

POLITECNICO DI TORINO

Master's degree in Civil Engineering

Master's degree thesis

**Structural optimisation of a parametric arch structure:
comparison between three different metaheuristic algorithms**



**Politecnico
di Torino**

Supervisors:

A. Manuello Bertetto

G. C. Marano

J. Melchiorre

Candidate:

V. Morganti

A.Y. 2021/2022

Abstract

Over the last few decades, sustainability and efficiency have become increasingly important in the architecture, engineering, and construction (AEC) industry due to the large amount of resources consumed. In view of this scenario, the primary requirement for designing large-scale structures is efficiency, achieving performance targets with the least amount of structural material. Since the advent and development of computational tools, structural optimization based on mathematical computation has become one of the most commonly used methods for designing sustainable and efficient buildings. Due to the complexity of optimization problems, classical and traditional techniques will not always be able to provide a global optimum solution efficiently. Metaheuristic algorithms can be applied to a wide range of situations, especially in the engineering field, where they have been extensively used. This thesis aims to investigate the efficiency of various metaheuristics algorithms, such as Genetic Algorithm (GA), Simulated Annealing (SA) and Estimation of Distribution Algorithm (EDA), when applied to size and shape optimization problems. Numerical results are obtained from the case study of a parametric arch structure. In particular, the optimization process is treated with the three different metaheuristic single-objective algorithms, by minimizing the structural weight and imposing a constraint condition on the Von Mises stresses along the arch. Cases with 2, 3, 4, and 5 parameters were studied based on the number of design variables considered. In order to have an objective evaluation of the performance of the three different algorithms and a comparative study that is robust and reliable, there have been 20 independent analyses conducted for each different algorithm in each case. In addition, a sensitivity analysis of the EDA algorithm was conducted for different alpha and beta values. Finally, comparative graphs of the averages and standard deviations were reported to assess the performance of the various algorithms tested objectively. The phases of parametric modelling, structural analysis and optimization have been developed entirely in the virtual environment provided by Rhino, Grasshopper, Karamba and Galapagos software packages. This interoperable software is chosen for this study due to its high versatility and customizability and, at the same time, the fact that it is widely used within engineering and architecture offices, so that procedures and results provided here can be helpful to design practitioners.

Summary

1.	Introduction.....	1
1.1.	General overview	1
1.2.	Main objectives	2
1.3.	The outline of the thesis	3
2.	Structural optimization	4
2.1.	Introduction	4
2.2.	The basic idea	4
2.3.	Three Types of Structural Optimization Problems	5
2.4.	The design process	6
2.5.	Process of Structural Optimization	7
3.	Metaheuristic Algorithms	11
3.1.	Introduction	11
3.2.	Simulated Annealing Algorithm	14
3.3.	Genetic Algorithm.....	17
3.4.	Estimation of distribution algorithms.....	23
4.	Integration between parametric design and structural optimization	27
4.1.	Introduction	27
4.2.	Parametric Design.....	28
4.3.	The first parametric design experiments since 1900	30
4.4.	Modern examples of parametric design	35
4.5.	Parametric software	38
4.5.1.	Introduction	38
4.5.2.	Rhinoceros and Grasshopper.....	40
1.1.1.	Karamba.....	41
1.1.2.	Galapagos.....	42
5.	Set-up of the optimisation problem applied to the case study of an arch structure	43

5.1.1.	Arch Structures.....	45
5.2.	The setting of the Optimisation Problem	46
5.2.1.	Fixed parameters	46
5.2.2.	Optimization choices and design variables	48
6.	Comparison between three different metaheuristic algorithms applied to the structural optimization of a Steel Arch Structure	50
6.1.	Introduction	50
6.2.	Optimisation Results	51
6.2.1.	Configuration with 2 design variables.....	51
6.2.1.1.	Introduction	51
6.2.1.2.	Genetic Algorithm.....	52
6.2.1.3.	Simulated Annealing	53
6.2.1.4.	EDA.....	54
6.2.1.5.	Comparison graphs and discussion of results.....	64
6.2.2.	Configuration with 3 design variables.....	67
6.2.2.1.	Introduction	67
6.2.2.2.	Genetic Algorithm.....	68
6.2.2.3.	Simulated Annealing	69
6.2.2.4.	EDA.....	70
6.2.2.5.	Comparison graphs and discussion of results.....	71
6.2.3.	Configuration with 4 design variables.....	73
6.2.3.1.	Introduction	73
6.2.3.2.	Genetic Algorithm.....	74
6.2.3.3.	Simulated Annealing	75
6.2.3.4.	EDA	76
6.2.3.5.	Comparison graphs and discussion of results	77
6.2.4.	Configuration with 5 design variables.....	79

6.2.4.1.	Introduction.....	79
6.2.4.2.	Genetic Algorithm.....	80
6.2.4.3.	Simulated Annealing.....	81
6.2.4.4.	EDA.....	82
6.2.4.5.	Comparison graphs and discussion of results	93
6.2.5.	Comparison graphs between different configurations and discussion of results	95
7.	Conclusion.....	97
	Bibliography	98
	Index of Figures.....	99
	Index of Graphs.....	100
	Index of Tables.....	102

1. Introduction

1.1. General overview

Over the last few decades, sustainability and efficiency have become increasingly important in the architecture, engineering, and construction (AEC) industry due to the large amount of resources consumed. In view of this scenario, the primary requirement for designing large-scale structures is efficiency, achieving performance targets with the least amount of structural material. Since the advent and development of computational tools, structural optimization based on mathematical computation has become one of the most commonly used methods for designing sustainable and efficient buildings. The fundamental problem of optimal structural design attracted the attention of Galileo (1638) and later, Bernoulli (1687), Newton (1687), D. Bernoulli (1733), Lagrange (1770), Saint-Venant (1864), Maxwell (1869), and Lévy (1873). Michell (1904) established the foundation for optimizing structural shape and topology. As a result of his pioneering studies in this field, he developed concepts originally demonstrated by Maxwell (1864) regarding the limits of the material economy in truss structures. When Schmit combined finite element analysis methods with nonlinear numerical optimization methods to create what he called *Structural Synthesis* in 1960, it marked the beginning of modern structural optimization (Vanderplaats, 1993). A new branch of structural engineering is emerging that formulates the structural design problem as a decision-making problem. *Decision-making* is a cognitive process that tries to select the best action among several alternatives. A decision-making problem is a challenging task that involves the use of mathematical models, simulations, statistical analysis, and mathematical optimization techniques to determine the optimal solution. In the context of decision-making problems, an objective function representing the quality of the solution under given constraints; the optimal solution is a solution that identifies the best values of the decision variables such that the objective function gets its extremum value and all constraints are satisfied simultaneously. Operations research is developed to help decision makers in an engineering business, public systems, manufacturing and service industries. Applying the operations research methods to structural design caused emergence of structural optimization. In the mathematical modelling of the structural design process, the decision variables are taken as the cross-sectional properties of structural members. The constraints are imposed on stresses and displacements in the structure under the applied loads. The objective function is generally considered to minimise the structure's overall or material cost. In the past, designers had to find the required cross-sections for the members of structural frames by trial and error method prior to formulating the structural design problem. A structural design problem involves a large number of design parameters that have to be taken into consideration. Furthermore, the possible combinations of design parameters result in thousands of different design options, which makes the total number of trials so large that no designer has the time to try all these possible combinations. For this reason, structural designers have welcomed the emergence of structural optimization

because the design process has become a decision-making problem that can be formulated mathematically to obtain the optimum solution to the design problem. As a result of all these factors, it can be concluded that decision-making is a critical element in the early stages of design since the decisions made during the early phases of the design process have a major impact on the subsequent stages of the design process. Consequently, they influence the overall performance and the appearance of the constructed building. At this point, computational optimization techniques became a necessity in the structural design of a large-scale structure. Regarding computational optimization techniques, metaheuristic algorithms can play a significant role in presenting promising design alternatives and dealing with the design's complexity. Furthermore, these algorithms do not guarantee that the optimal global solution will be found. However, they are capable of providing near-optimal results within a reasonable time. Providing a near-optimal solution in a reasonable timeframe can be more advantageous for a decision-maker than presenting the optimal solution in a long time. Some of these metaheuristic are Harmony search (HS) (Zong Woo Geem et al., 2001), Particle swarm optimization (PSO) (Kennedy and Eberhart, 1995), Differential evolution (DE) (Rainer Storn and Kenneth V. Price, 1995), Genetic algorithm (GA) (Holland, 1975), Ant colony optimization (ACO) (Dorigo et al., 1992), Simulated annealing (SA) (Kirkpatrick et. Al, 1983), Estimation of distribution algorithm (EDA) (Mühlenbein and Paab, 1996), and Evolutionary algorithm (EA) (Bäck, T, 1996). Three of these algorithms: Genetic algorithm, Simulated annealing and Estimation of distribution algorithm were compared to each other in the present work by applying them to the case study of structural optimization of an arch structure.

1.2. Main objectives

The purpose of this work is to investigate the efficiency of various metaheuristics algorithms, such as Genetic Algorithm (GA), Simulated Annealing (SA) and Estimation of Distributed Algorithm (EDA), when applied to structural size and shape optimization problems. The comparison between these three algorithms was carried out on the case study of structural optimisation of a parametric arch structure. The phases of parametric modelling, structural analysis and optimization have been developed entirely in the virtual environment provided by Rhino, Grasshopper, Karamba and Galapagos software packages. In particular, the novel implementation of EDA presented in (Rosso et al., 2022) was developed in the Grasshopper environment (Gabriele Rosi, 2022), in order to comparing it in terms of efficiency with respect the optimization algorithms present in Grasshopper via the component Galapagos: Genetic Algorithm and Simulated Annealing. This interoperable software is chosen for this study due to its high versatility and customizability and, at the same time, the fact that it is widely used within engineering and architecture offices, so that procedures and results provided here can be helpful to design practitioners. Numerical results are obtained from the case study of a such parametric arch structure. In particular, the optimization process is treated with the three different metaheuristic single-objective algorithms, by minimizing the structural weight and imposing a constraint

condition on the Von Mises stresses along the arch. Cases with 2, 3, 4, and 5 parameters were studied based on the number of design variables considered. In order to have an objective evaluation of the performance of the three different algorithms and a comparative study that is robust and reliable, there have been 20 analyses conducted for each different algorithm in each case. In addition, a sensitivity analysis of the EDA algorithm was conducted for different alpha and beta values. Finally, comparative graphs of the averages and standard deviations were reported to assess the performance of the various algorithms tested objectively.

1.3. The outline of the thesis

The content of this thesis is divided into 6 chapters, without considering the current one. A background on structural optimization is provided in Chapter 2, where the introduction emphasizes the importance of this method in structural engineering. After presenting the basic idea of structural optimization, the three optimization problems used in civil engineering are discussed: size, shape, and topology optimization. Finally, the iterative process of structural optimization is illustrated, and the mathematical formulation of the problem is presented. The third chapter focuses on metaheuristic algorithms. The introduction provides a background on the history of metaheuristic algorithms and a distinction between the metaheuristic algorithms and the so-called exact algorithms, describing the pros and cons of each. Afterwards, the three meta-heuristic algorithms used in this thesis are discussed in detail: Genetic algorithm, Simulated annealing and Estimation of Distribution Algorithms. A discussion of parametric design follows in chapter 4 and the potential of the visual coding commercial software is presented. The software package used in this work and its interaction are finally discussed. In Chapter 5, the optimisation case study of an arch structure is finally presented. The geometry and the material used are presented. Finally, the methodology of the optimisation process used is explained. Chapter 6 deals with the comparison between the three different metaheuristic algorithms, cases with 2, 3, 4, and 5 parameters were studied depending on the number of design variables considered. There have been 20 analyses conducted for each different algorithm in each case. In addition, a sensitivity analysis of the EDA algorithm was conducted for different alpha and beta values. Finally, comparative graphs of the averages and standard deviations were reported to assess the performance of the various algorithms tested objectively. Conclusively, chapter 7 summarising the results of the preceding research and discussing some ideas and suggestion for future developments.

2. Structural optimization

2.1. Introduction

Nowadays, environmental sustainability has become an integral part of nearly every aspect of an individual's life on the planet. For example, it is widely recognized that construction has a significant environmental impact worldwide (Spence and Mulligan, 1995); buildings contribute 40% of global greenhouse gas emissions, which is why the design of lightweight structures is crucial today. The building sector has a significant impact on global warming. The way in which a building behaves in terms of energy consumption and emissions associated with the material and construction varies greatly; for example, the energy and carbon embodied in the materials used. Due to the significant amount of CO_2 emissions in the building sector, structural optimization's primary objective is to reduce the environmental impact. In light of the above scenario, the primary requirement for designing large-scale structures is efficiency, achieving performance targets with the least amount of structural material. As a result of structural optimization, one of the main purposes is to minimize the total structural cost by reducing the total weight and reusing part of it.

In addition, architects are increasingly embracing a freeform architectural language. Although these may seem contradictory at first glance, the use of parametric models and tools can be very effective in addressing these issues, especially in the early stages of the design process, providing the opportunity for the efficient and resource-efficient structure to be designed. The relations between aesthetic quality and structural efficiency in a computational approach involve several design possibilities where computational algorithms generate sustainable, highly efficient, innovative, and creative building designs. However, exploring such options is challenging and requires moving away from the current paradigm of geometric modelling, analyzing natural systems to control structural shapes and forces, and minimizing material quantities.

2.2. The basic idea

Structural optimization is loosely defined by (Olhoff and Taylor, 1983) as the rational establishment of structural design that is the best of all possible designs within a prescribed objective and a given set of geometrical and/or behavioural limitations. In this context, the term "best" indicates making the structure as light as possible, i.e., to minimize weight. Another specification of "best" could be making the structure as stiff as possible, and yet another to make it as insensitive to buckling or instability as possible. Such maximization or minimization cannot be performed without any constraints. The variables usually constrained in structural optimization problems are stresses, displacements, and/or geometry. Most quantities used as constraints can also be used as measures of "best," i.e., as an objective function. Therefore, it is possible to put down several structure measures like weight, stiffness, critical load, stress, displacement, and geometry, and a structural

optimization problem is then formulated by choosing one of these as the objective function to minimize or maximize and using some of the other measures as a constraints (Christensen and Klarbring, 2009).

2.3. Three Types of Structural Optimization Problems

Structural efficiency results from an optimization process that can be more or less explicit and more or less rigorously at a different level. Depending on the degree of complexity, structural optimization problems are divided into four types, namely sizing optimization, shape optimization, topology optimization and layout optimization, as described in (Mei and Wang, 2021):

- *Sizing optimization* treats the cross-sectional areas of structures or structural members as the design variables;
- *Shape optimization* also known as configuration optimization which treats the nodal coordinates of structures as the design variables;
- *Topology optimization* focuses on how nodes or joints are connected and supported, aiming to delete unnecessary structural members to achieve the optimal design;
- *Layout optimization*: simultaneously consider two or more of the above optimization objectives for better optimization results, an optimization involving size, shape, and topology at the same time is also known as layout optimization.

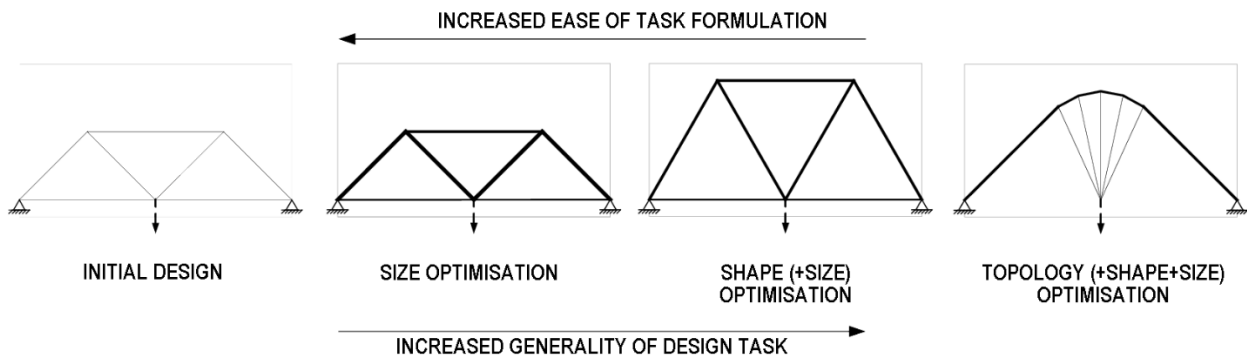


Figure 1: Example of structural size, shape and topology optimisation

2.4. The design process

It is relevant to note that the structural performance measures indicated in the previous paragraph such as displacement, stresses are purely mechanical in nature, and therefore parameters such as functionality, economy or aesthetics were not considered. However, as a way to clarify the position of structural optimization concerning such, usually not mathematically defined factors, it is helpful to provide a short outline of the main step involved in designing a product in general as described by (Kirsch, 1993):

1. *Functionality*: What is the use of the product? The space required in an industrial building, the number of lanes required in a bridge, expected load to be carried on a truss bridge etc. These are the examples of functional requirements, which are often established before entering the design process.
2. *Conceptual design*: This is the critical part of the design stage, because the designer is expected to do material selection, select the overall topology, and the type of structure. For example in a bridge design, the designer will decide whether it should be a truss bridge, an arch bridge or perhaps a cable stayed bridge with selected material.
3. *Optimization*: Within the selected conceptual design considering desired constraints that satisfy the functional requirements achieving the optimal design to making the product as good as possible. A typical example for a bridge design it would be natural to minimize cost by using the least possible amount of materials.
4. *Details*: After completion of the optimization stage, the results must be checked and modified if necessary. This step is usually controlled by market, social or aesthetic factors. In the bridge case, perhaps choosing an interesting colour.

According to (Christensen and Klarbring, 2009) Step 3 is an iterative process in nature that can be implemented in two different ways: *iterative-intuitive* and *mathematical design optimization*. In the *iterative-intuitive* method, the iterative process is primarily based on intuitive foundations; a series of designs are created, which hopefully converges to an acceptable final design.

The *mathematical design optimization* method differs conceptually from the *iterative-intuitive* method. This method consists of formulating a mathematical optimization problem, where requirements due to the functions impose constraints, and the concept of “as good as possible” is formulated mathematically in precise terms. In this thesis, a case study will be presented on how to optimize a steel arch using the *mathematical design optimization* method.

2.5. Process of Structural Optimization

(Mei and Wang, 2021) contend that structural optimization is generally carried out based on the following four factors:

1. Modelling technique for structural analysis and design, based on which structural optimization can be divided into discrete optimization and continuous optimization;
2. Formulation of optimization problem, including the definition of variables, objective function(s), and constraints;
3. Optimization method, referring to the mathematical programming methods applied to achieve structural optimization;
4. Computational tool and design platform, referring to the software platform used to run the optimization codes and conduct structure design.

Modelling Techniques for Structural Analysis

Structural optimization is an iterative process in nature. During the optimization process, structural analysis needs to be repeated many times to evaluate the improvement of each design until convergence is reached, which is an enormous computational task. Therefore, it is necessary to select a structural analysis method that is computationally inexpensive, especially for large, complex civil engineering structures. In this regard, structural optimization is most commonly performed at the early stages of a design project since considering all structural details would significantly increase the computational effort. Generally, structural analysis is conducted based on finite element method (FEM).

According to (Ohsaki, Makoto, 2002) structural optimization can be divided into two broad category, namely discrete optimization and continuum optimization. In discrete optimization, the structure is modelled with discrete structural elements, while the structure system is treated as a solid continuum with variable topology in continuum optimization. Continuum optimization is usually applied in topology optimization, dealing with material distribution problems.

Formulation of Optimization Problem

The mathematical definition of a structural optimization problem refers to the determination of three fundamental components in the problem's search space: the design variables, the objective function(s), and the constraints. When performing structural optimization, it is assumed that some structural parameters can be

changed. The parameters used to represent the change in these attributes are generally called design variables and usually is described by a vector. It is clear that structural optimization is an iterative process in nature and, therefore, the structural analysis must be repeated many times to evaluate the improvement of each analysis until convergence is reached. Design variables can be split into two categories depending on their value: a continuous design variable and a discrete design variable. The values of continuous design variables range within a specific interval, whereas discrete design variables only have isolated values.

As the name suggests, objective functions are functions or sets of functions that can be used as a measure of the optimum solution to an optimization problem and that return a number for every possible solution that indicates the degree of goodness of that design.

Imperative design requirements for structure as safety and serviceability are considered constraints that must be satisfied during the optimization process. Constraints can be divided into two categories, equality constraints, and inequality constraints. They can be interconverted to satisfy the requirements of different optimization methods. A corresponding mathematical function of a structural optimization problem with equality and inequality constraints and mixed discrete-continuous design variables is as follows:

<i>Minimize:</i>	$f(\mathbf{X})$	$\mathbf{X} = (x_1, x_2, \dots, x_n)^T$	<i>objective function</i>
<i>Subject to:</i>	$g_i(\mathbf{X}) \leq 0, i = 1, 2, 3, \dots, m$		<i>inequality constraints</i>
	$h_j(\mathbf{X}) = 0, j = 1, 2, 3, \dots, p$		<i>equality constraints</i>
	$x_i \in D_i$	$D_i = (d_{i1}, d_{i2}, \dots, d_{iq_i}); i = 1, \dots, n_d$	
<i>discrete variable set</i>			
	$\mathbf{X} \in S$		
<i>Where f is the objective function of \mathbf{X}, a set of n design variables, n_d of which are discrete, the remainder being continuous, q_i is the number of available discrete values within D_i for each x_i, g_i is the set of m inequality constraints (including bounds on continuous variables), h_j is the set of m equality constraints.</i>			

In summary, structural optimization involves the following according to (Papalambros e Wilde, 2000):

1. The selection of a set of variables to describe the design alternatives.
2. The selection of an objective (criterion), expressed in terms of the design variables, which we seek to minimize or maximize.

3. The determination of a set of constraints, expressed in terms of the design variables, which must be satisfied by any acceptable design.
4. The determination of a set of values for the design variables, which minimize (or maximize) the objective, while satisfying all the constraints.

Optimization Techniques and Methods

There are generally two categories of optimization methods employed in civil engineering structural optimization: gradient-based approaches and heuristic approaches (Aldwaik and Adeli, 2014).

Gradient-based approaches require a predefined search direction, which is called gradient, before searching the optimal solution. As stated by (Aldwaik and Adeli, 2014) this type of optimization approach can be further divided into four categories: linear programming methods, non-linear programming methods, optimality criteria methods, and feasible direction methods. The gradient-based optimization methods are also known as conventional methods, and they were widely used in the early studies on civil engineering structural optimization. Several limitations of gradient-based algorithms have been identified in previous studies. These limitations can be summarized as follows:

1. In structural optimization, gradient-based algorithms have good performance, but convergence to the global optimum can be hard to achieve (Topping, 1983). Due to the fact that structural optimization problems typically have multiple local minima, these gradient-based algorithms may converge to one of these local optimums if the initial design and the search direction are not well defined. Therefore, these algorithms are often trapped into a local optimum rather than reaching the global optimum.
2. As a result of the computation gradients requirement, these algorithms are difficult to implement and inefficient (Sigmund, 2011). Consequently, gradient-based methods are not suitable for solving optimization problems involving large structures with highly nonlinear, implicit, and discontinuous constraints.
3. In general, gradient-based algorithms have limited application scope due to the lack of comprehensive optimization constraints (Topping, 1983).

To address the limitations of the gradient-based algorithms, a new type of mathematical programming methods is proposed and adopted to meet the requirements in structural optimization, which is known as heuristic methods. Heuristic methods refer to problem-solving approaches that obtain the solution by trial and error. This kind of optimization methods include lots of machine learning techniques, such as artificial neural network (Ganjefar S. et al., 2018) and support vector machines (Yuan Y. et al., 2009), that aim to improve the accuracy of solutions by iterations. Although heuristic methods are relatively easy to program with high

computational speed, these methods are problem-dependent and may be trapped in local optimum. Therefore, researchers have developed other heuristic methods, namely metaheuristic methods, for better optimization. However, it is essential to realize that metaheuristic methods are not problem-dependent and are based on specific trade-off randomisation to move from local search to global search. The computational tools and the platform used to run the optimization codes and conduct structure design will be discussed in detail in Chapter 4.

3. Metaheuristic Algorithms

3.1. Introduction

Optimization algorithms are roughly divided into two categories: exact and heuristic algorithms. The main difference between the two categories is that exact algorithms such as linear programming (LP), non-linear programming (NPL), integer programming (IP), and dynamic programming (DP) are designed in such a way as to guarantee that they will find the optimal solution in a finite time. Heuristic types are conceived to produce hopefully reasonable solutions without a priori certainty of proximity to the optimum. The exact algorithm must guarantee to find the optimum; to do this, they have exhaustively examined every solution in the solution space, except if they can explicitly determine that it is unnecessary to examine it. Various techniques manage to eliminate many solutions at each iteration, but the fact remains that many real-life optimization problems where the size and/or the complexity is large are not easily tackled by the exact method. Due to the impracticality of exhaustive research, heuristics for challenging problems have been developed throughout human history. The first research on heuristics often focused on human intuition as a problem-solving method. For example, Simon and Newell (1958) propose that a heuristic theory of problem-solving (as opposed to algorithm or exactness) focuses on intuition, insight, and learning. In his influential book “*How to Solve It?*” (Pólya, 1945) George Pólya discussed many techniques to create efficient heuristics, and some of the heuristic principles are actively being used by heuristics researchers today to develop a practical algorithm. These mathematical principles can be summarised as follows:

- The *Analogy* principle is sort of similarity. Similar objects agree with each other in some respect, and analogous objects agree in certain relations of their respective parts.
- The *Induction* principle is the process of discovering general laws by the observation and combination of particular instances.
- *Auxiliary problem* is the principle that allows finding the solution of a problem by solving another problem, not for its own sake, but because its consideration may help to solve another problem, the original problem.

Based on the above principles, in order to design a heuristic algorithm, one will search in the literature for similar problems for which a solution technique is known (analogy) and try to solve a problem by deriving a generalization from some example in order to find an intelligent solution strategy (induction), and again trying to break down the problem into, for example, the main problem and a subproblem, and develop specialized techniques for both (auxiliary problem).

The term *metaheuristic* was first defined by Glover (1986) as follows: "A metaheuristic is a high-level problem-independent algorithm framework that provides a set of guidelines or strategies to develop heuristic optimization algorithms. The term is also used to refer to a problem-specific implementation of a heuristic optimization algorithm according to the guidelines expressed in such a framework". The idea behind the development of metaheuristic algorithms was that a heuristic should not be wholly problem-dependent but that general optimization techniques could be developed and applied to solve broader problems of any nature. In general, a metaheuristic is not an algorithm, i.e., it is not a sequence of actions that must be followed; instead, it is a consistent set of ideas, concepts, and operators that can be used to design heuristic optimization algorithms. Many of the early metaheuristics were closely related to the way humans use their intelligence to solve problems; however, from the early 1970s, metaheuristics began to be developed based not on insight into the problem structure or on how an intelligent human would solve it, but on the process that at first sight seemed to have very little to do with optimization.

Meta-heuristic optimization algorithms are subdivided into single and population-based categories. The principle of single-based meta-heuristic algorithms is generating a single solution at each execution. Some of the well-known single-based meta-heuristics are: Simulated Annealing (SA) (Kirkpatrick et. Al, 1983), Guided Local Search (GLS) (Voudouris and Tsang, 1996), Tabu Search (TS) (Glover F., 1989), Variable Neighbourhood Search (VNS) (Mladenović & Hansen in 1997), Iterated Local Search (ILS) (Stützle T. and Ruiz, R., 2017), Stochastic Local Search (SLS) (H.H. Hoos and T. Stützle, 2004) and Greedy Randomized Adaptive Search Procedure (GRASP) (Feo and Resende, 1995). In contrast to single-based meta-heuristic algorithms, population-based meta-heuristic algorithms generate multiple solutions (population) at each execution. The class of population-based meta-heuristics can be classified into four main categories: evolution-based, swarm intelligence-based, event-based and physics-based. The first category of population-based algorithms is Evolutionary Algorithms (EA) (Bäck, T, 1996), which are inspired by natural evolutionary phenomena, using three leading operators: selection, recombination, and mutation. Some well-known evolutionary algorithms are Genetic Algorithm (GA) (Holland, 1995), Estimation of Distribution Algorithms (Mühlenbein and Paab, 1996), Differential Evolution (DE) (Rainer Storn and Kenneth V. Price, 1995), Evolutionary Programming (EP) (Lawrence J. Fogel, 1960), Genetic Programming (GP) (Koza, J. R., 1990), Evolution Strategy (ES) (Beyer and Schwefel, 2002), and Biogeography-Based Optimizer (BBO) (D. Simon, 2008). The second category includes Swarm Intelligence (SI) approaches (Banerjee et al., 2022), where the source of information is collective behaviour in nature. (e.g., birds, ants, bees). Particle Swarm Optimisation (PSO) (Kennedy and Eberhart, 1995) and the Artificial Bee Colony Algorithm (ABC) (Karaboga, 2005) are among the most popular algorithms in this category. Finally, in the third family, the source of inspiration is not nature but rather human-related actions. For example, the Teaching Learning-Based Algorithm (TLBA) (R.V. Rao et al., 2011) mimics the teaching and learning process in classrooms, Imperialist Competitive Algorithm

(ICA) (E. Atashpaz-Gargari and C. Lucas, 2007) is inspired by imperialism in societies, and musical concepts inspire Harmony Search (HS) (Geem et al., 2001). The last family of meta-heuristics is Physics-based Algorithms (PA). For example, Multi-Verse Optimiser (MVO) (Mirjalili et al., 2016) is inspired by some theories of multiple universes, and Gravitational Search Algorithm (GSA) (Rashedi et al., 2009) mimics gravitational forces between masses.



Figure 2: The mosaic of metaheuristic optimization algorithms

The original figure with the classification of the algorithms appears in (Meraihi Y et al., 2021) and includes 45 algorithms; instead, in the new version reported above it is added one additional algorithm (EDA). Three of them, Simulated Annealing (SA), Genetic Algorithm (GA), and Estimation Distribution Algorithm (EDA), have been selected to be confronted with the challenging problem of optimizing an arch steel structure and are examined in detail in the next section.

3.2. Simulated Annealing Algorithm

The Simulated Annealing algorithm was first presented by Kirkpatrick et al. (1983) and is based on the Metropolis algorithm (Metropolis et al., 1953) to generate simple states of a thermodynamic system and was one of the earliest metaphor-based metaheuristics. It is considered to be the oldest among the metaheuristic algorithms and was one of the first to have an explicit strategy to avoid getting trapped in local optima. The main idea of Simulate Annealing is to accept, in certain cases, transitions that correspond to improvements in the objective function and those transitions that lead to deteriorations in the value of this evaluation function. The probability of accepting such deterioration varies throughout the search process, and slowly descends toward zero. Towards the end of the search. However, the possibility of transiting to points in the search space that deteriorate the current optimal solution, makes it possible to abandon local minima and better explore the set of admissible solutions. Precisely, the strategy underlying Simulated Annealing is inspired by the physical process of Annealing: the atoms in the molten material move randomly, but as the temperature decreases, they attempt to drop into the lowest possible energy state. If the system is at thermal equilibrium at a given temperature T , then the probability $n_T(s)$ that it is in a given configuration s depends on the energy of the state $E(s)$ and follows the Boltzmann distribution:

$$n_T(s) = \frac{e^{-E(s)/kT}}{\sum_w e^{-E(w)/kT}} \quad (3.1)$$

Where k is the Boltzmann constant and the summation extends to all possible states w . It was Metropolis in 1953 that first proposed a method for calculating the distribution of a system of particles at thermal equilibrium using a computer simulation method. In this method assuming that the system is in a configuration q having energy $E(q)$, a new state r having energy $E(r)$ is generated, by moving one of the particles from its position: the new configuration is then compared with the old one, if $E(r) \leq E(q)$ the new state is accepted; if $E(r) > E(q)$ it is not rejected, but is accepted with a probability of:

$$e^{-(E(r)-E(q))/kT} \quad (2.2)$$

According to this method, there is therefore a non-zero probability of reaching states of higher energy states, i.e. to bypass the energy barriers that separate global minima from local minima. Note that the exponential function expresses the ratio between the probability of being in configuration r and the probability of being in q . Kirkpatrick used the Simulate Annealing scheme for combinatorial optimization problems. To do this, he replaced the energy with an objective function and the states of physical system of particles with the solutions of a minimization problem. Minimization is achieved by first heating the system, and then slowly cooling it until it reaches the crystallization situation in a stable state. Is to emphasize the fact that in order to reach low-energy configuration, it is not sufficient simply to lower the temperature; it is necessary on the contrary to wait for each transition temperature a time sufficient to reach the thermal equilibrium of the system. Otherwise, the probability of reaching low-energy states is significant reduced.

