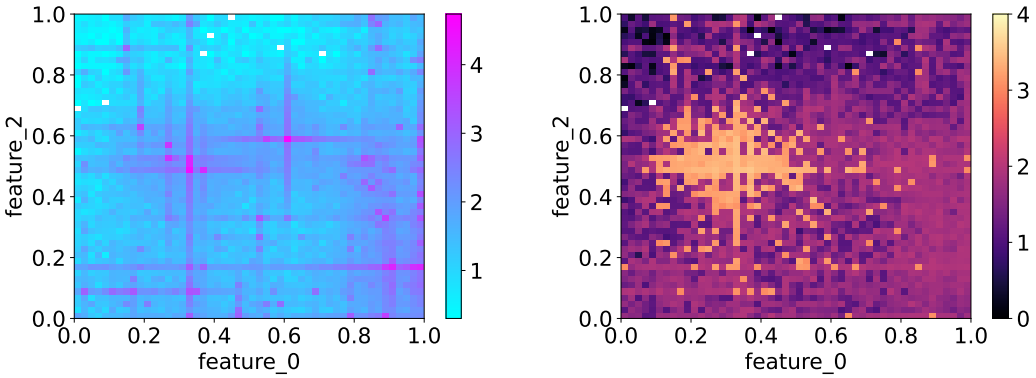


(a) Activity plot.

(b) Fitness heatmap.

Figure A.25: On the right we have a projection of features 2 and 3 of the grid archive for the best MAP-Elites seed. On the left is the activity plot, to help us understand the areas that were more searched. The activity plot uses a logarithmic scale of base 10.

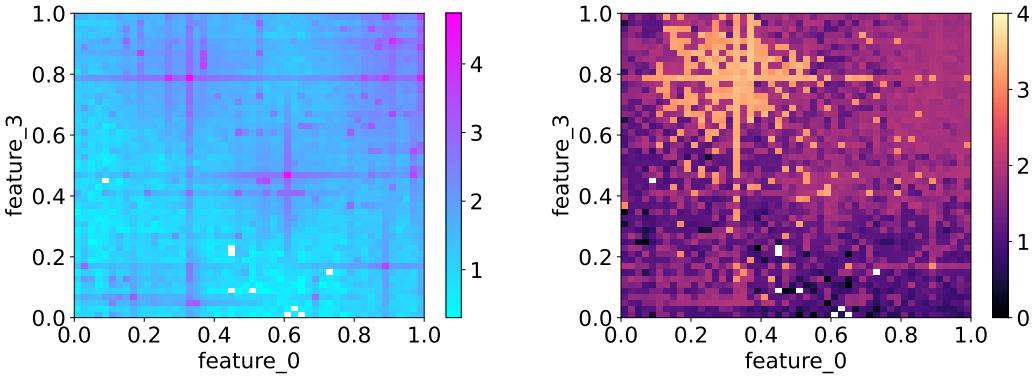
A.6 Hybrid



(a) Activity plot.

(b) Fitness heatmap.

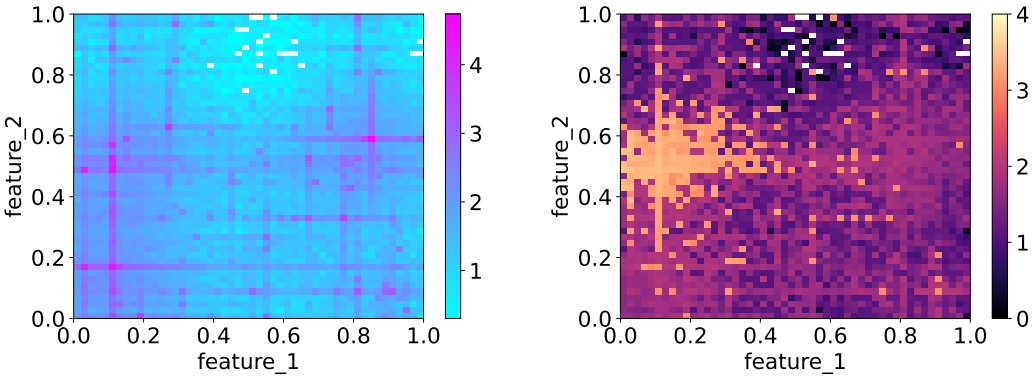
Figure A.26: On the right we have a projection of features 0 and 2 of the grid archive for the best Hybrid seed. On the left is the activity plot, to help us understand the areas that were more searched. The activity plot uses a logarithmic scale of base 10.



(a) Activity plot.

(b) Fitness heatmap.

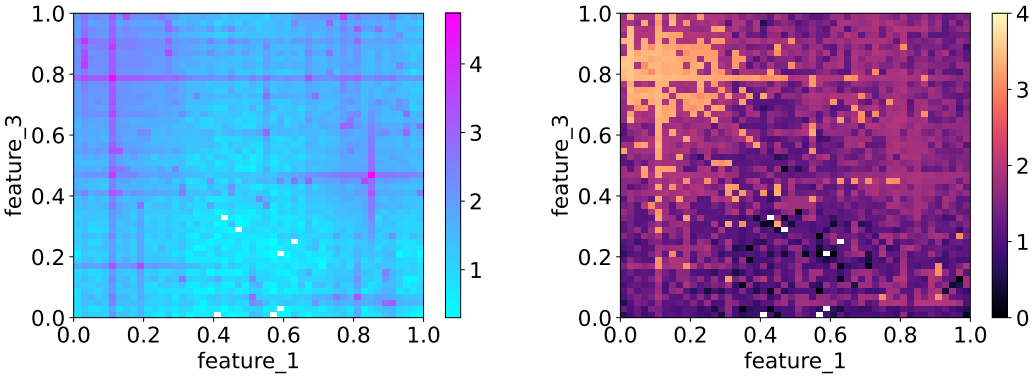
Figure A.27: On the right we have a projection of features 0 and 3 of the grid archive for the best Hybrid seed. On the left is the activity plot, to help us understand the areas that were more searched. The activity plot uses a logarithmic scale of base 10.



(a) Activity plot.

(b) Fitness heatmap.