Implementation of the Simulated Annealing Process

The Simulation of the Annealing process applied to optimization problems requires several preparatory steps. First of all, the optimization problem must identify analogies with physical concepts: energy becomes the objective function, configurations of the particle become the configuration of the design variables, searching for a state of minimum energy means searching for a solution that minimized the objective function and, temperature becomes a parameter control. Therefore, an appropriate annealing scheme must be chosen consisting of the adjusting the parameters on which the optimization process depends, i.e., it is a matter of establishing the temperature decay law and the length of time required for the reaching thermal equilibrium at each temperature. In the end, it is necessary to introduce a method of perturbing the system that allows the search space to be explored generating new configuration. The annealing algorithm described by Kirkpatrick consists in the repeated execution of the Metropolis algorithm for each temperature of transition, until the optimal configuration of the system is reached, obtained in proximity of the prefixed crystallization temperature. The Metropolis algorithm, which accepts configuration of lower quality than the current best solution, provides the probabilistic mechanism that avoids entrapment in local minima.

Based on the above consideration it is possible to identify three major components to a Simulated Annealing Implementation:

- 1) Problem Representation:
 - A means to represent the solution space
 - Function to measure the quality of any given solution

2) Transition mechanism:

- Identify a simple transition mechanism to slightly modify the current solution

3) Annealing Schedule:

- Initial system temperature: typically $t_1 = \infty$ or a high value
- Temperature decrement function: typically $t_k = \alpha \cdot t_{k-1}$, where $0,8 \leq \alpha \leq 0,99$
- Number of iteration between temperature change – typically between 100 to 1000 iterations
- Acceptance criteria
- Stopping criteria

An essential feature of the SA algorithm is the annealing schedule or cooling schedule. During the entire time the algorithm runs the temperature parameter, T , is gradually reduced. Initially, T is set to a high value or infinity and is decreased at each step according to the annealing schedule, which is usually specified by the user, but must end with $T = 0$ at or close to the end of the allotted time budget. In this way, the system is expected to initially wander over a broad region of the search space containing good solutions and ignoring small features of the energy function. As the temperature cools, the system then drift towards low-energy regions that become narrower and narrower, finally making only downhill moves.

A diagram of the Simulated Annealing algorithm is shown below:

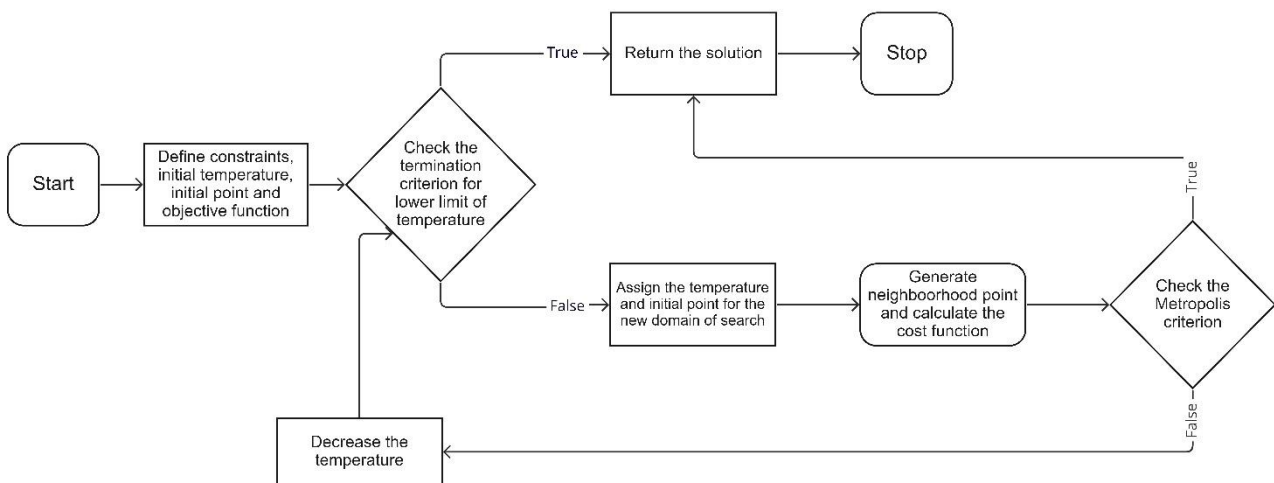


Figure 3: Flowchart of the Simulated Annealing algorithm

3.3. Genetic Algorithm

Genetic Algorithms (G.A.s), proposed in 1975 by J.H Holland, are a computational model idealized on the evolutionary theories of Darwin, presented in his book "On the Origin of Species by Means of Natural Selection" of 1959, and are by far the most successful metaphor used in the development of optimization algorithms. This type of algorithm is based on the Darwinian principle that those elements most "adapted" to the environment have a greater chance of surviving and transmitting their characteristics to their successors; in practice, there is a population of individuals that evolve from generation to generation through mechanisms similar to sexual reproduction and to gene mutation. Each *individual* of the population has its properties externally manifested which constitute its *phenotype*. The *phenotype* is determined by the invisible genetic heritage or *genotype*, consisting of the genes, which are the fundamental units of the *chromosomes*. To each gene corresponds, in general, a characteristic *phenotype*. The two fundamental principles of evolution are *genetic variation* and *natural selection*. For the evolution of the population, the individuals that constitute it must first have a wide variety of phenotypes and this genotype. Selection can be started, which rewards the survival, longevity, and reproduction of the most suitable individuals. The mechanism generating genotype diversity are twofold: *crossover* and *mutation*. These operators, together with the number of maximum generations and the population size, constitute the basic parameters of the algorithm, and are applied with different probabilities until the new population has reached the desired size. To summarise a typically Genetic Algorithm consist of a finite *population of individuals*, representing the candidate solutions to the problem, an objective function which provides a measure of the individual's ability to adapt to the environment, i.e., it constitutes an estimate of the goodness of the solution and an indication of the most suitable individuals to reproduce by a series of operators that transform the current population into the next, by a criterion of termination and by a series of control parameters. A simple diagram of a genetic algorithm works is illustrated in the figure below.

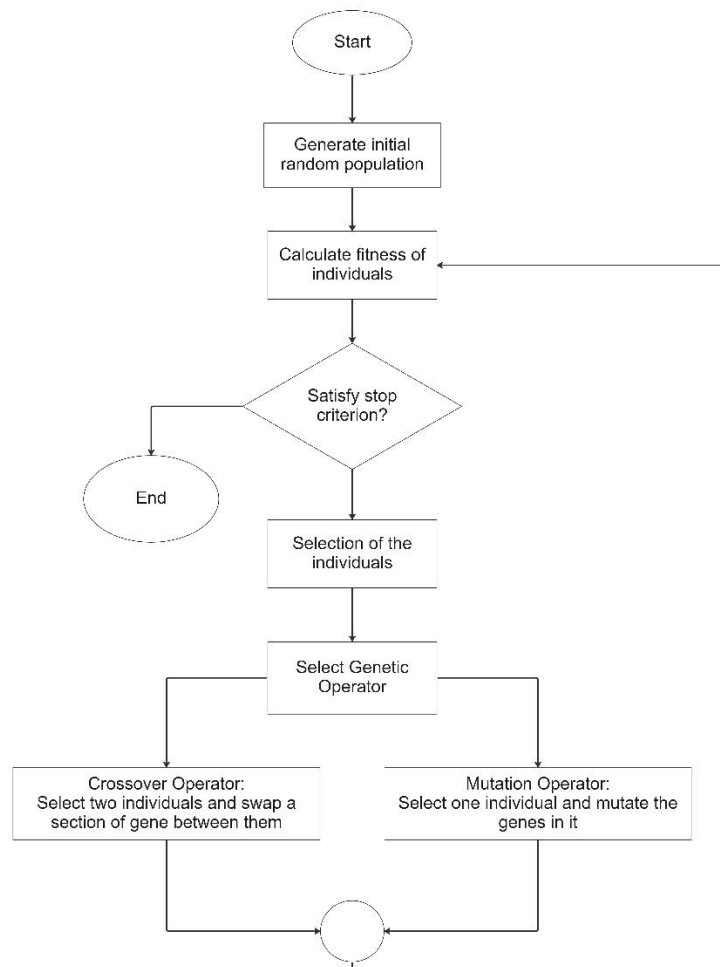


Figure 4: Flowchart of the Genetic Algorithm

Population

A basic element of genetic algorithms is the population that consists of a fixed number of individuals and is generated, at the start of the algorithm, at random. There are several variants of the genetic algorithm, dependent on the management of the population. If the entire population is replaced entirely by new elements, one speaks of *generational*, otherwise, one has *steady-state*. In the *island* model, on the other hand, one has a series of independently evolving populations with occasional migrations of individuals from one "island" to another. Each individual of the populations are the encoded solutions of the optimization problem. Encoding of chromosomes is the first step in solving the problem and it depends entirely on the problem itself. The process of representing the solution in the form of a string of bits that conveys the necessary information. Just as in a chromosome, each gene controls a particular characteristic of the individual, similarly, each bit in the string represents a characteristic of the solution. The most popular method is to use binary coding in which a

binary string corresponds to each solution. Each bit in the binary string stands for the gene. Continuing the similarity with genetic theories, the bit string of an individual represents the individual's *genotype*, while the *phenotype* indicates the individual's behaviour and is entirely domain-dependent.

Chromosome 1	110101110010
Chromosome 2	011010011101

Table 1: Example of Binary Encoding

Parent Selection

As discussed in previous paragraphs, chromosomes are selected from the population to be parents to crossover. The problem is how to select these chromosomes. According to Darwin's theory the best ones should survive and create new offspring. Parent selection is very crucial to the convergence rate of the GA as good parents drive individuals to a better and fitter solutions. However, care should be taken to prevent one extremely fit solution from taking over the entire population in a few generations, as this leads to the solutions being close to one another in the solution space thereby leading to a loss of diversity. Maintaining good diversity in the population is extremely crucial for the success of GA. This taking up of the entire population by one extremely fit solution is known as premature convergence and is an undesirable condition in a GA. In GA there is a trade-off between exploitation and exploration. Exploitation simply refers to the convergence quality of the algorithm. If we select an algorithm that always chooses the fittest individuals to survive, then we are exhibiting high exploitation but low exploration. High exploitation leads to quick convergence; but, it can also lead to premature convergence. In opposition to exploitation is Exploration, this selection type does not always choose the fittest individuals to mate, encouraging exploration of the input space. However, by encouraging exploration, the algorithm can have poor convergence qualities.

There are many methods how to select the best chromosomes for example: *tournament*, *roulette*, and *stochastic selection* and *elitism*.

Tournament selection: Tournament selection randomly selects two individuals from the population and compares their fitness value. The one with the best fitness is allowed to mate, the loser is replaced with the offspring. Another tournament is used to select the other parent. Two parent are selected using two two-member tournaments.

Roulette Wheel Sampling: Under Roulette Wheel Sampling, each member is assigned a slice of a roulette wheel where the size of the slice is proportional to the individual's normalized fitness. The wheel is spun up to

N times where N is equal to the number of members in the population. The parent is then selected based on the cumulative sum of fitness. Two parents are selected using two roulette wheel samplings. Member replacement is linear, that is, the GA replaces members 0 and 1 with the children of the first two parents selected, then 2 and 3, and so until the next generation is created.

Stochastic Remainder: Stochastic remainder is a probabilistic selection method – similar to a weighted roulette wheel. Each individual is assigned as many copies as the integer part of its expected number of instances. Then, for the unoccupied slots in the next generation, a roulette wheel-like selection is carried out based on the residues of the expected number of instances of each individual.

Elitism: Elitism's goal is to address the problem of losing the individuals with the best fitness during the optimization phase of the GA. Often, this is done by preserving the best individuals from the old population and combining them with the best of the new population. This is an opposition to the other methods that replace the entire old population with individuals created by recombining them.

Crossover

The crossover operation is the analogue of sexual reproduction in Darwin's evolutionary theory and provides to create new individuals or produce offspring to the next generation. The crossover operation randomly cuts two parents into 2 pieces and joins two offspring from these parts. Two new off-springs are then generated. The number of pieces can vary from one point to two point crossovers. The crossover happens by some probability, p_c , and by $1 - p_c$, where parents are directly copied to the next generation. The new off-spring have a better chance to be better than their parents. Usually, the value of p_c is between 0,6 and 0,8 and is known as the magnitude of crossover probability.

The one-point crossover randomly chosen a position of the string and generates children in the following way: assuming the string is of length L , we choose k it at random in the range $[1, L]$ then the first child will have genes from $[1, k]$ of the first parent and $[k + 1, L]$ of the other, and the second the reverse. Another widely used type is the two-point crossover, in which strings are exchanged, using a procedure similar to the previous one, between two points chosen at random. An example of one and two point crossover is shown in the figures below. The crossover is the driving force behind the genetic algorithm and is the operator that most influences convergence, although its too unscrupulous use could lead to premature convergence of the search on some local minimum.

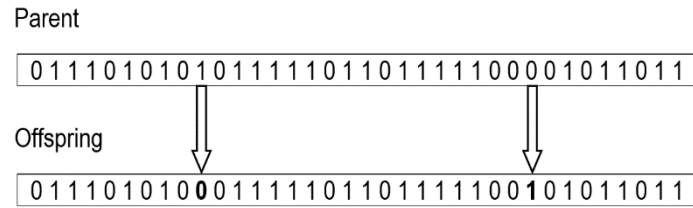


Figure 7: Example of mutation operator

The mutation operator is illustrated in the figure above. In this example, two bits of the parent individual are inverted by mutation. In general, for $p_m = 1/l$, one bit on average is expected to mutate per individual, but in principle also multiple mutations are possible with exponentially decreasing probability. It should be noticed that this is an important property of the mutation operator, because changing multiple bits at the same time at least in principle facilitates the algorithms escape from local optima, even if the probability of this to happen might be vanishingly small. Since the genetic algorithm is a stochastic search algorithm, it is possible that the best individual will not always survive. However, this can be prevented by using elitism in which the best individual is automatically copied to the next generation. For example, the worst individual in the population can be replaced by the best one from the previous generation. In the genetic algorithm, the initial population is usually selected randomly. The usual optimization criterion is to wait for a certain number of generation to be full and then optimization can be terminated. Other possibilities are to observe the fitness function value does or improve any-more or all individuals are too similar the optimization can be stopped.

3.4. Estimation of distribution algorithms

EDAs are non-deterministic, stochastic heuristic search strategies that form part of the evolutionary computation approaches, where number of solutions or individuals are created every generation, evolving once and again until a satisfactory solution is achieved. The characteristic that most differentiates EDAs from other evolutionary search strategies such as GAs is that the evolution from a generation to the next one is done by estimating the probability distribution of the fittest individuals, and afterwards by sampling the induced model. This avoids the use of crossing or mutation operators, and the number of parameters that EDAs require is reduced considerably. In EDAs, the individuals are not said to contain genes, but variables whose dependencies have to be analysed. Also, while in other heuristic methods from evolutionary computation the interrelations between the different variables representing the individuals are kept in mind implicitly, in EDAs the interrelations are expressed explicitly through the joint probability distribution associated with the individuals selected at each iteration. The task of estimating the joint probability distribution associated with the database of the selected individuals from the previous generation constitutes the hardest work to perform, as this requires the adaptation of methods to learn models from data developed in the domain of probabilistic graphical models. In the present work the algorithm used for the optimization of the steel arch compared successively with GA and SA is based on an iterative update of a Gaussian Mixture Model (GMM) (M.M. Rosso, J. Melchiorre, 2022). In structural design optimization problems, it is very common that the best solution is very close to the constraint function. The main advantage of applying the EDA for constrained optimization problems is that each generation of solutions is obtained starting from a PDF that is defined on the whole domain. This means that, for each generation, the solutions have a probability to be on the unfeasible domain space, maintaining the information about the objective function in the evolution process.

A simple diagram of EDA according to the methodology developed by (Rosso et al., 2022) is shown in the figure below, and can be summarised as follows:

1. At the beginning, the first population D_0 of N individuals is generated, usually by assuming a uniform distribution (either discrete or continuous) on each variable, and evaluating each of the individuals. (forse qui dovrei scrivere che viene utilizzato il Latin Hypercube)
2. Secondly, a number S_e ($S_e \leq N$) of individuals are selected, usually the fittest ones.
3. Thirdly, the n -dimensional probabilistic model that better expresses the interdependencies between the n variables is induced.
4. Next, the new population of N new individuals is obtained by sampling the probability distribution learned in the previous step.

Steps 2,3 and 4 are repeated until a stopping condition is verified. The most important step of this new paradigm is to find the interdependencies between the variables (step 3). This task will be done using techniques from the field of probabilistic graphical models.

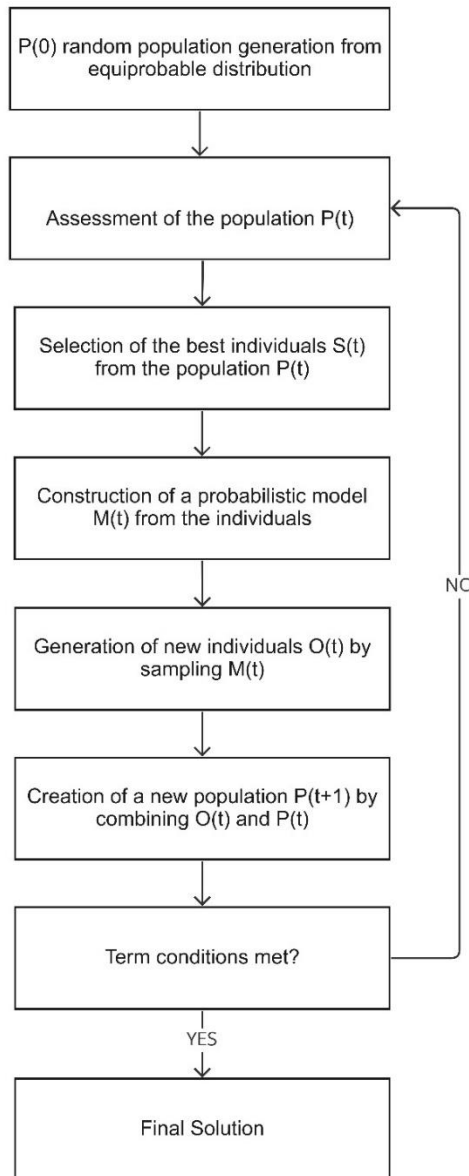


Figure 8: Flowchart of the EDA algorithm

Initial Population

In general, the initial population can be generated by using a random method but, using a population-based meta-heuristic algorithm, the final solution can be very dependent on the initial population. To reduce the randomness that can affect the solution due to a random initial population generation, it is possible to use the Latin Hypercube Sampling method to have a quite uniform distribution of the initial population on the search space. In this way, it is possible to avoid problems related to having all the individuals concentrated in some regions of the search space.

Constraint handling

To obtain a final solution that meets all imposed constraints and, thus, discard all unfeasible solutions, a penalty approach is used. This approach consists of applying a penalty to the score obtained by unfeasible individuals such that, during the sorting phase, feasible individuals are favoured. To push the out – of – constraints individuals toward the feasible region of the search space and, in particular toward the feasible region closest to them, a new method of calculating the penalty function is used. The penalty function is calculated by taking into account the number of violated constraints, the degree of violation of those constraints but, most importantly, by considering the distance between the unfeasible point and the feasible point closest to it.

Gaussian Mixture Model (GMM)

The main idea behind the EDA is to use a probabilistic model to generate the offspring for each iteration. The probabilistic model represents the probability to find the optimal value of the objective function in a certain region of the search space. The probabilistic model which is iteratively generated by the EDA on the population of candidate solutions has been chosen as the gaussian mixture model (GMM), The mixture models provide a semi-parametric tool able to model quite complex unknown distributional shapes, in which the parametric approach dreadfully fails. The GMM can be obtained by a linear superposition of n gaussian PDFs with non-negative mixing proportions or weight π_i , which sum to unity, applied to each PDF component densities p_i of the mixture.

$$p(\vec{x}) = \sum_{i=1}^n \pi_i \cdot p_i = \sum_{i=1}^n \pi_i \cdot N(\vec{x}|\mu_i, \Sigma_i) \quad \text{with} \quad \sum_{i=1}^n \pi_i = 1, \quad 0 \leq \pi_i \leq 1 \quad (3.3)$$

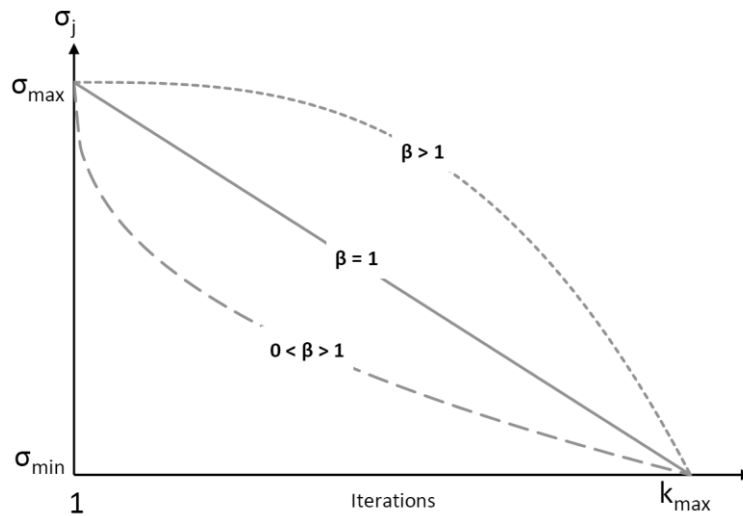
In general, each component density has its own mean and covariance matrix. In this method, to improve the optimization process toward the optima, the mixing weigh π_i , of GMM is related to the Objective Function of each candidate solution in the design space. Denoting k_{max} as the maximum number of iterations, to promote exploration in the early beginning of the iterative optimization process, and promoting exploitation towards the end of the iterative process, the current EDA adopts a dynamic covariance approach. Denoting with σ_j the variance diagonal term of the covariance matrix Σ referred to as the design variable j , the σ_j has been bounded between an admissible range:

$$\sigma_{j,max} = \alpha \frac{|x_j^{ub} - x_j^{lb}|}{m}; \quad \sigma_{j,min} = \alpha \frac{|x_j^{ub} - x_j^{lb}|}{m \cdot k_{max}} \quad (3.4)$$

where the parameter $\alpha > 0$ is a static user-defined hyperparameter that heuristically controls the exploration and exploitation: increasing alfa the exploration is promoted, on the contrary, the exploitation is enhanced, σ_{max} and σ_{min} represent the extremes of the dynamic variation of σ_j during iterations, as depicted in Graph 1. In particular, we will have $\sigma = \sigma_{max}$ at iteration $g = 1$ and $\sigma = \sigma_{min}$ for $g = g_{max}$. To further control these aspects, a continuous dynamic decreasing law of variation for σ_j during iterations $0 \leq k \leq k_{max}$ has been defined as:

$$\sigma_j(k) = \sigma_{j,max} - \frac{\sigma_{j,max} - \sigma_{j,min}}{k_{max}^\beta} k^\beta \quad (3.5)$$

where the parameter $\beta > 0$ another static user-defined hyperparameter which heuristically governs the decreasing law trend during iteration, having three possible outcomes as depicted in the Graph below.



Graph 1: Dynamic variation of σ as a function of β

After defining the GMM distribution, a number of offspring are sampled from this distribution by peaking randomly the PDF components of the GMM and sampling from that normal. Since the GMM is weighted according to the score of each individual, even the sampled offspring will have more chance to be sampled from normal components related to the current best individuals of the population. The entire population considering parents and offspring compete for survival and only the best individuals survive to the next iteration.

4. Integration between parametric design and structural optimization

4.1. Introduction

Previous chapters have discussed structural optimization methods and how metaheuristic algorithms play a crucial role in this context. It also discussed how optimization techniques and technological innovations could support the design and how the search for the optimal solution of a structure is the basis of the design space. It is essential to underline that a parametric definition of the design problem is indispensable within an optimisation process. For instance, the objective function and the constraint functions mainly depend on the set of design variables . This chapter aims to introduce the concepts of parametric design, its close correlation with structural optimization, and how high-performance design tools play a fundamental role in the preliminary phase of a structural optimization process.

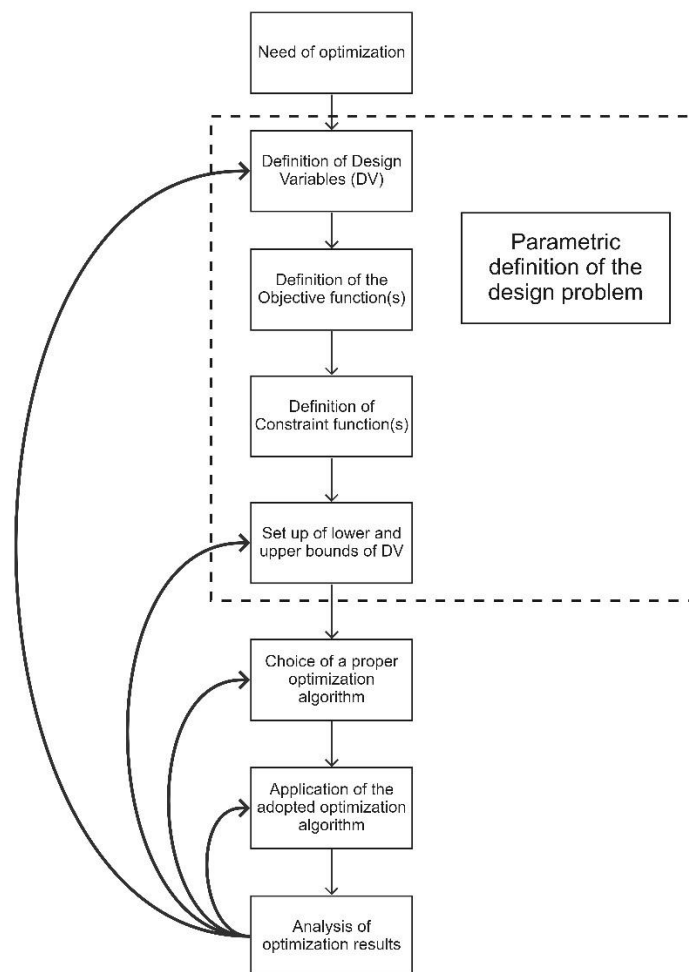


Figure 9: Parametric definition of the design problem

Recent years have seen the development of increasingly powerful modelling and analysis tools that allow architects and engineers to generate and analyse virtually any structural shape. As a result, generating complex geometries and performing advanced analyses require less manual effort and computational time than ever. However, such tools have generally not successfully integrated architectural and structural design into one tool. In particular, most of the current tools are evidence of a paradigm that excludes structural considerations from the earliest stages of architectural design. Such an approach will likely lead to structures with poor performance since the performance of a structure is highly dependent on its geometry. Further, integrating structural design as part of the conceptual design phase has the potential to achieve a synthesis between structural form and architectural geometry, resulting in building forms that are both efficient and expressive architecturally. The combination of parametric modelling and optimization algorithms has emerged as one of the most commonly used computational methodologies regarding conceptual structural design. As a result of the development of interrelated tools, it has organically grown over the past few years. In particular, 3D modelling software, such as Revit or Rhinoceros, can be combined with visual programming interfaces to form parametric modelling environments. These allow the user to script complex generative algorithms without prior programming knowledge and can help steer design space exploration. Furthermore, exploring different solutions can be done promptly as the parametric design process is essentially non-destructive, meaning that one model contains all the previously explored solutions and the ones yet to evaluate. Additionally, these parametric modelling environments can be combined with analysis and optimization components to constitute integrated design environments. Thus, such environments are not solely dedicated to computer-aided drawing but benefit from the numerous available plug-ins to assess the performance of architectural designs according to a wide range of criteria, from building envelope performance to daylighting availability. An integrated environment such as this constitutes a compelling common ground between architects and engineers.

4.2. Parametric Design

Assisted by digital optimisation, engineering and architectural expertise can promote structural awareness regarding design alterations in the conceptual design stages. Building geometry can be set up computationally to render it sensitive to structural input. Parametric design and algorithmic modelling are the keywords of a new paradigm capable of responding to the growing complexity of design problems and the need for optimisation and thus making engineers and architects interact at an early stage of the design process that through an alternative approach, which places the established roles of process and results in a different perspective and sees the computer as the natural ally. The use of parametric design and custom-developed scripts represent the linking of architectural design more closely to structural optimization by streamlining the connection between geometry generation, structural analysis and optimization. In the digital age, architects and designers have been able to refine their design processes through the use of prototyping in an exclusively

digital environment. Prior styles were characterized by using straight lines, sharp edges, and acute angles as their lifeblood. On the other hand, parametric architecture focuses on the principles of free-form architecture, where sweeping lines, curves, and irregular shapes give character and personality to works and buildings that might seem futuristic. Parametric architecture is a design process that is based on a scheme of algorithms, which allows parameters and rules that define and organise the relationship between design requirements and the final product of this process. Woodbury defines parametric design as a complex process that aims to define a design problem as a function of several parameters. For this purpose, the designer must establish the relationships between the parts of the project and define them as function of constant and variable parameters (Woodbury, 2010).

It is helpful to briefly examine the definitions of the terms "parametric" and "design" separately to better understand what parametric design means. Parametric is a derivative of the term "parameter", which originates from the Greek *para*, meaning a subsidiary or beside, and *metron*, as in to measure something (OED, 2002). In mathematics, a *parameter* is defined as a quantity constant in the case considered but varying in different cases. For example, the mathematical description of a particular circle can be expressed through two equations in which there is one parameter, the angle θ , and one constant, the radius of the circle r .

$$x = r\cos\theta$$

$$y = r\sin\theta$$

However, if r is a parameter, we have a potential family of circles with different radii. Other definitions describe a parameter as a characteristic or feature or a constant element or factor (OED, 2002). As a result of these definitions from the dictionary, it becomes possible to give meaning to the term "parameter" as used in this thesis; a parameter may be considered to be any measurable variable that defines a system or establishes the limits of a system. In architecture, design involves a response to a problem; often, the problem's nature is not clear; therefore, the design process also involves developing an understanding of the problem. (Lawson, 2006) recognises architectural design as a "matter of finding and solving problems". Subsequently, many alternative solutions may exist and designing, or finding a solution, becomes a selection process. For (Simon, 1996), this should be a rational choice, but architecture choices based on aesthetics may seem irrational to some. Gero describes the design process as one that "involves exploration, exploring what variables might be appropriate" (Gero, 1990). In summary "design" in this thesis is a task that involves defining a description of a problem, then generating and searching amongst alternatives to find a solution that satisfies the problem. "Parametric design" is understood as a process where a description of a problem is created using variables. By changing these variables a range of alternative solutions can be created, then based on some criteria a final solution selected. On this basis, it could be said all design is parametric. However, for this thesis, the definition needs to include contemporary architectural practice's context. Contemporary design practice is dependent on the

use of computers. The computer supports parametric design by providing a means of defining a model which represents the design problem. Such a model defines the relationship and parameters in the design problem, and by adjusting these parameters, alternative design can be configured. Consequently, parametric design is understood in this thesis as the process of creating a computer model or describing a design problem. This representation is based on the relationship between objects controlled by variables. Changes in the variables will result in alternative models when the variables are changed. The choice of a solution is then based upon a variety of factors, which includes performance, ease of construction, budget considerations, user needs, aesthetic considerations or a combination of these factors, as well as others. A parametric model can be constructed and defined by programming or writing code in a specific programming language. Alternatively, computer-aided design (CAD) applications can be extended to provide parametric functionality using their application programming interface (API). More recently, CAD applications have become available with parametric functionality that the user can control through a graphical user interface (GUI). This kind of application is described as parametric software; typically, these applications provide the option to use a scripting language to customise the parametric functionality further. Parametric software allows users to create relationships and associations between geometric objects and objects that are definitions of variables or functions. This is why parametric design is sometimes referred to as associative design.