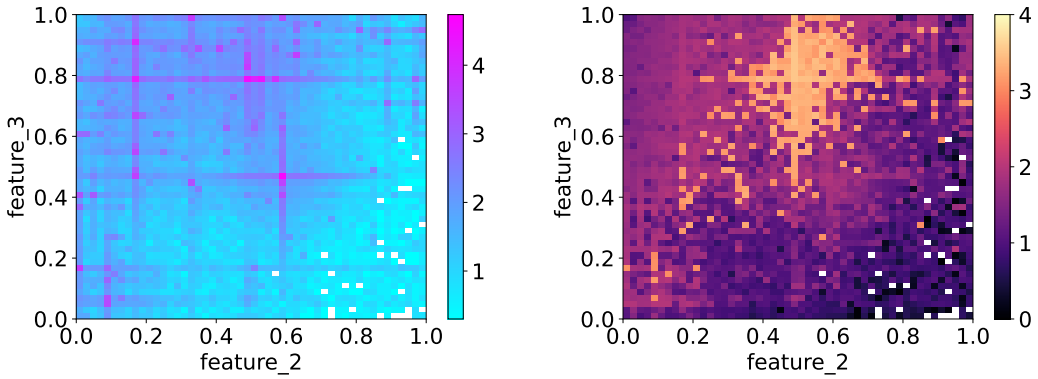
Figure A.28: On the right we have a projection of features 1 and 2 of the grid archive for the best Hybrid seed. On the left is the activity plot, to help us understand the areas that were more searched. The activity plot uses a logarithmic scale of base 10.



(a) Activity plot.

(b) Fitness heatmap.

Figure A.29: On the right we have a projection of features 1 and 3 of the grid archive for the best Hybrid seed. On the left is the activity plot, to help us understand the areas that were more searched. The activity plot uses a logarithmic scale of base 10.

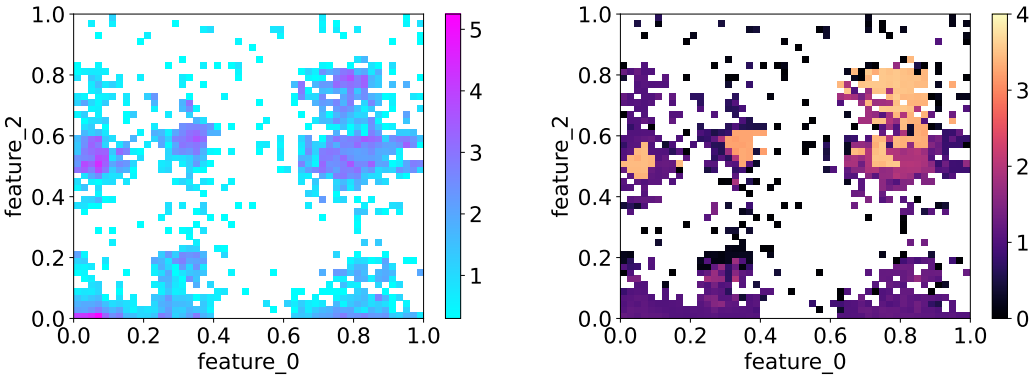


(a) Activity plot.

(b) Fitness heatmap.

Figure A.30: On the right we have a projection of features 2 and 3 of the grid archive for the best Hybrid seed. On the left is the activity plot, to help us understand the areas that were more searched. The activity plot uses a logarithmic scale of base 10.

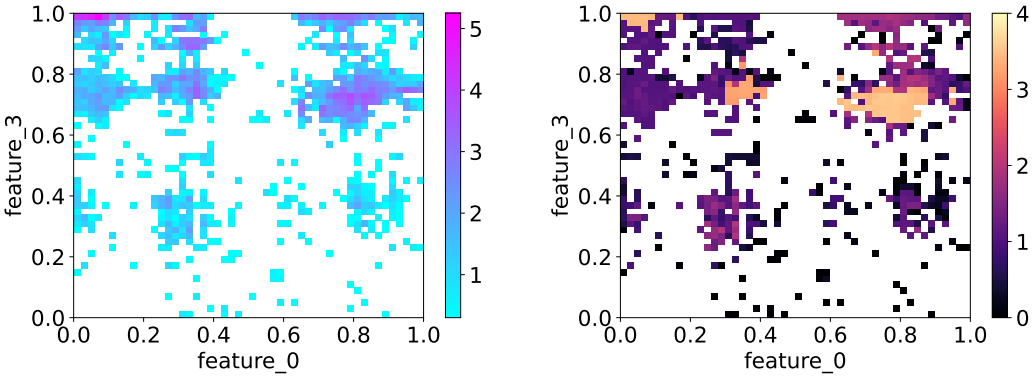
A.7 Non-dominated Sorting Genetic Algorithm II



(a) Activity plot.

(b) Fitness heatmap.

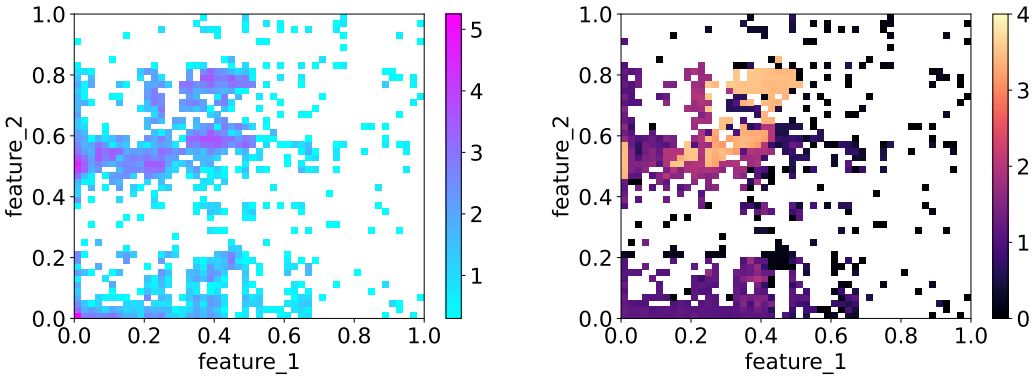
Figure A.31: On the right we have a projection of features 0 and 2 of the grid archive for the best NSGA-II seed. On the left is the activity plot, to help us understand the areas that were more searched. The activity plot uses a logarithmic scale of base 10.



(a) Activity plot.

(b) Fitness heatmap.

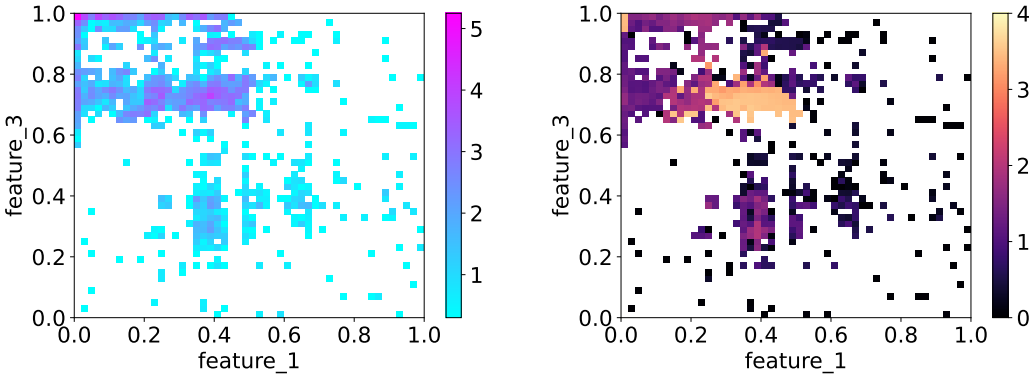
Figure A.32: On the right we have a projection of features 0 and 3 of the grid archive for the best NSGA-II seed. On the left is the activity plot, to help us understand the areas that were more searched. The activity plot uses a logarithmic scale of base 10.



(a) Activity plot.

(b) Fitness heatmap.

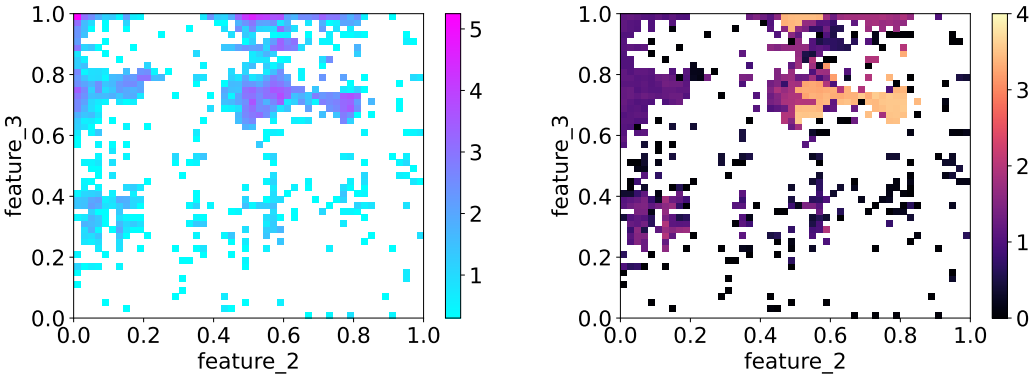
Figure A.33: On the right we have a projection of features 1 and 2 of the grid archive for the best NSGA-II seed. On the left is the activity plot, to help us understand the areas that were more searched. The activity plot uses a logarithmic scale of base 10.



(a) Activity plot.

(b) Fitness heatmap.

Figure A.34: On the right we have a projection of features 1 and 3 of the grid archive for the best NSGA-II seed. On the left is the activity plot, to help us understand the areas that were more searched. The activity plot uses a logarithmic scale of base 10.



(a) Activity plot.

(b) Fitness heatmap.

Figure A.35: On the right we have a projection of features 2 and 3 of the grid archive for the best NSGA-II seed. On the left is the activity plot, to help us understand the areas that were more searched. The activity plot uses a logarithmic scale of base 10.

Appendix B

Conference Paper

B.1 Reducing the Price of Stable Cable Stayed Bridges with CMA-ES

B.1.1 Remarks

In this paper, the cost of the bridges is said to be in the thousands of euros (k €), but in fact the cost is in the tens of thousands. The analysis of the results still holds because this mistake is verified in all the cost values.

Reducing the Price of Stable Cable Stayed Bridges with CMA-ES

Gabriel Fernandes[1 [0000−0001−6662−8675]], Nuno Lourenço[1 [0000−0002−2154−0642]], and João Correia[1 [0000−0001−5562−1996]]

University of Coimbra, CISUC, DEI, Coimbra, Portugal
gabrielf@student.dei.uc.pt, naml@dei.uc.pt, jncor@dei.uc.pt

Abstract. The design of cable-stayed bridges requires the determination of several design variables' values. Civil engineers usually perform this task by hand as an iteration of steps that stops when the engineer is happy with both the cost and maintaining the structural constraints of the solution. The problem's difficulty arises from the fact that changing a variable may affect other variables, meaning that they are not independent, suggesting that we are facing a deceptive landscape. In this work, we compare two approaches to a baseline solution: a Genetic Algorithm and a CMA-ES algorithm. There are two objectives when designing the bridges: minimizing the cost and maintaining the structural constraints in acceptable values to be considered safe. These are conflicting objectives, meaning that decreasing the cost often results in a bridge that is not structurally safe. The results suggest that CMA-ES is a better option for finding good solutions in the search space, beating the baseline with the same amount of evaluations, while the Genetic Algorithm could not. In concrete, the CMA-ES approach is able to design bridges that are cheaper and structurally safe.

Keywords: Genetic Algorithm · CMA-ES · Optimization · Cable Stayed Bridges

1 Introduction

Bridges are critical components of every transportation network infrastructure. They must be designed to be safe, robust and durable and simultaneously cost-effective and, sometimes, aesthetic pleasing, which are often competing objectives [18, 17, 27, 3]. The restrictions on the structural design standards vary from country to country and specify the requirements that bridges must satisfy, such as safety versus heavy vehicle loads, high-velocity winds and earthquakes. The bridge must also be within serviceability requirements which specify the maximum deflections, stresses and oscillations when subject to dynamic actions, such as pedestrians' movements. Each bridge is planned by a structural design firm or a consortium of several companies. The process usually starts with a tender for the services, in which the choice of the company is based on a set of evaluation criteria where, usually, the best commercial proposal (the lowest price)

wins. Design firms must deliver a solution using the lowest possible resources (man-hours). For this reason, structural designers do not have the time to evaluate all possible solutions, even with simpler designs. As such, any technique or mechanism to help automate and optimize the design of bridges is, therefore very valuable as even a small percentage of optimization (without compromising the bridge safety and requirements) constitutes large sums of money saved for the public treasury.

Cable-stayed bridges (CSB) are one of the most complex type of bridges to design, due to the fact that they are highly static indeterminate. Thanks to the progress that we have seen in computational technologies, we can now build CSBs that are longer but, at the same time, safer.

In this work, we extend the study conducted in [5, 4] which uses an Evolutionary Computation based approach to tackle the problem of designing CSBs. In concrete, the authors propose the use of a standard Genetic Algorithm (GA)[20] using a representation based on real numbers to represent each parameter that one most optimise to design a bridge. The results attained by the proposed approach were encouraging since the GA was able to optimize this type of bridge (with some variables fixed) in terms of the structural constraints. However, it was not able to reduce the costs when compared to a hand design without resorting to some tuning. For this work, we use the Covariance Matrix Adaptation Evolution Strategy algorithm (CMA-ES)[13] to see if it is able to surpass the results attained by the GA with the same amount of evaluations as well as the baseline solution.