4.3. The first parametric design experiments since 1900

Antoni Gaudi

Antoni Gaudi's model of upside-down churches is one of the earliest manifestations of parametric design, whereby the artist created intricate catenary arches using suspended ropes. In his design for the church of Sagrada Familia, he created a model of strings that were hung down with weights to create complex vaulted ceilings and catenary arches. By adjusting the position of the weights, he could alter the shape of the catenary arches and see how it influenced the whole model; he then placed the mirror on the bottom of the model to see how it should look upside down. This is an experiment in structural optimisation because the catenary and its spatial version (the catenoid) represent structures in which a minimum of material is required to obtain maximum performance from reinforced concrete. Gaudi quite literally invented a new kind of parametric design process which can be referred to as analogue computing.

Gaudi's analogue method includes the main features of a computational parametric model:

1. The length and weight of the string, as well as the location of the anchor point, are independent input parameters;
2. As a result of the model, the vertex location of the points on the strings is the outcome;
3. The outcomes are derived by explicit functions, in this case gravity.

Gaudi's work can be seen as an early example of designs based on these curves, which can be represented graphically based on a set of *parameters*, much like those that would create a parabola or other conical cross-section. Gaudi may not have used the mathematical approach to obtain these forms. However, he created the shapes of the Sagrada Familia based on the catenary curve he obtained by hanging chains from the ceiling of his studio and using the shapes he drew in space to define those of the Barcelona church.



Figure 10: Gaudi's upside-down physical model of the Sagrada Familia

Luigi Moretti

The first person who used the term *parametric architecture* was Architect Luigi Moretti in 1940. Moretti gave birth to an idea of architecture based on objective logical-mathematical relations defined in a parametric sense; it is, therefore, a matter of understanding complexity as a cybernetic system in which every alteration of boundary values implies a series of feedback loops that locally and globally modify the system itself. Moretti's stadium model is an early example of how a building's form can derive from 19 parameters. Moretti described Parametric Architecture in *Moebius* two years before his death (Moretti, 1971) as follows:

1. Rejection of empirical decision;
2. Assessment of traditional phenomena as objective facts based on interdependence of expressive, social and technical values;
3. Exact and complete definition of architectural themes;
4. Objective observation of all the conditioning elements (parameters) related to the architectural theme and identification of their quantitative values.
5. Definition of the relationships between the values of the parameters;
6. Indispensability of different skills and scientific methodologies according to the criteria of operational research to define conditioning elements and their quantities;
7. Affirmation of the Architect's freedom in decision and expression, only if it does not affect the characteristics determined by the analytical investigations;
8. Research of architectural forms towards a maximum, therefore definitive, exactness of relationships in their general "structure".

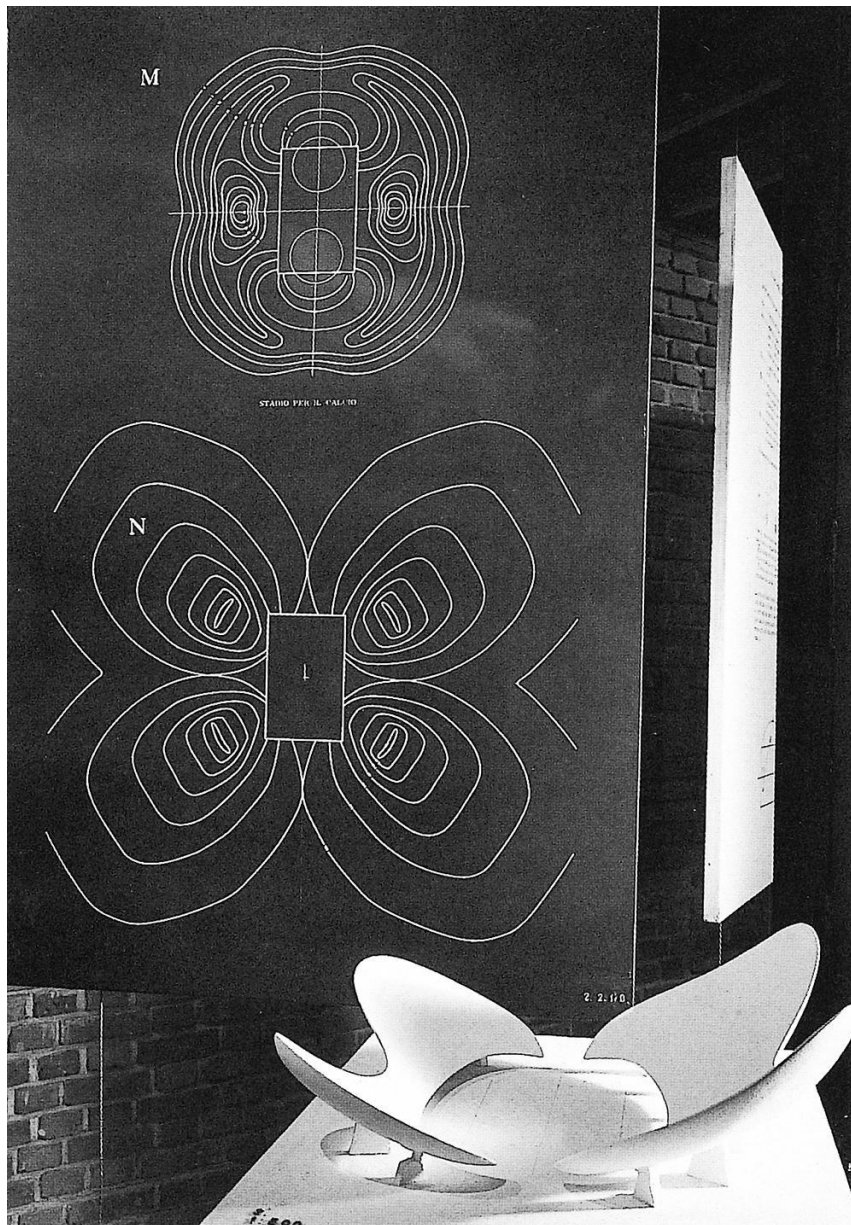


Figure 11: Moretti's Stadium model

Frei Otto

Gaudi's method of computing was enlarged by German Architect Frei Otto which included surfaces derived from soap films and paths found through wool dipped in liquid in the 1960s. Frei Otto was a German architect and engineer considered by many designers as the father of tensile structures. Frei Otto's main activity was experimentation in the laboratory at the Institute of Lightweight Structures in Stuttgart. The first major application of his research came in 1965, when Otto Frei, Rolf Gutbrod and tent maker Peter Stromeyer won the competition to construct the German pavilion for the 1967 Montreal Expo. Frei Otto always tried to use the minimum amount of material to serve the required functions, and the construction of the forms of the openly anti-classical pavilion was reduced to the essentials. Otto calls these experiments form-finding, a phrase that

sets the experimental nature of parametric modelling. For the definition of the shape of the pavilion, numerous physical models were used scale: soap film models for form finding, demonstration models and structural verification models in steel mesh, following a similar process to Gaudi's structural experiments.

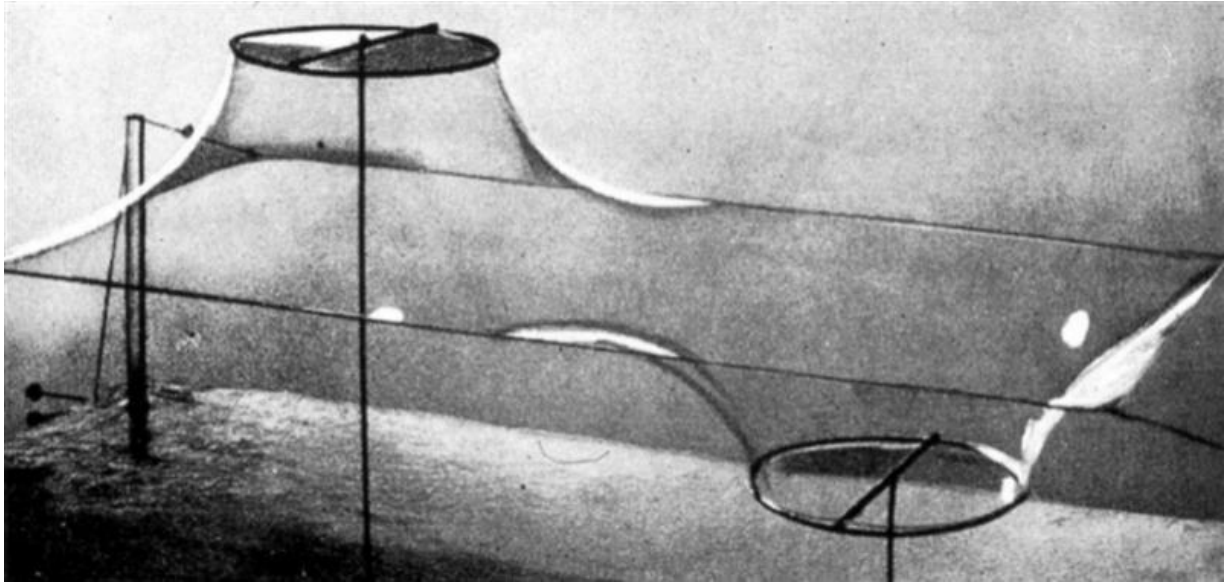


Figure 12: Frei Otto experimenting with soap bubble



Figure 13: German Pavilion, Montreal Expo '67 / Frei Otto and Rolf Gutbrod

4.4. Modern examples of parametric design

Frank Gehry

One of the modern examples of parametric design is the Guggenheim museum in Bilbao by Frank Gehry, performed using the parametric design software Catia. The “Fish” or Peix Olímpic in Barcelona, Spain, is another remarkable example of parametric buildings by the famed architect.



Figure 14: Guggenheim museum - Frank Gehry - Bilbao

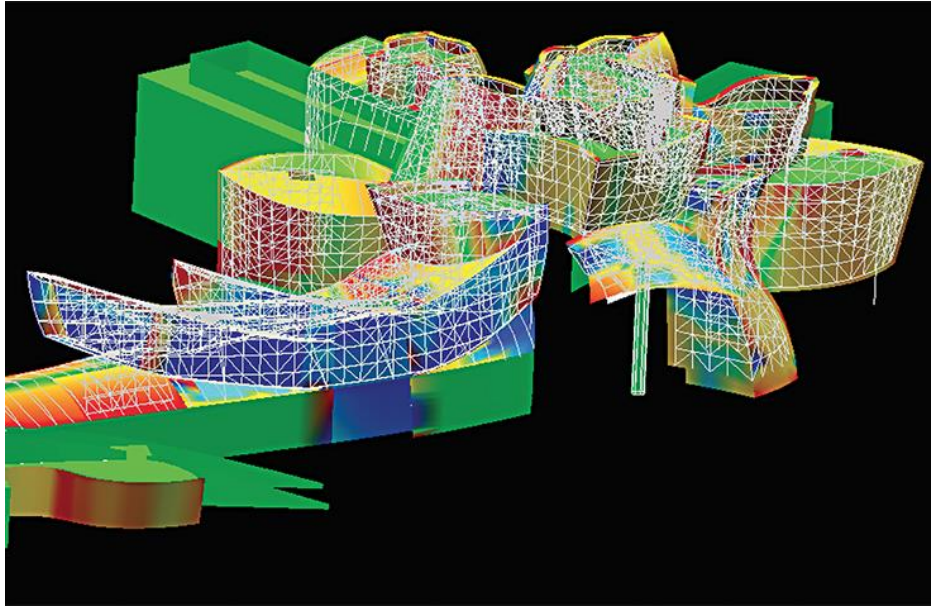


Figure 15: Parametric model of the Guggenheim museum



Figure 16: El Peix - Frank Gehry – Barcelona

Zaha Hadid

Another modern example is Zaha Hadid Architects, who make use of parametric design in virtually all of their design projects, which include the Guangzhou Opera House, the MAXXI Museum in Rome, and the London Aquatics Centre, built especially for the 2012 Olympics.



Figure 17: Guangzhou Opera House – Zaha Hadid - Cina



Figure 18: MAXXI Museum – Zaha Hadid – Rome



Figure 19: London Aquatics Centre - Zaha Hadid – London

4.5. Parametric software

4.5.1. Introduction

The digitalization of the first experiments of parametric design seen in the previous paragraph became available when Ivan Sutherland founded in 1963 the first interactive computer-aided design program: SketchPad, where designers could use a light pen to draw lines, arcs and geometries. For its time, it was a highly innovative system with a high degree of interactivity. The SketchPad was the first to introduce the GUI (Graphical User Interface), non-procedural programming and object programming: technologies used in computers and smartphones today. The final building block for the leap into the digital age is mainly due to the development of new drawing software and a reduction in the cost of computers and computation since the late 1980s. Of particular importance is the introduction of Pro/ENGINEER, a programme developed by Samuel Geisberg for drawing mechanical design. This allowed users to establish input parameters associated with the drawn components. Pro/ENGINEER has the merit of having reduced the cost of changes during design, and overcoming the rigid limitations of three-dimensional modelling. Many designers realised that more sophisticated programmes with the ability to structure certain repetitive practices for the construction of three-dimensional models would have the ability to handle geometries complex geometries beyond human capabilities, as well as saving time. This type of modelling is based on programming languages that express explicit instructions in a form that a computer can execute through a step-by-step procedure, i.e. the algorithm. The algorithm, a term derived from the name of the Persian mathematician Al-Khwarizmi, is a procedure used to obtain a solution to a question – or to accomplish a specific task – through a finite list of instructions.

Algorithms follow the human aptitude to break a problem into a series of simpler problems that can be computed. Although they are strongly linked to the computer world, they can be defined independently of a programming language. Examples of algorithms are cooking recipes, instructions on switching on a modem or even the series of steps to get a cup of coffee. Algorithms, however, remain the structure for the development of any software. When a computer performs algorithmic operations, these are entered into it via specific editors, which may be standalone or integrated within a programme. Examples of standalone editors include names such as C#, Python, and Java, while integrated editors are provided by programmes such as AutoCAD (AutoLisp), Rhinoceros (RhinoScript), Maya (MEL) or others. In recent times, many software houses have developed visual tools to make algorithmic programming more accessible to people without knowledge of programming languages. These programmes are VPL (Visual Programming Language) and are based on the graphical transcription of relations and rules using node diagrams. Examples of such systems are Grasshopper (McNeel), Dynamo (Autodesk), Generative Components (Bentley Systems) and others. Visual scripting favours a unidirectional process where a line can be modelled with two points, a rectangle connecting four lines, etc. This generative logic leads to the “possibility of constructing any network of relations, as long as it is within a domain of parameters describing the element”. Moreover, this parametric diagram can create associative models that reflect different configurations through the control of input parameters. In this thesis, the phases of parametric modelling, structural analysis and optimization with the purpose of investigate the efficiency of various metaheuristics algorithms, such as Genetic Algorithm (GA), Simulated Annealing (SA) and Estimation of Distributed Algorithm (EDA), applied to the case study of a steel arch have been developed entirely in the single virtual environment provided by Rhino by using the following plugins: Grasshopper, Karamba and Galapagos, which will be described in detail in the following paragraphs, instead the workflow and interoperability between the various packages will be described in detail In the next chapter. The parametric Computer-Aided Design frameworks Grasshopper, Karamba, and Galapagos enable architects and engineers to perform early schematic structural consideration and optimization. Nevertheless, the method is rarely used in instructional and design practice settings. In addition, the architectural schematic design process is often distinguished by free forms devoid of material and structural system requirements. On the other hand, materiality and structural system requirements are often enforced by structural engineers, who tend to be engaged as early as possible in the process. In this light, this study investigates the use of structural, investigates the accessibility of existing structural optimization tools, and investigates the interoperability and integration of parametric CAD tools and engineering analysis and optimization tools as well as usability.

4.5.2. Rhinoceros and Grasshopper

In 1969, Robert McNeil & Friends, a privately held and employee-owned U.S.-based company, introduced Rhinoceros, a 3D computer graphics and computer-aided design (CAD) application software suite widely used today by designers, technologists, and business professionals across the world. Rhinoceros is a 3D modelling environment with advanced algorithms and parametric modelling capabilities provided by Grasshopper (GH), a graphical algorithm editor tightly integrated with Rhino's 3-D modelling tools. Unlike RhinoScript, Grasshopper requires no knowledge of programming or scripting but still allows designers to build form generators from the simple to the awe-inspiring. More detailed, Grasshopper is a visual programming plug-in for Rhinoceros developed by David Rutten at Robert McNeel & Associates. It can build an iterative and immersive design process by parametrically modelling artefacts to create a more intuitive, visual, and collaborative design with advanced real-time computer simulations and visualizations that permits to create connections between a generated geometry and certain user-established parameters in a graphical interface. This relations (also known as algorithms) are based on geometric and mathematical concepts. Grasshopper offers a variety of ways to experiment with and construct geometry. It uses Rhino's powerful API (Application Programming Interface) to display geometry in the viewport. Grasshopper allows the integration of custom plugins that can then use the program's GUI elements and thus interact with other plugins in a well-defined way. Several third-party software applications are available, such as structural, energy, and daylight analysis, which can be installed as plugins or as easy-to-use interfaces to standalone applications.

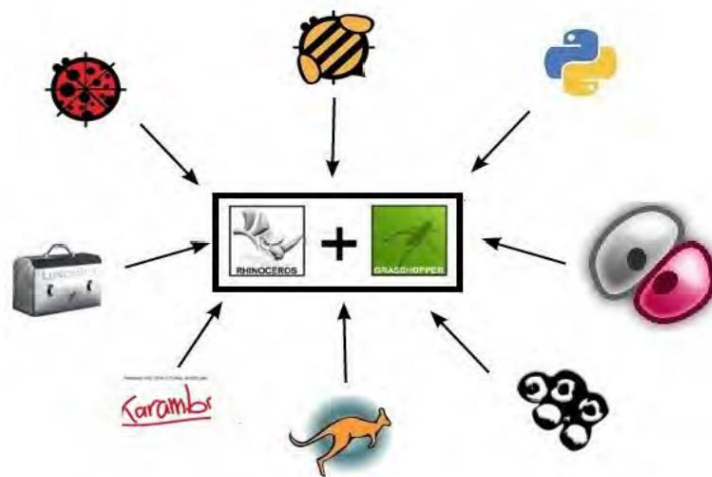


Figure 20: Example of Rhino-Grasshopper interaction with different plugins

Grasshopper's platform allows designers to plug and unplug different functions, depending on the specific design context, and works as a graphical associative logic modeller and algorithm editor; when one of the options is chosen, a part appears in the Grasshopper canvas. Each aspect serves a specific purpose. Most components have buttons for putting in and taking out values, which are plugs. Data is transferred to and from components through wires, resulting in a spaghetti-like script canvas. The workflow began with input and progressed by specifying geometry, load, and support conditions on the Grasshopper canvas, divided into sections and categories. Then the model was assembled and analysed.

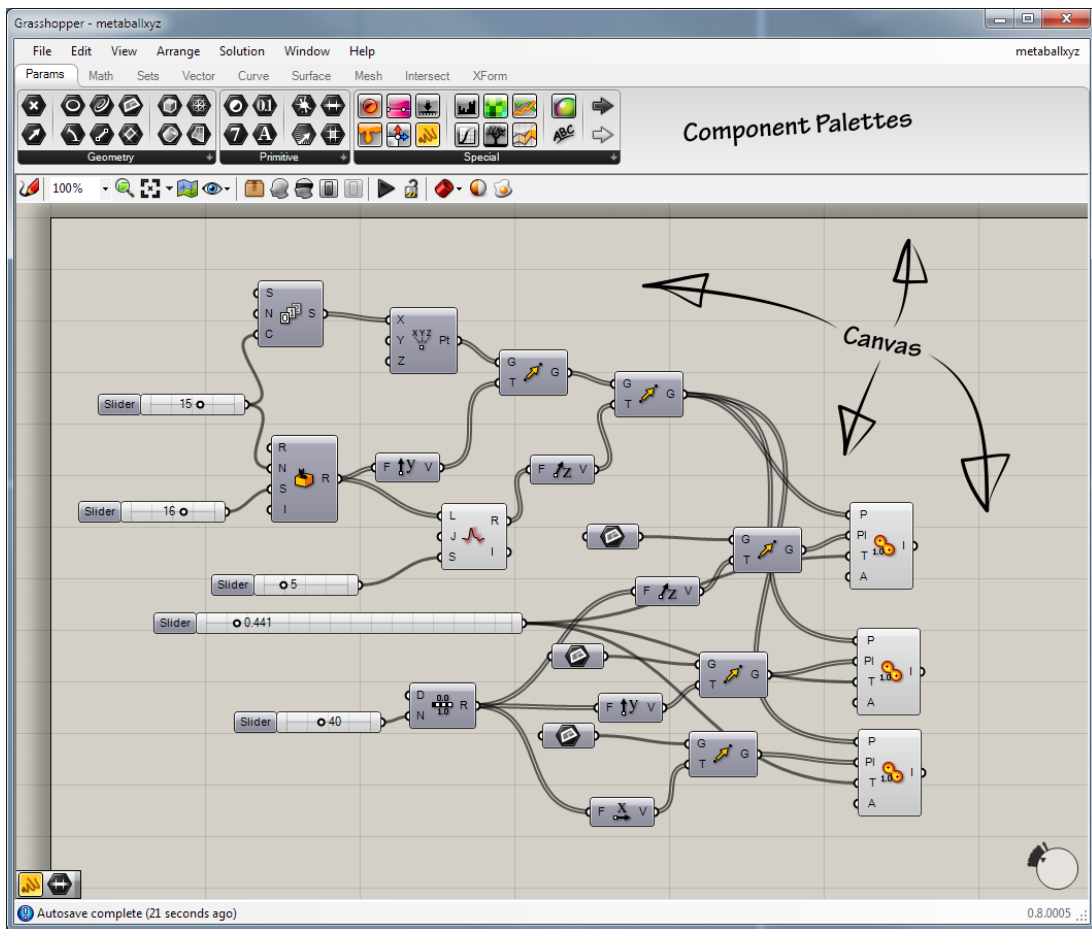


Figure 21: Grasshopper screenshot

1.1.1.Karamba

Among the considerable number of Grasshopper plug-ins, it can be found that several structural analysis tools can be implemented at early project stages. Kangaroo and Karamba shall be highlighted explicitly because of their high performance when calculating and showing results graphically.

Karamba is a very powerful finite element (FE) Grasshopper plug-in for predicting the behaviour of structures under external loads developed from a research project carried out at the University of Applied Arts Vienna in collaboration with Bollinger + Grohmann Engineers. Karamba models can be parametrized in many ways, so they are dependent on parametric input. In general, such a model comprises a group of visual components with the appearance of the native software's building blocks. It generates a model that reacts immediately to any change of input parameters, helping to understand the structural mechanism in the design process element calculations and optimization algorithms like Galapagos. The designer can instantly see the results and evaluate their reliability; conceptual mistakes can be easily found. The program engine allows one to perform calculations using beams and shells, applying any load condition, and determining materials and cross sections, among others. It also includes several calculation algorithms, such as first-order and second-order analysis.

1.1.2. Galapagos

Galapagos (Rutten, 2013) is one of the first released optimization plug-ins for Grasshopper. The tool provides heuristic optimization algorithms, which are the Genetic Algorithm (GA) (Holland, 1975) and Simulated Annealing (SA) (Kirkpatrick et al., 1983). The tool's author suggests SA for rough landscape navigation, whereas evolutionary solver is for finding reliable intermediate solutions. Examples of the design optimization papers that utilized the Galapagos tool in dealing with structure can be found in (Gerbo & Saliklis, 2014) (Sardone et al., 2020). It is necessary to note that in working with solvers, the experience demonstrates that when dealing with complex problems, often the solution provided by the solver is not exact but close to it (Rutten D., 2013) (Tedeschi A., 2014).

5. Set-up of the optimisation problem applied to the case study of an arch structure

The purpose of this chapter is to introduce the case study used in this thesis to achieve its objective: comparing in terms of performance three different meta-heuristic algorithms when applied to a structural optimisation case of a steel arch, as well as to set up the optimisation problem by defining what are: the input parameters, the design variables, the objective function and the constraints. The workflow followed is also presented, emphasising the interoperability of the software used throughout the process. The phases of parametric modelling, structural analysis and optimization have been developed entirely in the virtual environment provided by Rhino, Grasshopper, Karamba and Galapagos software packages. In particular, the novel implementation of EDA presented in (Rosso et al., 2022) was developed in the Grasshopper environment (Gabriele Rosi, 2022), in order to comparing it in terms of efficiency with respect the optimization algorithms present in Grasshopper via the component Galapagos: Genetic Algorithm and Simulated Annealing. For the sake of clarity, it is important to emphasise that the structural optimisation phase of the chosen arch structure requires the definition of the input parameters, the design variables, the objective function and the constraints. The input parameters are the fixed ones defined by the user at the beginning of the process and correspond to the design choices to be defined in the preliminary design phase. The design variables are the quantities that, together with the input parameters, define the geometry configuration of the arch and represent the variables that will be modified at each step by the chosen optimisation algorithm. The objective function is the quantity that the chosen algorithm will attempt to maximize or minimize by modifying the design variables. In this study, it was chosen to minimize the volume of material required to construct the arch structure. Finally, the constraints are equalities or inequalities representing mechanical limits for the optimized structure. In this thesis, the inequality constraints are the applied stresses along the arch structure, according to the following inequality:

$$\sigma_{max}^{VM} \leq f_y$$

The value of the maximum applied Von Mises stress must be smaller than the yield strength of the steel for each point of the structure.

The optimisation process sketched in the graph below was repeated for each of the three optimisation algorithms: Genetic Algorithm, Simulated Annealing and Estimation of Distribution Algorithm, in each of the scenarios described in detail below: 2,3,4 and 5 design variables considered.

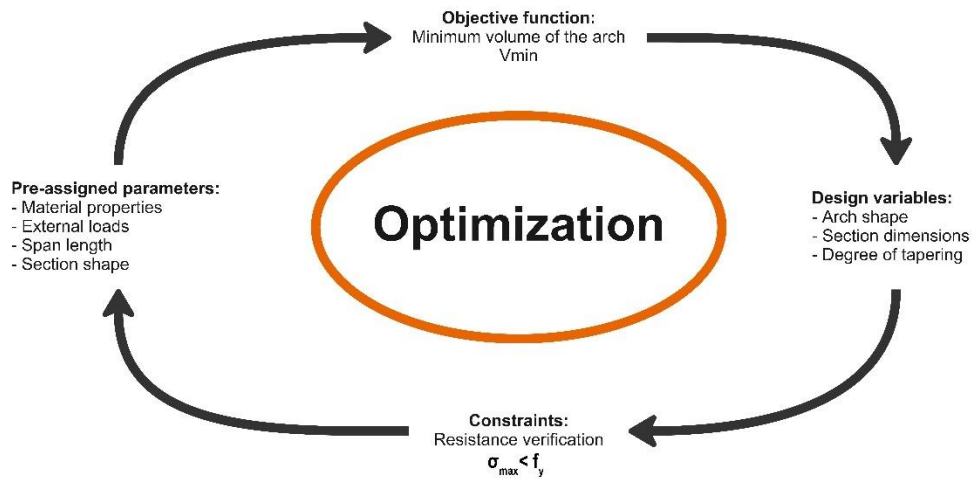
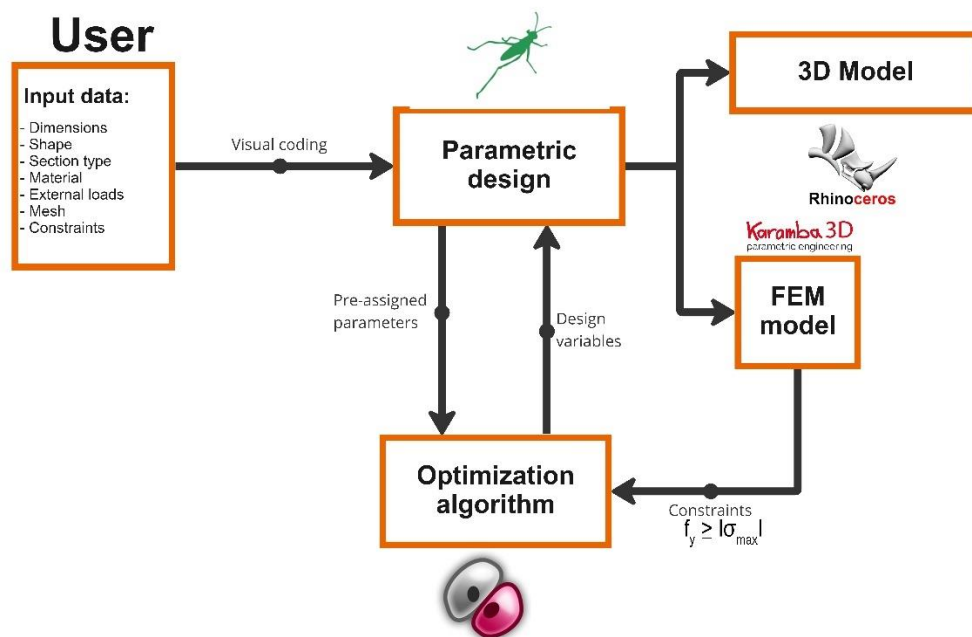


Figure 22: Process of structural optimization

The previous diagram represents the conceptual scheme followed in this thesis for the structural optimisation of the steel arch. In contrast, the diagram below shows the workflow of the entire process that was followed in this thesis to achieve the goal: comparing in terms of performance three different meta-heuristic algorithms when applied to a structural optimisation case of a steel arch. Furthermore, interoperability between the software used is explicitly highlighted in the diagram. Specifically, Grasshopper was used to construct the parametric wire-frame representing the structure's geometry. In contrast, the Karamba tool was used for load and constraint definition and the structural analysis phase. Finally, the optimisation phase was conducted through the Galapagos tool and user sub-routine based on visual scripting.



5.1.1. Arch Structures

The case study chosen for the present work is that of an arch structure because of its fundamental role in structural engineering and architecture, as structural elements provide aesthetics and strength by employing their geometry (Marano et al., 2014). Furthermore, they have been widely used in structural engineering since the early Roman Empire since they are one of the most appealing forms of bridge design and, at the same time, provide resistance and safety. Initially, they were designed with limited materials and a conservative criterion on the safety side, leading to material wasting and a limited maximum span length range. Nowadays, engineers deal with a vast range of solutions and materials; therefore, they must look for an optimal solution in terms of cost and resistance. Due to their importance, many researchers have been working on this topic from the earliest days of continuum mechanics, seeking to determine the optimal arches' shape under different load conditions. As with other structures, the efficiency of arch structures is strongly related to their shape: it is difficult to find another type of structure where the relationship between geometry and internal loading is so evident. In order to reduce structural volume, it is essential to select the appropriate arch geometry.



Figure 23: Hulme Arch Bridge – Manchester

5.2. The setting of the Optimisation Problem

5.2.1. Fixed parameters

Static Scheme

The static scheme considered is that of a fully restrained arch.

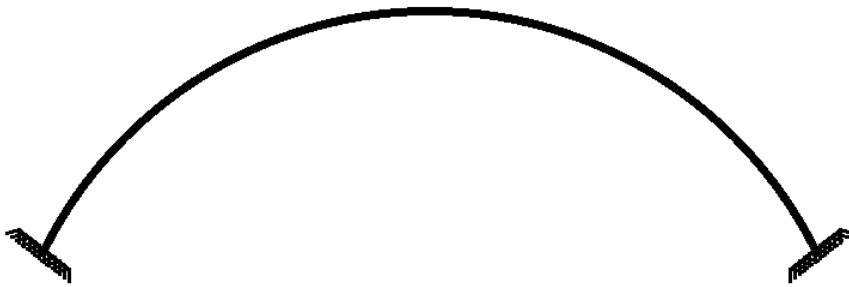
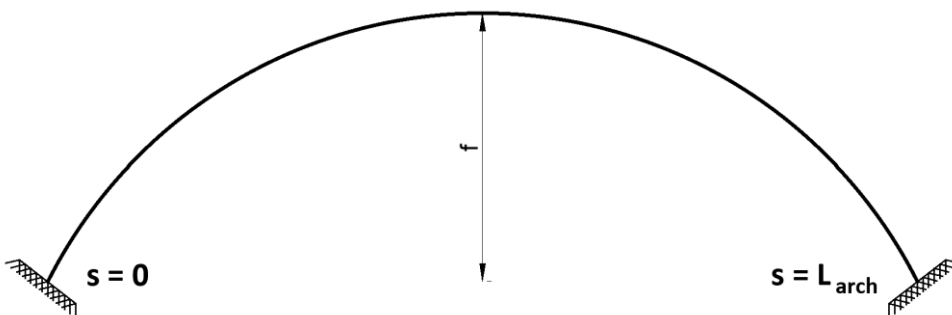


Figure 24: Fully restrained arch

Arch shape

The arch shape chosen for this work is that of a parabolic arch.