In terms of contributions, we enumerate the following: (i) a study of a more complex problem, due to the number of cables being also evolved, instead of being static; (ii) a comparison between two optimization algorithms, GA and CMA-ES; (iii) the results suggest that a standard GA might not be enough to find efficient solutions.

Additionally, the CMA-ES algorithm was able to discover a bridge with a cost that is 4.656k € less than the one of the baseline solution. Taking into account that the solution was discovered automatically, without any human input, the result is impressive and opens for further application of evolutionary approaches in the automatic design of bridges.

The remainder of the paper is divided as follows. Section 2 briefly presents the related work. In Section 3, the problem is defined and in Section 4, it is explained how the GA and CMA-ES experiments were modelled. The obtained results are shown in section 5, and our conclusions are listed in Section 6.

2 Related Work

The first works on the optimum design of CSBs focused on addressing the cable tensioning problem with fixed geometry and structural sections [23, 26, 1]. More recently, Genetic Algorithms (GA) have also been used to tackle this problem [14, 16].

Including dynamic loads creates additional constraints for the design problem. Previous researches have focused on earthquakes [25, 8], wind aerodynamics [2, 19, 22] and pedestrian induced action in cable stayed footbridges [9–11]. These dynamic loads may cause the bridge to vibrate, which is something that is detrimental. To mitigate this problem, there are some options, such as: (i) improve the sturdiness of the bridge by increasing its mass. This is something that is not desirable due to the potential increase in cost and the possibility of the resultant bridge not being as aesthetically pleasing as one might want; (ii) Including control devices like the ones used to retrofit the London Millennium Footbridge, [7, 6], for example, viscous dampers or tuned mass dampers (TMDs).

Gradient-based optimization techniques have been used to optimize the bridge's geometry, sizing and cable tensioning [21, 24]. GAs have also been used to optimize simultaneously these bridge properties [15], although with simpler models than the ones found in this and in the works on which this article is based [5, 4]. These are based on the works of Ferreira and Simões [10, 11], from where an already optimized solution (not necessarily a global optimum) was retrieved and then used as a baseline to help us understand if a given solution is in fact good. The literature tells us that gradient-based approaches are able to achieve more rapidly good results than the GA ones. However, compared to a GA, a gradient-based solution is far harder to parameterize, requiring more time and effort, while a GA is easier to get up and running.

As far as we know, there are no works applying CMA-ES to CSB design optimization. The CMA-ES ability to explore the search space via the exploitation of co-variance matrix properties of the genotype holds the potential to further optimize CSBs, beyond existing GA approaches.

3 Problem Definition

In this work, we are evolving configurations for cable-stayed footbridges, minimizing the overall cost of the structure while guaranteeing its structural safety. The structural safety of the bridge is accomplished when the values of the structural constraints are at most 1. For an in-depth explanation of this problem, the reader is advised to read [10].

Each individual is defined by an array composed of 22 variables, whose descriptions and domains can be seen in Table 2. These variables are then utilized to calculate the price of the bridge and the respective values for the structural constraints. Given that for a bridge to be considered secure, all of the constraints need to be less or equal to 1, we resort to using the maximum value of these constraints instead of the individual values. In addition to the variable parameters, the bridges also have fixed ones, which are presented in Table. 1.

In our experiments, we compare our results to a baseline solution, a bridge configuration optimized by the same approach used in Ferreira's et al. work [10] with the same fixed parameters. This bridge has a cost of 91.354 k€ and a maximum of structural constraints of 0.9962.

Table 1. Values for the fixed parameters.

Bridge Length (LTotal)	220 meters
Bridge Width	4 meters
Tower Height below deck	10 meters

Table 2. Cable-stayed bridges design variables description and domain values.

Variable Type	**Description**	**Domain Values**
Discrete		
DV0	Number of cables	3,4,5,6,7
Geometry		
DV1	Central span (tower to tower distance) of the structure	$[0.9, 1.2]$
DV2	Distance between the first and second cables anchorage in the lateral span of the deck	$[0.7, 1.3]$
DV3	Distance between the tower and the first cable in the central span	$[0.7, 1.3]$
DV4	Distance between the last cable anchorage and the bridge symmetry axis	$[0.7, 1.3]$
DV5	Height of the towers	$[0.1, 2.0]$
DV6	Distance where the cables are distributed in the top of the towers	$[0.1, 4.0]$
DV7	Distance between the top of each tower	$[0.1, 1.3]$
DV8	Distance between each tower at the base	$[0.1, 1.13]$
Control		
DV9	Transversal stiffness of the tower-deck connection	$[0.001, 1000]$
DV10	Vertical stiffness of the tower-deck connection	$[0.001, 1000]$
DV11	Transversal damping of the tower-deck connection	$[0.001, 1000]$
DV12	Vertical damping of the tower-deck connection	$[0.001, 1000]$
Sectional and tensioning		
DV13	Added mass of the concrete slab	$[0.1, 7.0]$
DV14	Deck section	$[0.1, 80.0]$
DV15	Deck section (triangular section)	$[0.5, 1.3]$
DV16	Tower section (rectangular hollow section)	$[0.4, 1.5]$
DV17	Tower section (rectangular hollow section)	$[0.1, 20.0]$
DV18	Tower section (rectangular hollow section)	$[0.3, 20.0]$
DV19	Tower section (rectangular hollow section)	$[0.3, 9.0]$
DV20	Cables pre-stress	$[0.7, 3.0]$
DV21	Cables cross section	$[0.5, 9.0]$

4 The Approach

To have a fair comparison with the previous works, we replicated the experiments conducted with the GA. The parameters used for the algorithm are almost the same as in [5], apart from the number of generations, and can be seen in Table 3.

The novelty that we add to this problem with this work is by experimenting with CMA-ES, implemented with Distributed Evolutionary Algorithms in Python (DEAP) framework [12]. The parameters used can be seen in Table 3. Since the sizes of the population are different, we ensure that the same number of evaluations is used, so that the results can be compared.

The initial individuals used to start the evolutionary process are created by uniformly sampling the domain intervals of each variable. This idea is also utilized in the mutation operator used in the Genetic Algorithm, meaning that when a gene is chosen to be mutated, the new value is also uniformly sampled from the domain of the specific variable. In order to deal with the unfeasible solutions generated by CMA-ES, we correct the specific variable to the minimum value of the domain if it is lower than it or to the maximum if it is greater than the maximum value. This process is not performed in the GA, because the variation operators ensure that the values of the variables are within the required domains, given that the crossover does not alter the values and the mutation operator, as previously stated, samples the new value from the domain. In CMA-ES, this correction is necessary because it is not guaranteed that the generated values are within the domain boundaries.