Material

A structural steel S355 with the following mechanical properties was considered for the analysis of the structure:

- Density: $\delta = 76,98 \text{ kN/m}^3$
- Elastic Modulus: $E = 210000 \text{ N/mm}^2$
- Yield Strength: $f_{yk} = 355 \text{ N/mm}^2$

Load Condition

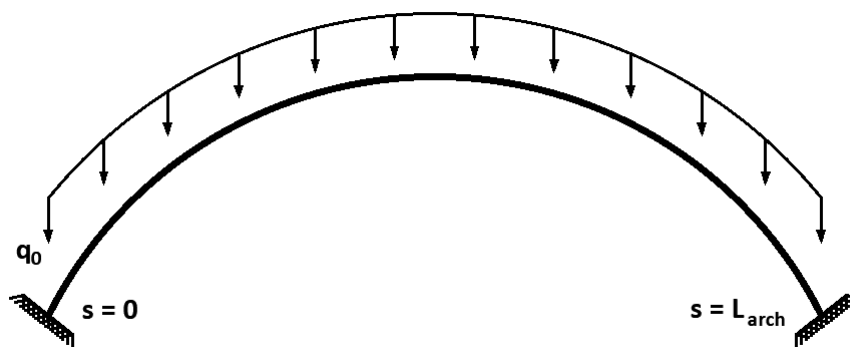


Figure 25:

The load combination considered is as follows:

$$D + q_0$$

Where D is the dead load and q_0 is a variable load equal to 250 kN/m.

5.2.2. Optimization choices and design variables

Possible cross-section

Currently, three different cross-sections of the optimization code are implemented at the time of writing. The cross-sections are chosen so that it is possible to add from one to three design variables as part of the optimization process. For example, the filled circular cross-section is the simplest, as it can be described by just one parameter, r , which represents the radius of the circular cross-section.

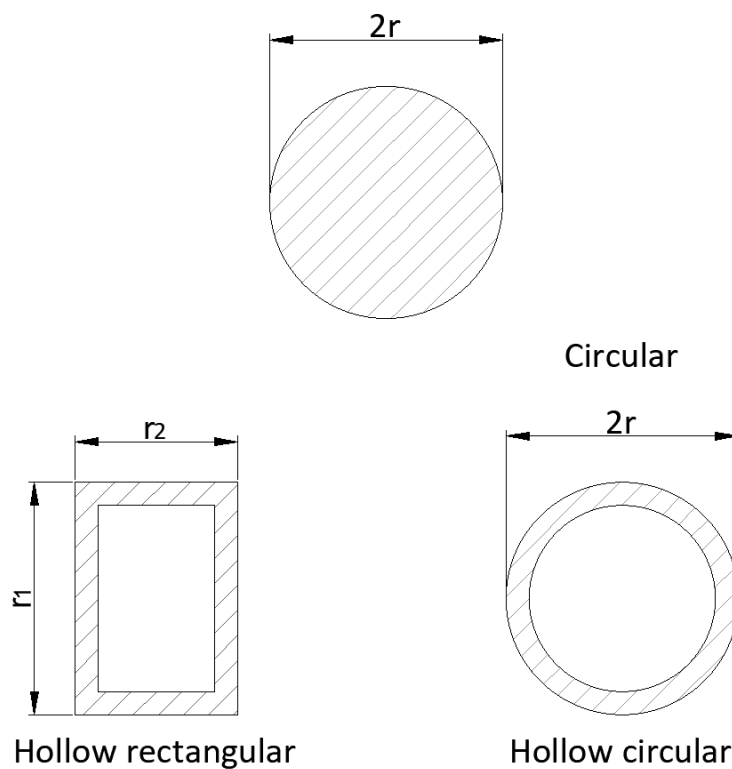


Figure 26: Possible cross-section implemented in the optimisation scripting

The user can choose to increment the number of variables and, consequently, the complexity of the optimization problem by picking between the hollow circular or rectangular cross-sections.

Possible tapering laws

The code also allows the user to choose to have constant or variable section dimensions along the arch. In particular, in case of constant cross-sections, there will be no additional design variables. Instead, the case of the variable cross-section along the arch will add complexity to the problem. The variation is defined by fixing the dimensions of the cross-sections at the base and in the mid-span and then defining a tapering law that allows defining all the cross-sections between these two automatically extremities. This procedure enables the possibility of optimizing the degree of tapering of the structure by adding just one design variable that, is the ration between the main dimension of the cross-section at the base and the cross-section in the mid-span

$$\eta = r_{base} / r_{mid}.$$

The user has the possibility to choose between two different configurations:

- Constant variation;
- Quadratic variation of the section dimensions along the arch.

6. Comparison between three different metaheuristic algorithms applied to the structural optimization of a Steel Arch Structure

6.1. Introduction

This section presents a comparison study on the efficiency of three metaheuristic algorithms such as: Genetic algorithm (Holland, 1975) , Simulated annealing (Kirkpatrick et. Al, 1983) and Estimation of Distribution Algorithm (Rosso et al., 2022). The main reason for comparing algorithms is based on the no-free lunch theorem (NFLT) (D. H. Wolpert and W. G. Macready, 1997) which argues that the performance of an optimization algorithm depends on the nature of the problem. In other words, one algorithm can outperform another algorithm in a specific problem. Thus, NFLT argues that no global optimization algorithm can present the best result for all real-world and benchmark problems (Cubukcuoglu et al., 2019).

For this comparative analysis, the efficiency of the three algorithms was tested on the structural optimisation case study of a steel arch structure, applying the chosen algorithm each time. Cases with 2, 3, 4, and 5 parameters were studied based on the number of design variables considered. In order to have an objective evaluation of the performance of the three different algorithms and a comparative study that is robust and reliable, there have been 20 analyses conducted for each different algorithm in each case. In addition, a sensitivity analysis of the EDA algorithm was conducted for different alpha and beta values (2.4),(2.5). Finally, comparative graphs of the averages and standard deviations were reported to assess the performance of the various algorithms tested objectively.

6.2. Optimisation Results

6.2.1. Configuration with 2 design variables

6.2.1.1. Introduction

The first configuration is characterized by two design variables. Indeed, a parabolic arch with a circular-filled cross-section that has constant dimension along the arch axis.

	User choices		
	Arch shape	Cross-section	Tapering law
Configuration	Parabolic	Filled Circular	Constant
Design variables	1	1	0
Total Design variables: 2			

Table 2: Configuration with 2 design variables

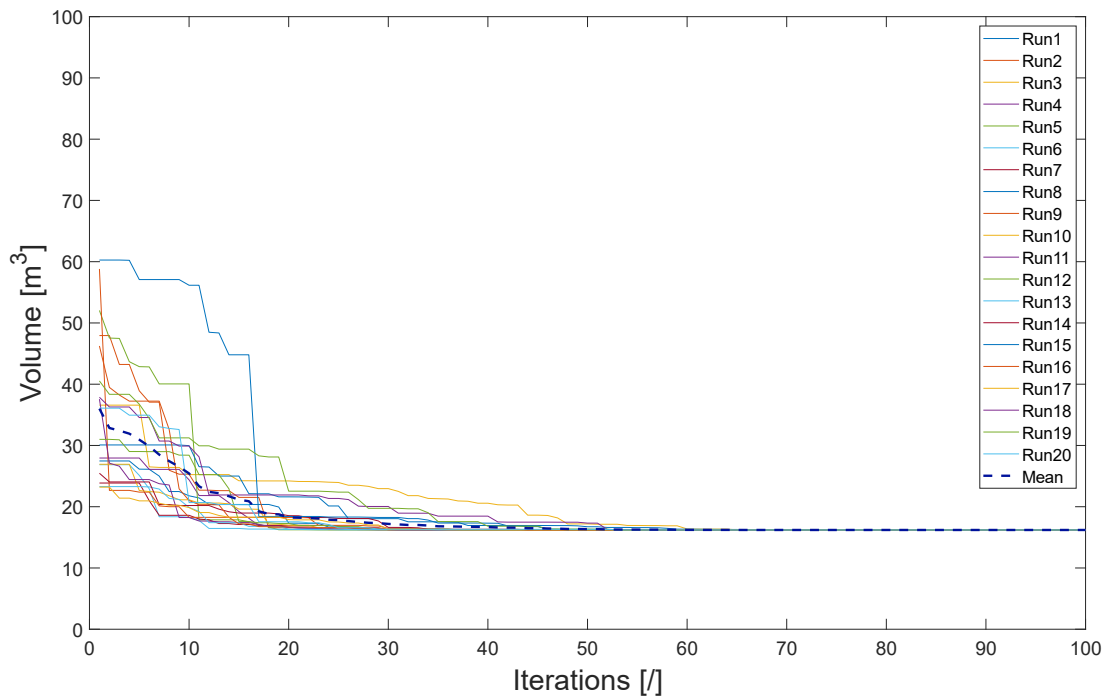
Specifically, in this configuration the design variables are:

- f : Height of the arch in the midspan
- r : Radius of filled-circular cross-section

The results for the three metaheuristics algorithm tested are reported in the following paragraphs. The results will be presented in graphical form and represent the evolution of the objective function as the number of iterations evolve. For this configuration the maximum number of iterations considered is equal to 100. In addition, graphs comparing the averages and standard deviations between the various algorithms will be shown.

6.2.1.2. Genetic Algorithm

For this configuration, 20 independent analyses were performed with a starting population of 50 individuals and a maximum number of iterations of 100.

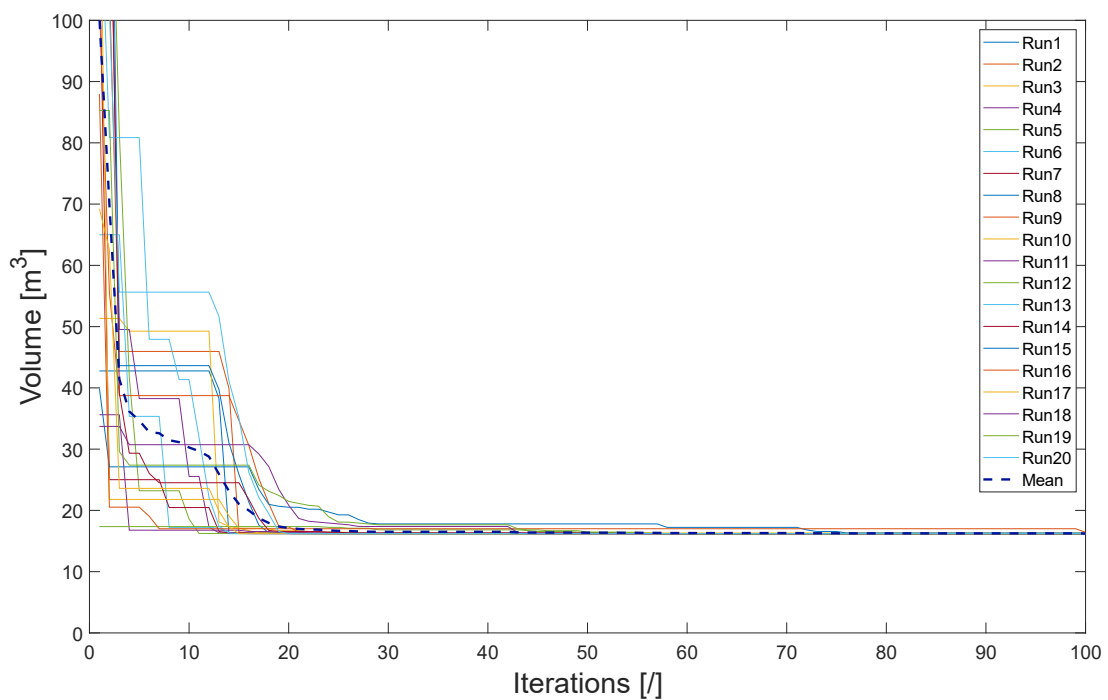


Graph 2: Genetic Algorithm with 2 design variables

Each curve in the above graph represents an independent analysis depicting the evolution of the objective function as the number of iterations changes. In the case under consideration, it can be seen that the curves up to a number of 20 iterations appear very dispersed, converging after approximately 50 iterations. Thus, it is assumed that generally, a good solution with the GA requires at least 50 iterations and an evaluation of the objective function 2500 times. The average value of the objective function evaluated on the 20 analyses after 100 iterations is approximately 16,20 m³.

6.2.1.3. Simulated Annealing

For this configuration, 20 independent analyses were performed with a starting population of 50 individuals and a maximum number of iterations of 100. Since the simulated annealing algorithm is not a population-based algorithm and therefore you do not have a population evolving, to make it comparable with the other algorithms, data were collected in groups of 50 individual. As with the previous algorithm, the average value of the objective function evaluated on the 20 analyses after 100 iterations is about 16,22 m³.

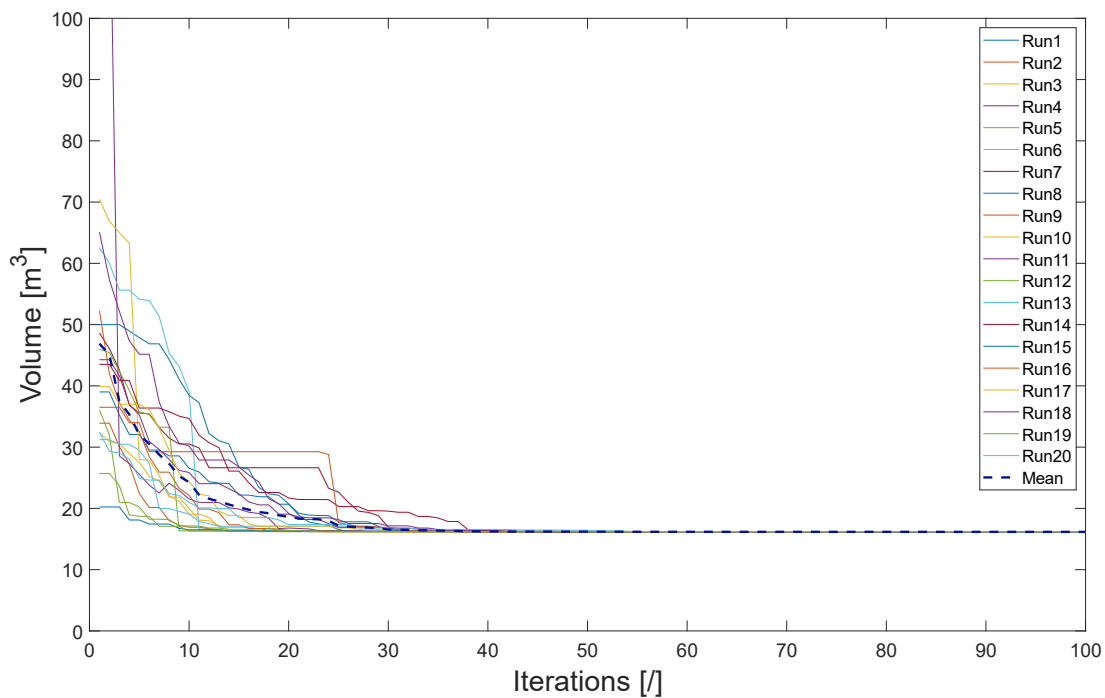


Graph 3: Simulated Annealing Algorithm with 2 design variables

In this case, it can be seen that at the beginning up to 5 iterations, the curves are slightly dispersed and then diverge from each other. Subsequently, after about 20 iterations, they all flatten out and converge. At first glance, the Simulated Annealing Algorithm seems to converge before the Genetic Algorithm.

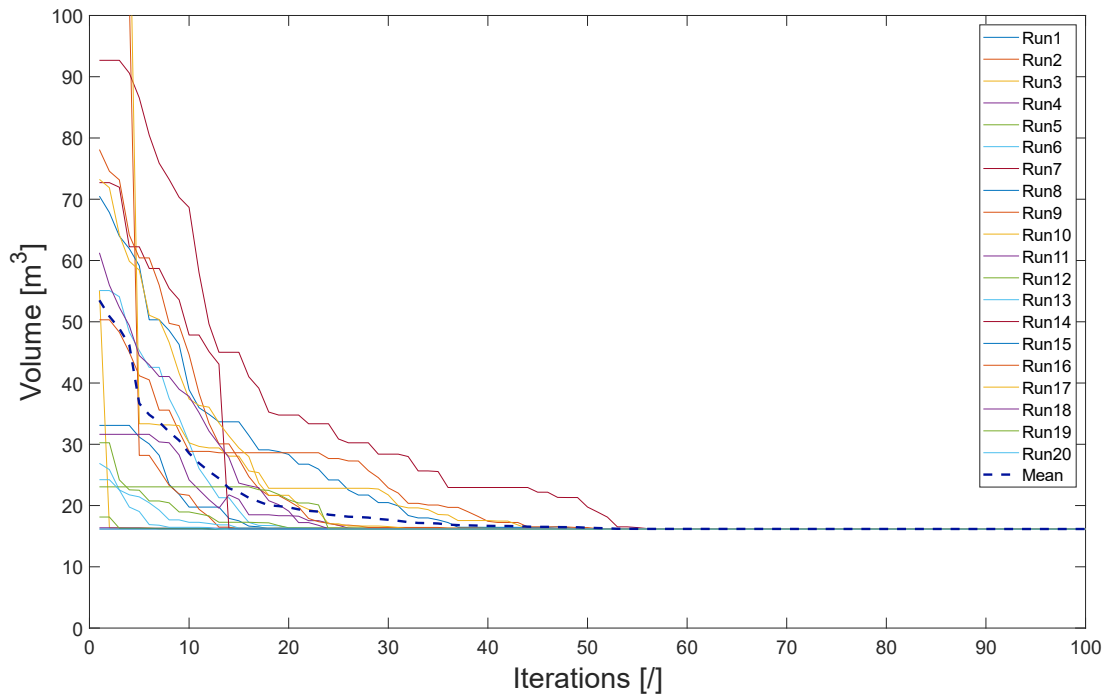
6.2.1.4. EDA

Since the proposed approach allows the user to vary the hyperparameters α and β to govern the dynamic covariance matrix adaptation during the optimization process, the problems have been tested with different values of α and β , exploring the effects of a covariance matrix with a sub-linear, linear or super-linear trends with number of iterations. In the following are reported the graphs of each tested combination of alfa and beta. As in the two previous cases, 20 independent analyses were carried out for each combination of alpha and beta to assess the algorithm's performance under consideration.



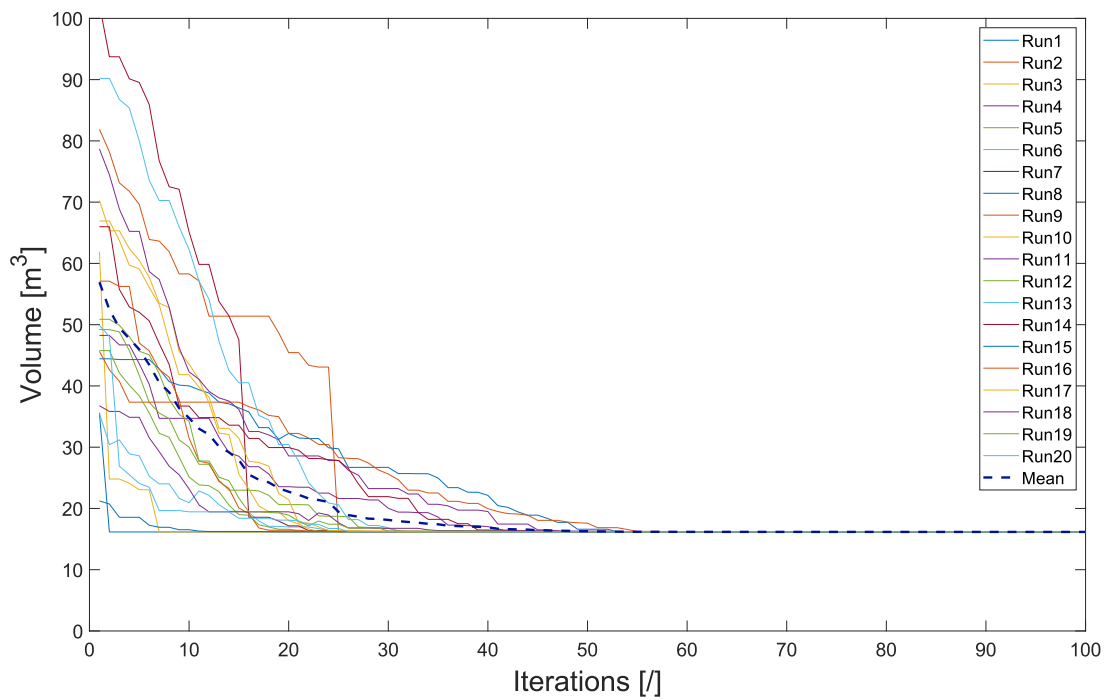
Graph 4: Estimation of Distribution Algorithm with 2 design variables ($\alpha = 0,5$ | $\beta = 0,5$)

The first combination tested is $\alpha = 0,5$ and $\beta = 0,5$, all curves converging on average after about 40 iterations. The results in terms of the objective function seem comparable with those of the two previous cases.



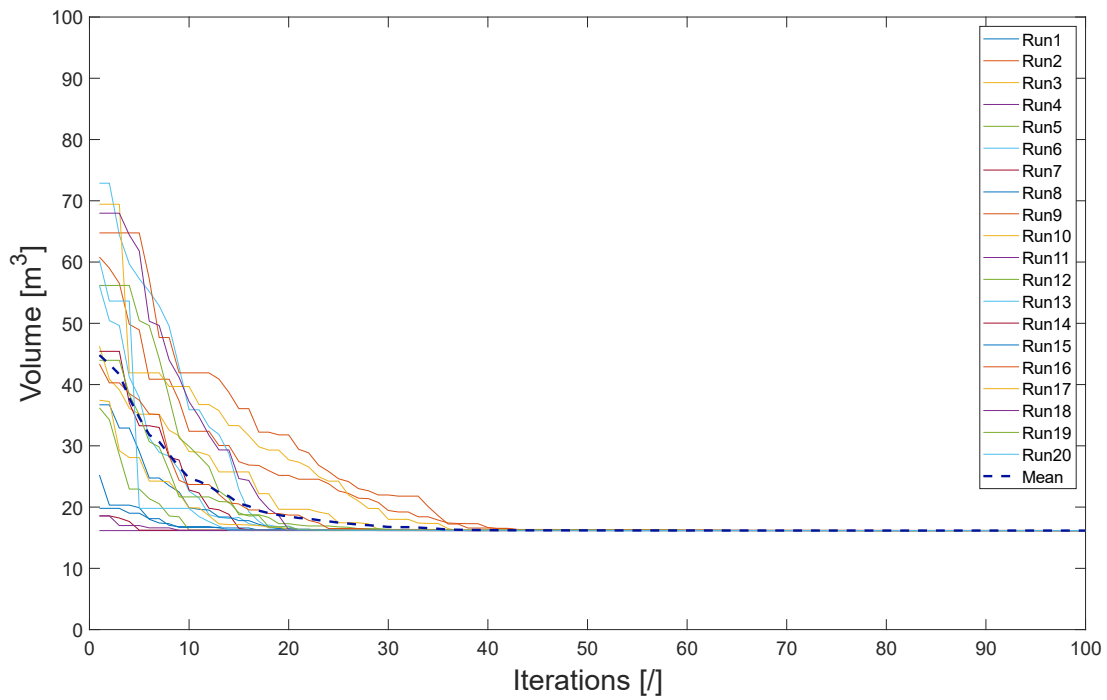
Graph 5: Estimation of Distribution Algorithm with 2 design variables ($\alpha = 0,5$ | $\beta = 1$)

The second combination tested is $\alpha = 0.5$ and $\beta = 1$. Compared to the first combination, the curves appear more dispersed and converge on average after about 45 iterations.

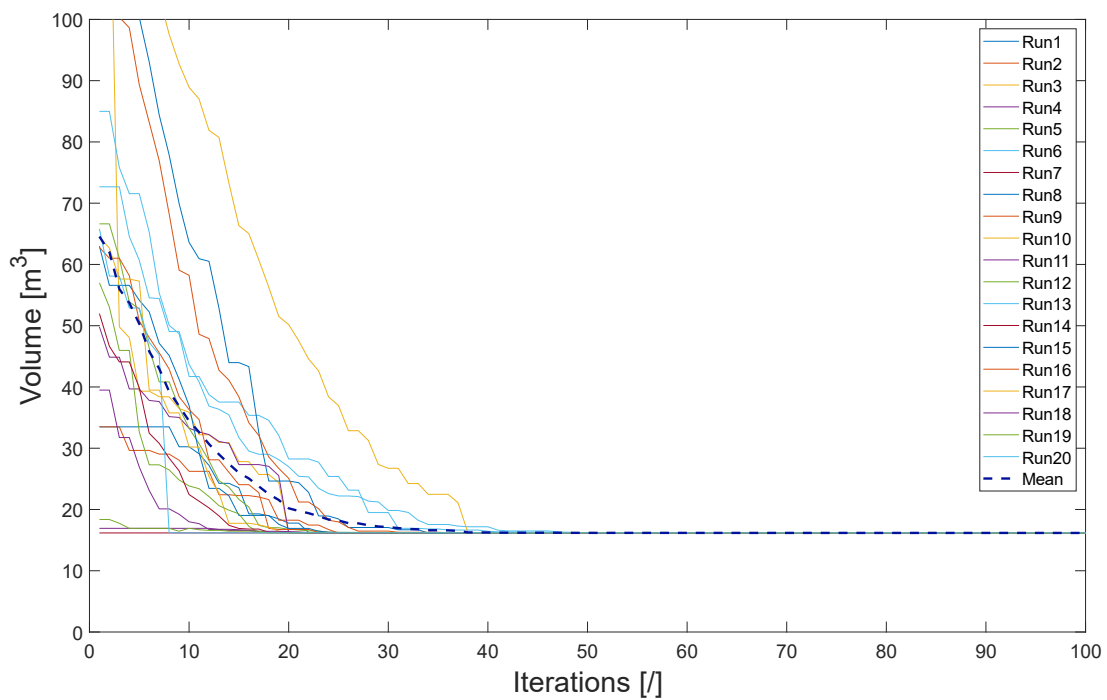


Graph 6: Estimation of Distribution Algorithm with 2 design variables ($\alpha = 0,5$ | $\beta = 1,5$)

The results of the combination $\alpha = 0,5$ and $\beta = 1,5$ appear more dispersed than the previous ones. The algorithm converges on average after about 55 iterations.

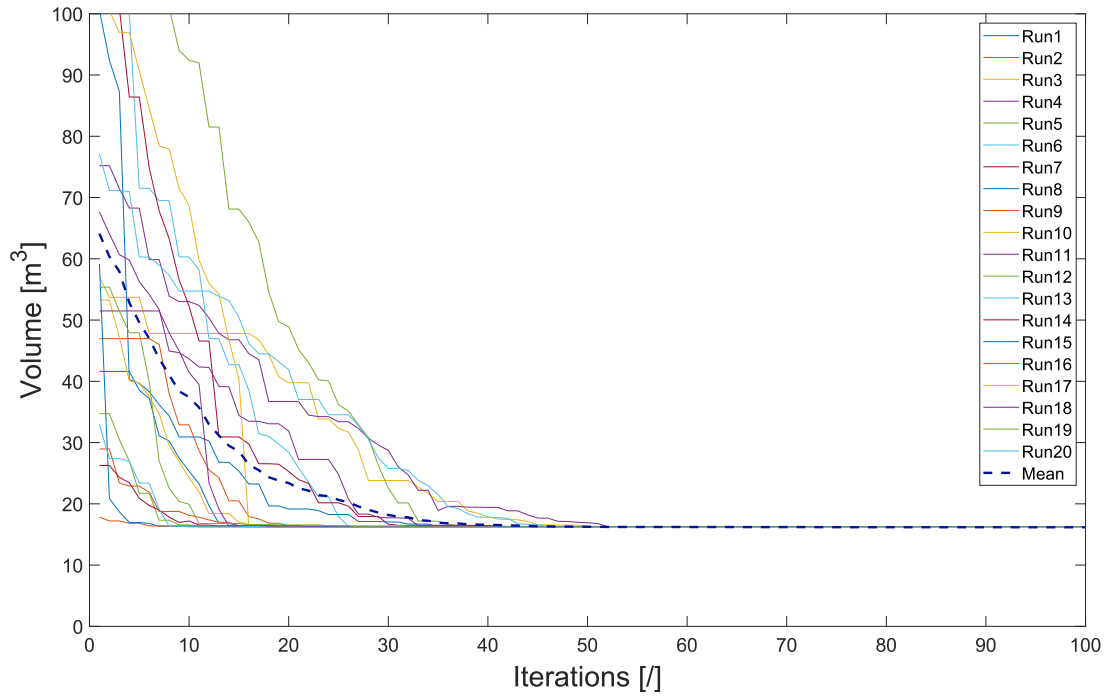


Graph 7: Estimation of Distribution Algorithm with 2 design variables ($\alpha = 1 \mid \beta = 0,5$)

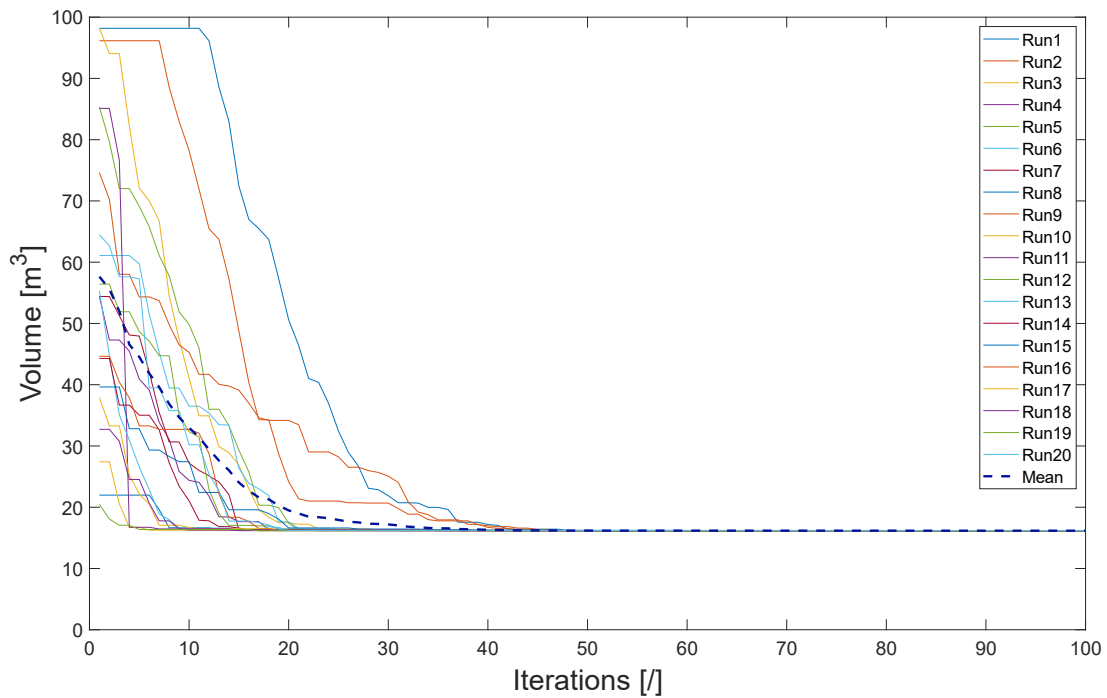


Graph 8: Estimation of Distribution Algorithm with 2 design variables ($\alpha = 1 \mid \beta = 1$)

The results of the combinations $\alpha = 1, \beta = 0,5$ and $\alpha = 1, \beta = 1$ appear very similar, although the latter has more scattered data. Both combinations converge after about 40 iterations