The fitness function used for both algorithms is presented in Eq. 1. $C(x)$ and $S(x)$ are the cost and the structural constraint value of individual x, respectively. The price returned by $C(x)$ is based on pre-determined pricing of the materials, and $S(x)$ returns the maximum value of the structural constraints of the individual [10]. In practice, we need to guarantee that the returned value of $S(x)$ is at most 1.0.

$$f(x) = \begin{cases} c_r/C(x), & \text{if } C(x) > c_r \\ 1 + 1/S(x), & \text{if } C(x) < c_r \wedge S(x) > 1.0 \\ 2 - (1.0 - S(x)) + c_r/C(x), & \text{if } C(x) < c_r \wedge S(x) \leq 1.0 \end{cases} \quad (1)$$

The fitness function aims to guide the population towards individuals that have a structural constraint of at most 1.0 and the lowest cost possible. First, by reducing the cost to a more acceptable value (c_r, it is fixed in our experiments, see Table 3, but can be changed), then search for individuals that are feasible structurally by rewarding individuals that have $S(x)$ values closer to 1.0, and finally find individuals that are both feasible and cost-effective (we want the lowest cost possible). Although we aim to minimize the cost of the structures, it is important to notice that we want to maximize the fitness value, thus defining this problem as a maximization problem. In the first branch, the fitness ranges from 0 to 1, in the second, from 1 to 2, and in the third, it is greater than 2.

Table 3. Algorithm's Parameters.

Parameter	**Value**
GA	
Generations	40 000
Population size	10
Tournament size	3
Crossover operator	Uniform Crossover
Crossover rate (per gene)	0.5
Mutation operator	per gene replacement
Mutation rate per gene	0.1
CMA-ES	Value
Generations	8 000
μ	25
λ	50
σ	0.5
Common	Value
Number of Runs	30
Elite size	1
c_r fitness constant	150
Number of evaluations	400 000

5 Experimental Results

The performance of the best individuals during the evolutionary process for the structural constraints $S(x)$ are presented in Fig. 1. Fig. 3 presents the results regarding the cost $C(x)$, whilst Fig. 5 presents the results for the fitness function, $f(x)$. Results are averages of 30 independent runs.

Looking at Fig.1, one can see that both approaches gradually improve the values of the structural constraints, by gradually getting close to the upper limit value of 1. Looking at the right panel, one can see that the GA has a steep descent in the value of the structural constraint, whilst CMA-ES performs a more slow descent. By the end of the optimization process, both approaches have reached approximately the same structural constraint values. To better understand the differences between the approaches we presented a boxplot of the results regarding the structural constraint for the best individuals. Looking at Fig. 2 one can see that CMA-ES is capable of optimizing this objective, but it has some runs where it fails by a relatively bigger margin, leading to a large mean value. The GA also fails (1 seed out of the 30), but by less. Both curves converge close to 1.0, which is the desired behavior, meaning that the structural safety of the bridge is being optimized.

In what concerns the cost, the results are depicted in Fig. 3. Looking at the results, one can see that during the first 1500 generations, the CMA-ES (left panel) does not seem to improve the values of the initial solutions. In fact, looking at the graph, one can see that it slightly increases cost. However, after generation 1500, the approach starts to rapidly improve the cost of the bridge.

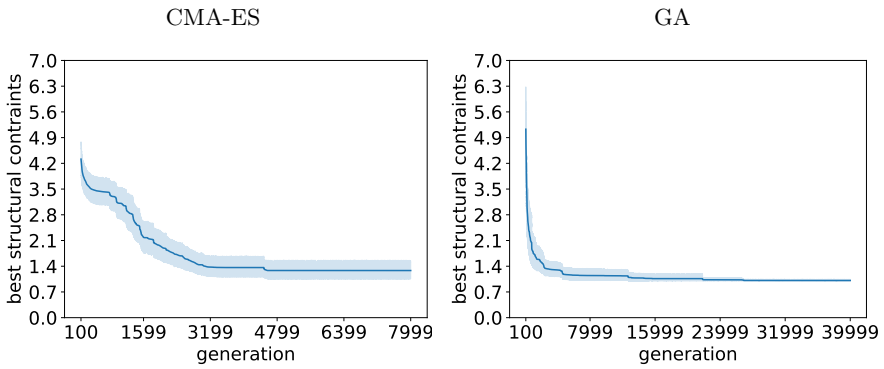


Fig. 1. Mean structural constraint values of the best individuals of 30 runs for CMA-ES (on the left) and for the GA (on the right) starting from generation 100 until generation 7 999 and 39 999 generations, respectively. We can see that CMA-ES stabilizes around the 4 800 generations mark and 28 000 for the GA. The first 100 generations ($[0 : 100[$) were not plotted due to the fact that the values of cost and structural constraints in these generations are extremely large, making the plots unreadable.

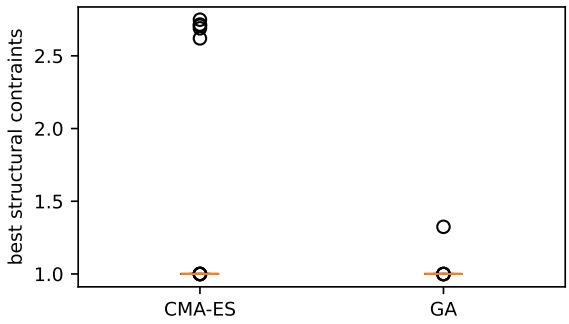


Fig. 2. Boxplot of the structural constraint of the best individual of each run (30 in total) for CMA-ES (on the left) and the GA (on the right).

The curve stabilizes around generation 6000, which might be an indication that the CMA-ES reaches an optimum. The GA (right panel) exhibits roughly the same behavior in what concerns the optimization trend. These results might be explained by the fact that both approaches, in the first generations, focus on obtained bridges that have a good value in terms of structural constraints. After having such bridges the approaches start to reduce the cost, without compromising the integrity and safety of the bridge.

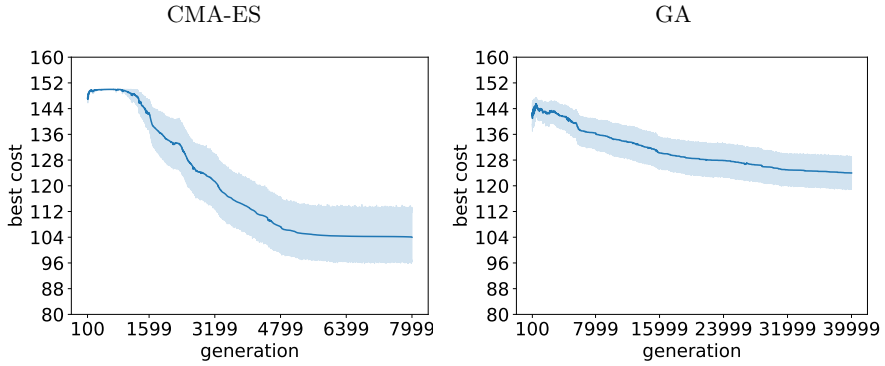


Fig. 3. Mean cost values of the best individuals of 30 runs for CMA-ES (on the left) and for the GA (on the right) starting from generation 100 until generation 7 999 and 39 999 generations, respectively. We can see that CMA-Es is able to achieve lower values of cost, however, it presents a higher variability between runs. It also appears to be stabilizing (we address this topic in the Experimental Results section). It appears that the GA is the opposite, continuing to optimize the cost even after the 40 000 generations. The explanation of why the first 100 generations are not plotted is presented in the caption of Fig.1.

Another interesting result is that by the end of the evolutionary process, the best solutions obtained by the CMA-ES have a much lower cost than the ones discovered by the GA. To help with this analysis, we created a boxplot of the cost values for both approaches and show them in Fig. 4. Whilst CMA-ES is not as good as the GA at optimizing the structural constraints, it reaches brilliant results in terms of cost. In fact, one can see that the CMA-ES approach can not only find bridges with lower costs but also finds them consistently given the lower variance obtained when compared to the GA.

We also show the fitness plots, Fig.5 and 6, to show the results for the $f(x)$ that combines both the cost $C(x)$ and the structural constraint $S(x)$.

Finally, Table 6 summarises the results. Looking at the values, it seems that CMA-ES, on average, does not differ that much from the GA. Even though the curves for the structural constraints are similar and the curves of the cost are very different, it appears that our fitness function is not very good at stretching

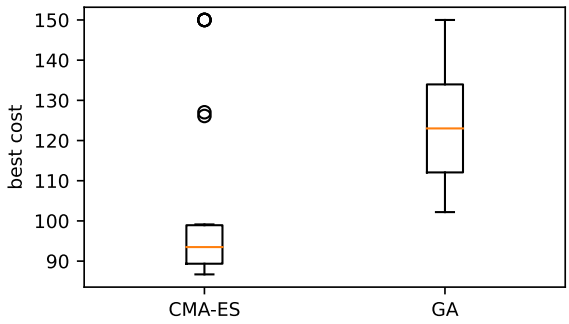


Fig. 4. Boxplot of the cost of the best individual of each run (30 in total) for CMA-ES (on the left) and the GA (on the right).

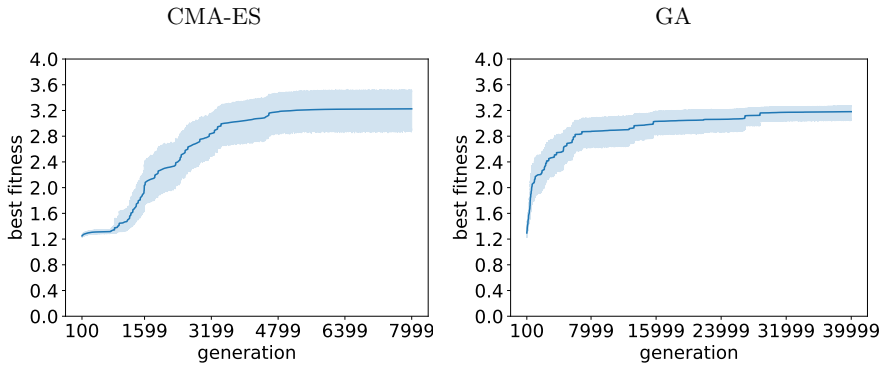


Fig. 5. Mean fitness values of the best individuals of 30 runs for CMA-ES (on the left) and for the GA (on the right) starting from generation 100 until generation 7 999 and 39 999 generations, respectively. It can be seen that CMA-ES is able to reach higher values of fitness, however, there is more variability between runs. The GA seems to be a more consistent algorithm, not showing as much variability. The explanation of why the first 100 generations are not plotted is presented in the caption of Fig.1.

the fitness values when the structural constraints are already satisfied, leading to similar values (but different) of fitness for candidate solutions that have a relatively different cost. For the optimization itself, it still classifies a better individual with higher fitness, however, when plotted, the difference is not very evident.

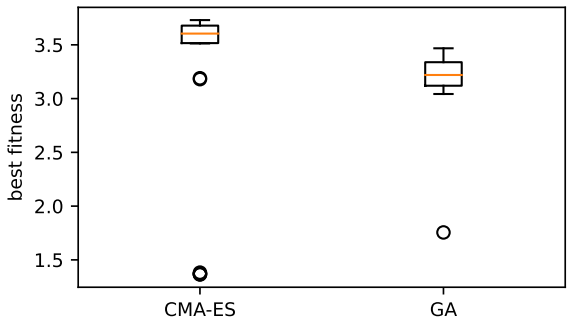


Fig. 6. Boxplot of the fitness of the best individual of each run (30 in total) for CMA-ES (on the left) and the GA (on the right).

To understand if there are meaningful differences between the two approaches, we performed a statistical analysis. Since the samples do not follow a normal distribution, we used the Mann-Whitney non-parametric test with a significance level of $\alpha = 0.05$. The effect sizes are presented in Table 4, and it can be observed that there is a large effect size in all the metrics, meaning that the differences between both approaches are significant.

Table 4. Results of the statistical analysis using the Mann Whitney U test with a significance level $\alpha = 0.05$.

Feature	Effect Size
fitness	-0.510
$C(x)$	0.510
$S(x)$	-0.702

Table 5 presents the cost and the value of the structural constraint of the best solution for every approach. CMA-ES was able to achieve multiple solutions that beat the baseline (see Improvement Rate in Table 4) (with a good level of diversity, because the solutions have different numbers of cables), while the GA could not do it once (see Table 6). With the help of the differences, one can see that both the approaches optimized the structural constraints, however, CMA-

ES was able to reduce the cost of the bridge by more than 4 k€, which is a substantial amount.