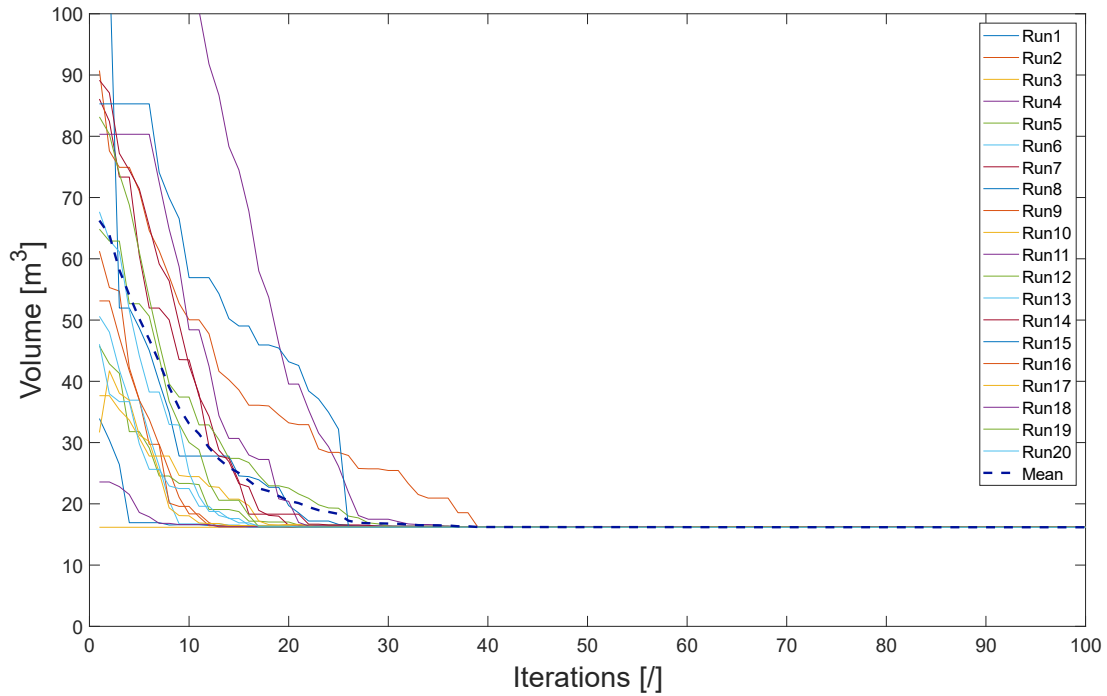


Graph 9: Estimation of Distribution Algorithm with 2 design variables ($\alpha = 1 \mid \beta = 1,5$)

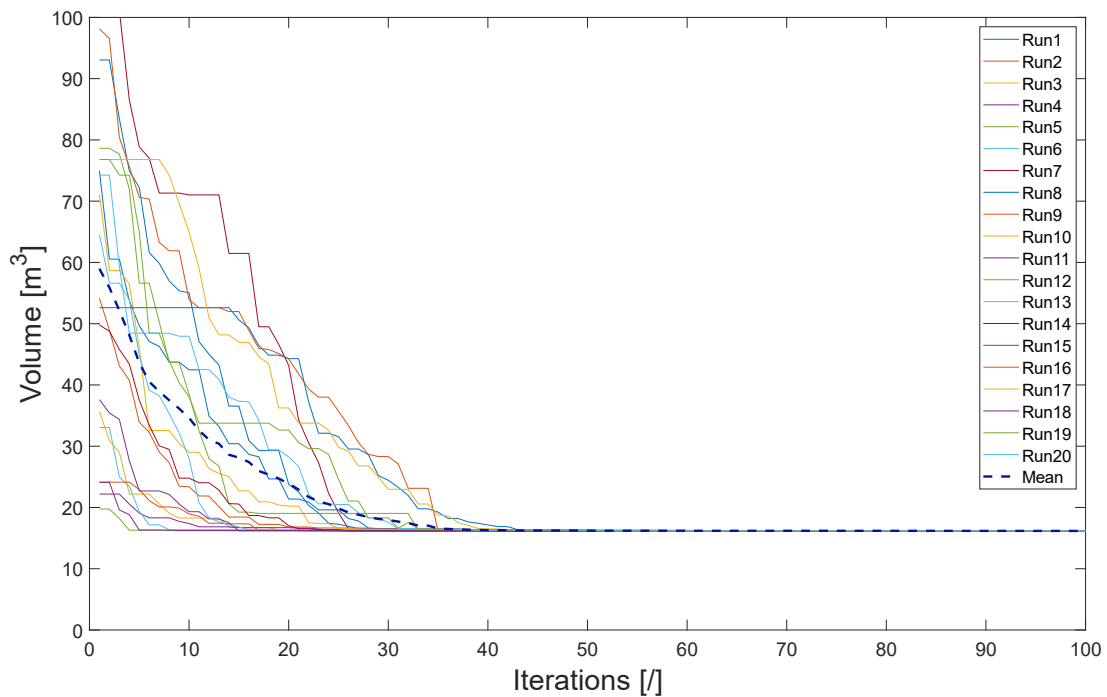


Graph 10: Estimation of Distribution Algorithm with 2 design variables ($\alpha = 1,5 \mid \beta = 0,5$)

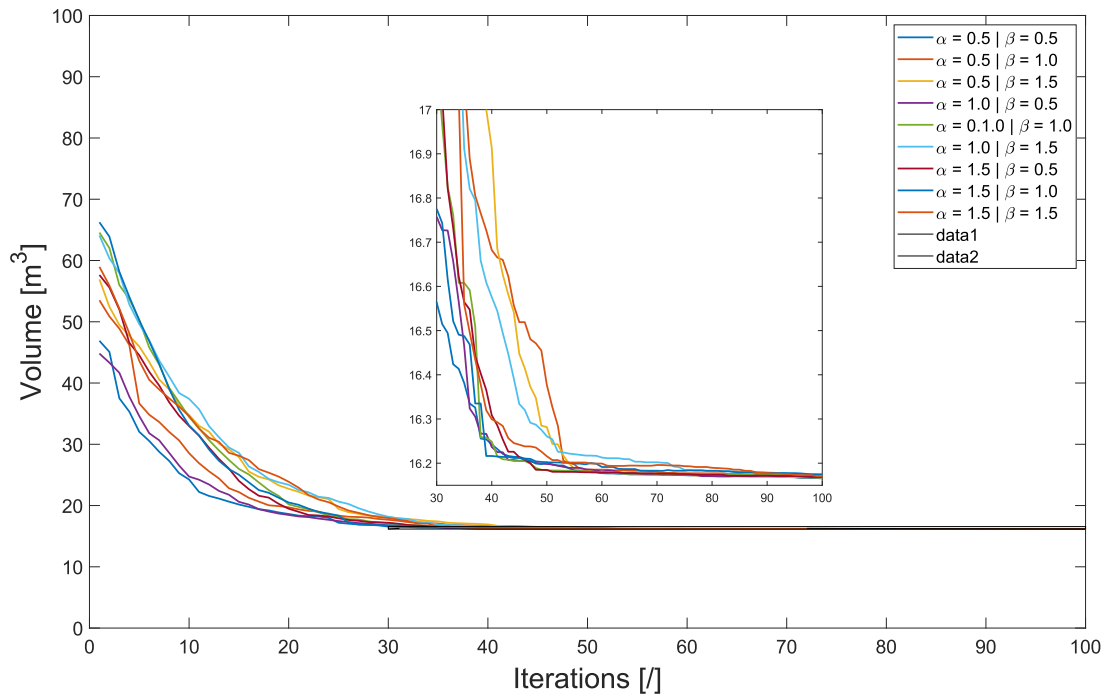
The two graphs above correspond to the combinations $\alpha = 1, \beta = 1.5$ and $\alpha = 1.5, \beta = 0.5$. Both show scattered curves, although they converge after about 50 iterations. In contrast, the two graphs below corresponding to the last two tested combinations $\alpha = 1.5 - \beta = 1$ and $\alpha = 1.5 - \beta = 1.5$, both converge after about 40 iterations.



Graph 11: Estimation of Distribution Algorithm with 2 design variables ($\alpha = 1,5 \mid \beta = 1$)



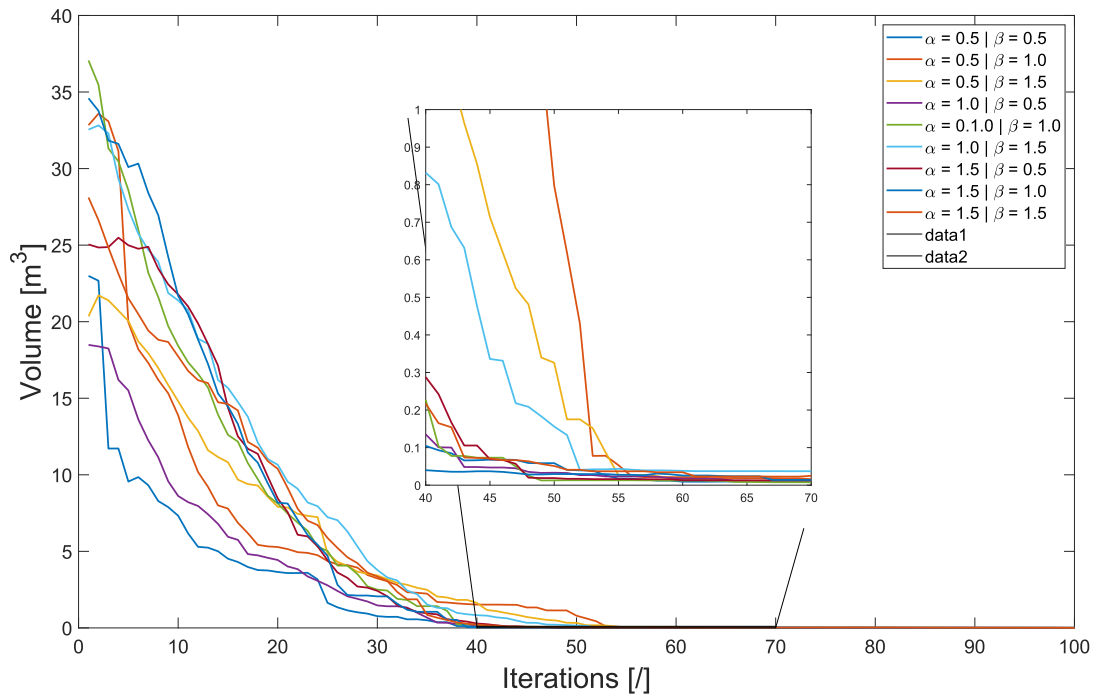
Graph 12: Estimation of Distribution Algorithm with 2 design variables ($\alpha = 1,5 \mid \beta = 1,5$)



Graph 13: Estimation of Distribution Algorithm with 2 design variables - Comparison of the means

The graph above shows the comparison of the averages for each combination tested. After about 40 iterations, all curves settle at an objective function value between 16,20 and 16,80 m^3 . At 100 iterations, all curves settle at a value between 16,15 and 16,20 m^3 .

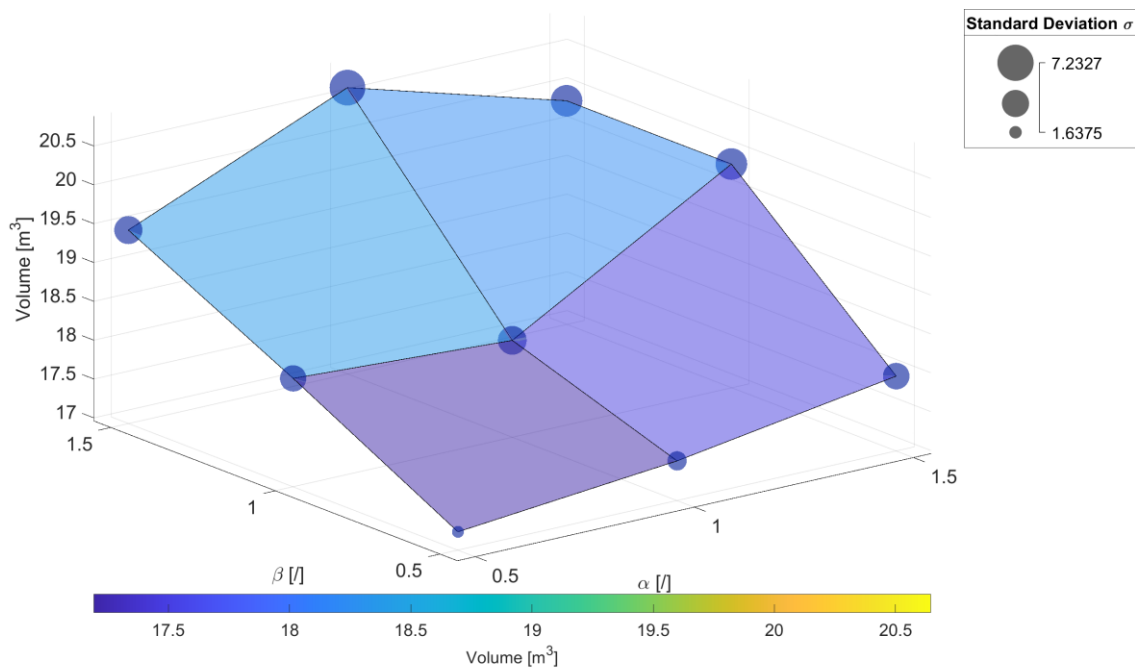
It is possible to state that from the statistical analysis of the results from 20 independent runs, it was observed not such remarkable changes with different values of the hyperparameter α and β , underlining its robustness with structural optimization of an arch structure considering 2 design variables.



Graph 14: Estimation of Distribution Algorithm with 2 design variables - Comparison of the Standard deviations

Similar to the previous graph, this compares standard deviations for each combination of values of the hyperparameter α and β tested. From the graph, it can be seen that for a number of iterations between 40 and 55, the standard deviation is of the order of 1 m³ while for more than 55 iterations, the value is of the order of 0,1 m³.

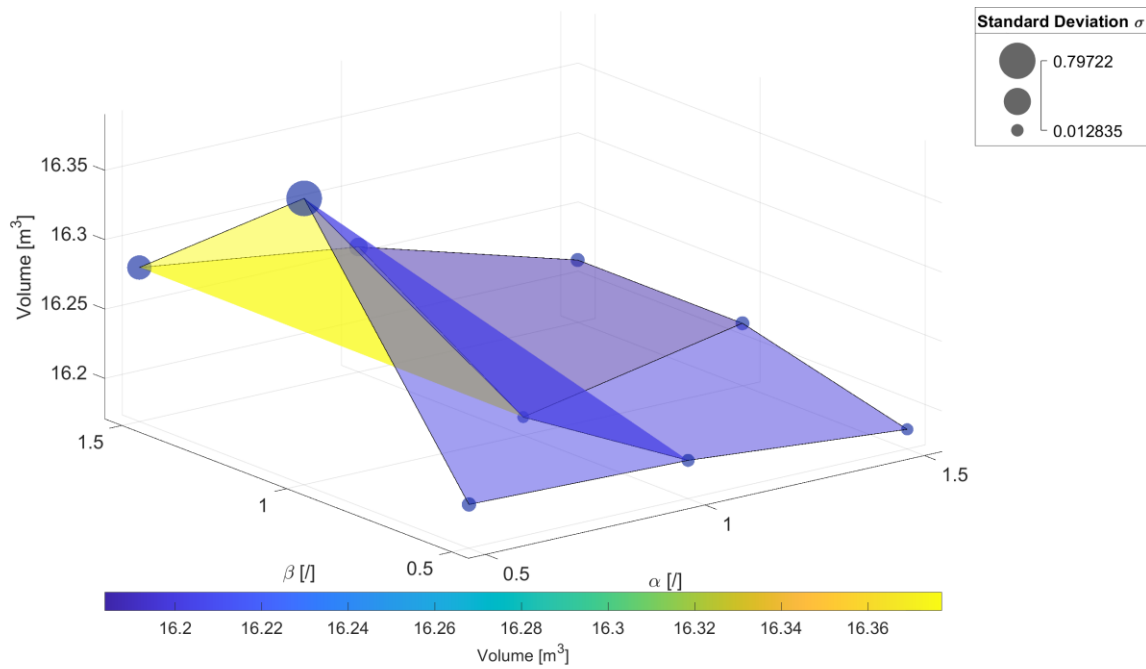
The bubble graphs below summarise the two previous graphs comparing averages and standard deviations in different forms. The x-axis and y-axis show all possible values of alpha and beta. While the z-axis shows the mean value taken by the objective function, the bubble size also represents the standard deviation value taken at a given combination. This graph has been reported for three different states: 25, 50 and 100 iterations.



Graph 15: EDA with 2 design variables: Comparison optimal solutions and standard deviations for different α and β values – (25 iterations)

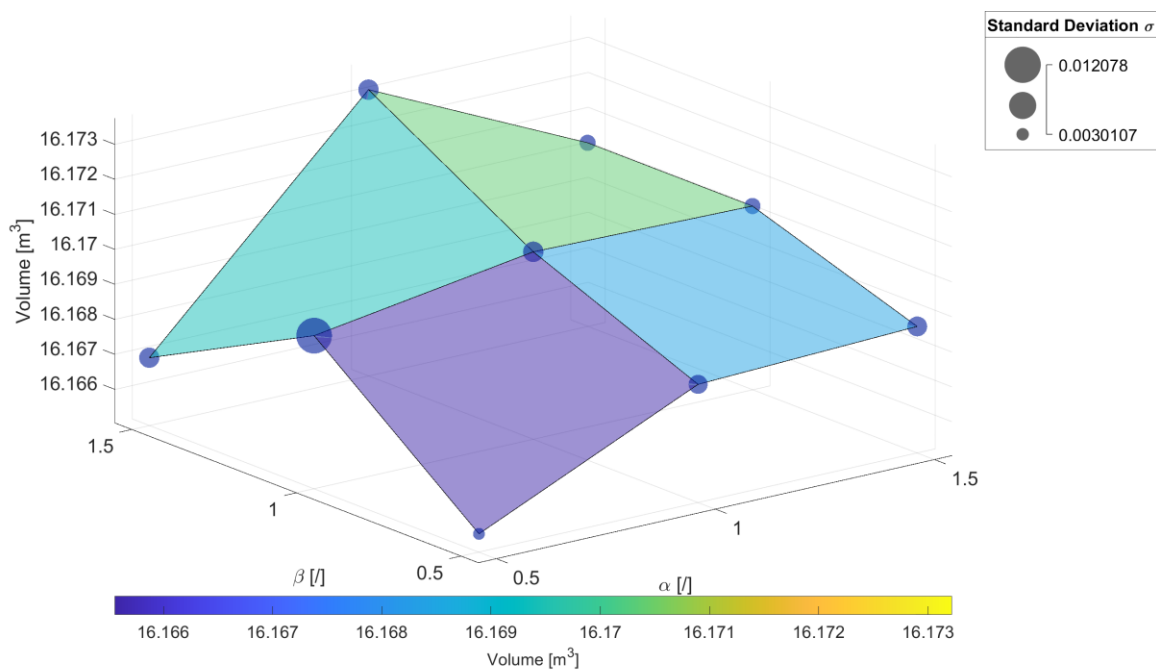
According to Graph 15, for $\alpha = 0,5$ and $\beta = 0,5$ after only 25 iterations, the objective function is close to that obtained at 100 iterations, with a standard deviation of 1,63 m³.

The first conclusion that can be drawn is that the combination of $\alpha = 0,5$ and $\beta = 0,5$ performs best both in terms of data dispersion and the value of the objective function. The difference in terms of the objective function value between the best and worst combination is about 3.5 m³, while in terms of standard deviation, it is about 5.6 m³. The worst combination after 25 iterations is the one with $\alpha = 1$ and $\beta = 1,5$.



Graph 16: EDA with 2 design variables: Comparison optimal solutions and standard deviations for different α and β values – (50 iterations)

After 50 iterations, the gap in the objective function value between all combinations narrows. All values are between 16,20 and 16,35 m³. The standard deviation value is also significantly lower for each combination with values between 0,01 and 0,80 m³.

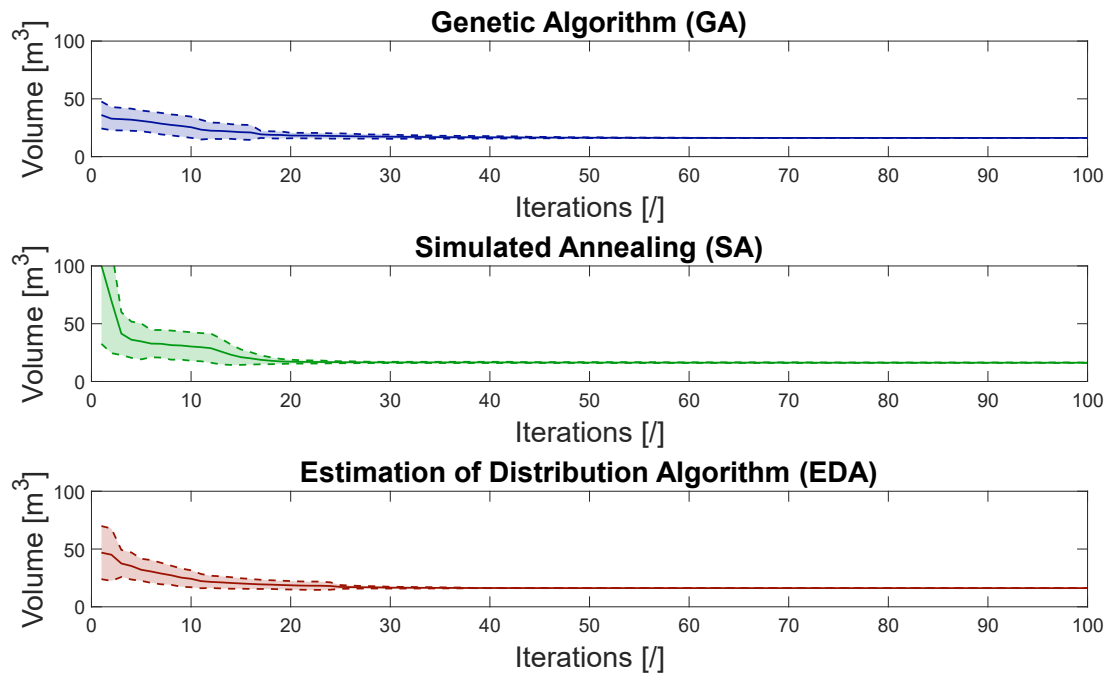


Graph 17: EDA with 2 design variables: Comparison optimal solutions and standard deviations for different α and β values – (100 iterations)

After 100 iterations, it can be observed that for each combination of α and β , objective function values are between 16,16 and 16,17 m³ with standard deviations in the order of magnitude of 10^{-3} m³.

From this, it is possible to emphasise again the robustness of this algorithm in the case of structural optimisation of a steel arch considering 2 design variables.

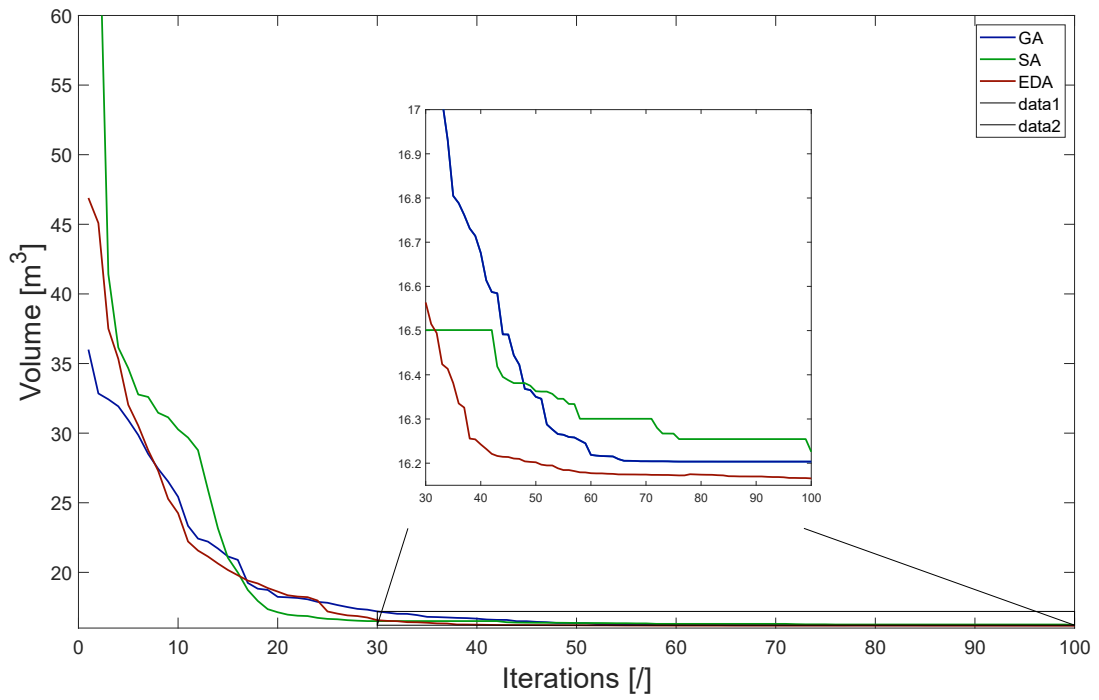
6.2.1.5. Comparison graphs and discussion of results



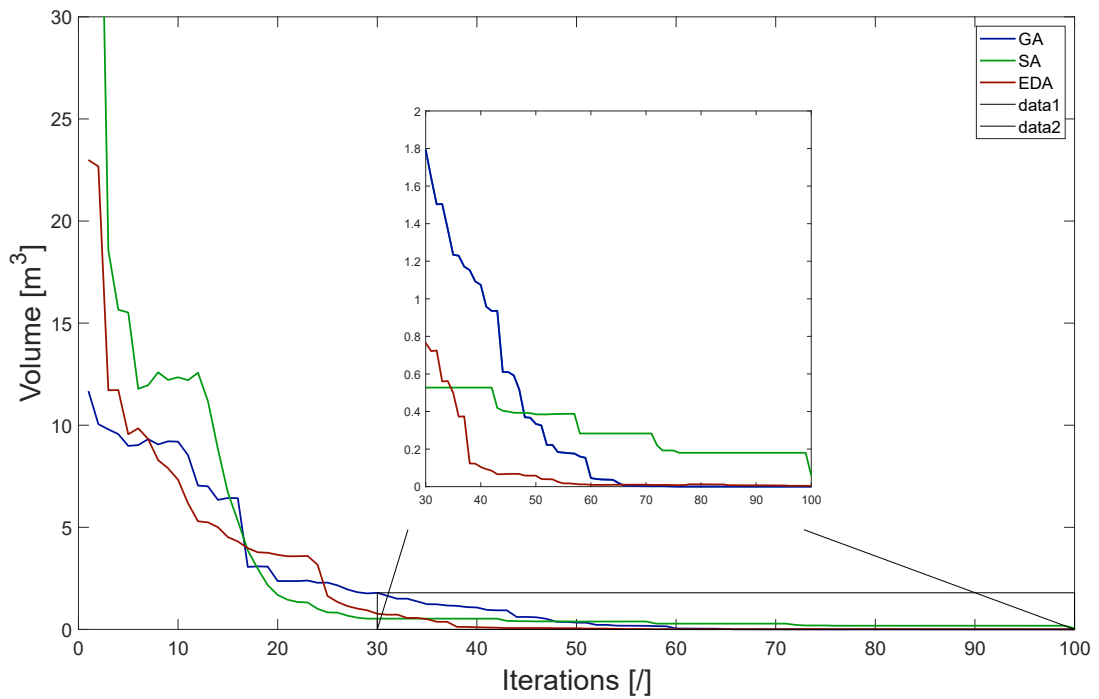
Graph 18: Comparison of means and standard deviations for different algorithms – Configuration with 2 design variables

This comparison graph shows the trends of the averages and standard deviations (shaded area). From the graph, it can be seen that all tested algorithms converge after approximately 30 iterations. Furthermore, the results obtained regarding the value taken by the objective function at the end of the iterative process are very similar, with a deviation of 0.2 m^3 .

It can be seen that the simulated annealing algorithm converges on average earlier than the others, at about 20 iterations, although just before 20 iterations, it has a high standard deviation value. In contrast, the EDA that converges after about 30 iterations has a low standard deviation value in the range between 20 and 30 iterations. Therefore, in the case of conducting a two-variable optimisation process on the case under consideration and evaluating the objective function a few times, applying the EDA algorithm appears to be the most appropriate.



Graph 19: Comparison of the averages – Configuration with 2 design variables



Graph 20: Comparison of standard deviations – Configuration with 2 design variables

Graph 19 compares the average values taken by the objective function over the 20 runs. Graph 19 compares the average values taken by the objective function over the 20 runs. It can be seen that the EDA is the one that, after 30 iterations, has a lower average than the others, and consequently, the result obtained is the best compared to the other two algorithms.

Choosing the EDA algorithm in this 2-variable case seems to be the more robust solution in terms of results since unless we know in advance when the Simulated Annealing goes to convergence, EDA gives more certainty, both in the stages immediately preceding the convergence of the Simulated Annealing and in general in the long term of finding a better solution than that of the Simulated Annealing and Genetic Algorithm. .

The same conclusions can be made for Graph 20, from which it can be seen that after 40 iterations, the standard deviation of EDA is by far the lowest compared to the other two algorithms.

In light of the above, EDA is the best-performing algorithm for the present case.

6.2.2. Configuration with 3 design variables

6.2.2.1. Introduction

The second configuration is characterized by three design variables. Indeed, a parabolic arch with a hollow – circular cross-section that has constant dimension along the arch axis.

	User choices		
	Arch shape	Cross-section	Tapering law
Configuration	Parabolic	Hollow circular	Constant
Design variables	1	2	0
Total Design variables: 3			

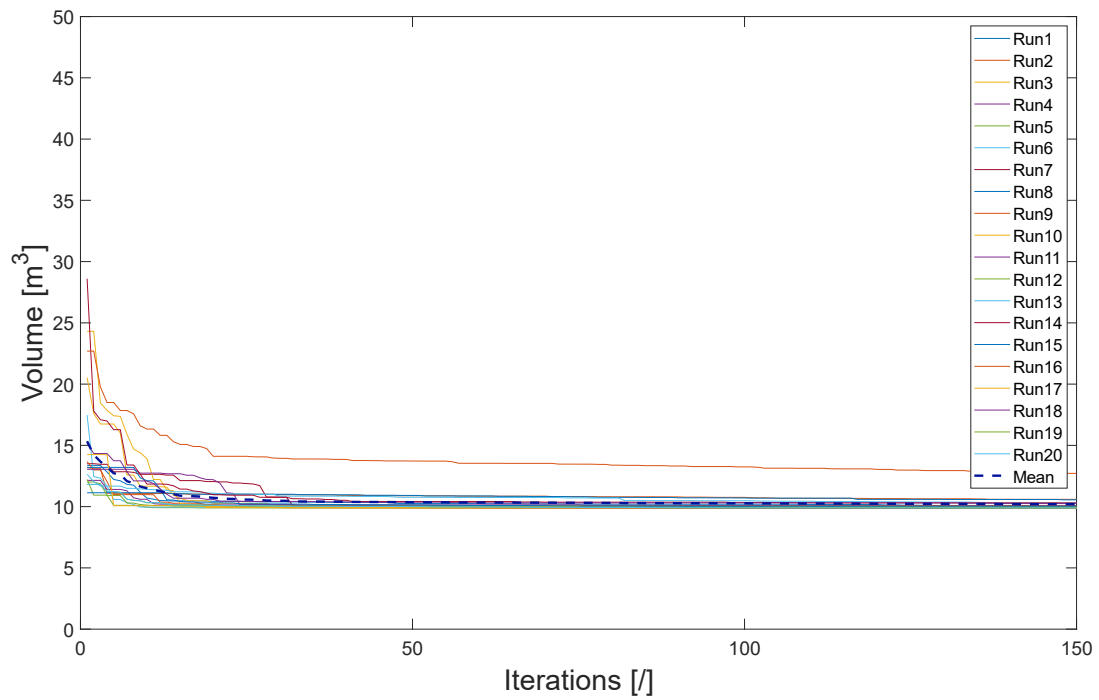
Table 3: Configuration with 3 design variables

Specifically, in this configuration the design variables are:

- f : Height of the arch in the midspan
- r : External radius of hollow-circular cross-section
- t : Thickness of the hollow circular cross-section

As for the previous configuration, the results for the three metaheuristics algorithm tested are reported in the following paragraphs. The results will be presented in graphical form and represent the evolution of the objective function as the number of iterations evolve. For this configuration the maximum number of iterations considered is equal to 150. In addition, graphs comparing the averages and standard deviations between the various algorithms will be shown.

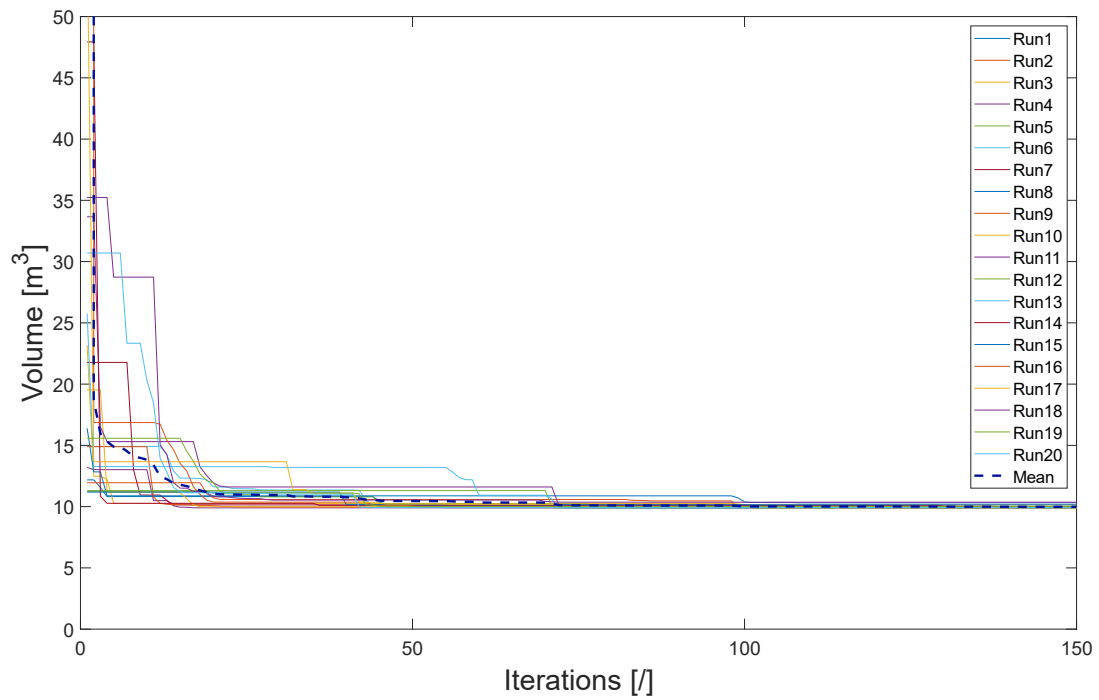
6.2.2.2. Genetic Algorithm



Graph 21: Genetic Algorithm with 3 design variables

From Graph 21, it can be seen that out of 20 analyses, 19 converge after about 30 iterations. At the same time, one analysis that results in a local minimum fails to find the optimal solution in line with the others.

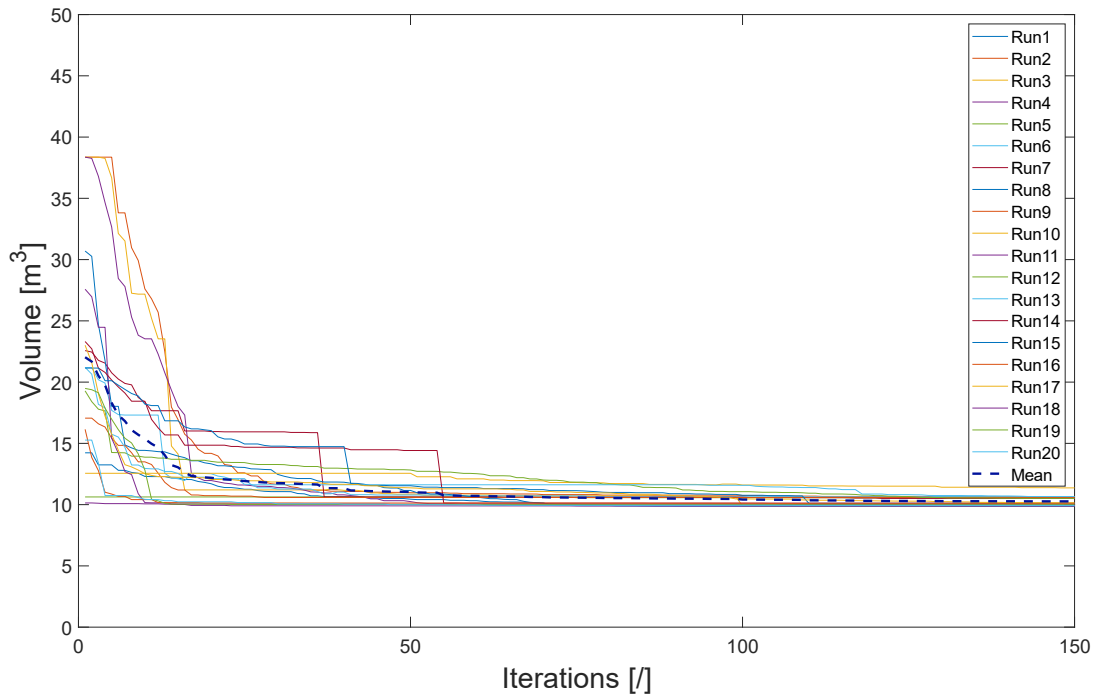
6.2.2.3. Simulated Annealing



Graph 22: Simulated Annealing Algorithm with 3 design variables

From the graph above, it can be seen that convergence is achieved on average after 75 iterations. As opposed to the previous case, where all the runs appear to converge on an optimal solution, it is necessary to carry out twice as many iterations as in the Genetic Algorithm to arrive at a solution.

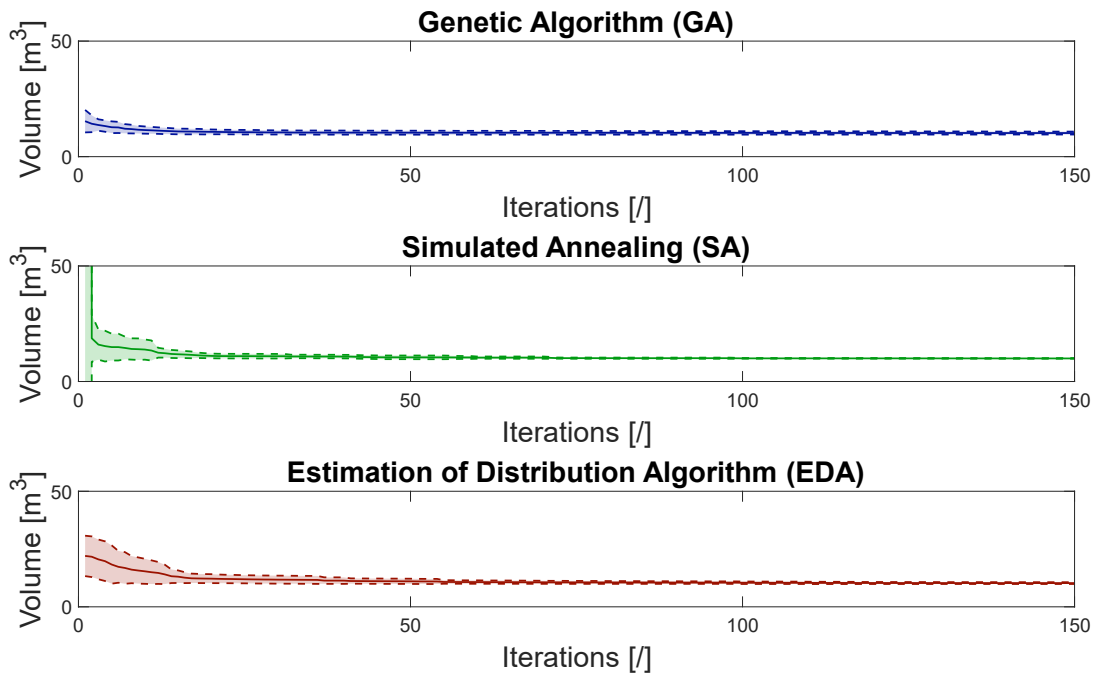
6.2.2.4. EDA



Graph 23: Estimation of Distribution Algorithm with 3 design variables ($\alpha = 0,5$ | $\beta = 0,5$)

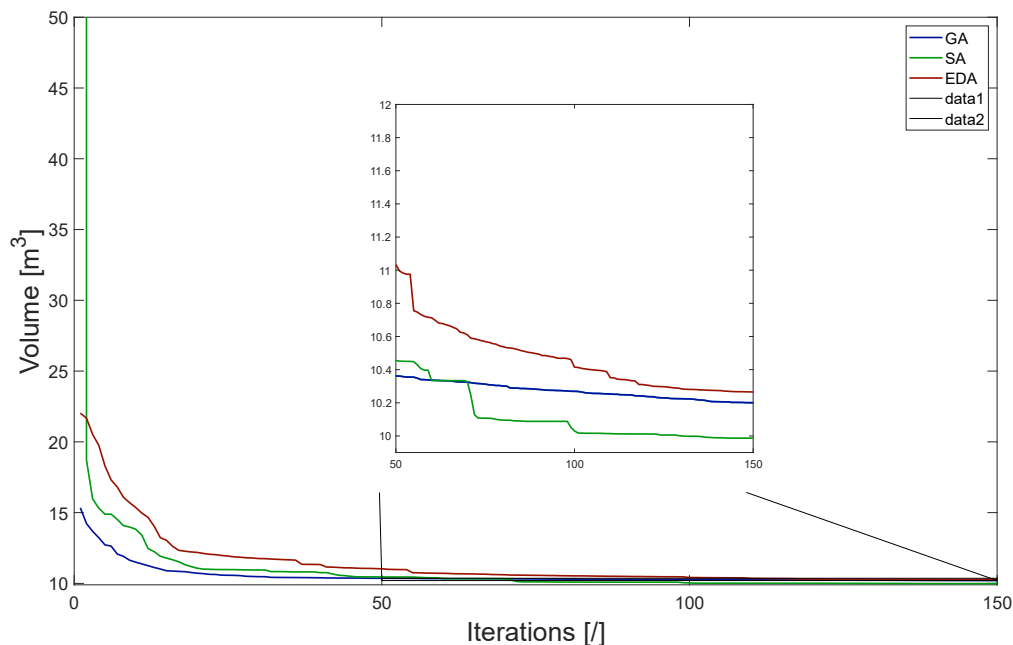
Only the combination of $\alpha = 0,5$ and $\beta = 0,5$ was tested for this configuration. In this graph, we find that EDA, in terms of performance, is intermediate between the Genetic Algorithm and the Simulated Annealing Algorithm as it converges for 60 iterations. The solutions appear to be more dispersed than the SA, but all the analyses converge, unlike the GA.

6.2.2.5. Comparison graphs and discussion of results

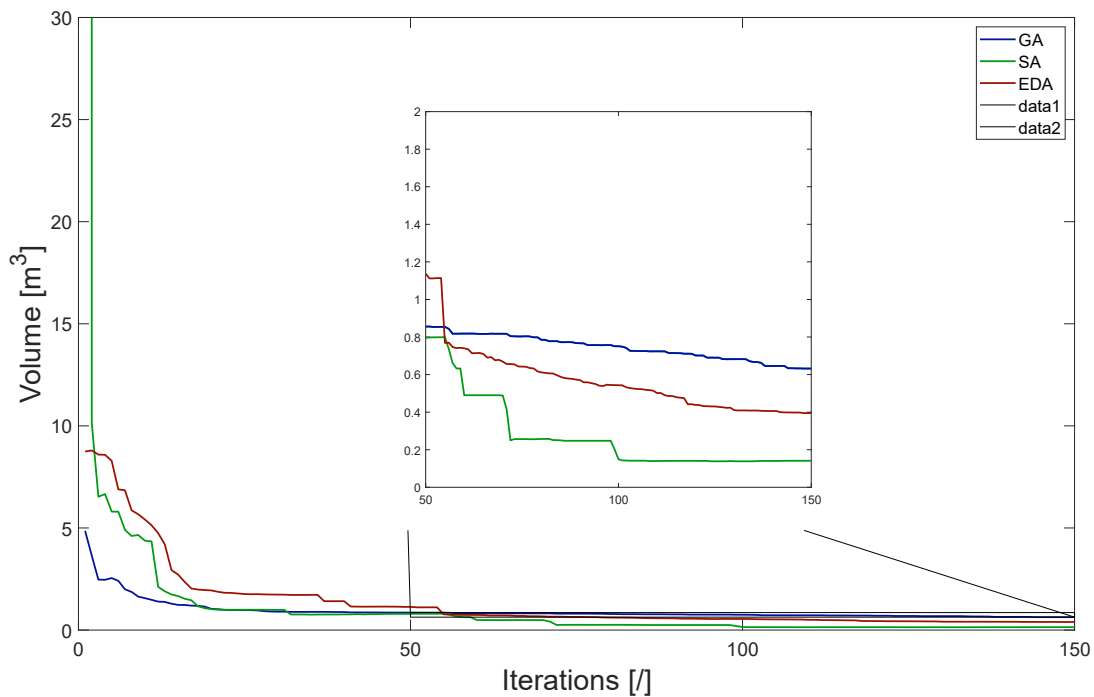


Graph 24: Comparison of means and standard deviations for different algorithms – Configuration with 3 design variables

From Graph 24, it can be seen that the Genetic Algorithm seems to converge earlier than the others, but at the same time, 5 % of the total solutions do not find an optimal result, so it is the fastest but the least accurate. On the other hand, simulated Annealing is slower than the Genetic Algorithm in reaching convergence but achieves a better solution. In this case, EDA ranks somewhere between the two in terms of performance.



Graph 25: Comparison of the averages – Configuration with 3 design variables



Graph 26: Comparison of standard deviations – Configuration with 3 design variables

In the two graphs above, the averages and standard deviations between the algorithms are compared, respectively. As far as the values taken by the objective function are concerned, Simulated Annealing is the one that manages to achieve the best solution with a dispersion of the data around the mean less than 0,2 m³. In this case, EDA obtains the worst solution; however, the data dispersion is around 0,4 m³. The Genetic Algorithm, in terms of the values taken by the objective function, ranks somewhere between the two. However, the data is slightly more dispersed than the EDA, with standard deviation values of around 0,6 m³. The latter is because an analysis of the 20 launched does not find the minimum solution but remains trapped in a local minimum.

The first conclusion that can be made is that by increasing the number of parameters from two to three, i.e. in this case, going from a filled section to a hollow section, the value of the objective function decreases and goes from 16 m³ to 10 m³ on average. As a result of increasing the number of iterations from 100 for the two-variable configuration to 150 for the three-variable configuration, which increases the computational effort resulting in a volume saving of approximately 6 m³. Thus a saving in terms of weight and money.

6.2.3. Configuration with 4 design variables

6.2.3.1. Introduction

The third configuration studied is characterized by four design variables. Indeed, a parabolic arch with a hollow – circular cross-section that has quadratic law variation of the dimensions along the arch axis.

	User choices		
	Arch shape	Cross-section	Tapering law
Configuration	Parabolic	Hollow circular	Quadratic variable
Design variables	1	2	1
Total Design variables: 4			

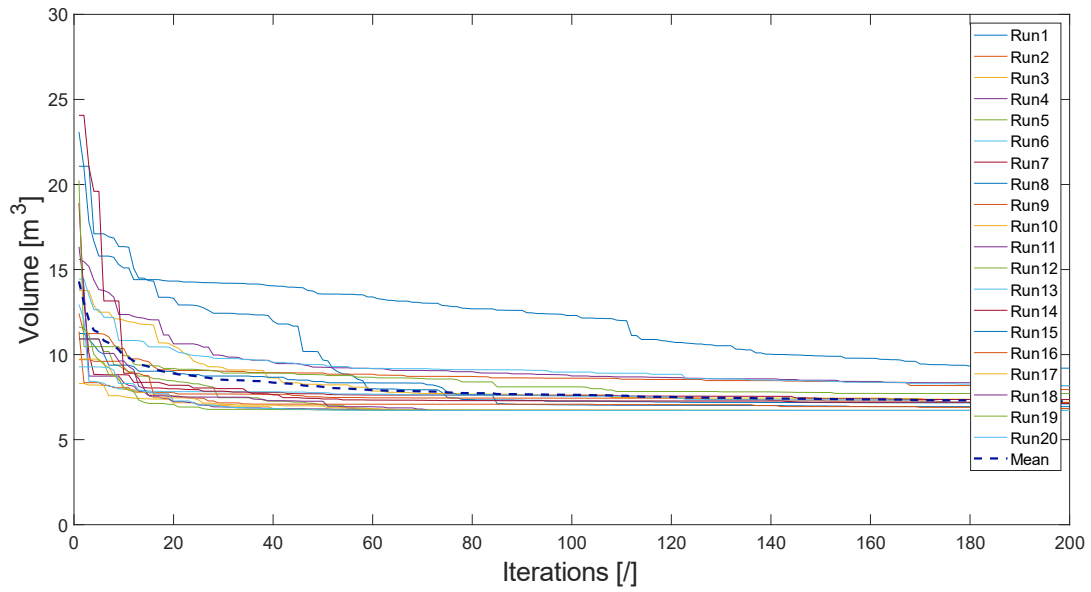
Table 4: Configuration with 4 design variables

Specifically, in this configuration the design variables are:

- f : Height of the arch in the midspan
- r : External radius of hollow-circular cross-section
- t : Thickness of the hollow circular cross-section
- η : Reduction factor (0,1)

As for the previous configuration, the results for the three metaheuristics algorithm tested are reported in the following paragraphs. The results will be presented in graphical form and represent the evolution of the objective function as the number of iterations evolve. For this configuration the maximum number of iterations considered is equal to 200. In addition, graphs comparing the averages and standard deviations between the various algorithms will be shown.

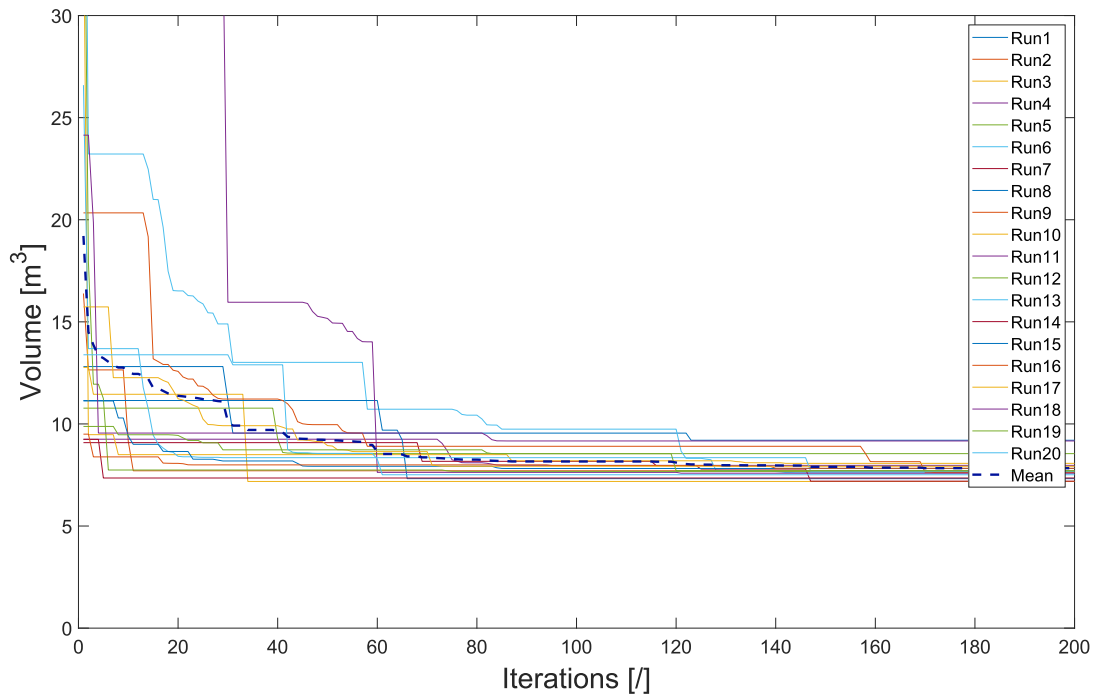
6.2.3.2. Genetic Algorithm



Graph 27: Genetic Algorithm with 4 design variables

From graph 27, it can be seen that almost all analyses converge on average after about 100 iterations, although they are also quite scattered on the graph's tail.

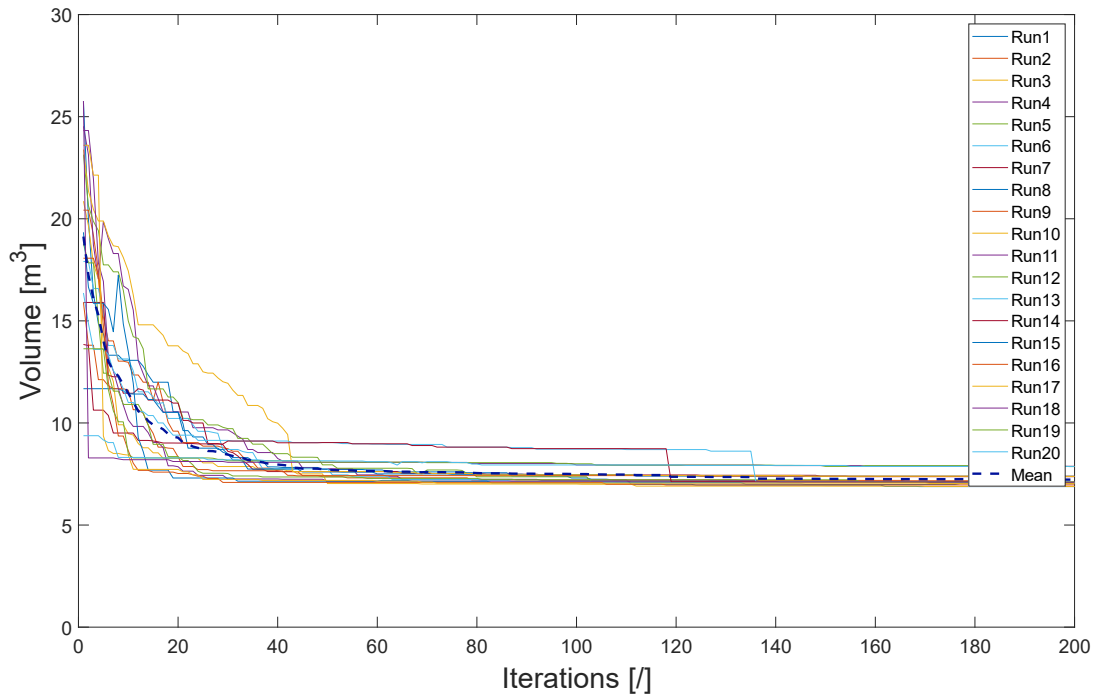
6.2.3.3. Simulated Annealing



Graph 28: Simulated Annealing Algorithm with 4 design variables

From graph 28, it can be seen that convergence is reached on average after 60 iterations. But, again, the data appear more dispersed than in the previously studied configurations.

6.2.3.4. EDA

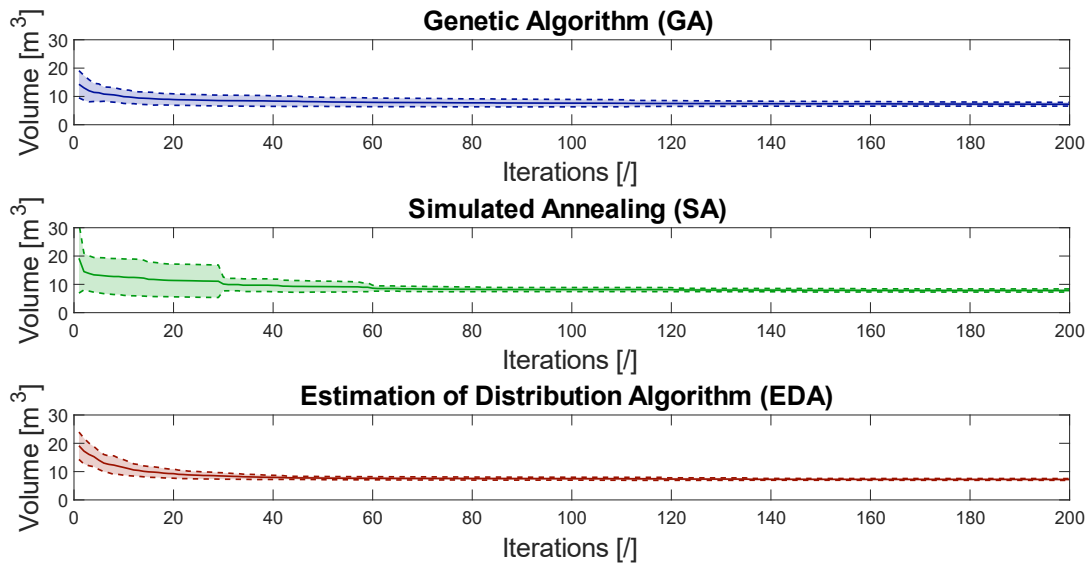


Graph 29: Estimation of Distribution Algorithm with 4 design variables ($\alpha = 0,5$ | $\beta = 0,5$)

Also, for this configuration, like the previous one only the combination of $\alpha = 0,5$ and $\beta = 0,5$ was tested.

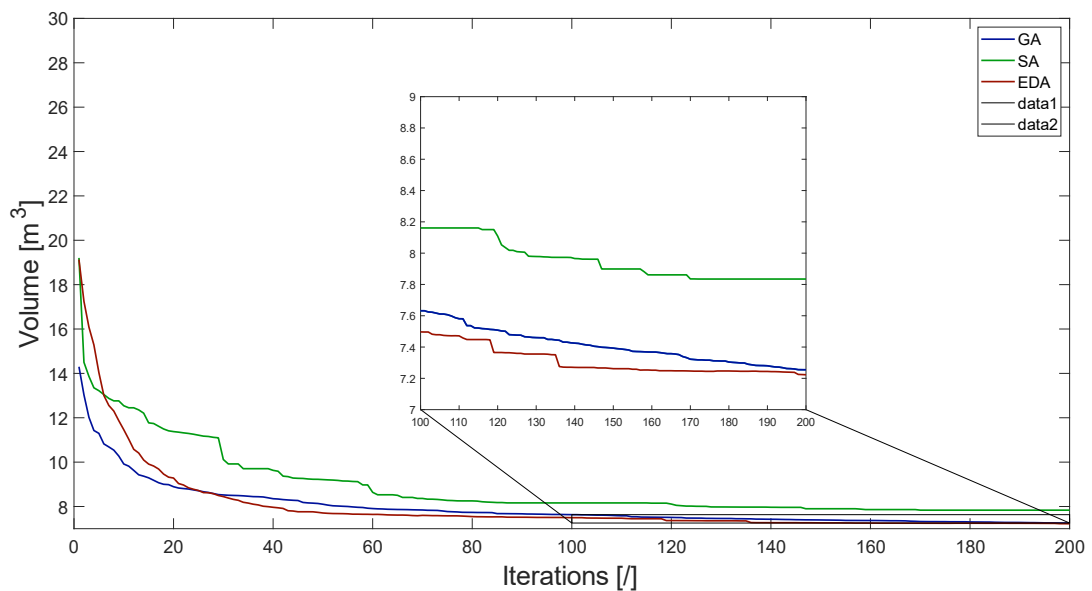
From this graph, one can see that, on average, all analyses converge after about 60 iterations except for three analyses that converge later. Another thing that can be noted is that the data are less dispersed compared to the two graphs seen above.

6.2.3.5. Comparison graphs and discussion of results

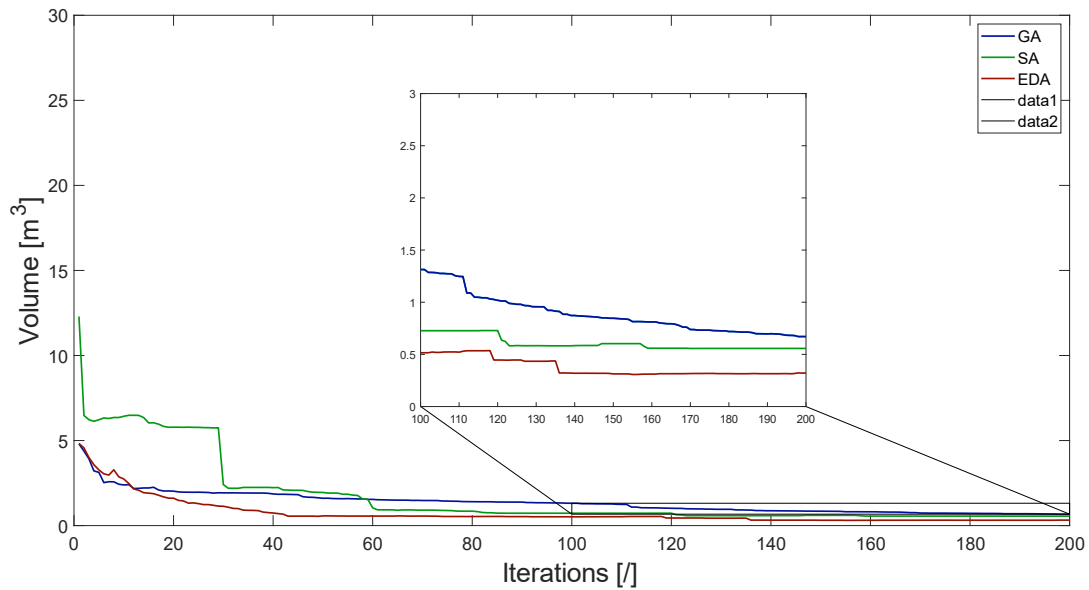


Graph 30: Comparison of means and standard deviations for different algorithms – Configuration with 4 design variables

From Graph 30, it can be seen that the genetic algorithm has a slow convergence compared to the other two; in fact, the Simulated Annealing converges on average after about 60 iterations, while the EDA after 40.



Graph 31: Comparison of the averages – Configuration with 4 design variables



Graph 32: Comparison of standard deviations – Configuration with 4 design variables

According to both of the last two graphs comparing the mean and standard deviation of each of the three algorithms, EDA, as in the two-variable configuration, is the algorithm that performs best in terms of both the value of the objective function, as well as the dispersion of the data for the objective function.

Comparing the value of the objective function at the end of the optimisation process results in an average volume value of between 7,2 and 8,2 m³ with a saving of about 2 m³ of material compared to the three-variable configuration.

So far, it has been seen that as the number of design variables and, thus, degrees of freedom increased, the value of the objective function dropped significantly.

6.2.4. Configuration with 5 design variables

6.2.4.1. Introduction

The last configuration studied is characterized by five design variables. Indeed, a parabolic arch with a hollow - rectangular cross-section that has quadratic law variation of the dimensions along the arch axis.