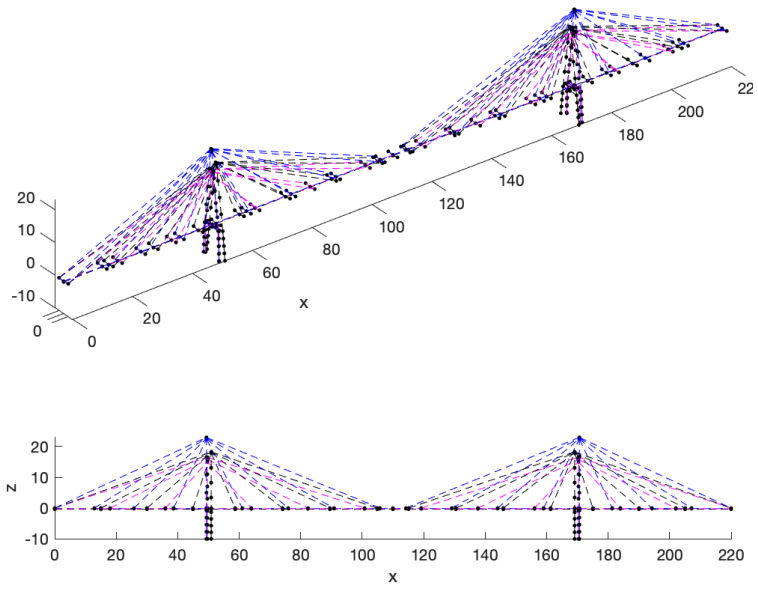


Fig. 7. Baseline bridge (black), versus the bridges optimized by the GA (pink) and by CMA-ES (blue).

Fig.7 shows how the best bridges evolved by the two algorithms compare to the baseline and to each other. We can see that the CMA-ES is visually distinct from the baseline solution due to its height, but the one evolved by the GA is significantly different from the rest because it uses 3 cables, while the others use 4.

Table 5. Best solution of every approach and the difference between cost and structural constraints against the baseline approach. The best values for both the cost and the structural constraints are in bold.

	Baseline(B)	diff(B, GA)	best GA	diff(B, CMA-ES)	best CMA-ES
$C(x)$	91.354	-10.840	102.194	4.656	**86.698**
$S(x)$	**0.996**	-0.004	1.000	-0.004	1.000

CMA-ES finds really good solutions to this problem but appears to be a little extreme, meaning that when it is able to find a good solution it is really good, but when it is not able to, the result is not satisfactory, being expensive and not even structurally safe.

Table 6 further cements what was previously said, showing that, on average, CMA-ES is worse in optimizing $S(x)$ but is able to greatly reduce the cost of the structures when compared to the GA. However, the GA is more consistent, presenting less variability, seen by the standard deviation values.

Table 6. Stats from the experiments. Mean(...) is the average of the 30 runs and Improvement Rate is the rate of runs that were able to beat the baseline. The average values are presented along with the respective standard deviation.

	Mean(fitness)	Mean($C(x)$)	Mean($S(x)$)	Improvement Rate
CMA-ES	3.225 ($\pm$ 0.862)	104.005 ($\pm$ 23.042)	1.282 ($\pm$ 0.652)	11/30
GA	3.181 ($\pm$ 0.296)	123.987 ($\pm$ 13.124)	1.01 ($\pm$0.059)	0/30

We decided to include Fig.8 because we wanted to show how much CMA-ES improved the cost in relation to the one of the baseline. For this image, only the results of the 11 seeds that beat the baseline were included. We do not present a figure for the structural constraints, because, as previously stated, we are only using the results of the seeds that beat the baseline, meaning that the values of the structural constraint are all at most 1.0, given the inverse nature between the cost and the structural constraint.

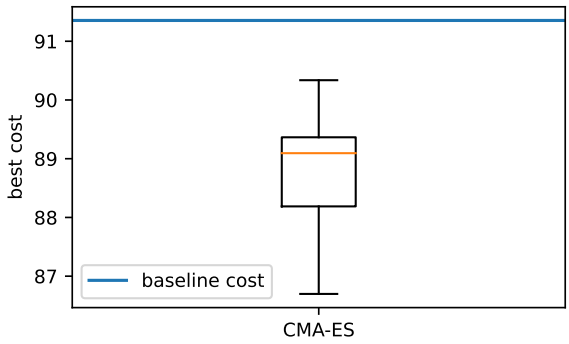


Fig. 8. Boxplot of the cost of the best 11 seeds of CMA-ES plotted with the baseline cost. This better highlights how much the CMA-ES was able to beat the baseline. All of the solutions that beat the baseline have a structural constraints value of at most 1.0, so these values are not plotted here because they would be basically all the same.

6 Conclusions

The design of Cable-stayed bridges (CSB) is one of the most complex designs in bridge engineering since they are highly static indeterminate and cannot be calculated by hand in a short amount of time. This task is mostly handled manually by Civil Engineers, where most of the research on CSB employs gradient-based optimization techniques which require programming the sensitivities of the problem. In this work, we perform a comparison of the performance of two evolutionary approaches, a standard GA and CMA-ES, in terms of cost and structural constraints. We further complement this analysis by comparing the results of both with a previously gradient-based optimized solution found in the literature.

In our results, CMA-ES was able to achieve a cost value of 86.698 k€, beating the baseline and GA costs, 91.354 k€ and 102.194 k€ respectively, while maintaining the structural constraints in acceptable values according to the safety codes. The behaviour of the GA and CMA-ES approach was analyzed in 30 different seeded runs. Under the same budget of evaluations, the GA was not able to beat the baseline solution not even once in terms of cost, despite being more consistent at optimizing the structural constraints when compared with the CMA-ES. The CMA-ES was able to beat the baseline 11 times by a significant margin. Statistical tests were performed with the results of the 30 runs of each algorithm, and the differences between the two were significantly different. The results suggest that CMA-ES performs better, under this setup, for this problem than a standard GA.

In future work, we intend to use quality-diversity algorithms, optimizing both the objective and exploring distinct solutions, to expand our knowledge of the search space as well as being able to retrieve multiple high-performing solutions from a single run and avoid local optimums that may exist.