	User choices		
	Arch shape	Cross-section	Tapering law
Configuration	Parabolic	Hollow rectangular	Quadratic variable
Design variables	1	3	1
Total Design variables: 5			

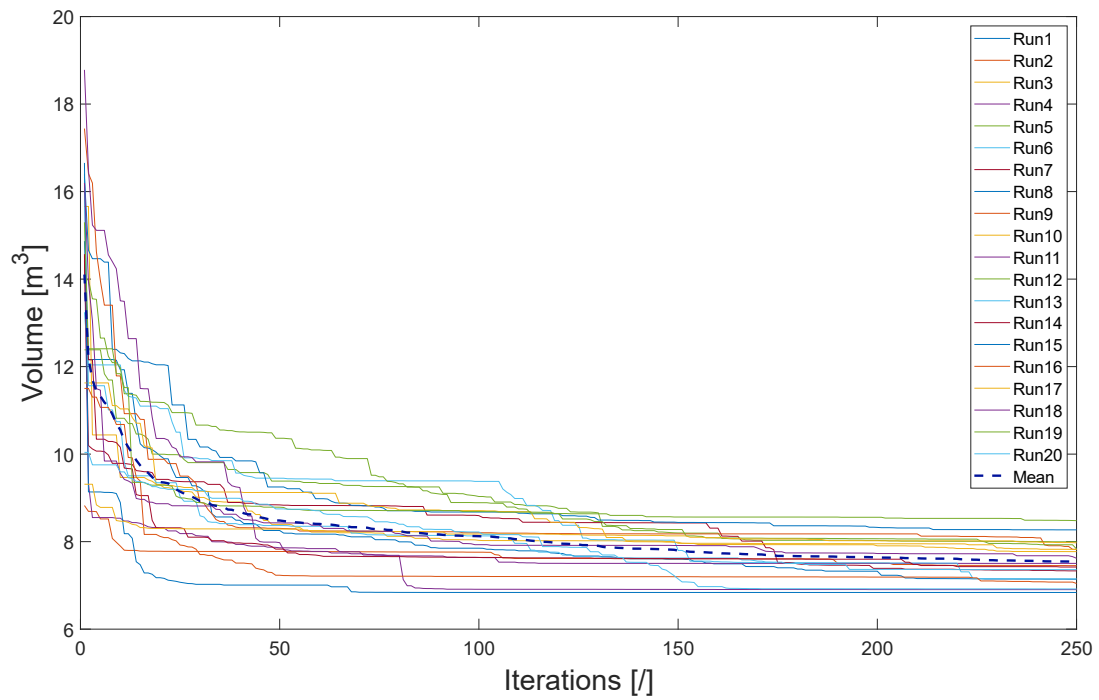
Table 5: Configuration with 5 design variables

Specifically, in this configuration the design variables are:

- f : Height of the arch in the midspan
- r_1 : Base of the hollow rectangular cross-section
- r_2 : Height of the hollow rectangular cross-section
- t : Thickness of the hollow rectangular cross-section
- η : Reduction factor (0,1)

As with all studied configurations, the results for the three metaheuristics algorithm tested are reported in the following paragraphs. The results will be presented in graphical form and represent the evolution of the objective function as the number of iterations evolve. For this configuration the maximum number of iterations considered is equal to 250. In addition, graphs comparing the averages and standard deviations between the various algorithms will be shown.

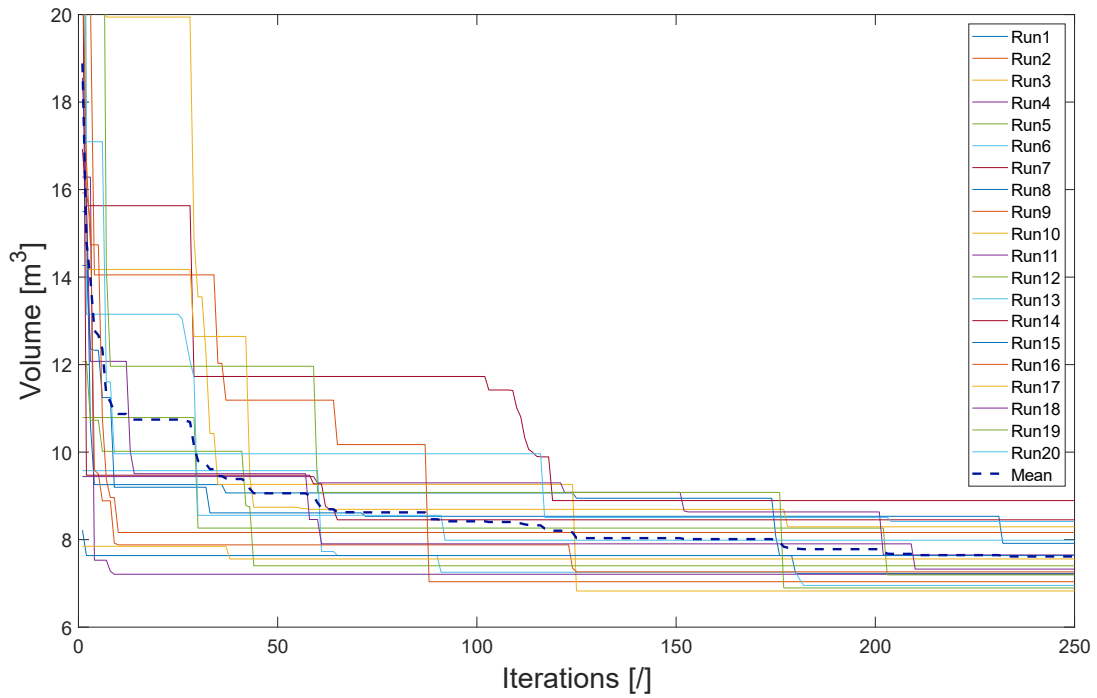
6.2.4.2. Genetic Algorithm



Graph 33: Genetic Algorithm with 5 design variables

At first glance, the data in Graph 33 appear very scattered. The dispersion of the data remains pronounced even at the tail end of the graph. All analyses converge on average after 150 iterations.

6.2.4.3. Simulated Annealing

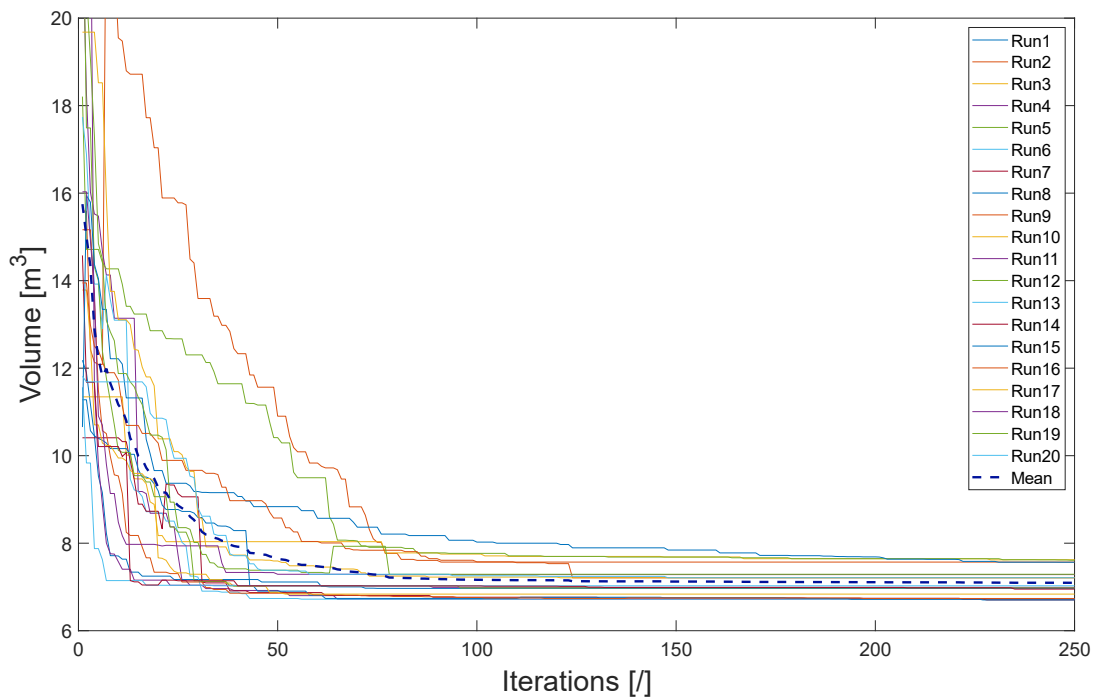


Graph 34: Simulated Annealing Algorithm with 5 design variables

At first glance, the data in Graph 33 appear very scattered. The dispersion of the data remains pronounced even at the tail end of the graph.

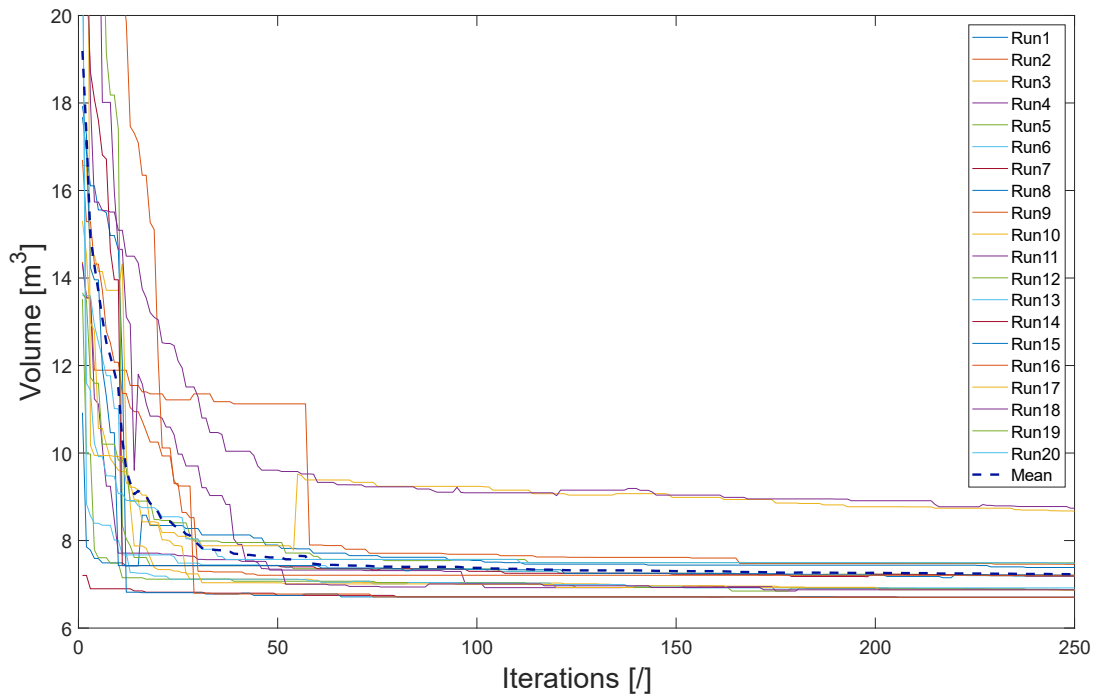
6.2.4.4. EDA

As for the two variables case the problem have been tested with different values of α and β , exploring the effects of a covariance matrix with a sub-linear, linear or super-linear trends with number of iterations. In the following are reported the graphs of each tested combination of alfa and beta. As in the previous cases, 20 independent analyses were carried out for each combination of alpha and beta to assess the algorithm's performance under consideration.



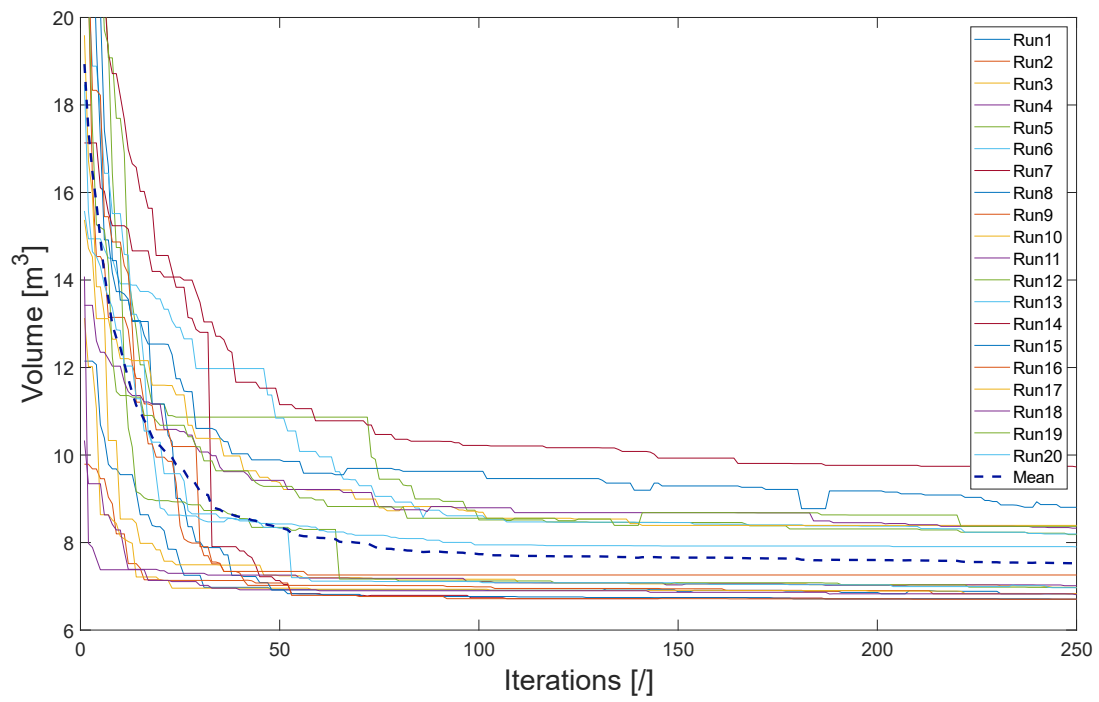
Graph 35: Estimation of Distribution Algorithm with 5 design variables ($\alpha = 0,5$ | $\beta = 0,5$)

The first combination tested is $\alpha = 0,5$ and $\beta = 0,5$, all curves converging on average after about 100 iterations. The data appear slightly scattered but to a lesser extent than in the other two algorithms.



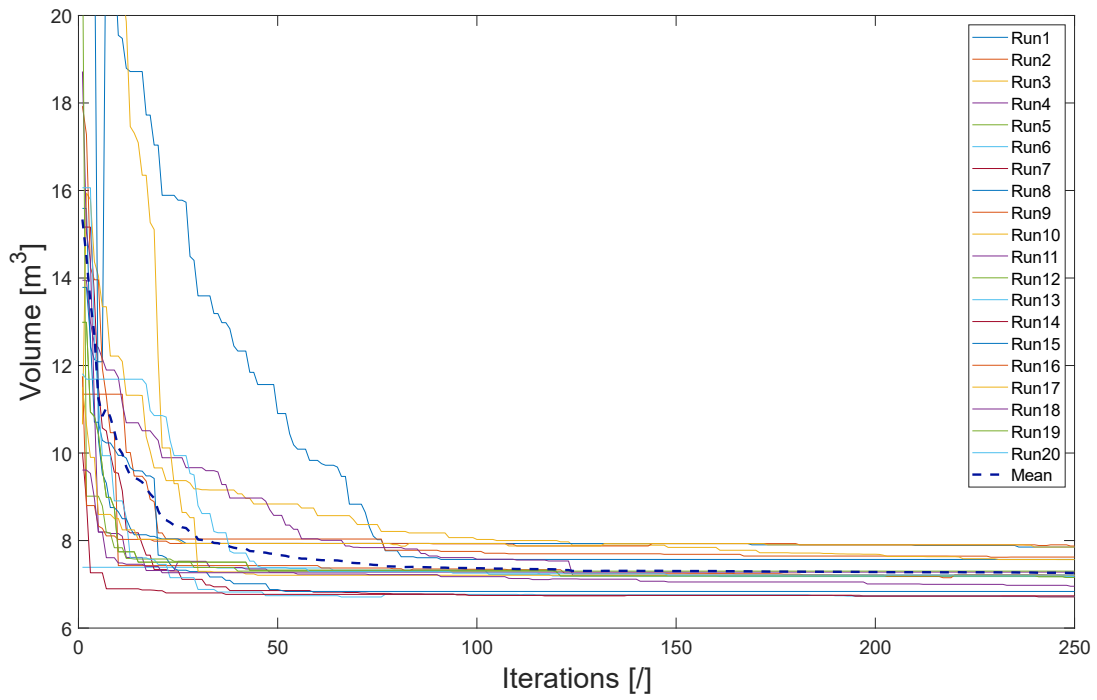
Graph 36: Estimation of Distribution Algorithm with 5 design variables ($\alpha = 0,5$ | $\beta = 1$)

The second combination tested is $\alpha = 0.5$ and $\beta = 1$. The graph in terms of dispersion looks very similar to the first combination, except that for two analyses, convergence is not achieved.



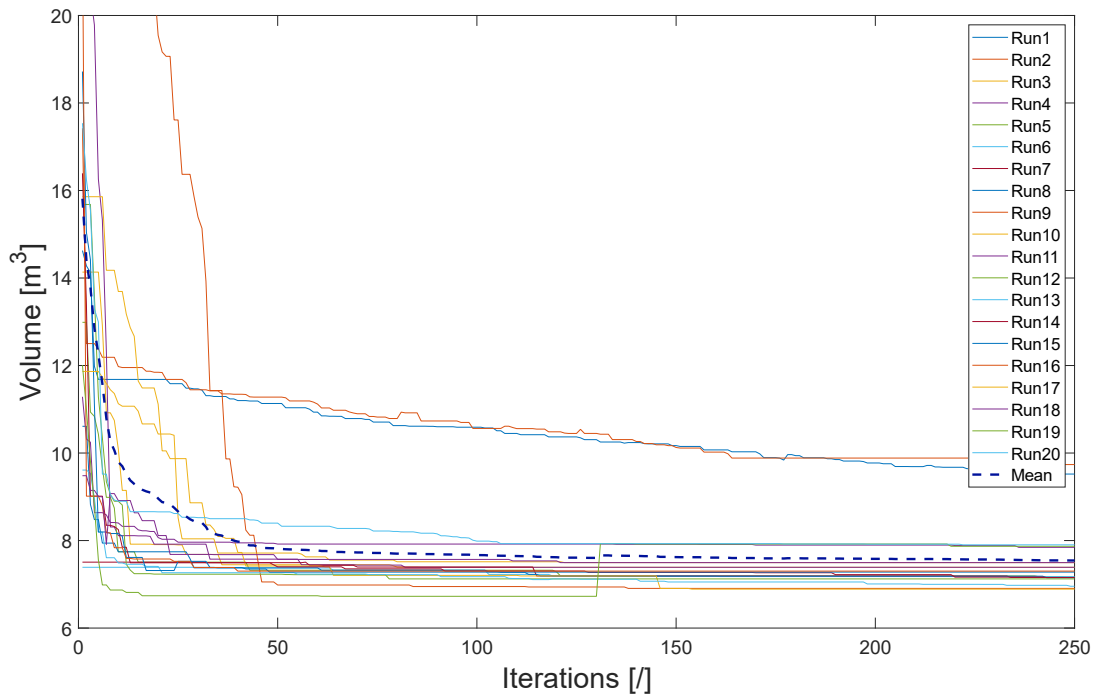
Graph 37: Estimation of Distribution Algorithm with 5 design variables ($\alpha = 0,5$ | $\beta = 1,5$)

In the tested combination shown in Figure 37, the data are very dispersed even at the end of the iterative process.



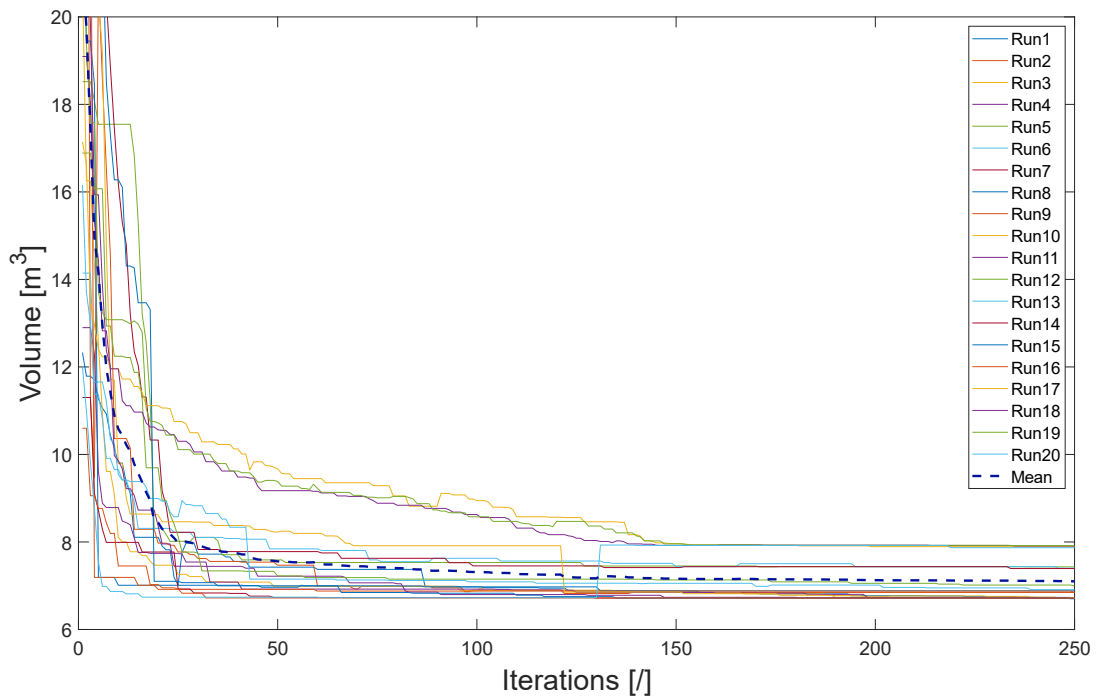
Graph 38: Estimation of Distribution Algorithm with 5 design variables ($\alpha = 1 \mid \beta = 0,5$)

The results obtained for the combination of $\alpha = 1$ and $\beta = 0,5$ shown in the Graph 38 appear less dispersed than the previous combinations, and all analyses converge on average after 100 iterations.



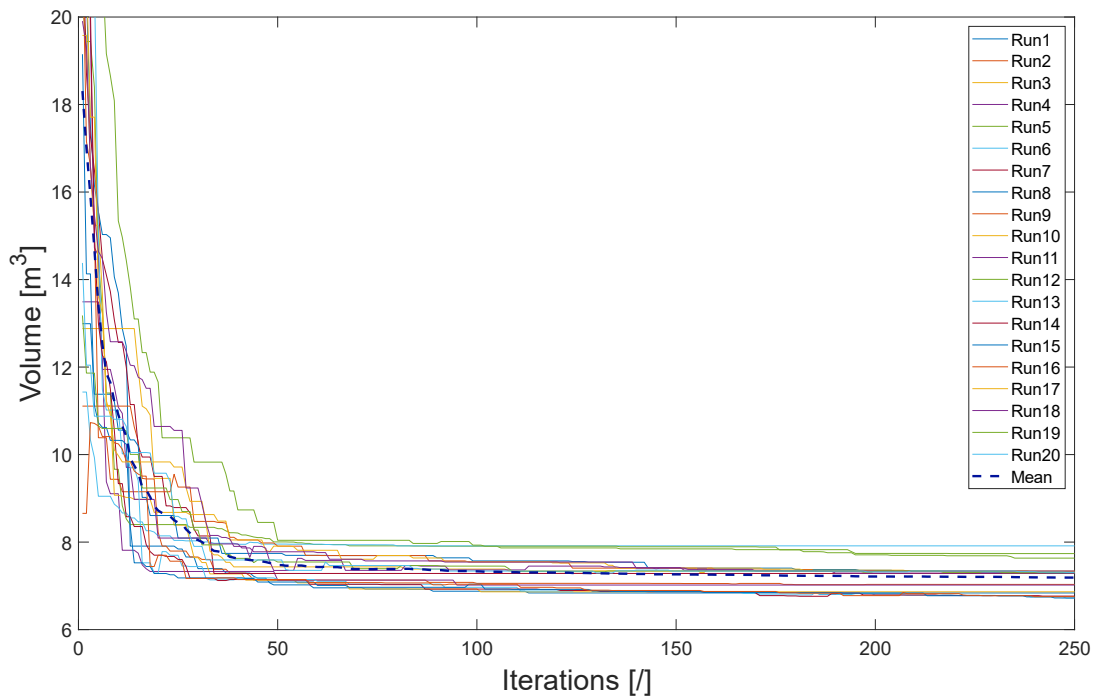
Graph 39: Estimation of Distribution Algorithm with 5 design variables ($\alpha = 1 \mid \beta = 1$)

Even in this case of $\alpha = 1$ and $\beta = 1$, there is little dispersion in the data, except that for two analyses, a global minimum was not reached.



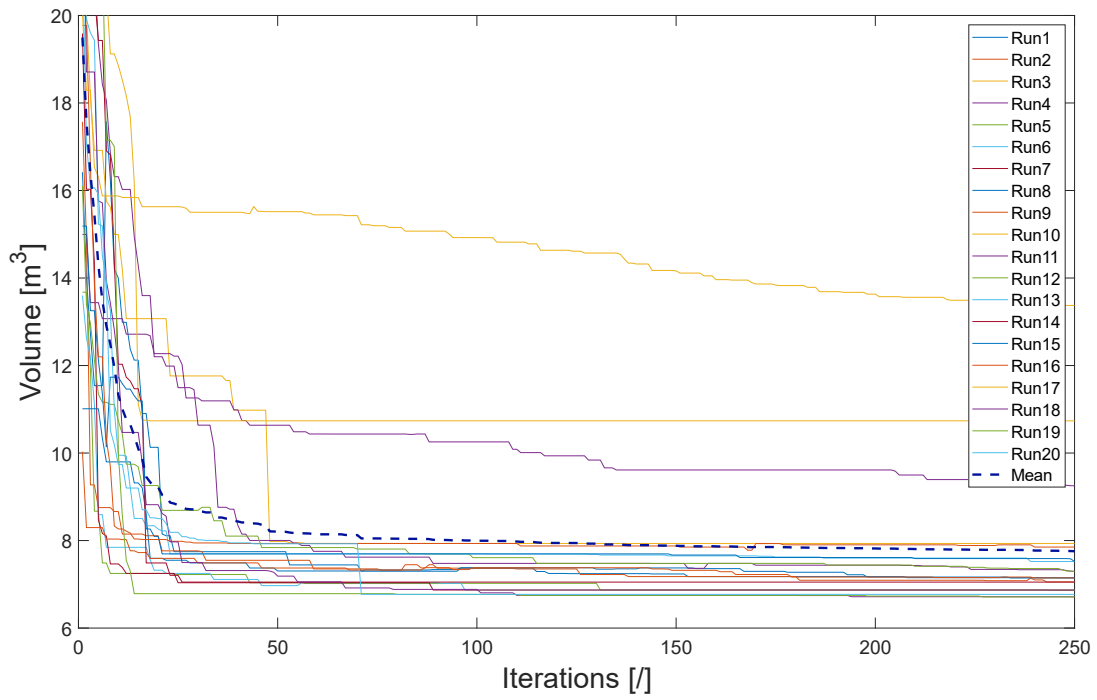
Graph 40: Estimation of Distribution Algorithm with 5 design variables ($\alpha = 1 \mid \beta = 1,5$)

Figure 40 shows the results obtained for the combination of $\alpha = 1$ and $\beta = 1,5$, all analyses converging after 150 iterations.



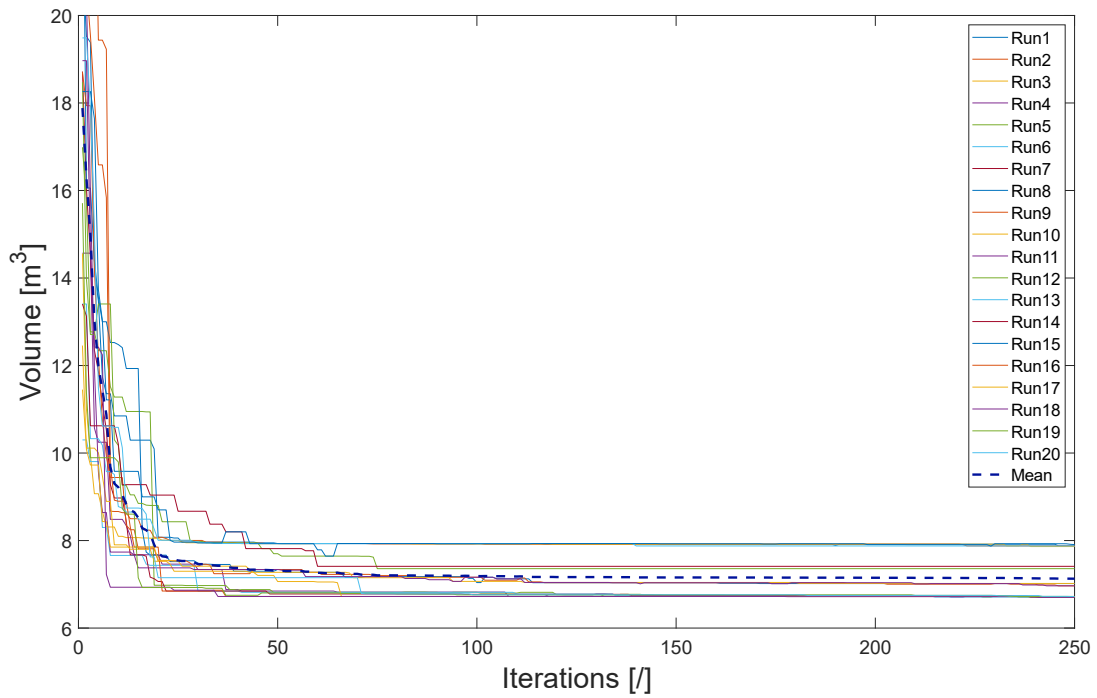
Graph 41: Estimation of Distribution Algorithm with 5 design variables ($\alpha = 1,5$ | $\beta = 0,5$)

This combination with $\alpha = 1,5$ and $\beta = 0,5$ values shown in the graph above seems to do very well in terms of convergence with little scatter in the data.



Graph 42: Estimation of Distribution Algorithm with 5 design variables ($\alpha = 1,5$ | $\beta = 1$)

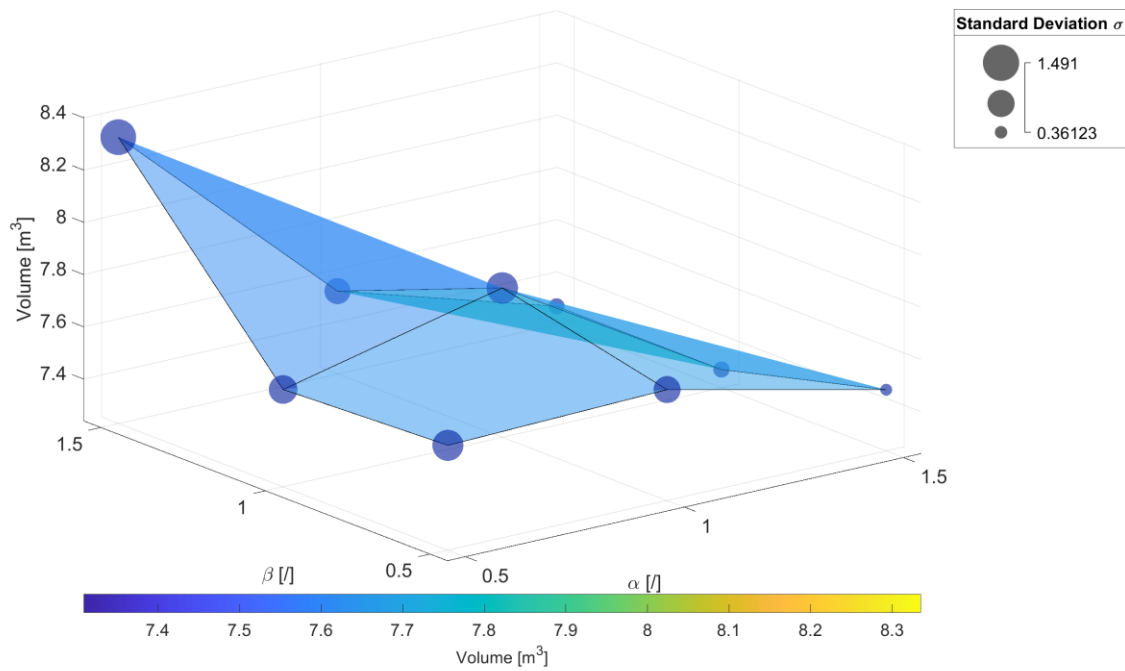
For the combination $\alpha = 1,5$ and $\beta = 1$, three of the 20 analyses failed to find a global minimum but remained trapped in a local minimum. All the others 17 analyses converge after about 80 iterations and remain constant until the end of the iterative process.