Acknowledgements

This research was partially funded by the project grant BEIS (Bridge Engineering Information System), supported by Operational Programme for Competitiveness and Internationalisation (COMPETE 2020), under the PORTUGAL 2020 Partnership Agreement, through the European Regional Development Fund (ERDF) and by the FCT - Foundation for Science and Technology, I.P./MCTES through national funds (PIDDAC), within the scope of CISUC R&D Unit - UIDB/00326/2020 or project code UIDP/00326/2020

References

1. Baldomir, A., Hernandez, S., Nieto, F., Jurado, J.: Cable optimization of a long span cable stayed bridge in La Coruña (Spain). Advances in Engineering Software **41**(7-8), 931–938 (jul 2010). https://doi.org/10.1016/J.ADVENGSOFT.2010.05.001

2. Baldomir, A., Kusano, I., Hernandez, S., Jurado, J.: A reliability study for the messina bridge with respect to flutter phenomena considering uncertainties in experimental and numerical data. Computers & Structures **128**, 91–100 (nov 2013). https://doi.org/10.1016/J.COMPSTRUC.2013.07.004
3. Chen, W.F., Duan, L. (eds.): Bridge Engineering Handbook. CRC Press (jan 2014). https://doi.org/10.1201/b16467
4. Correia, J., Ferreira, F.: Designing cable-stayed bridges with genetic algorithms. In: Castillo, P.A., Laredo, J.L.J., de Vega, F.F. (eds.) Applications of Evolutionary Computation - 23rd European Conference, EvoApplications 2020, Held as Part of EvoStar 2020, Seville, Spain, April 15-17, 2020, Proceedings. Lecture Notes in Computer Science, vol. 12104, pp. 228–243. Springer (2020). https://doi.org/10.1007/978-3-030-43722-0_15, https://doi.org/10.1007/978-3-030-43722-0_15
5. Correia, J., Ferreira, F., Maçãs, C.: Cable-stayed bridge optimization solution space exploration. In: Coello, C.A.C. (ed.) GECCO '20: Genetic and Evolutionary Computation Conference, Companion Volume, Cancún, Mexico, July 8-12, 2020. pp. 261–262. ACM (2020). https://doi.org/10.1145/3377929.3390033, https://doi.org/10.1145/3377929.3390033
6. Dallard, P.: The London millennium footbridge. Structural Engineer **79**(22), 17–21 (2001)
7. Dallard, P., Fitzpatrick, T., Flint, A., Low, A., Smith, R.R., Willford, M., Roche, M.: London Millennium Bridge: Pedestrian-Induced Lateral Vibration. Journal of Bridge Engineering **6**(6), 412–417 (dec 2001). https://doi.org/10.1061/(ASCE)1084-0702(2001)6:6(412)
8. Ferreira, F., Simoes, L.: Optimum design of a controlled cable stayed bridge subject to earthquakes. Structural and Multidisciplinary Optimization **44**(4), 517–528 (oct 2011). https://doi.org/10.1007/s00158-011-0628-9
9. Ferreira, F., Simões, L.: Optimum cost design of controlled cable stayed footbridges. Computers & Structures **106-107**, 135–143 (sep 2012). https://doi.org/10.1016/J.COMPSTRUC.2012.04.013
10. Ferreira, F., Simões, L.: Optimum design of a cable-stayed steel footbridge with three dimensional modelling and control devices. Engineering Structures **180**, 510–523 (feb 2019). https://doi.org/10.1016/j.engstruct.2018.11.038, https://linkinghub.elsevier.com/retrieve/pii/S0141029618314275
11. Ferreira, F., Simões, L.: Optimum Design of a Controlled Cable-Stayed Footbridge Subject to a Running Event Using Semiactive and Passive Mass Dampers. Journal of Performance of Constructed Facilities **33**(3), 04019025 (jun 2019). https://doi.org/10.1061/(ASCE)CF.1943-5509.0001285
12. Fortin, F.A., De Rainville, F.M., Gardner, M.A., Parizeau, M., Gagné, C.: DEAP: Evolutionary algorithms made easy. Journal of Machine Learning Research **13**, 2171–2175 (jul 2012)
13. Hansen, N., Ostermeier, A.: Completely derandomized self-adaptation in evolution strategies. Evol. Comput. **9**(2), 159–195 (2001). https://doi.org/10.1162/106365601750190398, https://doi.org/10.1162/106365601750190398
14. Hassan, M.M.: Optimum design of cable-stayed bridges. Phd, Western Ontario University (2010)
15. Hassan, M.M., El Damatty, A.A., Nassef, A.O.: Database for the optimum design of semi-fan composite cable-stayed bridges based on genetic algorithms. Structure and Infrastructure Engineering **11**(8), 1054–1068 (aug 2015). https://doi.org/10.1080/15732479.2014.931976

16. Hassan, M.: Optimization of stay cables in cable-stayed bridges using finite element, genetic algorithm, and B-spline combined technique. Engineering Structures **49**, 643–654 (apr 2013). https://doi.org/10.1016/J.ENGSTRUCT.2012.11.036
17. Hibbeler, R.C., Kiang, T.: Structural analysis. Pearson Prentice Hall (2015)
18. Holgate Alan: The art of structural engineering : the work of Jorg Schlaich and his team. Edition Axel Menges Stuttgart ; London (1997)
19. Jurado, J.Á., Nieto, F., Hernández, S., Mosquera, A.: Efficient cable arrangement in cable stayed bridges based on sensitivity analysis of aeroelastic behaviour. Advances in Engineering Software **39**(9), 757–763 (sep 2008). https://doi.org/10.1016/J.ADVENGSOFT.2007.10.004
20. Mitchell, M.: An Introduction to Genetic algorithms. MIT Press, London (1996)
21. Negrão, J., Simões, L.: Optimization of cable-stayed bridges with three-dimensional modelling. Computers & Structures **64**(1-4), 741–758 (jul 1997). https://doi.org/10.1016/S0045-7949(96)00166-6
22. Nieto, F., Hernández, S., Jurado, J.Á., Mosquera, A.: Analytical approach to sensitivity analysis of flutter speed in bridges considering variable deck mass. Advances in Engineering Software **42**(4), 117–129 (apr 2011). https://doi.org/10.1016/J.ADVENGSOFT.2010.12.003
23. Qin, C.: Optimization of cable-stretching planning in the construction of cable-stayed bridges. Engineering Optimization **19**(1), 1–20 (feb 1992). https://doi.org/10.1080/03052159208941217
24. Simões, L., Negrão, J.: Optimization of cable-stayed bridges with box-girder decks. Advances in Engineering Software **31**(6), 417–423 (jun 2000). https://doi.org/10.1016/S0965-9978(00)00003-X
25. Simões, L.M.C., Negrão, J.H.J.O.: Optimization of cable-stayed bridges subjected to earthquakes with non-linear behaviour. Engineering Optimization **31**(4), 457–478 (mar 1999). https://doi.org/10.1080/03052159908941382
26. Sung, Y.C., Chang, D.W., Teo, E.H.: Optimum post-tensioning cable forces of Mau-Lo Hsi cable-stayed bridge. Engineering Structures **28**(10), 1407–1417 (aug 2006). https://doi.org/10.1016/J.ENGSTRUCT.2006.01.009
27. Zienkiewicz, O., Taylor, R., Fox, D.: The Finite Element Method for Solid and Structural Mechanics. Elsevier (2014). https://doi.org/10.1016/C2009-0-26332-X