Graph 43: Estimation of Distribution Algorithm with 5 design variables ($\alpha = 1,5$ | $\beta = 1,5$)

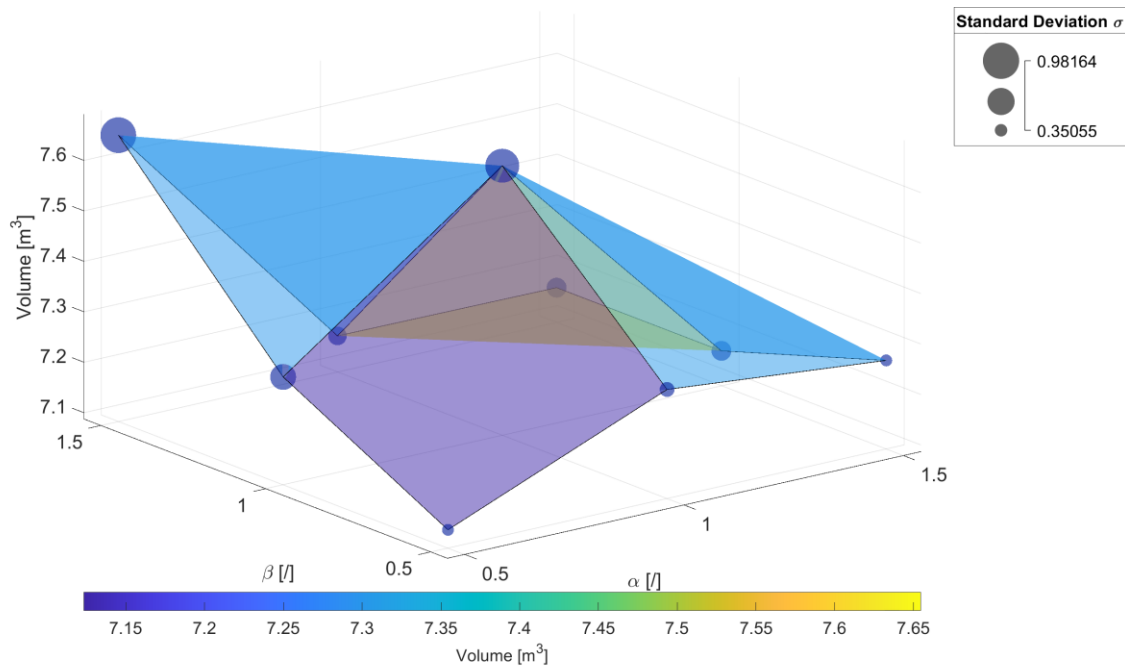
The last tested combination corresponding to $\alpha = 1,5$ and $\beta = 1,5$, shown in Graph 43, gives good results in terms of data dispersion. All analyses converge after about 80 iterations.

The bubble graphs below summarise the two previous graphs comparing averages and standard deviations in different forms. The x-axis and y-axis show all possible values of alpha and beta. While the z-axis shows the mean value taken by the objective function, the bubble size also represents the standard deviation value taken at a given combination. This graph has been reported for three different states: 50, 150 and 250 iterations.



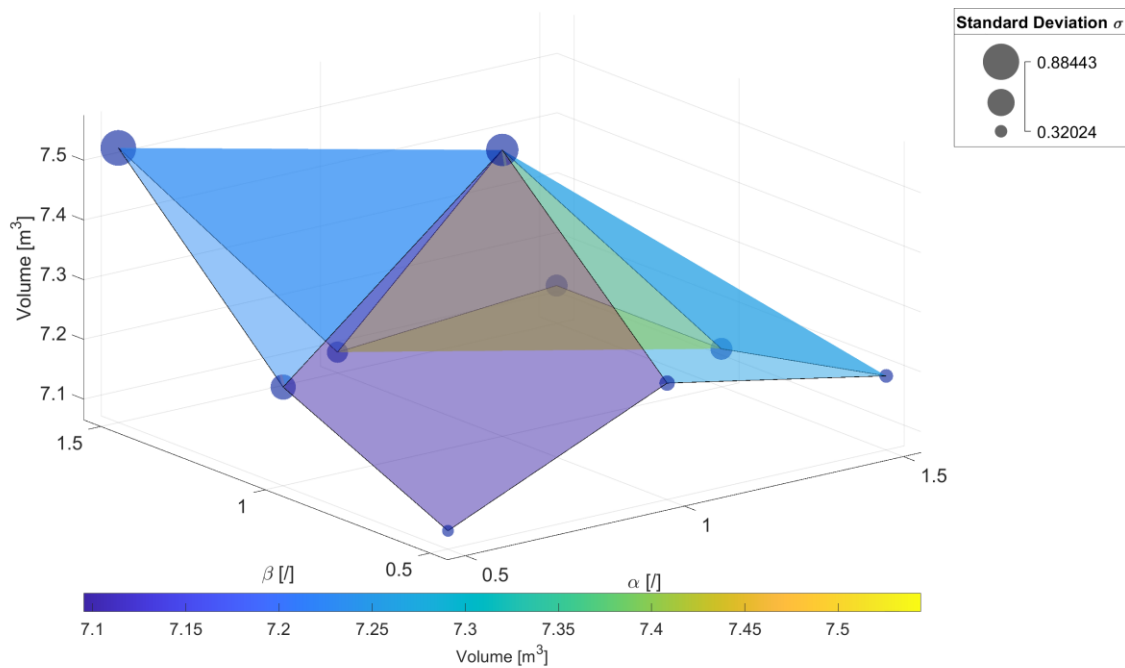
Graph 44: EDA with 5 design variables: Comparison optimal solutions and standard deviations for different α and β values – (50 iterations)

According to Graph 44, after 50 iterations the best values in terms of both objective function and standard deviation are obtained for $\alpha = 1,5$. This is a result in line with expectations as initially, the GMM Gaussians with alpha 1,5 are wider in preference to exploration, and consequently, better solutions are found in the early stages, having explored more of the domain. This result was not obtained for the two-variable configuration as the domain to be explored is small.



Graph 45: EDA with 5 design variables: Comparison optimal solutions and standard deviations for different α and β values – (150 iterations)

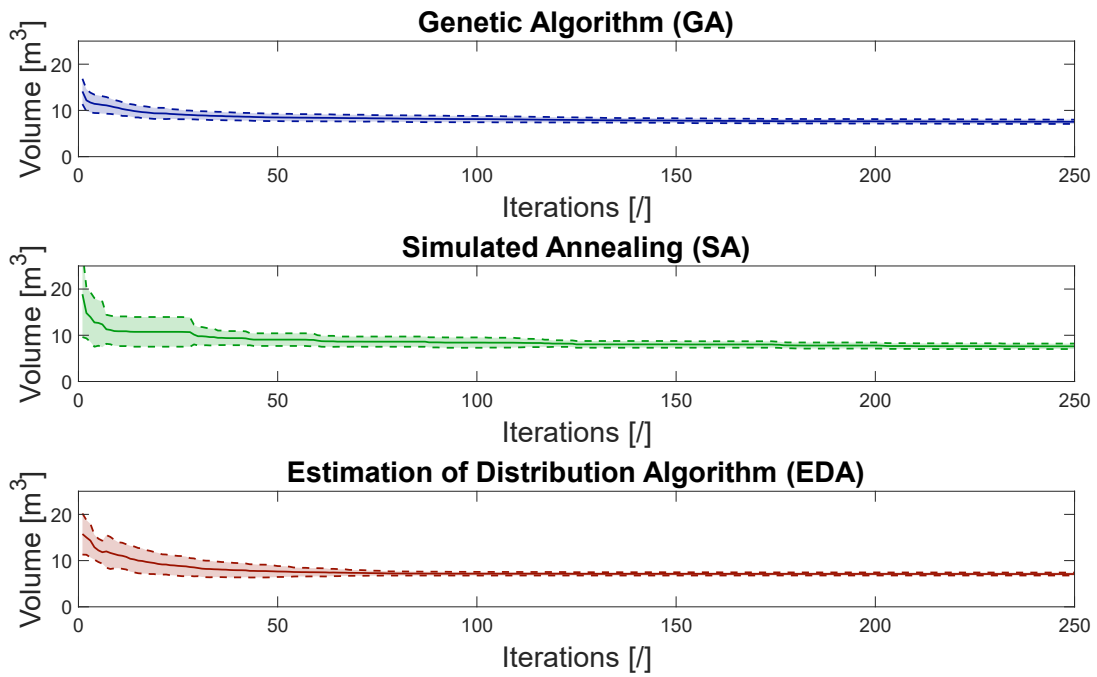
By looking at the graph at 150 iterations, the best solution is for $\alpha = 0,5$ and $\beta = 0,5$. After 150 iterations for each combination of α and β , the value of the objective function is between 7.1 and 7.6 m³.



Graph 46: EDA with 5 design variables: Comparison optimal solutions and standard deviations for different α and β values – (250 iterations)

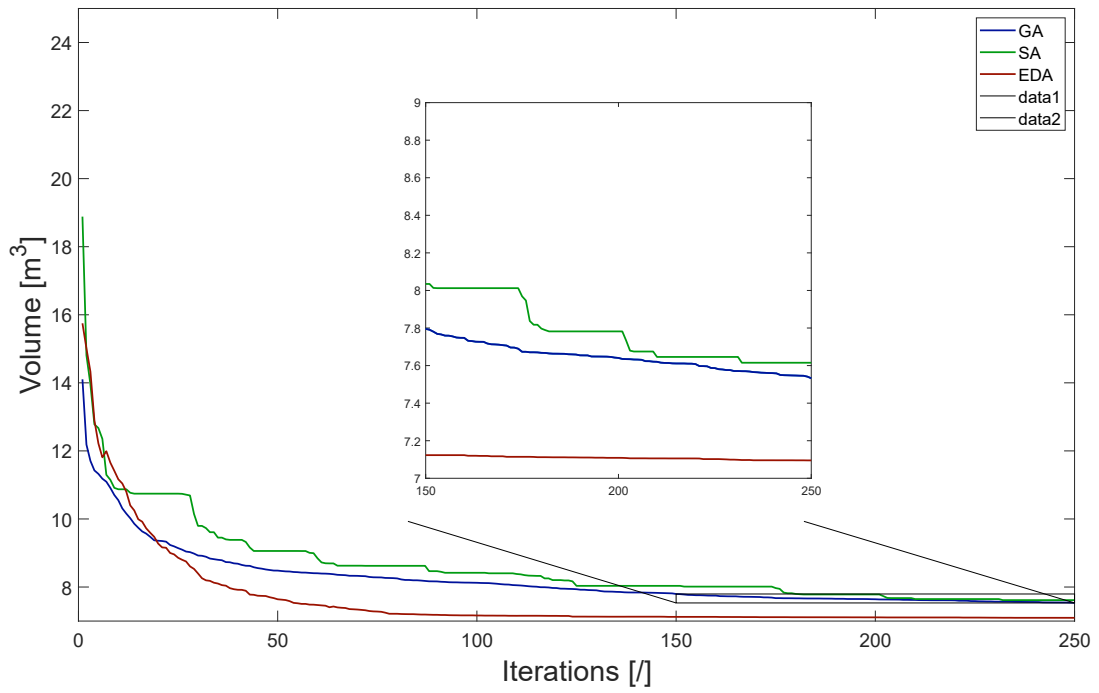
At the end of the iterative process, the best combination always remains that of $\alpha = 0,5$ and $\beta = 0,5$. From the analyses performed, it is clear that having a parametric algorithm such as EDA allows the user to choose between alpha and beta parameters and whether to prefer exploration or search. Therefore, if an evaluation of an objective function is required a few times, it is better to set large alphas. On the other hand, if more iterations are possible, it is better to set small alpha and small beta to favour the exploitation phase.

6.2.4.5. Comparison graphs and discussion of results

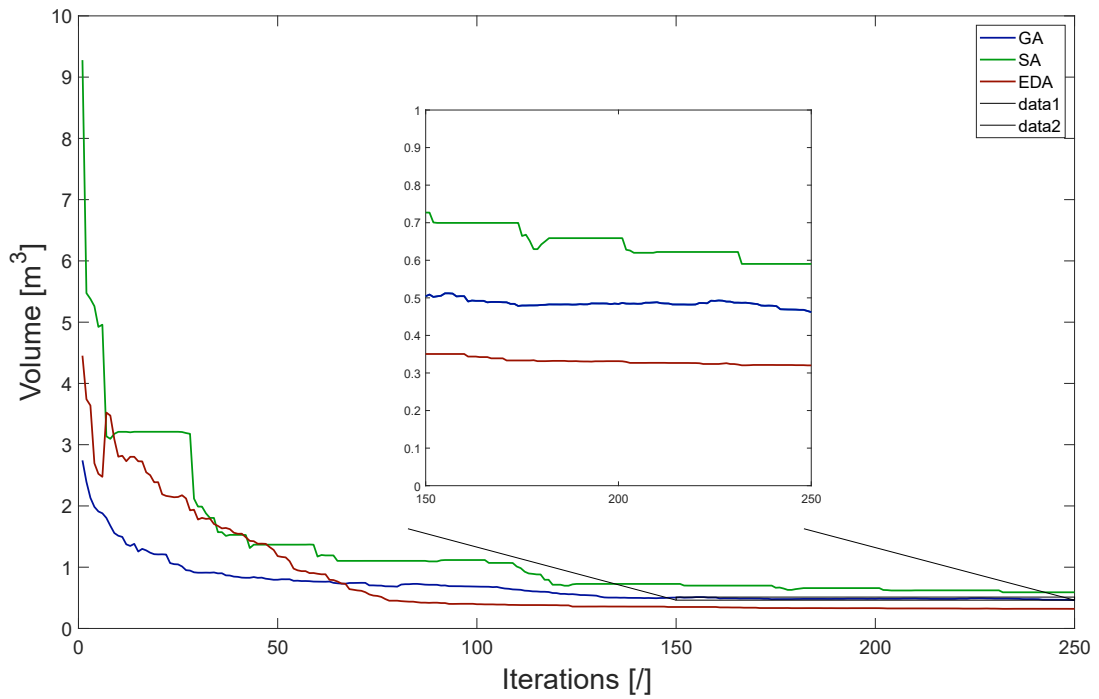


Graph 47: Comparison of means and standard deviations for different algorithms – Configuration with 5 design variables

This comparison graph shows the trends of the averages and standard deviations (shaded area). From the graph, it can be seen that all tested algorithms converge. It can be seen that EDA converges for fewer iterations than the other two and has shallow standard deviation values.



Graph 48: Comparison of the averages – Configuration with 5 design variables

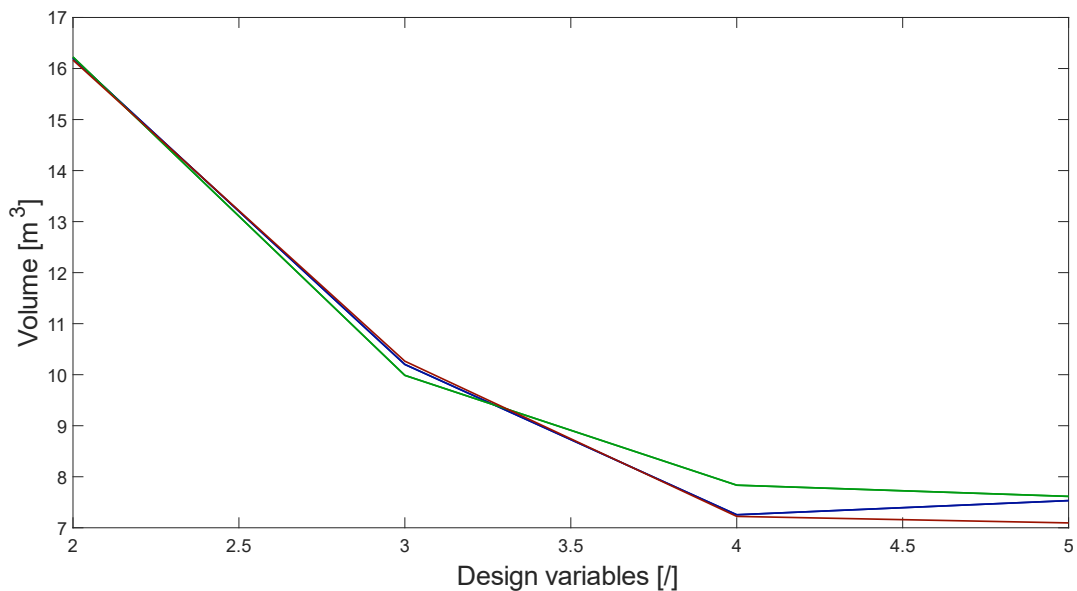


Graph 49: Comparison of standard deviations – Configuration with 5 design variables

According to both of the last two graphs comparing the mean and standard deviation of each of the three algorithms, EDA, as in the two and fourth variables configuration, is the algorithm that performs best in terms of both the value of the objective function, as well as the dispersion of the data for the objective function.

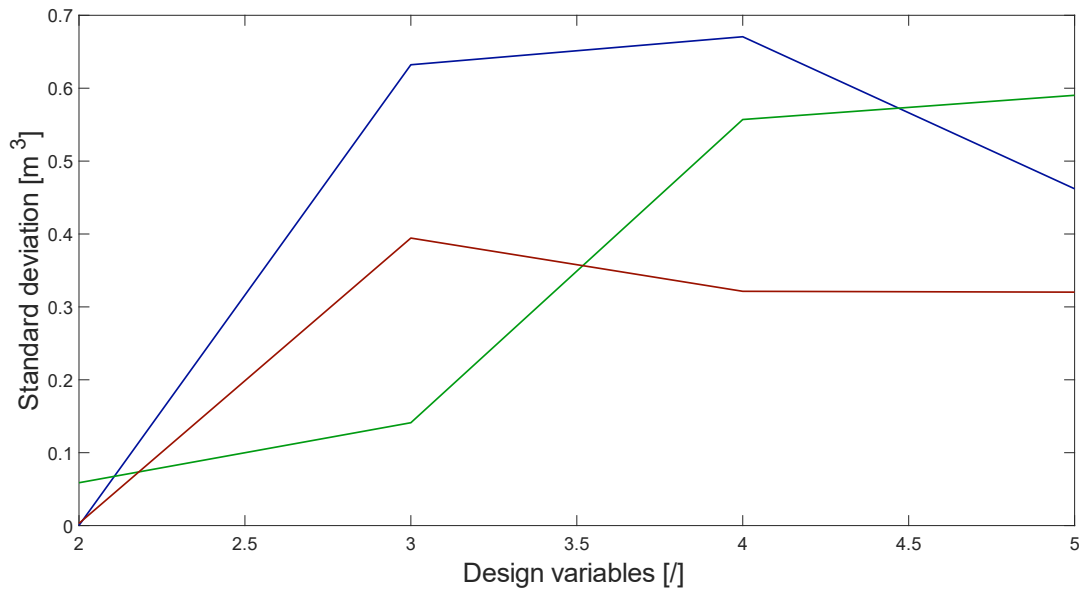
6.2.5. Comparison graphs between different configurations and discussion of results

The graph below shows the minimum value averaged over the 20 analyses conducted and reached by the objective function at the end of the iterative process for the three different algorithms tested when varying the number of design variables.



Graph 50: Comparison between different algorithms best values increasing the number of design variables

Similarly, the value of the standard deviation on the 20 analyses conducted at the end of the iterative process for the three different algorithms tested by varying the number of design variables is shown in Graph 51.



Graph 51: Comparison between different algorithms standard deviations increasing the number of design variables

Based on these results, some conclusions can be made:

- As the number of design variables increases, the value of the objective function tends to decrease except for the Genetic Algorithm when going from 4 to 5 variables, the number of iterations should probably have been increased to find a lower value;
- In terms of the value of the objective function, the EDA finds a better solution for 3 out of 4 cases;
- The two-variable configuration is the one with the lowest standard deviation values for all three algorithms, with almost equal values.
- As the number of design variables increases, the standard deviation values increase for all three algorithms, presenting values between 0,1 and 0,7.

7. Conclusion

In this master's degree thesis, a comparative study was conducted between three different metaheuristic algorithms: Genetic Algorithm (GA), Simulated Annealing (SA) and Estimation of Distribution Algorithm (EDA). Numerical results are obtained from the case study of a parametric arch structure. In particular, the optimization process was treated with the three different metaheuristic single-objective algorithms, by minimizing the structural weight and imposing a constraint condition on the Von Mises stresses along the arch. Configurations with 2, 3, 4, and 5 design variables were considered. The second objective of this thesis was to frame the entire optimisation process by creating work-flows with parametric design software and plug-ins for the structural finite element analysis and optimisation phase. In the present work the phases of parametric modelling, structural analysis and optimization have been developed entirely in the virtual environment provided by Rhino, Grasshopper, Karamba software packages. This software made it possible to conduct the entire optimisation process with the three different algorithms in a single virtual environment, eliminating the need for a script to interact between the different systems and ensuring seamless data exchange. In order to have an objective evaluation of the performance of the three different algorithms and a comparative study that is both robust and reliable, four different configurations were studied, and some 1200 independent analyses were launched. From the analyses carried out, an initial result obtained was that there was a tendency for the value of the objective function obtained to be lower and lower as the number of design variables considered increased. From the analyses carried out, an initial result was that there was a tendency for the value of the objective function obtained to be lower and lower as the number of design variables considered increased. This result is in line with expectations since as the number of design variables increases, the optimisation process has more degrees of freedom and can perform more combinations between variables to achieve a better result. On the other hand, as the number of variables considered increases, it is necessary to increase the number of iterations and, thus, the number of times the objective function is evaluated, which leads to an increase in the computational cost. In a structural optimisation process, the main problem is the fact that the evaluation of the objective function can require a high computational effort, especially in real problems with a large number of parameters. The aim is, therefore, to run as few analyses as possible, which will result in evaluating the objective function as few times as possible and yet still achieve good results. In light of the above, the new version of the EDA algorithm presented (Rosso et al., 2022) in comparison with the Genetic Algorithm and Simulated Annealing reported comparable results and on three configurations out of the four studied reported better values than the others.

Bibliography

- Aldwaik, M., Adeli, H., 2014. Advances in optimization of highrise building structures. *Struct. Multidiscip. Optim.* 50, 899–919. <https://doi.org/10.1007/s00158-014-1148-1>
- Banerjee, A., Singh, D., Sahana, S., Nath, I., 2022. Chapter 3 - Impacts of metaheuristic and swarm intelligence approach in optimization, in: Mishra, S., Tripathy, H.K., Mallick, P.K., Sangaiah, A.K., Chae, G.-S. (Eds.), *Cognitive Big Data Intelligence with a Metaheuristic Approach*. Academic Press, pp. 71–99. <https://doi.org/10.1016/B978-0-323-85117-6.00008-X>
- Beyer, H.-G., Schwefel, H.-P., 2002. Evolution strategies – A comprehensive introduction. *Nat. Comput.* 1, 3–52. <https://doi.org/10.1023/A:1015059928466>
- Christensen, P.W., Klarbring, A., 2009. *An introduction to structural optimization, Solid mechanics and its applications*. Springer, Dordrecht London.
- Cubukcuoglu, C., Ekici, B., Tasgetiren, M.F., Sariyildiz, S., 2019. OPTIMUS: Self-Adaptive Differential Evolution with Ensemble of Mutation Strategies for Grasshopper Algorithmic Modeling. *Algorithms* 12, 141. <https://doi.org/10.3390/a12070141>
- E. Atashpaz-Gargari, C. Lucas, 2007. Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition, in: 2007 IEEE Congress on Evolutionary Computation. Presented at the 2007 IEEE Congress on Evolutionary Computation, pp. 4661–4667. <https://doi.org/10.1109/CEC.2007.4425083>
- Feo, T.A., Resende, M.G.C., 1995. Greedy Randomized Adaptive Search Procedures. *J. Glob. Optim.* 6, 109–133. <https://doi.org/10.1007/BF01096763>
- Marano, G.C., Trentadue, F., Petrone, F., 2014. Optimal arch shape solution under static vertical loads. *Acta Mech.* 225, 679–686. <https://doi.org/10.1007/s00707-013-0985-0>
- Mirjalili, S., Mirjalili, S.M., Hatamlou, A., 2016. Multi-Verse Optimizer: a nature-inspired algorithm for global optimization. *Neural Comput. Appl.* 27, 495–513. <https://doi.org/10.1007/s00521-015-1870-7>
- Rashedi, E., Nezamabadi-pour, H., Saryazdi, S., 2009. GSA: A Gravitational Search Algorithm. *Spec. Sect. High Order Fuzzy Sets* 179, 2232–2248. <https://doi.org/10.1016/j.ins.2009.03.004>
- Rosso, M.M., Melchiorre, J., Cucuzza, R., Manuello, A., Marano, G.C., n.d. ESTIMATION OF DISTRIBUTION ALGORITHM FOR CONSTRAINED OPTIMIZATION IN STRUCTURAL DESIGN 12.
- Sigmund, O., 2011. On the usefulness of non-gradient approaches in topology optimization. *Struct. Multidiscip. Optim.* 43, 589–596. <https://doi.org/10.1007/s00158-011-0638-7>
- Topping, B.H.V., 1983. Shape Optimization of Skeletal Structures: A Review. *J. Struct. Eng.* 109, 1933–1951. [https://doi.org/10.1061/\(ASCE\)0733-9445\(1983\)109:8\(1933\)](https://doi.org/10.1061/(ASCE)0733-9445(1983)109:8(1933))

Index of Figures

Figure 1: Example of structural size, shape and topology optimisation.....	5
Figure 2: The mosaic of metaheuristic optimization algorithms	13
Figure 3: Flowchart of the Simulated Annealing algorithm.....	16
Figure 4: Flowchart of the Genetic Algorithm	18
Figure 5: Example of one point crossover	21
Figure 6: Example of two point crossover	21
Figure 7: Example of mutation operator	22
Figure 8: Flowchart of the EDA algorithm.....	24
Figure 9: Parametric definition of the design problem	27
Figure 10: Gaudi’s upside-down physical model of the Sagrada Familia	31
Figure 11: Moretti's Stadium model	33
Figure 12: Frei Otto experimenting with soap bubble.....	34
Figure 13: German Pavilion, Montreal Expo '67 / Frei Otto and Rolf Gutbrod	34
Figure 14: Guggenheim museum - Frank Gehry - Bilbao	35
Figure 15: Parametric model of the Guggenheim museum	36
Figure 16: El Peix - Frank Gehry – Barcelona	36
Figure 17: Guangzhou Opera House – Zaha Hadid - Cina.....	37
Figure 18: MAXXI Museum – Zaha Hadid – Rome	37
Figure 19: London Aquatics Centre - Zaha Hadid – London	38
Figure 20: Example of Rhino-Grasshopper interaction with different plugins.....	40
Figure 21: Grasshopper screenshot	41
Figure 22: Process of structural optimization	44
Figure 23: Hulme Arch Bridge – Manchester	45
Figure 24: Fully restrained arch	46
Figure 25:	47
Figure 26: Possible cross-section implemented in the optimisation scripting.....	48

Index of Graphs

Graph 1: Dynamic variation of σ as a function of β	26
Graph 2: Genetic Algorithm with 2 design variables	52
Graph 3: Simulated Annealing Algorithm with 2 design variables	53
Graph 4: Estimation of Distribution Algorithm with 2 design variables ($\alpha = 0,5$ $\beta = 0,5$)	54
Graph 5: Estimation of Distribution Algorithm with 2 design variables ($\alpha = 0,5$ $\beta = 1$)	55
Graph 6: Estimation of Distribution Algorithm with 2 design variables ($\alpha = 0,5$ $\beta = 1,5$)	55
Graph 7: Estimation of Distribution Algorithm with 2 design variables ($\alpha = 1$ $\beta = 0,5$)	56
Graph 8: Estimation of Distribution Algorithm with 2 design variables ($\alpha = 1$ $\beta = 1$)	56
Graph 9: Estimation of Distribution Algorithm with 2 design variables ($\alpha = 1$ $\beta = 1,5$)	57
Graph 10: Estimation of Distribution Algorithm with 2 design variables ($\alpha = 1,5$ $\beta = 0,5$)	57
Graph 11: Estimation of Distribution Algorithm with 2 design variables ($\alpha = 1,5$ $\beta = 1$)	58
Graph 12: Estimation of Distribution Algorithm with 2 design variables ($\alpha = 1,5$ $\beta = 1,5$)	58
Graph 13: Estimation of Distribution Algorithm with 2 design variables - Comparison of the means	59
Graph 14: Estimation of Distribution Algorithm with 2 design variables - Comparison of the Standard deviations	60
Graph 15: EDA with 2 design variables: Comparison optimal solutions and standard deviations for different α and β values – (25 iterations)	61
Graph 16: EDA with 2 design variables: Comparison optimal solutions and standard deviations for different α and β values – (50 iterations)	62
Graph 17: EDA with 2 design variables: Comparison optimal solutions and standard deviations for different α and β values – (100 iterations)	62
Graph 18: Comparison of means and standard deviations for different algorithms – Configuration with 2 design variables.....	64
Graph 19: Comparison of the averages – Configuration with 2 design variables	65
Graph 20: Comparison of standard deviations – Configuration with 2 design variables.....	65
Graph 21: Genetic Algorithm with 3 design variables	68
Graph 22: Simulated Annealing Algorithm with 3 design variables	69
Graph 23: Estimation of Distribution Algorithm with 3 design variables ($\alpha = 0,5$ $\beta = 0,5$)	70
Graph 24: Comparison of means and standard deviations for different algorithms – Configuration with 3 design variables.....	71

Graph 25: Comparison of the averages – Configuration with 3 design variables	71
Graph 26: Comparison of standard deviations – Configuration with 3 design variables.....	72
Graph 27: Genetic Algorithm with 4 design variables	74
Graph 28: Simulated Annealing Algorithm with 4 design variables	75
Graph 29: Estimation of Distribution Algorithm with 4 design variables ($\alpha = 0,5$ $\beta = 0,5$).....	76
Graph 30: Comparison of means and standard deviations for different algorithms – Configuration with 4 design variables.....	77
Graph 31: Comparison of the averages – Configuration with 4 design variables	77
Graph 32: Comparison of standard deviations – Configuration with 4 design variables.....	78
Graph 33: Genetic Algorithm with 5 design variables	80
Graph 34: Simulated Annealing Algorithm with 5 design variables	81
Graph 35: Estimation of Distribution Algorithm with 5 design variables ($\alpha = 0,5$ $\beta = 0,5$).....	82
Graph 36: Estimation of Distribution Algorithm with 5 design variables ($\alpha = 0,5$ $\beta = 1$).....	83
Graph 37: Estimation of Distribution Algorithm with 5 design variables ($\alpha = 0,5$ $\beta = 1,5$).....	84
Graph 38: Estimation of Distribution Algorithm with 5 design variables ($\alpha = 1$ $\beta = 0,5$).....	85
Graph 39: Estimation of Distribution Algorithm with 5 design variables ($\alpha = 1$ $\beta = 1$).....	86
Graph 40: Estimation of Distribution Algorithm with 5 design variables ($\alpha = 1$ $\beta = 1,5$).....	86
Graph 41: Estimation of Distribution Algorithm with 5 design variables ($\alpha = 1,5$ $\beta = 0,5$).....	87
Graph 42: Estimation of Distribution Algorithm with 5 design variables ($\alpha = 1,5$ $\beta = 1$).....	88
Graph 43: Estimation of Distribution Algorithm with 5 design variables ($\alpha = 1,5$ $\beta = 1,5$).....	89
Graph 44: EDA with 5 design variables: Comparison optimal solutions and standard deviations for different α and β values – (50 iterations)	90
Graph 45: EDA with 5 design variables: Comparison optimal solutions and standard deviations for different α and β values – (150 iterations)	91
Graph 46: EDA with 5 design variables: Comparison optimal solutions and standard deviations for different α and β values – (250 iterations)	91
Graph 47: Comparison of means and standard deviations for different algorithms – Configuration with 5 design variables.....	93
Graph 48: Comparison of the averages – Configuration with 5 design variables	94
Graph 49: Comparison of standard deviations – Configuration with 5 design variables.....	94
Graph 50: Comparison between different algorithms best values increasing the number of design variables.....	95

Graph 51: Comparison between different algorithms standard deviations increasing the number of design variables..... 96

Index of Tables

Table 1: Example of Binary Encoding..... 19

Table 2: Configuration with 2 design variables..... 51

Table 3: Configuration with 3 design variables..... 67

Table 4: Configuration with 4 design variables..... 73

Table 5: Configuration with 5 design variables..... 79